Randomise Alone, Reach as a Team

Léonard Brice[1], Thomas A. Henzinger[1], Alipasha Montaseri[1], Ali Shafiee[1], and K. S. Thejaswini[2(✉)]

[1] Institute of Science and Technology Austria, Klosterneuburg, Austria
{leonard.brice,tah,alipasha.montaseri, ali.shafiee}@ista.ac.at

[2] Université libre de Bruxelles, Bruxelles, Belgium
thejaswini.raghavan@ulb.be

Abstract. We study concurrent graph games where n players cooperate against an opponent to reach a set of target states. Unlike traditional settings, we study distributed randomisation: team players do not share a source of randomness, and their private random sources are hidden from the opponent and from each other.

We show that memoryless strategies are sufficient for the threshold problem (deciding whether there is a strategy for the team that ensures winning with probability that exceeds a threshold), a result that not only places the problem in the Existential Theory of the Reals ($\exists\mathbb{R}$) but also enables the construction of value iteration algorithms. We additionally show that the threshold problem is NP-hard. For the almost-sure reachability problem, we prove NP-completeness.

We introduce Individually Randomised Alternating-time Temporal Logic (IRATL). This logic extends the standard ATL framework to reason about probability thresholds, with semantics explicitly designed for coalitions that lack a shared source of randomness. On the practical side, we implement and evaluate a solver for the threshold and almost-sure problem based on the algorithms that we develop.

1 Introduction

Multi-agent systems provide a robust mathematical framework for modelling complex interactions across a diverse array of domains. From the formal verification of reactive systems [4,19] and cyber-physical architectures [32,37] to the study of epidemic processes [30] and distributed probabilistic programs [1], the interplay between autonomous entities is central to ensuring system correctness. In these models, such entities are typically represented as *players*, repeatedly and concurrently choosing among some set of *actions*. While classical verification often considers a single controller against an adversarial environment, there are plenty of applications that demand multi-agent models with access to varying degrees of cooperation and information.

A traditional tool to capture what coalitions of players can achieve in such settings is Alternating Temporal Logic (ATL) [2]. ATL captures specifications of

E. Darulova et al. (Eds.): CAV 2026, LNCS 16682, pp. 215–236, 2026.
https://doi.org/10.1007/978-3-032-32519-8_12

the following form: "Can a coalition achieve a given temporal objective against all strategies of the other players?" In such a setting, one need only consider deterministic strategies, since randomising among several actions does not bring any additional power. Important extensions, however, are Randomised ATL (RATL) [17] and Probabilistic ATL (PATL) [16], both of which capture notions such as almost-sure and limit-sure winning—and the latter considers threshold winning, that is, guaranteeing that the probability of winning exceeds a given threshold t. For those types of specifications, randomised strategies are strictly more powerful than deterministic ones. Nevertheless, the semantics of RATL and PATL allow coalitions to randomise their actions jointly, implicitly assuming that they can resort to shared sources of randomness, or equivalently, that they can communicate about their random choices privately. Such a hypothesis simplifies significantly the setting, since the coalition can then be seen as a single player; but it is not realistic in many real-life distributed and multi-agent systems, where challenges often lie, precisely, in the inability of the agents to communicate. This paper aims at bridging this gap.

An Example. Picture an object that needs to be moved to the other side of a sliding door. Two robots, named R2D2 and C3PO, can choose at each step to move the object either to the left or to the right, while an adversarial environment moves the sliding door, such that one among the left and the right side is open at each time step. If both R2D2 and C3PO choose the same direction as the environment, then the object moves to the other side and they win. If R2D2 and C3PO choose the same side, but the environment opens the other side then the object remains stationary, and they have to try again. If they choose different sides, the object breaks due to the opposing forces. This situation is illustrated in Fig. 1. With what probability can R2D2 and C3PO ensure reaching s_{goal}?

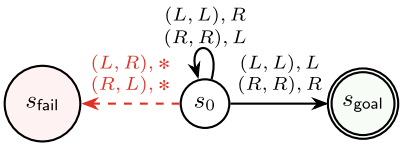

Fig. 1. The two agents choose among actions $\{L, R\}$ and the environment chooses among actions $\{L, R\}$.

Consider first the setting where R2D2 and C3PO have access to a shared source of randomness, which can't be observed by the adversary. Such games can be viewed as standard two-player concurrent games, where the team acts as a single meta-player. By playing the joint actions (L, L) or (R, R) with probability $\frac{1}{2}$ each, R2D2 and C3PO ensure that for any environmental choice, the probability of advancing is $\frac{1}{2}$, and the probability of reaching the failure state is 0. By repeating this, it is almost sure that the target will eventually be reached.

However, when R2D2 and C3PO do not share a source of randomness nor a private communication channel, they have to randomise their actions independently. One can show that against any such strategy profile, the environment

can respond in such a way that the probability of eventually reaching the target does not exceed $\frac{1}{3}$. Observe here that the severity of this result relies on the order of the quantifiers: we assume that the team fixes a collective strategy first, and that the environment responds to it, hence the value $\frac{1}{3}$ is what is called in the literature the *max-min value*. If, instead, the environment must commit to a strategy first, then R2D2 and C3PO can respond deterministically and reach the target state almost surely (with probability 1), which is called the *min-max value*. It is because the team can always coordinate on a single deterministic action (e.g., both constantly playing L or R) that the environment plays with positive probability. By this, they can avoid the failure state and ensure they eventually progress. We assume the burden of pessimism and focus on the max-min value.

Our Results. We first start with the following problem for team games.

Threshold Problem: Given a game, given a rational threshold $t \in [0, 1]$, does there exist a *collective* strategy for the team that ensures the target is reached with probability strictly greater than t?

Our first main result establishes that when there exists such a collective strategy, there exists a memoryless one (Theorem 1). This implies that these games can be fully characterised by local, state-dependent strategies. Therefore, given an instance of the threshold problem, we can construct a formula in the Existential Theory of the Reals (ETR) that is satisfiable if and only if we have a positive instance. This proves that the threshold problem lies in the complexity class $\exists\mathbb{R}$ (itself contained in PSPACE [11,36]). We also prove that the problem is NP-hard using a reduction from k-clique problem. This is unlike the situation for two-player concurrent games, where no such hardness results are known. In fact, for two-player concurrent games (or for games where the team can share randomness, which reduces to the latter) the best known lower bound for the threshold problem is SQRTSUM-hardness [18] and P-hardness derived from turn-based reachability games [24] and the best known upper bound is the complexity class $\exists\mathbb{R}$ [12]. While our encoding into ETR provides an algorithm, such large logical sentences are notoriously difficult for existing solvers to solve. To provide a more computationally accessible approach, we also develop a Value Iteration (VI) algorithm that converges to the optimal max-min value—even though it might never reach it.

Secondly, we prove NP-completeness of almost-sure reachability.

Almost-sure Problem: Given a game, does there exist a collective strategy for the team that ensures that the target is reached with probability 1?

While this problem lies in P in a two-player setting, we show that it is NP-hard even with three players. Membership in NP is obtained by proving that memoryless strategies also suffice in this setting—but the proof requires techniques different from that of the threshold problem.

Thirdly, we implement solvers for the threshold and almost-sure problems, and evaluate them on existing benchmarks that we modify to fit our setting. For the threshold problem, we first encode the entire game as a single formula in $\exists\mathbb{R}$ and use SMT solvers; while this offers strong theoretical guarantees, the formula's complexity often leads to timeouts. In contrast, we employ VI algorithms using various termination criteria and heuristics. These heuristics provide *sound under-approximations*, and terminate with results close to the actual max-min values in our benchmarks. Because VI methods query the $\exists\mathbb{R}$ solver using smaller formulas, they terminate significantly faster than solving the whole encoding. For the almost-sure case, given its NP-completeness, we utilise SAT solvers on the succinct SAT encodings that we construct. We use state-of-the-art PRISM-games to solve the threshold and the almost-sure version of the strict subcase of the games with shared randomness as a baseline. Despite solving a computationally harder game, our solvers achieve runtimes comparable to PRISM-games [27].

Finally, we propose a new logic called *Individually Randomised ATL (IRATL)* to capture the inability of a coalition to randomise collectively. The syntax of IRATL extends ATL with operators capturing the fact that a coalition can enforce an objective without using any shared randomness. For example, considering our robot team, we would write the formula $\langle\!\langle \mathsf{R2D2}, \mathsf{C3PO} \rangle\!\rangle^{\mathsf{sh}}_{\mathsf{almost}}(\Diamond\{s_{\mathsf{goal}}\})$ to check if the team can reach s_{goal} almost-surely using *shared* (sh) randomness. In contrast, to query if the team can reach the target with a probability strictly greater than 0.3 using *independent* (ind) sources of randomness, we write $\langle\!\langle \mathsf{R2D2}, \mathsf{C3PO} \rangle\!\rangle^{\mathsf{ind}}_{>0.3}(\Diamond\{s_{\mathsf{goal}}\})$. As established previously, both sentences hold true in their respective settings. We effectively solve the model-checking problem for a key fragment of this logic.

Related Work

Logics for Stochastic Team Games. Alternating-time Temporal Logic (ATL) was first presented in the work of Alur et al. [2], and has inspired a rich ecosystem of extensions. While the ATL semantics considers only deterministic moves, several of those extensions introduce randomness into the picture, such as Randomised ATL (RATL) [17], PATL [16], PAMC [38], rPATL [15], and SGL [5]. SGL and rPATL are defined only on turn-based games, where the distinction between shared and individual randomisation is not relevant. RATL, PATL, and PAMC assume that players in a coalition can resort to shared randomisation, making it possible to reduce the setting to a standard two-player zero-sum game.[1]

[1] PATL and PAMC were actually defined without this hypothesis, but a careful reading of the proofs in the cited papers shows that it is then implicitly introduced, perhaps inadvertently. This ambiguity has been propagated across the subsequent literature: all results seem to assume shared randomisation, but the definitions are sometimes consistent with that hypothesis [15] and sometimes not [41]. Let us also note that the work introducing PATL [16] claims that model-checking for PATL is in $\mathsf{NP} \cap \mathsf{coNP}$ by invoking a reduction from LTL objectives to parity ones; however, such a reduction is not polynomial-time.

Strategy logic [13] is another well-known extension of ATL, that considers only deterministic strategies. Its extension Probabilistic Strategy Logic (PSL) [3] is very expressive, and since strategies are represented as variables, individual randomisation is implicitly implied. But this expressivity comes at the cost of undecidability in the general case, and very high complexities even when the players are restricted to memoryless strategies: the easiest known fragment that subsumes our setting is only known to be in EXPSPACE.

Finally, probabilistic Resource-Bounded ATL (pRB-ATL, [34]) is another logic that subsumes our setting, with a broader semantics including limited resources for the agents. No complexity upper bound is known for pRB-ATL model-checking.

Other Multiplayer Game Settings. Several works including those on population games [6], concurrent parameterised games [7], and synthesis of coalition strategies [8], work on settings with an unknown number of players as opposed to the fixed number of players in our case. Fixing the number of players enables us to obtain value-iteration-like algorithms, more amenable to implementation. A recent work on one-shot games with partially shared randomness (various sources of randomness, each of them being accessible to a given subset of players) shows that the threshold problem with a non-strict inequality (instead of strict, as in our setting) is $\exists\mathbb{R}$-complete [10] for one-shot games. However, neither the hardness nor the easiness results extend to our setting.

Fixed Environments and Multi-agent Learning. As a side observation, we consider settings where the environment is memoryless (modelling fixed but unknown stochastic process). This setting reduces to multi-agent reinforcement learning, where agents must learn to coordinate in a stochastic environment [20,25,42].

Tools. The landscape of automated verification for multi-agent systems is rich with tools, yet, to the best of our knowledge, none natively support the independent randomisation constraint central to our work. The most prominent tool, PRISM-games [14,27], extends probabilistic model checking to the two-player version. While PRISM-games can verify properties for a coalition of players (e.g., using the logic rPATL), it treats it as a single "meta-player" capable of correlated strategies. Tools such as MCMAS [31] and EVE [21] enable strategic reasoning via ATL and Nash equilibrium synthesis. However, MCMAS focuses primarily on qualitative (Boolean) verification and epistemic properties, while EVE and related Nash solvers (e.g., PRISM-Nash) target non-zero-sum equilibria.

2 Preliminaries

2.1 Probabilities

Given a finite set S, a *probability distribution* over S is a mapping $d : S \to [0, 1]$ that satisfies the equality $\sum_{x \in S} d(x) = 1$. The set of probability distributions over S is denoted by $\mathsf{Dist}S$.

2.2 Games

We use the word *game* for team concurrent stochastic reachability games.

Definition 1 (Game Structure, Game). *A* game structure *is a tuple* $G = (\Sigma, S, (A_p)_{p\in\Sigma}, (\mathsf{Av}_p)_{p\in\Sigma}, \delta)$*, consisting of:*

- *a set* Σ *of* players*;*
- *a set* S *of* states*;*
- *for each player* p*, a set* A_p *of* actions*;*
- *for each player* p *and state* s*, a set* $\mathsf{Av}_p(s) \subseteq A_p$ *of actions available at state* s *(always assumed to be nonempty);*
- *a* transition function $\delta : S \times \prod_{p\in\Sigma} A_p \to \mathsf{Dist}(S)$ *that maps each tuple* $(s, (a_p)_p)$ *with* $a_p \in \mathsf{Av}_p(s)$ *for each* p *to a probability distribution over* S*.*

A game *is a tuple* $\mathcal{G} = (G, \Pi, T, s_0)$*, that consists of a game structure equipped with a* team $\Pi \subseteq \Sigma$*, a set* $T \subseteq S$ *of* target states*, and an* initial state $s_0 \in S$*.*

When referring to a game $\mathcal{G}$ or a game structure G, we often use the notations S, δ, etc., without recalling them. Given a set $P \subseteq \Sigma$ and a vertex s, we often write $\mathsf{Av}_P(s)$ (or $\mathsf{Av}(s)$ if $P = \Sigma$) to denote the product $\mathsf{Av}_P(s) = \prod_{p\in P} \mathsf{Av}_p(s)$. For technical reasons, we assume that the target set T is *absorbing*, i.e. that for every state $t \in T$ and action profile $\bar{a}$ we have $\delta(t, \bar{a})(t) = 1$.

We call *play* (resp. *history*) in the game structure G an infinite (resp. finite) word over the alphabet S. We sometimes say that the team *wins* the play π if that play contains a target state $s \in T$, and *loses* it otherwise.

2.3 Strategies, and Strategy Profiles

A *strategy* for player $p \in \Sigma$ is a mapping σ_p that maps each history hs to a probability distribution $\sigma_p(hs) \in \mathsf{Dist}\,\mathsf{Av}_p(s)$. A *strategy profile* for the set $P \subseteq \Sigma$ is a tuple $\bar{\sigma}_P = (\sigma_p)_{p\in P}$, where each σ_p is a strategy for player p. A *collective strategy* is a strategy profile $\bar{\sigma}_\Pi$ for the team. A *complete* strategy profile is a strategy profile $\bar{\sigma}_\Sigma$ for Σ, often written $\bar{\sigma}$, without subscript. Given two strategy profiles $\bar{\sigma}_P$ and $\bar{\tau}_Q$ with $P \cap Q = \emptyset$, we write $(\bar{\sigma}_P, \bar{\tau}_Q)$ for the strategy profile $\bar{\eta}_{P\cup Q}$, where $\eta_p = \sigma_p$ if $p \in P$, and τ_p if $p \in Q$. Given two strategy profiles denoted by $\bar{\sigma}_P$ and $\bar{\sigma}_{\Sigma\setminus P}$, we simply write $\bar{\sigma}$ for the complete strategy profile $(\bar{\sigma}_P, \bar{\sigma}_{\Sigma\setminus P})$. Given a strategy profile $\bar{\sigma}_P$ and a history h, we write $\bar{\sigma}(h)$ for the induced distribution over joint actions, i.e., for the distribution $\bar{\sigma}(h) : \bar{a}_P \mapsto \prod_{p\in P} \sigma_p(h)(a_p)$.

A complete strategy profile $\bar{\sigma}$ defines a probability distribution $\mathbb{P}_{\bar{\sigma}}$ over plays, where for each history hs and state t, we have:

$$\mathbb{P}_{\bar{\sigma}}(hstS^\omega) = \sum_{\bar{a}\in\mathsf{Av}(s)} \bar{\sigma}(hs)(\bar{a}) \cdot \delta(s, \bar{a})(t).$$

A strategy σ_p is *deterministic* if for every history h, there exists an action a such that $\sigma_p(h)(a) = 1$. It is *memoryless* if for every state s and every history h, we have $\sigma_p(hs) = \sigma_p(s)$.

2.4 Problems

Throughout this paper, our goal is to design collective strategies for teams that guarantee a certain property against every strategy of the other players. We always assume that the team players fix their strategies first, and the other players (the *opponents*) respond to it. From the opponents' perspective, responding to a collective strategy amounts to finding optimal strategies in a Markov decision process. It is a well-known result that deterministic strategies are sufficient in such a setting. Consequently, contrary to the team players, the fact that the opponents do not share a common source of randomness is not a restriction, and the opponents can be considered as a single player. For convenience, we will therefore always assume that the set $\Sigma \setminus \Pi$ is a singleton $\{\mathsf{O}\}$, where player O is called *opponent*. In such a setting, a natural question is to ask whether the team players can win the game with a probability greater than a given threshold.

Problem 1 (Threshold Problem). Given a game $\mathcal{G}$ and a threshold $t \in \mathbb{Q} \cap [0, 1]$, does there exist a collective strategy $\bar{\sigma}_\Pi$ such that for every strategy σ_O, the set T is reached with probability strictly greater than t?

A less ambitious question is to ask whether they can win with probability 1.

Problem 2 (Almost-sure Problem). Given a game $\mathcal{G}$, does there exist a collective strategy $\bar{\sigma}_\Pi$ such that for every strategy profile $\bar{\sigma}_\mathsf{O}$, the set T is reached with probability 1?

In the two next sections, we design algorithms for those two problems.

3 The Threshold Problem

In this section, we analyse the structure of the team's strategies and establish the theoretical foundations for solving Problem 1. The main result of this section is Theorem 1, which establishes that memoryless strategies are sufficient.

Building on this result, we provide a decision procedure by encoding the threshold problem as an ETR formula. Finally, we end the section with a result on NP-hardness for the threshold problem, which adds to the SQRTSUM-hardness result that is already known from the two-player case [18].

3.1 Optimality of Memoryless Strategies

In this subsection, we prove that memoryless strategies suffice to exceed a given threshold. The structure of the proof is similar to that of the same result in standard two-player zero-sum concurrent games [12]. However, the latter heavily relies on the equality of the max-min and min-max values, which no longer holds in our setting. Our proof requires therefore more technical effort.

Value Iteration. As a step toward our result, and as an algorithmic tool, we define a value iteration sequence. A *valuation* is a mapping $v : S \to [0, 1]$. We define the *predecessor operator* based on this intuition: for a fixed valuation v and a state s, we consider a *local one-shot game* played in s. In this local game, the team players choose a joint distribution over their available actions, and the opponent responds with an action as well. If t is the subsequent state, then the team gets *payoff* $v(t)$. The team players want to maximise the expected payoff, while the opponent wants to minimise it. To avoid confusion, strategies in this one-shot game are *selectors*. The predecessor operator is defined in three steps: given fixed strategies for both the team and the opponent, given a fixed team strategy against an optimal opponent, and finally, the optimal one-step value.

First, given a valuation v, a collective selector $\bar{\xi}_{\Pi}$, and an opponent selector ξ_{O}, we define the expected payoff at state s as:

$$\mathsf{Pre}_{\bar{\xi}}(v)(s) \;=\; \sum_{\bar{a} \in \mathsf{Av}_{\Pi}(s)} \sum_{b \in \mathsf{Av}_{\mathsf{O}}(s)} \sum_{t \in S} \bar{\xi}_{\Pi}(s)(\bar{a}) \cdot \xi_{\mathsf{O}}(s)(b) \cdot \delta(s, \bar{a}, b)(t) \cdot v(t).$$

We now define the value guaranteed by a fixed collective selector $\bar{\xi}_{\Pi}$:

$$\mathsf{Pre}_{\bar{\xi}_{\Pi}}(v)(s) \;=\; \inf_{\xi_{\mathsf{O}} \in \Lambda_{\mathsf{O}}(s)} \mathsf{Pre}_{\bar{\xi}}(v)(s)$$

where $\Lambda_{\mathsf{O}}(s) = \mathsf{Dist}(\mathsf{Av}_{\mathsf{O}})$ is the set of selectors for the opponent in state s.

Finally, we define the *predecessor operator* Pre, which represents the maximum value the team can guarantee in one step against the opponent:

$$\mathsf{Pre}(v)(s) \;=\; \sup_{\bar{\xi}_{\Pi} \in \Lambda_{\Pi}(s)} \inf_{\xi_{\mathsf{O}} \in \Lambda_{\mathsf{O}}(s)} \mathsf{Pre}_{\bar{\xi}}(v)(s)$$

where $\Lambda_{\Pi}(s)$ is the set of collective selectors for the team in state s.

We can then define a sequence $(v_n)_n$ of valuations as follows: the valuation v_0 maps each target state to the value 1 and each other state to the value 0, and for each n, we have $v_{n+1} = \mathsf{Pre}(v_n)$. We then have the following result.

Lemma 1. *For every vertex s, the sequence $(v_n(s))_n$ converges to the max-min value of the game, when initialised in s.*

As we will show in Sect. 5, this value iteration sequence also entails a sound under-approximation algorithm. For now, let us use this tool to prove our result.

Memoryless Strategies Are Sufficient.

Theorem 1. *Let $\mathcal{G}$ be a game and let t be a threshold. If there is a collective strategy $\bar{\sigma}_{\Pi}$ that guarantees victory with probability strictly greater than t against every opponent's strategy, then there is a memoryless one.*

Proof (Sketch). Using Lemma 1, there exists an index n such that the nth valuation, on s_0, takes a value that is greater than t. We assign a "rank" to each state corresponding to the iteration index where its value was determined. The memoryless strategy is then constructed to ensure that at each step, the next state has either a higher valuation or a lower rank, which guarantees convergence to the states with valuation 1 and with rank 0: the target states. □

A Variant: Playing Against a Memoryless Opponent. Before proceeding to the study of the threshold problem, we make a remark about a variant of our problem: the case where the opponent is restricted to memoryless strategies. This modelling choice is well-motivated by the physical reality of autonomous systems, whose environments are often fixed (but unknown) stochastic processes rather than strategic adversaries. We show here that Theorem 1 does not extend to this setting: surprisingly, restricting the opponent to memoryless strategies can force the team players to use memory to play optimally. The following example illustrates that solving this variant would require significantly different techniques from the field of multi-agent learning.

Theorem 2. *There exists a game $\mathcal{G}$ such that against an opponent restricted to memoryless strategies, the team can achieve a strictly higher probability of reaching the target using a strategy with memory than with any memoryless collective strategy.*

Proof (Sketch). We give an example in the game in Fig. 2. Intuitively, the team players can agree to wait arbitrarily long to *learn* the probability distribution during the waiting stage and then play the appropriate action to maximise the probability of reaching the target state. □

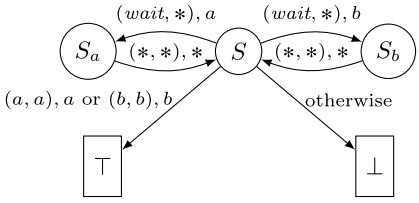

Fig. 2. An example game for opponent with no memory.

3.2 Complexity Bounds

After a short detour, we return to the setting where the opponent has (potentially) unbounded memory. We establish complexity bounds.

Membership in the Class $\exists\mathbb{R}$. We first show that the problem is in $\exists\mathbb{R}$ by constructing an equivalent ETR formula. Our construction relies on Theorem 1.

Theorem 3. *The threshold problem lies in the class $\exists\mathbb{R}$.*

Proof (Sketch). To help our reduction, we introduce λ-discounted games. These games share the same structure as concurrent games but employ a discounted payoff function parameterised by a *discount factor* $\lambda \in (0, 1)$. These games are

quantitative: if the team reaches the target set T after k steps, the team gets the payoff λ^k (instead of 1). We construct a polynomial-size ETR formula $\Psi(\lambda)$ which is satisfiable if and only if there exists a collective strategy such that the λ-discounted value exceeds t. Since this only captures the discounted version of the problem, we then show that it is equivalent to its qualitative counterpart, in the sense that there exists λ such that the formula $\Psi(\lambda)$ is satisfiable if and only if we have a positive instance of the threshold problem.

This intermediary reduction to the λ-discounted setting is necessary. A direct encoding of the reachability problem faces a hurdle: the Bellman optimality equations for reachability generally admit multiple fixed points. Consider, for example, a simple loop $s \to s$ with no path to the target. The true reachability probability is 0. However, the assignment $v(s) = 1$ trivially satisfies the Bellman equations one would write, making it a valid but spurious fixed point. One standard solution is to impose the restriction that it is the least fixed point. However, enforcing a "least fixed point" constraint would require the min operator, which is not permitted in the standard $\exists\mathbb{R}$ framework. Alternatively, identifying the non-zero support set requires guessing (if there exists a subset of vertices from which the target can be reached with nonzero probability), which puts the problem in a larger complexity class $\mathsf{NP}^{\exists\mathbb{R}}$. This makes our intermediary detour via λ-discounted settings necessary. □

NP-hardness. We finally end with a proof of NP-hardness for this problem. Note that it is also SQRTSUM-hard, following from the two-player case [18]. For the sake of completeness, we also note that the threshold problem with non-strict inequalities was recently shown to be complete for the class $\exists\mathbb{R}$ [10, Lemma 5.7].

Theorem 4. *The threshold problem is* NP*-hard, even with three states and three players.*

Proof (Sketch). We establish NP-hardness via a reduction from the *clique problem*. Given a graph G and an integer k, we construct a game with two team players and three states: an initial state, a target state, and a trap state. The team players select actions consisting of a tuple $(u, i) \in V \times \{1, \ldots, k\}$, claiming that they have a clique whose i-th vertex is u. Simultaneously, the opponent selects a pair of indices (i, j) to verify. Transitions from s_0 are defined as follows: the play reaches the target if the team plays $((u, i), (v, j))$ and the opponent selects (i, j) such that the selection is consistent with a clique ($u = v$ if $i = j$, and $(u, v) \in E$ if $i \neq j$). The game transitions to the trap state if the team matches the opponent's indices (i, j) but fails the clique consistency check ($u \neq v$ when $i = j$, or $(u, v) \notin E$ when $i \neq j$). If the team's chosen indices do not match the opponent's query, the game loops back to repeating the round. We show that if G contains a clique of size k, the team can reach the target state with probability 1. Otherwise, the maximum probability of reaching the target is bounded by $\frac{3}{4}$. The threshold problem is therefore NP-hard, even with threshold $\frac{3}{4}$. □

4 The Almost-Sure Problem

In this section, we focus on Problem 2: deciding whether the team can win almost surely. We show that this problem is NP-complete. First, in Sect. 4.1, we show that memoryless strategies are also optimal for this problem. Second, in Sect. 4.2, we prove the upper and the lower bound.

4.1 Optimality of Memoryless Strategies

In this subsection, we show that if the team can guarantee reaching the target almost-surely, it can do so using a memoryless collective strategy. Formally, we show the correctness of the following theorem. Note that it is not a consequence of Theorem 1, since the threshold problem is defined with strict inequalities.

Theorem 5. *Let $\mathcal{G}$ be a game. If there is a collective strategy $\bar{\sigma}_\Pi$ that guarantees victory with probability 1 against every opponent's strategy, then there is a memoryless one.*

Proof (Sketch). We assign a rank (a value from $\mathbb{N} \cup \{\infty\}$) to each vertex, that captures "distance" from the target: the target set T has rank 0, and the states that have rank $k+1$ are exactly those where the team can force a transition to states with rank k with positive probability, without visiting a vertex of infinite rank. We construct a memoryless collective strategy as follows. At every state, the team plays to guarantee a positive probability of moving to a lower rank, while staying within the winning set (the states with finite rank). Since the probability of reducing the rank is always strictly positive, the game is forced to eventually progress and reach rank 0 (the target) with probability 1.

4.2 Complexity Bounds

SAT Encoding for NP Membership. We show the following theorem.

Theorem 6. *The almost-sure problem is in* NP.

Proof (Sketch). Based on the results of Sect. 4.1, we know that if an almost-sure winning collective strategy exists, there exists a memoryless one. We then give an explicit SAT formula Φ that is satisfiable if and only if there exists a memoryless strategy that guarantee almost-sure reachability. The construction of Φ relies on the fact that almost-sure reachability is a qualitative property: it depends only on the *support* of the strategies, not on the precise probability values. The formula Φ encodes the search for a winning set W and the set of support of strategy for each player. The constraints enforce the existence of the ranking function as defined in the proof of Theorem 5. We also apply some optimisations to reduce the number of variables, like using a binary encoding of the rank. □

We end this section with a hardness result. The following theorem is an immediate consequence of the construction presented in the proof of Theorem 4.

Corollary 1 (of Theorem 4). *The almost-sure problem is* NP*-hard.*

5 Experimental Results

We implement two algorithms to compute the max-min value and one for the almost-sure case. The algorithm for the value problem is a global reduction that maps the game to a single ETR formula. The second algorithm uses a value iteration (VI) scheme to compute the value. For the almost-sure case, we implement one algorithm based on its SAT encoding. To establish a baseline and contextualise our runtime, we also compare our approach with the state-of-the-art PRISM-games [27], which solves the two-player version of concurrent games. PRISM-games is particularly relevant as it handles both the quantitative (max-min value) and qualitative (almost-sure) versions of this problem, but where the team players have access to shared randomness. PRISM-games, therefore, solves a strict and computationally easier subcase of our setting. The source code is available at https://github.com/alipashamontaseri/Team-Concurrent-Game.

5.1 Algorithms

We evaluate three algorithms for the max-min value. The first is a global reduction. The second is a value iteration framework, for which we provide three distinct implementations. Each implementation provides a *sound under-approximation* of the value. The third is a baseline based on PRISM-games [27].

Algorithm 1: ETR-Direct. This method implements the reduction from Sect. 3.2. It translates the entire game into a single ETR formula. It then computes the game value using binary search combined with SMT solvers.

Algorithm 2: Value iteration (VI). ETR-Direct scales poorly on games with many states. Therefore, we use the value iteration scheme from Sect. 3.1. This method updates values iteratively by solving local one-shot games. In these local games, we calculate the highest probability the team can guarantee to gain in a single step, based on the current value estimates of the states. Based on this, we update the estimates of state values. The value iteration algorithm is a lower bound for the max-min value of the game, and improves it every step. Therefore, the values act as a sound under-approximation. We propose three implementations for solving these local instances:

1. **VI-ETR:** This implementation uses an SMT solver to compute each local instance reliably. This guarantees precision but may take longer on computationally difficult instances.
2. **VI-OPT:** This implementation treats each one-shot game as a non-linear optimisation problem to be faster. We use Sequential Least Squares Programming (SLSQP) [26] to solve them. Because the constraints are smooth (in C^∞), the solver uses analytical gradients. This method yields an under-approximation of the value because the solver may converge to a local optimum rather than the global one. However, our experiments show that the computed value remains close to the true value.

3. **VI-Hybrid:** This implementation combines SLSQP optimisation with exact solving. For each local game, we first generate a candidate value v^* using SLSQP. We then query the SMT solver to verify if a value $v \geq v^* + \varepsilon$ is feasible. If the query is unsatisfiable, we accept v^* as the global optimum. If satisfiable (indicating v^* is a local optimum), we solve the instance with binary search using v^* as a lower bound. Thus, this reliably computes the value of the game.

Algorithm 3: Quantitative PRISM (Baseline). We include the state-of-the-art algorithm from PRISM-games [27]. It assumes that team players share randomness (whereas our setting assumes that team players do not share randomness). Our setting is more general, as shared randomness can be modelled by a single player.

The optimisation strategy used in VI-OPT and VI-Hybrid (SLSQP) is explained in detail in the full version of the paper [9].

For the almost-sure reachability problem, we evaluate two algorithms:

Algorithm 4: SAT-Direct. This method implements the encoding described in Sect. 4.2. It translates the almost-sure problem into a SAT. We then use a SAT solver to determine if an almost-surely winning collective strategy exists. This algorithm always gives a verification guarantee.

Algorithm 5: Qualitative PRISM (Baseline). We use the qualitative verification engine of PRISM-games [27] as a baseline. Similar to Algorithm 3, this tool solves the almost-sure problem under the assumption of shared randomness, which is only a comparison point for our general setting.

Implementation. We implemented a prototype solver in Python 3.11.14 to evaluate these algorithms. We conducted all experiments on a machine running Ubuntu 24.04 LTS, equipped with an Intel Core Ultra 5 225U processor and 16 GB of RAM. We set a timeout of 600 s for each algorithm on each instance. For the max-min problem, we perform binary search until we achieve a precision of $\varepsilon = 10^{-4}$.

1. **SMT solver:** We use the Z3 SMT solver [33] for all ETR queries. This handles Algorithm 1, and the verification steps in the value iteration variants.
2. **SLSQP optimiser:** We use the SLSQP algorithm provided by SciPy [40]. This optimiser solves the local games in VI-OPT and generates the candidate values for VI-Hybrid.
3. **Stopping criteria:** We terminate the value iteration when the values stabilise. We stop the process when the maximum change across all states drops below $\varepsilon' = 10^{-4}$ and report the final values.
4. **SAT solver:** For the almost-sure reachability problem (Algorithm 4), we use the `minisat22` SAT solver [39] provided by the PySAT library.

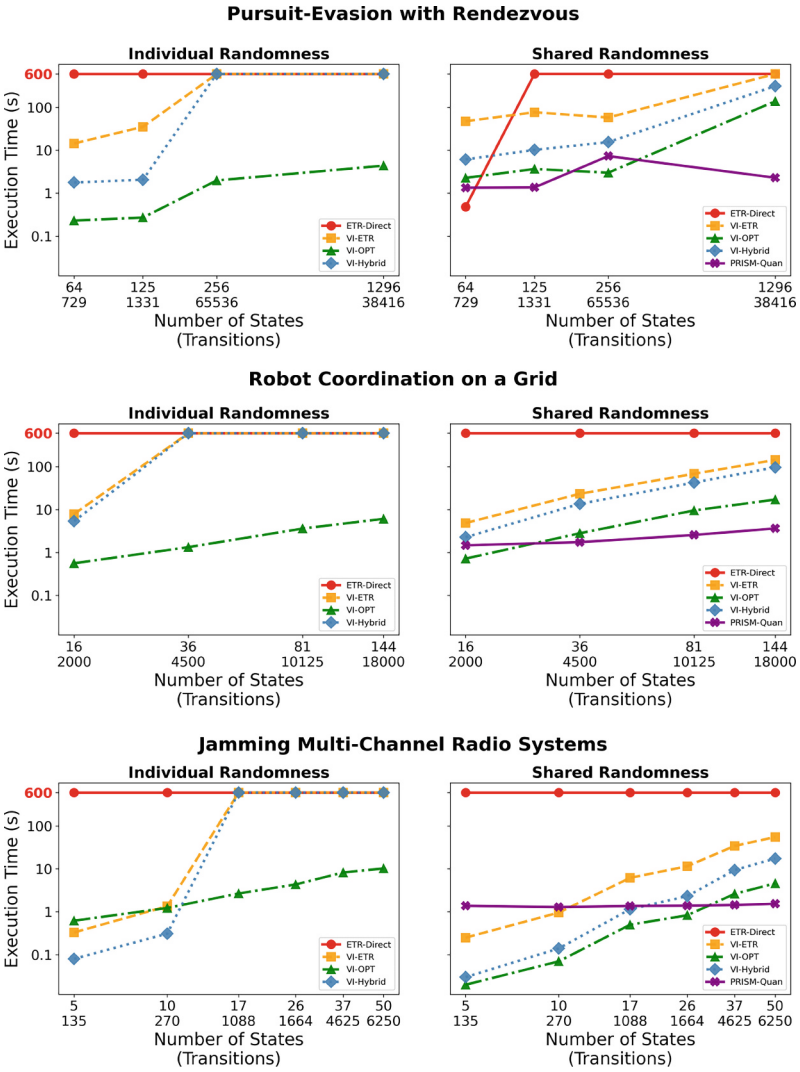

Fig. 3. Execution time of the algorithms across three benchmarks. The plots report execution time in seconds vs. the state space size: both in log scale.

5.2 Benchmarks

We evaluate our algorithms on three distinct benchmarks. The first is Pursuit-Evasion with Rendezvous, a variant of the game from [35]. In this setting, a cooperative team of agents must meet at *one vertex* in a directed graph while avoiding a pursuer. The second is Robot Coordination, adapted from PRISM-games [28], where a team of robots navigates a grid against adversarial wind conditions to reach a target configuration. The third is Jamming Multi-Channel

Radio Systems, also adapted from PRISM-games [28], which models sensors transmitting packets over frequency channels subject to adversarial jamming and collisions. We provide formal definitions, dynamics, and objective functions for all three benchmarks in the full version.

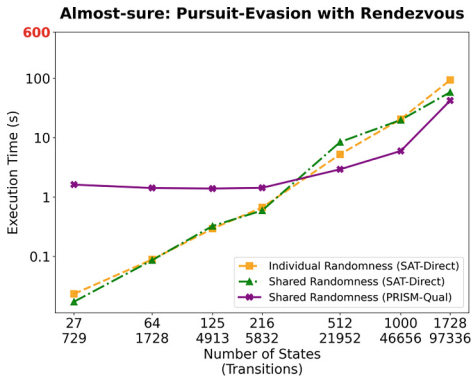

Fig. 4. The plot compares the execution time (in seconds, logarithmic scale) of SAT-Direct under individual and shared randomness constraints against the baseline qualitative solver (which assumes shared randomness).

5.3 Results

We evaluate our algorithms in two settings. The first is the general case of *individual randomness*. The second is the special case of *shared randomness*. In the latter, we model the team as a single "meta-player" with joint actions, which allows us to compare our results against PRISM-games. Detailed numerical data is provided in the full version [9].

Max-Min Problem. Across both settings, **ETR-Direct** fails to solve even small instances within the timeout. In the case of individual randomness (Fig. 3, left), the exact and hybrid VI approaches **VI-ETR**, **VI-Hybrid** solve small instances, but their execution time increases drastically as the state space grows. Conversely, the optimisation-based variant **VI-OPT** scales effectively, solving the largest instances across all benchmarks within the time limit.

In terms of precision, **VI-OPT** yields tight under-approximations in both settings, consistently converging to values very close to the exact solutions. While PRISM outperforms our general-purpose solvers in the special case of shared randomness (Fig. 3, right), **VI-OPT** stays competitive.

Almost-Sure Problem. We analyse the Pursuit-Evasion benchmark. We selected this instance because it is the only one where the team has non-trivial strategies to win with probability 1. Figure 4 presents the results.

First, we look at the case of *shared randomness*. Here, our **SAT-Direct** algorithm performs well compared to PRISM-games. On smaller instances (Scenarios 1 to 4), our method is actually faster than PRISM because it has less overhead. In the largest instances (Scenarios 6 and 7), PRISM scales slightly better, but our solver remains competitive, despite solving a more difficult problem.

Second, we look at the case of *individual randomness*. This problem is much harder because the players cannot coordinate their actions. As expected, the time needed to find a solution grows much faster than in the shared case. Despite this added difficulty, **SAT-Direct** is successful. It manages to solve large games with over 97,000 transitions within the time limit.

6 Towards Individually Randomised ATL

We conclude this paper by proposing a new logic to capture our concept of individual randomisation. A classical tool to define specifications over reactive systems is the *Alternating Temporal Logic* (ATL) [2], in which one can encode the fact that a coalition of players can guarantee surely an objective that is expressible in Linear Temporal Logic (LTL). Since only sure guarantees are considered here, the semantics of ATL need not consider randomised strategies. This changed with the introduction of *Randomised ATL* (RATL) [17] (and PATL [16]). There, sure guarantees are extended with almost-sure and limit-sure ones, and the semantics allows players to use randomness. For example, the formula $\langle\!\langle C \rangle\!\rangle_{\mathsf{almost}} \Box q$ means that the coalition C has a "collective strategy" to ensure that every state in the play satisfies q with probability 1. However, the RATL semantics defines a collective strategy for the coalition C as a mapping from histories to distributions over the whole set of tuples of actions available for the players in C, implicitly assuming that those players can randomise based on a secret common source of randomness—which can then be reduced to a 2-player setting. Our notion of collective strategy, where players in a coalition can design their strategy together but randomisation is an individual process, cannot be captured by this logic. In this section, we introduce *Individually Randomised ATL* (or IRATL), whose semantics bridge this gap.

Syntax. Our logic is defined to extend the logic RATL.

Definition 2 (IRATL Formula). *Let Q be a set of atoms. Let Σ be a set of players. An* IRATL formula *is either:*

- *a proposition $q \in Q$;*
- *or a formula $\neg\varphi$ or $\varphi \vee \psi$, where φ and ψ are IRATL formulas;*
- *or a formula $\langle\!\langle C \rangle\!\rangle^r_w \circ \varphi$, $\langle\!\langle C \rangle\!\rangle^r_w \Box \varphi$, or $\langle\!\langle C \rangle\!\rangle^r_w \varphi \mathsf{U} \psi$, where $C \subseteq \Sigma$ is a coalition of players, where $r \in \{\mathsf{sh}, \mathsf{ind}\}$ is a* randomisation type*, and where $w \in \{\mathsf{sure}, \mathsf{almost}, \mathsf{limit}\} \cup \{> t, \geq t \mid t \in [0, 1)\}$ is a* winning condition*, and where φ and ψ are IRATL formulas.*

The operators $\langle\!\langle\rangle\!\rangle^r_w$ are called *path quantifiers*, the operators $\bigcirc$, $\Box$, and U are called *temporal operators*, and subformulas of the form $\bigcirc\varphi$, $\Box\varphi$ or $\varphi\mathsf{U}\psi$ are called *path formulas*.

Semantics. Given a game structure G, a set of atomic propositions Q, and a labelling function $L : S \mapsto 2^Q$, we define the semantics of IRATL by cross-induction as follows. Given a state $s \in S$, we define:

- $s \models q$ if we have $q \in L(s)$;
- $s \models \neg\varphi$ if we do not have $s \models \varphi$, and $s \models \varphi \vee \psi$ if we have $s \models \varphi$ or $s \models \psi$;
- $s \models \langle\!\langle C\rangle\!\rangle^{\mathsf{ind}}_{\mathsf{sure}}\theta$ if in the game structure G, from the state s, there exists a strategy profile $\bar{\sigma}_C$ such that for every strategy profile $\bar{\sigma}_{\Sigma\setminus C}$ and every play π compatible with the strategy profile $\bar{\sigma}$, we have $\pi \models \theta$;
- $s \models \langle\!\langle C\rangle\!\rangle^{\mathsf{ind}}_{\mathsf{almost}}\theta$ if in the game structure G, from the state s, there exists a strategy profile $\bar{\sigma}_C$ such that for every strategy profile $\bar{\sigma}_{\Sigma\setminus C}$, we have $\mathbb{P}_{\bar{\sigma}}\{\pi \in S^\omega \mid \pi \models \theta\} = 1$;
- $s \models \langle\!\langle C\rangle\!\rangle^{\mathsf{ind}}_{\mathsf{limit}}\theta$ if in the game structure G, from the state s, for every $\varepsilon > 0$, there exists a strategy profile $\bar{\sigma}_C$ such that for every strategy profile $\bar{\sigma}_{\Sigma\setminus C}$, we have $\mathbb{P}_{\bar{\sigma}}\{\pi \in S^\omega \mid \pi \models \theta\} > 1 - \varepsilon$;
- $s \models \langle\!\langle C\rangle\!\rangle^{\mathsf{ind}}_{>t}\theta$ (resp. $\langle\!\langle C\rangle\!\rangle^{\mathsf{ind}}_{\geq t}\theta$) if in the game structure G, from the state s, there exists a strategy profile $\bar{\sigma}_C$ such that for every strategy profile $\bar{\sigma}_{\Sigma\setminus C}$, we have $\mathbb{P}_{\bar{\sigma}}\{\pi \in S^\omega \mid \pi \models \theta\} > t$ (resp. $\geq t$);
- $s \models \langle\!\langle C\rangle\!\rangle^{\mathsf{sh}}_{w}\theta$ is defined analogously, but in the game structure G_C, in which all C players have been merged into one player who can pick an action profile at each state.

On the other hand, given a play $\pi = \pi_0\pi_1\ldots$, we define:

- $\pi \models \bigcirc\varphi$ if we have $\pi_1\pi_2\ldots \models \varphi$;
- $\pi \models \Box\varphi$ if for every $n \in \mathbb{N}$, we have $\pi_n \models \varphi$;
- $\pi \models \varphi\mathsf{U}\psi$ if there exists $n \in \mathbb{N}$ such that we have $\pi_n \models \psi$, and $\pi_k \models \varphi$ for every $k < n$.

Other classical symbols can be derived: for example, the formula $\top$ is short for $q \vee \neg q$ for some q, and the formula $\Diamond\varphi$ for $\top\mathsf{U}\varphi$.

Model-Checking. The algorithms presented in the previous sections enable us to define a decidable fragment of IRATL.

Theorem 7. *Given a game structure G, a state s and a formula φ that does not contain the symbols $\langle\!\langle C\rangle\!\rangle^{\mathsf{ind}}_{\mathsf{limit}}$, $\langle\!\langle C\rangle\!\rangle^{\mathsf{ind}}_{\geq t}$, or $\langle\!\langle C\rangle\!\rangle^{\mathsf{ind}}_{>t}\Box$, deciding whether in the game G we have $s \models \varphi$ can be done in* PSPACE*, and in polynomial time if the size of the game structure is fixed.*

Proof. As a preliminary remark, let us remind that when a coalition C seeks to achieve an objective θ against all strategies of the other players, those can be modelled by a single opponent, as they do not need any form of randomisation to respond to the coalition's strategy. We can then apply the following recursive algorithm. Given an IRATL formula φ:

- if $\varphi = q, \neg\psi$, or $\psi \vee \chi$, treat it recursively as expected.
- If $\varphi = \langle\!\langle C \rangle\!\rangle^{\mathsf{sh}}_{w}\theta$, then apply the same recursive operations as for RATL [17] and PATL [16].
- If $\varphi = \langle\!\langle C \rangle\!\rangle^{\mathsf{ind}}_{\mathsf{sure}}\theta$: since the coalition must surely reach the target, randomising is useless, and the coalition can then be seen as a single player. We can therefore apply the algorithms cited at the previous point.
- If $\varphi = \langle\!\langle C \rangle\!\rangle^{\mathsf{ind}}_{\mathsf{almost}}\Box\psi$: this formula is semantically equivalent to $\langle\!\langle C \rangle\!\rangle^{\mathsf{ind}}_{\mathsf{sure}}\Box\psi$. We can therefore apply the previous point.
- If $\varphi = \langle\!\langle C \rangle\!\rangle^{\mathsf{ind}}_{\mathsf{almost}}\mathsf{O}\psi$ (resp. $\langle\!\langle C \rangle\!\rangle^{\mathsf{ind}}_{>t}\mathsf{O}\psi$), first compute the set T of states that satisfy the formula ψ. Then, consider the game in which the team C wants to reach the set T, and all states except s are made absorbing. Then, decide whether the team can guarantee reaching the set T almost surely (resp. with probability greater than t), using Theorem 3 (resp. Theorem6).
- If $\varphi = \langle\!\langle C \rangle\!\rangle^{\mathsf{ind}}_{\mathsf{almost}}\psi\mathsf{U}\chi$ (resp. $\langle\!\langle C \rangle\!\rangle^{\mathsf{ind}}_{>t}\psi\mathsf{U}\chi$), first compute the sets U and T of states that satisfy the formulas ψ and χ, respectively. Consider the game in which all states $s' \in S \setminus U$ are made absorbing. Then, decide whether in this game, the team C can guarantee reaching the set T almost surely (resp. with probability greater than t), using Theorem 3 (resp. Theorem 6).

The size of the recursion stack is bounded by the size of the formula, and each recursion consists in a polynomial-space algorithm, hence the conclusion. □

The decidability of the full logic IRATL will be obtained if one can handle the two following problems, which we leave as open questions.

Problem 3 (Limit-sure Problem). Given a game $\mathcal{G}$, does it hold that for every $\varepsilon > 0$, there exists a collective strategy $\bar{\sigma}_{\Pi}$ such that for any strategy σ_{O}, the probability of reaching the set T under the strategy profile $\bar{\sigma}$ is greater than $1-\varepsilon$?

Problem 4 (Safety Problems). Given a game $\mathcal{G}$, can the team guarantee (with probability greater than a given threshold, or limit-surely) that the set T is *not* eventually reached?

The reader may already be convinced that the limit-sure problem requires new techniques, but believe that the safety threshold problem can be treated in a way analogous to the reachability threshold problem. This is not the case, because Theorem 1 does not extend to safety games: as shown in the work of de Alfaro et al. [17, Theorem 8]. Even in a two-player setting, infinite memory may be required to guarantee that a safety objective is satisfied with positive probability. However, in that setting, this problem can be decided by taking the perspective of the opponent—by asking whether the opponent has a strategy to make the player lose with probability at least $1 - t$. Such arguments no longer work in our setting, since the max-min value does not equal the min-max value. Hence, the safety threshold problem seems to be more challenging than its reachability counterpart.

7 Discussion

We introduced the logic IRATL, that extends the well-studied ATL to support cases where players randomise independently without a common coin. We showed that this distinction is not just semantic but requires new algorithms.

IRATL describes games where players form teams, but must randomise individually. The exact complexity of solving these games is left as an open problem. We show $\exists\mathbb{R}$-easiness and NP-hardness for the threshold problem, which is also SQRTSUM-hard, following from the two-player case [18]. It would be interesting to know the exact complexity of this problem.

Second, from a practical perspective, our current value iteration method relies on asymptotic convergence without a certified stopping criterion. Even in the simpler two-player concurrent setting, it is known that a double-exponential number of iterations might be required to approximate the value to within an ε-approximate value [22]. A promising avenue for future work is to extend recent results on stopping criteria for concurrent games [29]. By developing techniques to approximate the value from both above and below, we could rigorously guarantee ε-optimality upon termination. We leave as future work the integration into established model checkers such as PRISM-games [27] or STORM [23] enabling model checking tools that accurately model decentralised systems.

Acknowledgments. This work is a part of project VAMOS that has received funding from the European Research Council (ERC), grant agreement No 101020093. Part of this work was realised when the first author was an FNRS aspirant at Université libre de Bruxelles.

Data Availibility Statement. The artifact can be accessed at the link: https://doi.org/10.5281/zenodo.19680359.

The source code is available at: https://github.com/alipashamontaseri/Team-Concurrent-Game.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. de Alfaro, L., Henzinger, T.A., Jhala, R.: Compositional methods for probabilistic systems. In: CONCUR 2001 – Concurrency Theory, pp. 351–365. Springer, Berlin Heidelberg, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44685-0_24
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002). https://doi.org/10.1145/585265.585270
3. Aminof, B., Kwiatkowska, M., Maubert, B., Murano, A., Rubin, S.: Probabilistic strategy logic, pp. 32–38 (2019). https://doi.org/10.24963/ijcai.2019/5
4. Attoui, A.: Multi-agent based method for reactive systems formal specification and validation. IFAC Proc. Vol. **29**(2), 1–6 (1996). https://doi.org/10.1016/S1474-6670(17)43768-2. 5th IFAC/IFIP/GI/GMA Workshop on Experience With the Management of Software Projects 1995 (MSP '95)

5. Baier, C., Brázdil, T., Größer, M., Kučera, A.: Stochastic game logic. Acta Inf. **49**(4), 203–224 (2012). https://doi.org/10.1007/s00236-012-0156-0
6. Bertrand, N., Bouyer, P., Lapointe, L., Mascle, C.: Reach together: how populations win repeated games. CoRR abs/2510.02984 (2025). https://doi.org/10.48550/ARXIV.2510.02984
7. Bertrand, N., Bouyer, P., Majumdar, A.: Concurrent parameterized games. In: FSTTCS 2019. LIPIcs, vol. 150, pp. 31:1–31:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPICS.FSTTCS.2019.31
8. Bertrand, N., Bouyer, P., Majumdar, A.: Synthesizing safe coalition strategies. In: FSTTCS 2020. LIPIcs, vol. 182, pp. 39:1–39:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPICS.FSTTCS.2020.39
9. Brice, L., Henzinger, T.A., Montaseri, A., Shafiee, A., Thejaswini, K.S.: Randomise alone, reach as a team. CoRR abs/2603.07094 (2026). https://doi.org/10.48550/ARXIV.2603.07094
10. Brice, L., Henzinger, T.A., Thejaswini, K.S.: Dicey games: shared sources of randomness in distributed systems (2026). https://doi.org/10.48550/arXiv.2601.18303
11. Canny, J.: Some algebraic and geometric computations in PSPACE. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing STOC, pp. 460–467. STOC '88, Association for Computing Machinery (1988). https://doi.org/10.1145/62212.62257
12. Chatterjee, K., de Alfaro, L., Henzinger, T.A.: Strategy improvement for concurrent reachability and safety games. CoRR abs/1201.2834 (2012). https://doi.org/10.1016/j.jcss.2012.12.001, https://arxiv.org/abs/1201.2834. preprint version of the combined QEST/SODA results
13. Chatterjee, K., Henzinger, T.A., Piterman, N.: Strategy logic. Inf. Comput. **208**(6), 677–693 (2010). https://doi.org/10.1016/j.ic.2009.07.004. https://www.sciencedirect.com/science/article/pii/S0890540110000192. special Issue: 18th International Conference on Concurrency Theory (CONCUR 2007)
14. Chen, T., Forejt, V., Kwiatkowska, M.Z., Parker, D., Simaitis, A.: Prism-games: a model checker for stochastic multi-player games. In: TACAS 2013. Lecture Notes in Computer Science, vol. 7795, pp. 185–191. Springer (2013). https://doi.org/10.1007/978-3-642-36742-7_13
15. Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 315–330. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28756-5_22
16. Chen, T., Lu, J.: Probabilistic alternating-time temporal logic and model checking algorithm. In: Lei, J. (ed.) FSKD 2007, pp. 35–39. IEEE Computer Society (2007). https://doi.org/10.1109/FSKD.2007.458
17. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. Theoret. Comput. Sci. **386**(3), 188–217 (2007). https://doi.org/10.1016/j.tcs.2007.07.008. expressiveness in Concurrency
18. Etessami, K., Yannakakis, M.: Recursive markov decision processes and recursive stochastic games. J. ACM **62**(2) (2015). https://doi.org/10.1145/2699431
19. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems, pp. 53–113. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21455-4_3
20. Guestrin, C., Koller, D., Parr, R.: Multiagent planning with factored MDPs. In: Advances in Neural Information Processing Systems (NIPS). vol. 14, pp. 1523–1530. MIT Press (2002). https://doi.org/10.5555/2980539.2980737

21. Gutierrez, J., Najib, M., Perelli, G., Wooldridge, M.J.: EVE: a tool for temporal equilibrium analysis. In: Lahiri, S.K., Wang, C. (eds.) Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, 7–10 October 2018, Proceedings. Lecture Notes in Computer Science, vol. 11138, pp. 551–557. Springer (2018). https://doi.org/10.1007/978-3-030-01090-4_35
22. Hansen, K.A., Ibsen-Jensen, R., Miltersen, P.B.: The complexity of solving reachability games using value and strategy iteration. In: Computer Science - Theory and Applications. Springer (2011). https://doi.org/10.1007/978-3-642-20712-9_7
23. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker storm. Int. J. Softw. Tools Technol. Transf. **24**(4), 589–610 (2022). https://doi.org/10.1007/s10009-021-00633-z
24. Immerman, N.: Number of quantifiers is better than number of tape cells. J. Comput. Syst. Sci. **22**, 384–406 (1981). https://doi.org/10.1016/0022-0000(81)90039-8, https://api.semanticscholar.org/CorpusID:40753086
25. Kok, J.R., Vlassis, N.: Collaborative multiagent reinforcement learning by payoff propagation. J. Mach. Learn. Res. (JMLR) **7**(65), 1789–1828 (2006). https://jmlr.org/papers/v7/kok06a.html
26. Kraft, D.: A software package for sequential quadratic programming. Tech. Rep. Forschungsbericht FB 88–28, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (DFVLR), Köln, Germany (1988). https://degenerateconic.com/uploads/2018/03/DFVLR_FB_88_28.pdf
27. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Prism-games 3.0: stochastic game verification with concurrency, equilibria and time. In: Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, 21–24 July 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12225, pp. 475–487. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_25
28. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Automatic verification of concurrent stochastic systems. Formal Methods Syst. Design, 188–250 (2021). https://doi.org/10.1007/s10703-020-00356-y
29. Křetínský, J., Meggendorfer, T., Weininger, M.: Stopping criteria for value iteration on stochastic games with quantitative objectives. In: 2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 1–14 (2023). https://doi.org/10.1109/LICS56636.2023.10175771
30. Lefèvre, C.: Optimal control of a birth and death epidemic process. Oper. Res. **29**(5), 971–982 (1981). https://doi.org/10.1287/opre.29.5.971, http://www.jstor.org/stable/170234
31. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: an open-source model checker for the verification of multi-agent systems. Int. J. Softw. Tools Technol. Transf. **19**(1), 9–30 (2017). https://doi.org/10.1007/S10009-015-0378-X
32. Lyu, G., Fazlirad, A., Brennan, R.W.: Multi-agent modeling of cyber-physical systems for IEC 61499 based distributed automation. Procedia Manufact. **51**, 1200–1206 (2020). https://doi.org/10.1016/j.promfg.2020.10.168. 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)
33. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
34. Nguyen, H., Rakib, A.: A probabilistic logic for resource-bounded multi-agent systems, pp. 521–527 (2019). https://doi.org/10.24963/ijcai.2019/74

35. Parsons, T.D.: Pursuit-evasion in a graph. In: Theory and Applications of Graphs: Proceedings, Michigan 11–15 May 1976, pp. 426–441. Springer (2006). https://doi.org/10.1007/BFb0070400
36. Schaefer, M., Cardinal, J., Miltzow, T.: The existential theory of the reals as a complexity class: a compendium. CoRR abs/2407.18006 (2024). https://doi.org/10.48550/ARXIV.2407.18006
37. Shi, D., Elliott, R.J., Chen, T.: On finite-state stochastic modeling and secure estimation of cyber-physical systems. IEEE Trans. Autom. Control **62**(1), 65–80 (2016). https://doi.org/10.1109/TAC.2016.2541919
38. Song, F., Chen, T., Tang, Y., Xu, Z.: Probabilistic alternating-time μ-calculus. In: Proceedings of the AAAI Conference on Artificial Intelligence vol. 33, pp. 6179–6186 (2019). https://doi.org/10.1609/aaai.v33i01.33016179
39. Sorensson, N.: Minisat 2.2 and minisat++ 1.1. http://baldur.iti.uka.de/sat-race-2010/descriptions/solver_25+26.pdf (2010). https://www.semanticscholar.org/paper/MINISAT-2.2-and-MINISAT%2B%2B-1.1-S%C3%B6rensson/e46a90899c7ca924c30983cf4cede5804df7ab7f
40. Virtanen, P., et al.: SciPy 1.0 Contributors: SciPy 1.0: Fundamental algorithms for scientific computing in python. Nat. Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
41. Zhang, C., Pang, J.: On probabilistic alternating simulations. In: Calude, C.S., Sassone, V. (eds.) TCS 2010. IAICT, vol. 323, pp. 71–85. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15240-5_6
42. Zhang, K., Yang, Z., Başar, T.: Multi-agent reinforcement learning: a selective overview of theories and algorithms. In: Handbook of Reinforcement Learning and Control, pp. 321–384 (2021). https://doi.org/10.1007/978-3-030-60990-0_12