



# Quantitative verification and strategy synthesis for stochastic games<sup>☆</sup>



Mária Svorenová, Marta Kwiatkowska<sup>\*</sup>

Department of Computer Science, University of Oxford, Oxford, UK

## ARTICLE INFO

### Article history:

Received 19 January 2016

Received in revised form

22 April 2016

Accepted 23 April 2016

Recommended by A. Astolfi

Available online 11 May 2016

### Keywords:

Stochastic games

Controller synthesis

Quantitative verification

Temporal logic

Multi-objective properties

## ABSTRACT

Design and control of computer systems that operate in uncertain, competitive or adversarial, environments can be facilitated by formal modelling and analysis. In this paper, we focus on analysis of complex computer systems modelled as turn-based  $2\frac{1}{2}$ -player games, or stochastic games for short, that are able to express both stochastic and non-stochastic uncertainties. We offer a systematic overview of the body of knowledge and algorithmic techniques for verification and strategy synthesis for stochastic games with respect to a broad class of quantitative properties expressible in temporal logic. These include probabilistic linear-time properties, expected total, discounted and average reward properties, and their branching-time extensions and multi-objective combinations. To demonstrate applicability of the framework as well as its practical implementation in a tool called PRISM-games, we describe several case studies that rely on analysis of stochastic games, from areas such as robotics, and networked and distributed systems.

© 2016 European Control Association. Published by Elsevier Ltd. All rights reserved.

## 1. Introduction

Since the dawn of the information age, correctness and safety of computer systems have been central to their design and analysis. Computer systems typically operate in uncertain environments. The uncertainty can be stochastic due to, e.g., unreliable communication media, faulty components or simply due to the use of randomisation. Moreover, if components that cannot be controlled are present in the environment, their adversarial or competitive behaviour results in additional, non-stochastic uncertainty. Examples of such systems appear in many domains, from robotics and autonomous transport, to security, networked and distributed systems, and power management.

It is natural to view such complex systems as games between the controllable computer system and its (uncontrollable) environment. In this work, we present a comprehensive overview of techniques used in verification and controller (also called a strategy) synthesis for systems modelled as  $2\frac{1}{2}$ -player games, or stochastic games for short. In every step of a stochastic game, the two players, Player 1 and Player 2, choose their moves and, based on their choices, the next state of the game is determined, possibly in a probabilistic fashion. Controller synthesis can then be viewed as finding a winning strategy for Player 1, where Player 2 may play

adversarially. Stochastic games have been employed, for example, to support decision making and synthesise controllers for aircraft power distribution [6], sensor network management in renewable energy production plants [73], in human-in-the-loop UAV planning [46], and autonomous driving in the presence of hazards such as pedestrians [81,33]. They arise naturally in the context of security and defence, where they have been used in patrol planning [82], port defence [72], infrastructure protection [18], to generate countermeasures for DNS bandwidth attacks [43] and to analyse complex attack-defence scenarios in RFID goods management system [4]. Through the use of abstraction and discretisation, high-level control of hybrid and continuous systems can also be addressed.

Stochastic games were first introduced by Shapley in 1953 [71]. Various classes and modifications of these games have been extensively studied since then and surveyed in, e.g., [47,61,22,48,54,21,62]. In this survey, we focus on turn-based games, where players choose their moves in turns rather than concurrently as in [71,47,61]. More specifically, we restrict our attention to turn-based, finite, complete-observation, stochastic, discrete-time, zero-sum games. We also consider a generalisation of these games to multiple players. Compared to existing surveys of these games such as [22], the distinguishing feature of our survey is a comprehensive coverage of algorithms for temporal logic properties, including reward and multi-objective properties not covered in [22], and an illustration of their practical application on a tutorial-style example. Other surveys typically focus on related classes of games, to mention concurrent games [71,47,61], or only on a subclass of properties, e.g., single-objective [22].

<sup>☆</sup>This work was supported by ERC Advanced Investigators Grant VERIWARE and EPSRC Mobile Autonomy Programme Grant EP/M019918/1.

<sup>\*</sup> Corresponding author.

E-mail addresses: [maria.svorenova@cs.ox.ac.uk](mailto:maria.svorenova@cs.ox.ac.uk) (M. Svorenová), [marta.kwiatkowska@cs.ox.ac.uk](mailto:marta.kwiatkowska@cs.ox.ac.uk) (M. Kwiatkowska).

We study various classes of properties of stochastic games expressible in temporal logic. First, we consider quantitative probabilistic properties over linear time, expressed as formulas of probabilistic linear temporal logic. Examples of such properties include ‘the maximum probability of the airbag failing to deploy within 0.02 s is at most  $10^{-6}$ ’, or ‘the minimum probability of the car to reach its destination without colliding with pedestrians, while obeying traffic rules, is at least  $1-10^{-10}$ ’. Next, properties reasoning about rewards associated with states of the game are introduced. Namely, we consider the expected total and discounted cumulative reward as well as long-run average reward. These can be used to state properties such as ‘the minimum expected profit that the investor can guarantee within a year is at least 1000’, or ‘the expected number of requests served per time unit in the network is at least 5’. Finally, we allow to combine the above linear-time and reward properties to express requirements over branching time, thus allowing analysis of properties such as ‘the probability that the network recovers from a bad decision to a state from which a consensus can be reached with probability at least 0.9 is at least 0.95’.

Given a stochastic game and a property, verification and strategy synthesis problems, respectively, focus on the existence and construction of a strategy for Player 1 that guarantees satisfaction of the property against all strategies of Player 2. In this work, we discuss general findings for the two problems and overview the existing algorithmic solutions for various classes of properties. The solutions typically rely on a reduction to simpler games or properties, and the computation of optimal values and strategies, for which a value iteration algorithm is typically utilised. In the multi-player case, a coalition of players aims to cooperatively enforce a property. Intuitively, multi-player stochastic games can be seen as stochastic games with two players, where the coalition acts as Player 1 and the remaining set of players as Player 2. Finally, we analyse stochastic games with respect to multi-objective properties that require simultaneous satisfaction of multiple linear-time and reward properties. Here, the properties can be conflicting and the techniques reduce to the computation of an  $\varepsilon$ -approximation of the Pareto set of optimal trade-offs between the individual properties.

While a number of software tools exist with partial support for stochastic games, see Section 5 for a summary, they only allow a subclass of such games, e.g., with one player or without stochasticity, or perform analysis of stochastic games against single-objective properties only. On the other hand, most of the over-viewed algorithms have been implemented within the tool called PRISM-games [36,55] for modelling, verification, synthesis and simulation of stochastic games, an extension of the PRISM model checker [56]. We briefly overview the features and functionality of the tool and offer a number of case studies, where complex computer systems have been modelled as stochastic games and their properties analysed in PRISM-games.

Contributions of this paper can be summarised as follows:

- we present a comprehensive framework for analysis of stochastic games, focusing on high-level temporal logic specifications;
- we overview the existing body of knowledge and algorithmic solutions for verification and strategy synthesis problems for stochastic games, and identify open problems;
- we offer a list of case studies of control systems that are modelled and analysed through stochastic games.

The remainder of this paper is organised as follows. In Section 2, we introduce stochastic games and define a specification language for linear-time and reward properties. In Section 3, we formulate the verification and strategy synthesis problems for

single-objective properties, present general findings for the problems, as well as algorithmic solutions, and their extensions to branching-time properties and multi-player games. Multi-objective combinations of properties are then discussed in Section 4. We overview existing tools for games in Section 5 and briefly describe the functionality of PRISM-games, which currently provides the most comprehensive support for stochastic games. Finally, we list several case studies that rely on stochastic games in Section 6. We finish with concluding remarks in Section 7. To demonstrate the framework, an illustrative example modelled and analysed in PRISM-games is used throughout the paper.

## 2. Preliminaries

### 2.1. Notation

We use  $\mathcal{D}(X)$  to denote the set of all probability distributions over a set  $X$ . Given a finite or infinite sequence  $\lambda$  of elements of  $X$ , we use  $\lambda_i$  to denote its  $i$ th element for  $i \geq 0$ . For a finite sequence  $\lambda = x_0x_1\dots x_k$  of elements of  $X$ , we use  $|\lambda| = k+1$  to denote the length of the sequence and  $\text{last}(\lambda) = x_k$  denotes its last element.

### 2.2. Stochastic games

**Definition 1 (Stochastic game).** A turn-based  $2\frac{1}{2}$ -player game or simply a stochastic game is a tuple  $\mathcal{G} = (S, (S_1, S_2, S_p), \Delta)$ , where  $S$  is a finite set of states partitioned into sets  $S_1, S_2$  and  $S_p$  of Player 1, Player 2 and probabilistic states, respectively, and  $\Delta : S \times S \rightarrow [0, 1]$  is a probabilistic transition function such that, for states  $s \in S_1 \cup S_2$ , it holds that  $\Delta(s, s') \in \{0, 1\}$  for every  $s' \in S$ , where we assume that  $\Delta(s, s') = 1$  for at least one  $s' \in S$ , and for states  $s \in S_p$ , it holds  $\sum_{s' \in S} \Delta(s, s') = 1$ .

Intuitively, the game is played as follows. The state of the game is always determined uniquely and, in every step, the next state is chosen according to the transition function. When the current state of the game is a Player 1 state, i.e.,  $s \in S_1$ , then Player 1 chooses the next state  $s' \in S$  such that  $\Delta(s, s') = 1$ , and similarly for Player 2. When the current state is probabilistic, i.e.,  $s \in S_p$ , the next state of the game is sampled according to the distribution  $\Delta(s, \cdot)$ .

Formally, a *path* of a game  $\mathcal{G}$  is an infinite sequence  $\lambda = s_0s_1\dots$  such that  $\Delta(s_i, s_{i+1}) > 0$  for all  $i \geq 0$ . A finite path of  $\mathcal{G}$  is a finite prefix of a path. We use  $\text{Path}_{\mathcal{G},s}$  to denote the set of all paths originating in a state  $s \in S$  and  $\text{Path}_{\mathcal{G}} = \bigcup_{s \in S} \text{Path}_{\mathcal{G},s}$ . The sets  $\text{FPath}_{\mathcal{G},s}$ ,  $\text{FPath}_{\mathcal{G}}$  of finite paths are defined analogously. Given two states  $s, s' \in S$ , we say that  $s'$  is *reachable* from  $s$  if and only if there exists a finite path  $\lambda$  such that  $\lambda_0 = s$  and  $\text{last}(\lambda) = s'$ .

**Definition 2 (Labelling function).** Given a finite set of atomic propositions  $\text{AP}$ , a labelling function  $L : S \rightarrow 2^{\text{AP}}$  assigns to each state  $s \in S$  of the game a set of atomic propositions that hold true in  $s$ .

**Definition 3 (Reward structure).** Given a game  $\mathcal{G} = (S, (S_1, S_2, S_p), \Delta)$ , a reward structure for  $\mathcal{G}$  is a function  $r : S \rightarrow \mathbb{R}_{\geq 0}$  or  $r : S \rightarrow \mathbb{R}_{\leq 0}$ .

While the term reward intuitively suggests that the goal will be to maximise functions over these values, we use a reward structure as a general value assignment and consider minimisation problems as well. In such a case, the values are often referred to as costs rather than rewards. By inverting the signature of all rewards, the resulting function is again a reward structure, and this will allow us to translate minimisation problems to maximisation. Note that, unless stated otherwise, in this work we do not consider reward structures that assign both negative and positive values.

Stochastic games as defined above were first studied with respect to reward properties in [50,58], as a special case of games originally defined by Shapley [71]. With respect to reachability properties, simple stochastic games were studied in [40]. In a simple stochastic game, every state has exactly two successors and all transitions from probabilistic states have probability 0.5, simulating a coin toss. More complex, temporal properties of stochastic games were then introduced in [23].

Our definition of a stochastic game (Definition 1) is based on the definition used, e.g., in [40,22]. An alternative way to define such games is to partition the state space only into Player 1 and Player 2 states and introduce a finite set of actions. The transition function then defines at most one probability distribution for each pair of a state and an action. A reward structure assigns values to pairs of states and actions. Note that this definition of games, appearing e.g., in [71,50], is equivalent to Definition 1.

Stochastic games include as a subclass many interesting and widely studied models. If there are no probabilistic states, i.e.,  $S_p = \emptyset$ , the game is called a 2-player game or a non-stochastic game. Similarly, if the game only has one player and probabilistic states, i.e.,  $S_1 = \emptyset$  or  $S_2 = \emptyset$ , it is called a  $1\frac{1}{2}$ -player game or a Markov decision process (MDP). Moreover, if also  $S_p = \emptyset$ , the game is a transition system. Finally, if both  $S_1 = \emptyset$  and  $S_2 = \emptyset$ , the game reduces to a Markov chain.

**Definition 4 (Strategy).** A Player 1 strategy is a tuple  $\pi = (M, \pi_u, \pi_n, \pi_{\text{init}})$ , where  $M$  is a countable set of memory elements,  $\pi_u : M \times S \rightarrow M$  is a memory update function,  $\pi_n : M \times S_1 \rightarrow \mathcal{D}(S)$  is a next move function such that  $\pi_n(m, s)(s') > 0$  only if  $\Delta(s, s') > 0$ , and  $\pi_{\text{init}} : S \rightarrow M$  is an initial memory element function. A Player 2 strategy  $\sigma = (M, \sigma_u, \sigma_n, \sigma_{\text{init}})$  is defined analogously.

Intuitively, strategies prescribe the behaviour of players as follows. Given a Player 1 strategy  $\pi$ , first, an initial memory element is chosen according to the function  $\pi_{\text{init}}$ . Then, in every step of the game, Player 1 updates the current memory element based on the current state of the game, using the memory update function  $\pi_u$ . Moreover, if the game is in a Player 1 state, Player 1 chooses the next state of the game using the next move action  $\pi_n$ . Player 2 strategies are applied in an analogous way.

We use  $\Pi$  and  $\Sigma$  to denote the set of all Player 1 and Player 2 strategies, respectively. A (finite) path under strategies  $\pi \in \Pi, \sigma \in \Sigma$  is any (finite) path resulting from Player 1 playing according to strategy  $\pi$  and Player 2 playing according to strategy  $\sigma$ . We use notation  $\text{Path}_{\mathcal{G},s}^{\pi,\sigma}, \text{Path}_{\mathcal{G},s}^{\pi,\sigma}, \text{FPath}_{\mathcal{G},s}^{\pi,\sigma}$  and  $\text{FPath}_{\mathcal{G},s}^{\pi,\sigma}$  with obvious meaning.

Generally, a Player 1 strategy  $\pi$  is randomised. It is called pure if the next move function  $\pi_n$  is of type  $\pi_n : M \times S_1 \rightarrow S$ . Similarly,  $\pi$  is a memoryless strategy if  $M$  is a singleton, a finite memory strategy if  $M$  is finite, and an infinite memory strategy in the general case. For simplicity, we consider pure memoryless strategies to be functions of type  $\pi : S_1 \rightarrow S$ . Player 2 strategies are classified in the same way.

Let  $s \in S$  be a state of a game  $\mathcal{G}$  and let  $\pi \in \Pi, \sigma \in \Sigma$  be a Player 1 and Player 2 strategy, respectively. Given a finite path  $\lambda \in \text{FPath}_{\mathcal{G},s}^{\pi,\sigma}$ , the cylinder set  $\text{Cyl}(\lambda)$  is the set of all paths in  $\text{Path}_{\mathcal{G},s}^{\pi,\sigma}$  that have  $\lambda$  as a prefix. Consider the  $\sigma$ -algebra  $\Sigma$  of paths generated by the set of all such cylinder sets. According to classical probability and measure theory [3], there exists a unique probability measure  $\text{Pr}_{\mathcal{G},s}^{\pi,\sigma}$  over  $\Sigma$  such that

$$\text{Pr}_{\mathcal{G},s}^{\pi,\sigma}(\text{Cyl}(s)) = 1,$$

$$\text{Pr}_{\mathcal{G},s}^{\pi,\sigma}(\text{Cyl}(\lambda)) = \prod_{i=0}^{|\lambda|-1} \Delta(\lambda_i, \lambda_{i+1})$$

for all  $\lambda \in \text{FPath}_{\mathcal{G},s}^{\pi,\sigma}$ . Given a random variable  $\rho$  over the probability

space  $(\text{Path}_{\mathcal{G},s}^{\pi,\sigma}, \Sigma, \text{Pr}_{\mathcal{G},s}^{\pi,\sigma})$ , the expected value of  $\rho$  is defined as

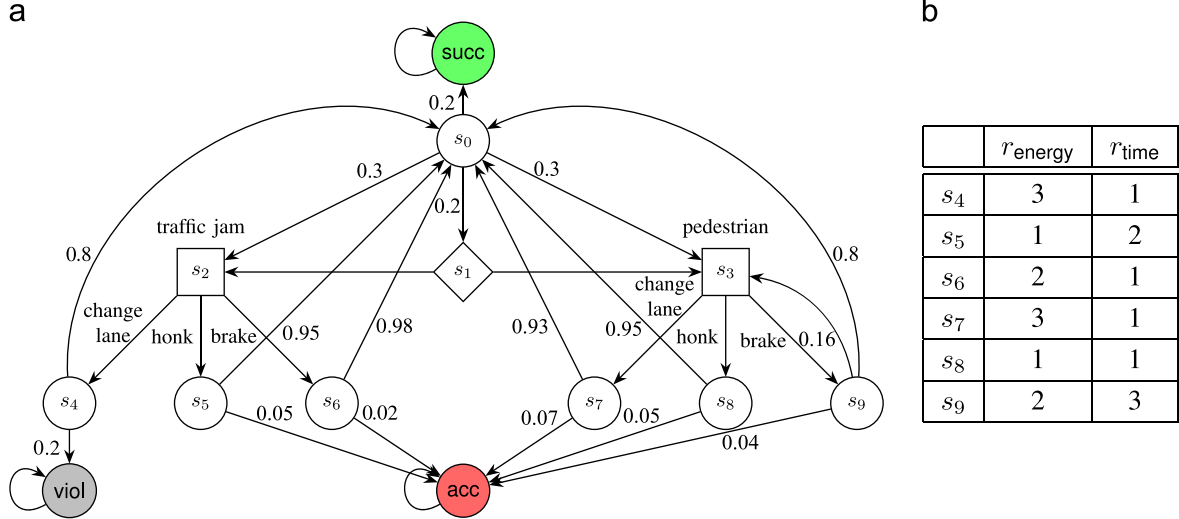
$$\mathbb{E}_{\mathcal{G},s}^{\pi,\sigma}(\rho) = \int_{\Omega} \rho \, d\text{Pr}_{\mathcal{G},s}^{\pi,\sigma}.$$

**Definition 5 (Stopping game).** A state  $s_f \in S$  of a stochastic game  $\mathcal{G} = (S, (S_1, S_2, S_p), \Delta)$  is called terminal if and only if  $\Delta(s_f, s_f) = 1$  and  $\Delta(s_f, s') = 0$  for all  $s' \neq s_f, s' \in S$ . A game is called stopping if it has at least one terminal state and if it holds that, for every pair of strategies  $\pi \in \Pi, \sigma \in \Sigma$  and every initial state, with probability 1 the game eventually stops, i.e., a terminal state is reached.

The principle of stopping games was introduced in [71], where games were first studied with respect to reward properties, to avoid infinite accumulation of rewards. The original definition imposed a stronger assumption, not required for the results discussed here, that in every step the game stops with non-zero probability. The more general notion of stopping as in Definition 5 appears, for example, in [40].

Note that we do not consider partial observability and assume the games are finite. Several related, more general stochastic game models exist, which include concurrent games [71,47,61], partial-observation games [21] and uncertain (or bounded-parameter) MDPs [62]. In particular, uncertain MDPs generalise stochastic games to infinite games by considering Player 2 with an infinite set of actions, typically defined through convex uncertainty sets. The motivation comes from the fact that, in many practical problems, estimating transition probabilities of systems with stochastic uncertainty from data may be very difficult but they can be over-approximated using sets. These models have a strong connection to partial-observation and concurrent games, see e.g., [57,80,67].

**Example 1.** Consider the stochastic game  $\mathcal{G} = (S, (S_1, S_2, S_p), \Delta)$  depicted in Fig. 1(a). It can be seen as a simplified version of the autonomous driving case study presented in [33]. The game models a car driving in an urban area that is given a route to navigate through. While navigating the route, it has to autonomously react to hazards. We consider two hazards, a traffic jam and a pedestrian. In order to avoid a hazard, the car chooses to perform one of the three reactions, namely, change lane, honk or brake. The game proceeds as follows. Starting from the probabilistic state  $s_0$ , a hazard is encountered or the car successfully finishes its route by entering the terminal state labelled with atomic proposition *succ*, shown in green in Fig. 1(a). In the former case, each of the two hazards appears with probability 0.3 and, with probability 0.2, the choice of a hazard is left to Player 2. The outcome of the three reactions to each hazard is indicated in Fig. 1(a). For example, honking in the event of a traffic jam results with probability 0.05 in an accident, i.e., entering the red terminal state labelled with proposition *acc*, and with probability 0.95 in successfully resolving the hazard, i.e., returning back to state  $s_0$ . The results of braking in a traffic jam, and changing lane and honking when approaching a pedestrian are similar, with varying probabilities of the individual transitions. In addition, braking when approaching a pedestrian may have no effect on the hazard, i.e., with probability 0.16 the pedestrian remains a hazard and the car can make a new choice of a reaction by returning back to state  $s_3$ . Finally, changing lane in a traffic jam results in resolving the hazard with probability 0.8 and, with probability 0.2, it results in a violation of road rules, i.e., entering the grey terminal state labelled with proposition *viol*. Note that the game  $\mathcal{G}$  is a stopping game since, regardless of Player 1 and Player 2 choices in their respective



**Fig. 1.** (a) Stochastic game modelling a car autonomously reacting to hazards on a road. Player 1, Player 2 and probabilistic states are depicted as squares, diamonds and circles, respectively. Transition probabilities are indicated unless equal to 1. The set of atomic propositions is  $AP = \{\text{succ}, \text{viol}, \text{acc}\}$  and, for convenience, states with a non-empty set of labels are also shown in colour. Hazards and reactions are indicated as annotations. (b) Two reward structures over  $\mathcal{G}$ . States not shown in the table are assigned value 0 in both reward structures. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

states, the game ends with probability 1 in one of the terminal states labelled with proposition *succ*, *viol* or *acc*.

We consider two reward structures,  $r_{\text{energy}}$  and  $r_{\text{time}}$ , over  $\mathcal{G}$  defined in Fig. 1(b) that represent the energy and time demands of individual reactions to hazards, respectively.

We modelled the game in the tool PRISM-games and analysed it with respect to several properties. The corresponding input files for PRISM-games can be found in [52], and we report on the results in the following sections.

### 2.3. Properties

In this work, we are interested in both temporal and reward properties of stochastic games. The specification language defined below allows us to formulate such properties over linear time, namely probabilistic linear-time properties, and various expected reward functions. The language is motivated by the language used by the PRISM model checker [56] and its extension for stochastic games known as PRISM-games [36,55], based on probabilistic Computation Tree Logic (PCTL) and its extension PCTL\* [8], which combines Linear Temporal Logic (LTL) together with the probabilistic and reward operators in a CTL-like branching-time fashion.

**Definition 6 (Property).** A property is a formula  $\phi$  in the following grammar:

$$\begin{aligned} \phi &::= P_{\bowtie p}[\psi] \mid R_{\bowtie x}^l[\rho], \\ \psi &::= \text{true} \mid a \mid \neg\psi \mid \psi \wedge \psi \mid \text{X}\psi \mid \psi \cup \psi, \\ \rho &::= c \leq k \mid c \mid F^*a \mid D^\beta \mid S, \end{aligned}$$

where  $a \in AP$  is an atomic proposition,  $\bowtie \in \{\leq, <, \geq, >\}$ ,  $p \in [0, 1]$ ,  $r$  is a reward structure,  $x \in \mathbb{R}$ ,  $k \in \mathbb{N}_0$ ,  $*$   $\in \{0, \infty, c\}$  and  $\beta \in (0, 1)$ .

The semantics of the properties is listed in Table 1. In particular, formulas  $\phi$ , i.e., the probabilistic and reward operator, are interpreted over states of the game, and linear temporal formulas  $\psi$  and reward functions are interpreted over paths of the game. Below, we give a brief description grouping properties from Definition 6 into categories according to their syntactic form.

**Table 1**

Semantics of properties defined in Definition 6. Here,  $\mathcal{G}$  is a stochastic game,  $s \in S$  is its state,  $L$  is a labelling function over a set of atomic propositions  $AP$ ,  $\Pi$  and  $\Sigma$  are the sets of Player 1 and Player 2 strategies, respectively,  $\lambda = \lambda_0 \lambda_1 \dots \in \text{Path}_{\mathcal{G}}$  is a path,  $r$  is a reward structure on  $\mathcal{G}$  and  $\beta \in (0, 1)$  is a discount factor.

$s \models P_{\bowtie p}[\psi]$	$\iff \exists \pi \in \Pi$ such that
$s \models R_{\bowtie x}^l[\rho]$	$\iff \forall \sigma \in \Sigma : \Pr_{\mathcal{G},s}^{\pi,\sigma}(\psi) = \Pr_{\mathcal{G},s}^{\pi,\sigma}(\{\lambda \in \text{Path}_{\mathcal{G},s} \mid \lambda \models \psi\}) \bowtie x$
$\lambda \models \text{true}$	$\iff \exists \pi \in \Pi$ such that $\forall \sigma \in \Sigma : \mathbb{E}_{\mathcal{G},s}^{\pi,\sigma}(\text{rew}(r, \rho)) \bowtie x$
$\lambda \models a$	$\iff a \in L(\lambda_0)$
$\lambda \models \neg\psi$	$\iff \lambda \not\models \psi$
$\lambda \models \psi_1 \wedge \psi_2$	$\iff \lambda \models \psi_1 \wedge \lambda \models \psi_2$
$\lambda \models \text{X}\psi$	$\iff \lambda_1 \lambda_2 \dots \models \psi$
$\lambda \models \psi_1 \cup \psi_2$	$\iff \exists i \geq 0 : (\lambda_i \lambda_{i+1} \dots \models \psi_2 \wedge \forall j < i : (\lambda_j \lambda_{j+1} \dots \models \psi_1))$
$\text{rew}(r, c \leq k)(\lambda)$	$= \sum_{i=0}^k r(\lambda_i)$
$\text{rew}(r, c)(\lambda)$	$= \sum_{i=0}^{\infty} r(\lambda_i)$
$\text{rew}(r, F^*a)(\lambda)$	$= \begin{cases} * & \text{if } \forall i \geq 0 : a \notin L(\lambda_i) \text{ and } * \in \{0, \infty\} \\ \sum_{i=0}^{\infty} r(\lambda_i) & \text{if } \forall i \geq 0 : a \notin L(\lambda_i) \text{ and } * = c \\ \sum_{i=0}^{k-1} r(\lambda_i) & \text{otherwise, where } k = \min\{i \mid a \in L(\lambda_i)\} \end{cases}$
$\text{rew}(r, D^\beta)(\lambda)$	$= \sum_{i=0}^{\infty} \beta^i r(\lambda_i)$
$\text{rew}(r, S)(\lambda)$	$= \liminf_{k \rightarrow \infty} \frac{\text{rew}(r, c \leq k)(\lambda)}{k+1}$

#### Probabilistic reachability

Properties of the form  $P_{\bowtie p}[F a]$ , where  $F a = \text{true} \cup a$ , are called probabilistic reachability properties. Given a state  $s$ , the property requires that there exists a Player 1 strategy  $\pi \in \Pi$  such that, for all Player 2 strategies  $\sigma \in \Sigma$ , the probability of reaching a state labelled with atomic proposition  $a$ , starting from  $s$ , under the two strategies satisfies the bound  $\bowtie p$ . Probabilistic reachability properties represent a simple but fundamental class of properties since many properties of games can be reduced to probabilistic reachability. A step-bounded probabilistic reachability is a property of the form  $P_{\bowtie p}[F^{\leq k} a]$ , where  $k \in \mathbb{N}_0$ ,  $F^{\leq k} a = \bigvee_{i=0}^k \text{X}^i a$  and  $\text{X}^i$  is an abbreviation for a sequence of  $i$  consecutive instances of the operator  $\text{X}$ . In this case, the goal is to visit a state labelled with atomic property  $a$  within the first  $k$  steps of a path. Finally, properties of the form  $P_{\geq 1}[F a]$  are called almost-sure reachability properties.

#### Probabilistic LTL

Properties of the form  $P_{\bowtie p}[\psi]$  are generally referred to as probabilistic LTL properties since, according to Definition 6,  $\psi$  can



be an arbitrary LTL formula [64]. Recently, LTL has been increasingly often used in various areas of control as it is expressive enough to describe many interesting properties of systems and, at the same time, it resembles natural language statements. Examples of properties that can be expressed in LTL include reachability  $\mathsf{F} a$ , safety  $\mathsf{G} a = \neg \mathsf{F} \neg a$ , liveness  $\mathsf{G}(a \Rightarrow \mathsf{F} b)$ , persistent surveillance  $\mathsf{GF} a$  or stability  $\mathsf{FG} a$ . Similar to probabilistic reachability, almost-sure LTL properties are properties of the form  $\mathsf{P}_{\geq 1}[\psi]$ .

#### Total reward properties

Properties of the form  $\mathsf{R}_{\text{pxx}}^r[\rho]$ , where  $\rho$  is equal to  $\mathsf{C}$ ,  $\mathsf{C}^{\leq k}$  or  $\mathsf{F}^* a$ , are called total reward properties. They are concerned with the expected cumulative reward collected in states of the game over infinite time horizon ( $\mathsf{C}$ ), in the first  $k$  steps of a path for  $k \in \mathbb{N}_0$  ( $\mathsf{C}^{\leq k}$ ), or until a state labelled with proposition  $a$  is reached ( $\mathsf{F}^* a$ ). In the latter case, if such a state is never visited, we allow to treat the cumulative reward in different ways through the use of the flag  $*$ . Namely, we consider the reward being zero ( $*$  = 0), infinity ( $*$  =  $\infty$ ), or we allow the reward to accumulate indefinitely ( $*$  =  $\mathsf{C}$ ). Just as reachability properties are fundamental probabilistic properties, total reward properties are fundamental reward properties.

#### Discounted reward properties

Properties of the form  $\mathsf{R}_{\text{pxx}}^r[\mathsf{D}^\beta]$  analyse the expected cumulative reward over infinite time horizon, where the collected rewards are increasingly discounted by the discount factor  $\beta \in (0, 1)$ .

#### Average reward properties

Finally, properties of the form  $\mathsf{R}_{\text{pxx}}^r[\mathsf{S}]$  analyse the expected average reward collected in states of the game over infinite time horizon.

### 3. Single-objective game solving

In this section, we first formulate the problem of solving a stochastic game with respect to a property and discuss general findings for this problem. Next, we overview the algorithmic solutions for different types or properties. Finally, we discuss extensions of these techniques to properties over branching time and multi-player stochastic games.

#### 3.1. Problem formulation

**Problem 1 (Verification).** Given a stochastic game  $\mathcal{G}$  with an initial state  $s \in S$ , a set of atomic propositions AP and a labelling function  $L$ , and a property over AP from Definition 6, i.e.,  $\phi = \mathsf{P}_{\geq p}[\psi]$  or  $\phi = \mathsf{R}_{\text{pxx}}^r[\rho]$ , does it hold that  $s \models \phi$ ?

**Problem 2 (Strategy synthesis).** Given a stochastic game  $\mathcal{G}$  with an initial state  $s \in S$ , a set of atomic propositions AP and a labelling function  $L$ , and a property over AP from Definition 6, i.e.,  $\phi = \mathsf{P}_{\geq p}[\psi]$  or  $\phi = \mathsf{R}_{\text{pxx}}^r[\rho]$ , construct a Player 1 strategy  $\pi \in \Pi$  (if it exists) that is a witness to the satisfaction  $s \models \phi$ .

A Player 1 strategy that is a solution to Problem 2 is called a winning Player 1 strategy. Conversely, Player 2 aims to violate the property  $\phi$  and a winning Player 2 strategy is such that, for all Player 1 strategies, the property is not satisfied. Games with this semantics are called *zero-sum games* since the objectives of the two players are complementary.

In order to solve both verification and strategy synthesis problems, we consider the *optimal values* of path formulas  $\psi$  and

reward functions  $\rho$  from Definition 6 defined as follows:

$$\begin{aligned} \mathsf{Pr}_{\mathcal{G},s}^{\min}(\psi) &= \inf_{\pi \in \Pi} \sup_{\sigma \in \Sigma} \mathsf{Pr}_{\mathcal{G},s}^{\pi,\sigma}(\psi), \\ \mathsf{Pr}_{\mathcal{G},s}^{\max}(\psi) &= \sup_{\pi \in \Pi} \inf_{\sigma \in \Sigma} \mathsf{Pr}_{\mathcal{G},s}^{\pi,\sigma}(\psi), \\ \mathbb{E}_{\mathcal{G},s}^{\min}(\text{rew}(r, \rho)) &= \inf_{\pi \in \Pi} \sup_{\sigma \in \Sigma} \mathbb{E}_{\mathcal{G},s}^{\pi,\sigma}(\text{rew}(r, \rho)), \\ \mathbb{E}_{\mathcal{G},s}^{\max}(\text{rew}(r, \rho)) &= \sup_{\pi \in \Pi} \inf_{\sigma \in \Sigma} \mathbb{E}_{\mathcal{G},s}^{\pi,\sigma}(\text{rew}(r, \rho)). \end{aligned} \quad (1)$$

A Player 1 strategy  $\pi \in \Pi$  starting from state  $s$  is called *optimal* if it achieves the optimal value, e.g.,  $\sup_{\sigma \in \Sigma} \mathsf{Pr}_{\mathcal{G},s}^{\pi,\sigma}(\psi) = \mathsf{Pr}_{\mathcal{G},s}^{\min}(\psi)$ . Similarly, the strategy is called  $\varepsilon$ -optimal, for  $\varepsilon > 0$ , if it achieves a value deviating by at most  $\varepsilon$  from the optimum, e.g.,  $\sup_{\sigma \in \Sigma} \mathsf{Pr}_{\mathcal{G},s}^{\pi,\sigma}(\psi) \geq \mathsf{Pr}_{\mathcal{G},s}^{\min}(\psi) + \varepsilon$ .

A stochastic game is called *determined* with respect to a chosen optimality criterion if the corresponding equality holds:

$$\begin{aligned} \mathsf{Pr}_{\mathcal{G},s}^{\min}(\psi) &= \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathsf{Pr}_{\mathcal{G},s}^{\pi,\sigma}(\psi), \\ \mathsf{Pr}_{\mathcal{G},s}^{\max}(\psi) &= \inf_{\sigma \in \Sigma} \sup_{\pi \in \Pi} \mathsf{Pr}_{\mathcal{G},s}^{\pi,\sigma}(\psi), \\ \mathbb{E}_{\mathcal{G},s}^{\min}(\text{rew}(r, \rho)) &= \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathbb{E}_{\mathcal{G},s}^{\pi,\sigma}(\text{rew}(r, \rho)), \\ \mathbb{E}_{\mathcal{G},s}^{\max}(\text{rew}(r, \rho)) &= \inf_{\sigma \in \Sigma} \sup_{\pi \in \Pi} \mathbb{E}_{\mathcal{G},s}^{\pi,\sigma}(\text{rew}(r, \rho)). \end{aligned}$$

Determinacy also guarantees existence of  $\varepsilon$ -optimal strategies for all  $\varepsilon > 0$  for both players from every state. A deep result in [60] established determinacy for a large class of games including stochastic games with respect to any Borel measurable property and, in particular, with respect to all properties in Definition 6.

Note that determinacy does not necessarily imply the existence of optimal strategies. However, for all classes of properties described in Section 2.3 it has been shown that both players have optimal strategies and pure memoryless strategies suffice, except for the step-bounded properties, the class of general probabilistic LTL properties and total reward properties with  $\rho = \mathsf{F}^0 a$ , where pure finite-memory strategies may be required, see [2,22,35] and references therein. The optimal values and strategies can be used to solve the verification problem stated in Problem 1 in the following way. For example, to solve the verification problem for  $\mathcal{G}, s$  and property  $\mathsf{P}_{\geq p}[\psi]$ , it suffices to verify that  $\mathsf{Pr}_{\mathcal{G},s}^{\max}(\psi) \geq p$ . The remaining properties in Definition 6 can be addressed in an analogous way. To solve the strategy synthesis problem stated in Problem 2, we compute an optimal or a suitable  $\varepsilon$ -optimal Player 1 strategy. Together with the existence of optimal strategies, this implies that, for every property in Definition 6, there exists a winning strategy for one of the players.

To directly address the computation of optimal values and strategies, we extend the syntax of properties to include numerical queries for computing optimal strategies.

**Definition 7 (Numerical query).** Let  $\psi$ ,  $r$  and  $\rho$  be as defined in Definition 6. Numerical queries  $\mathsf{P}_{\min} = ?[\psi]$ ,  $\mathsf{P}_{\max} = ?[\psi]$ ,  $\mathsf{R}_{\min}^r = ?[\rho]$  and  $\mathsf{R}_{\max}^r = ?[\rho]$  aim to compute the optimal values defined in Eq. (1), respectively, together with the corresponding optimal Player 1 strategies.

As discussed above, the problem of computing the optimal values for states, called *quantitative query solving*, and constructing an optimal Player 1 strategy, called *strategic query solving*, are two separate problems utilised to solve the verification and strategy synthesis problems for games. In [2], it has been shown that all problems of quantitative and strategic query solving for probabilistic reachability, total, discounted as well as average reward properties are polynomially equivalent, i.e., there exists a polynomial-time reduction between the problems. The computational complexity of these problems is  $\text{NP} \cap \text{coNP}$ . No polynomial-time algorithm is known, even for reachability objectives, and the

$$v_n^*(s) = \begin{cases} 1 & \text{if } a \in L(s), \\ 0 & \text{if } a \notin L(s) \text{ and } n = 0, \\ \max_{s' \in S} \{\Delta(s, s') \cdot v_{n-1}^*(s')\} & \text{if } a \notin L(s), n > 0 \text{ and } s \in S_1, \\ \min_{s' \in S} \{\Delta(s, s') \cdot v_{n-1}^*(s')\} & \text{if } a \notin L(s), n > 0 \text{ and } s \in S_2, \\ \sum_{s' \in S} \Delta(s, s') \cdot v_{n-1}^*(s') & \text{if } a \notin L(s), n > 0 \text{ and } s \in S_p. \end{cases}$$

**Fig. 2.** Value iteration algorithm for the probabilistic reachability numerical query  $\mathbb{P}_{\max} = ?[\mathbb{F} a]$ .

widely used exponential-time algorithm for the corresponding quantitative query solving problems, i.e., computing the optimal values for states, is a value iteration algorithm, presented in detail later in this section. It is not necessarily true that the strategic solution can be easily derived from the quantitative solution, i.e., an optimal strategy might not be easily constructed from the optimal values. For the case of probabilistic reachability, total and discounted reward this nevertheless is the case, and optimal strategies can be constructed from the optimal values in linear time. For average reward properties, the existence of a similar (even polynomial-time) algorithm remains an open question [2].

In the following Sections 3.2–3.6, we discuss algorithmic solutions to the verification and strategy synthesis problems for properties in Definition 6 as classified in Section 2.3. Firstly, note that from determinacy we get

$$\Pr_{\mathcal{G},s}^{\min}(\psi) = 1 - \Pr_{\mathcal{G},s}^{\max}(\neg\psi),$$

$$\mathbb{E}_{\mathcal{G},s}^{\min}(\text{rew}(r, \rho)) = \mathbb{E}_{\mathcal{G},s}^{\max}(\text{rew}(-r, \rho)),$$

and hence it suffices to discuss maximisation numerical queries. In Section 3.7, we discuss a generalisation of these techniques to a logic that combines properties from Definition 6 in a PCTL\*-like fashion to obtain properties over branching time. Finally, Section 3.8 overviews an extension of verification and strategy synthesis for stochastic games with two players to general, multi-player stochastic games.

### 3.2. Probabilistic reachability

Consider the numerical reachability query  $\mathbb{P}_{\max} = ?[\mathbb{F} a]$  for a game  $\mathcal{G}$  with an initial state  $s \in S$ . To quantitatively solve the query, we can use an adaptation of the value iteration algorithm that first appeared in [40] for simple stochastic games defined in Section 2.2. The algorithm computes the optimal values for Player 1 states  $s \in S_1$  as

$$\Pr_{\mathcal{G},s}^{\max}(\mathbb{F} a) = v^*(s) = \lim_{n \rightarrow \infty} v_n^*(s), \quad (2)$$

where  $v_n^*(s)$  is iteratively computed as indicated in Fig. 2. While the limit may not converge in finite time, a precision threshold  $\alpha$  can be computed such that, if the value iteration algorithm is terminated once the maximum difference between  $v_n^*(s)$  and  $v_{n+1}^*(s)$ , for  $s \in S$ , is not more than  $\alpha$ , the limit values can be obtained by simple rounding [25]. Moreover, using this procedure, the algorithm always stops in a number of iterations that is at most exponential in the size of the game. It was proven in [2] that, given the quantitative solution to the query, the necessary and sufficient conditions for the strategic solution, i.e., a pure memoryless optimal Player 1 strategy  $\pi^*$ , are the following. First, an optimal strategy  $\pi^*$  satisfies, for every  $s \in S_1$ , the set membership

$$\pi^*(s) \in \arg \max_{s' \in S} \Delta(s, s') \cdot v^*(s'), \quad (3)$$

and second, under  $\pi^*$ , the game reaches a state labelled with proposition  $a$  with non-zero probability starting from any state  $s \in S$  such that  $v^*(s) > 0$ , under any Player 2 strategy. An optimal Player 1 strategy can be constructed from the optimal values in

time linear in the size of the game using a technique called retrograde analysis, first introduced in the artificial intelligence community to solve chess endgames [74]. Intuitively, the strategy is computed from states labelled with  $a$ , using backward propagation. For details of the construction, see [2].

The minimisation queries  $\mathbb{P}_{\min} = ?[\mathbb{F} a]$  can be solved analogously. The value iteration algorithm can also be used for a fixed number of iterations to approximate the optimal values and strategy. For example, to solve the strategy synthesis problem for properties of type  $\mathbb{P}_{\geq p}[\mathbb{F} a]$ , the maximisation query is considered and value iteration is terminated once the value  $v_n^*(s)$  is greater or equal to  $p$  for the chosen state  $s \in S$  and the corresponding Player 1 strategy is then computed in a way similar to the optimal case above.

Besides the value iteration algorithm, one can adapt the equations from Fig. 2 to design a quadratic program and a strategy iteration algorithm that iterates over pure memoryless strategies [41]. For a chosen Player 1 strategy  $\pi \in \Pi$ , the strategy iteration algorithm computes the optimal values  $v_\pi^*$  obtainable by Player 2 if Player 1 plays according to  $\pi$ , and then the algorithm locally improves  $\pi$  to achieve better values for Player 1. While the best known bound for the number of iterations is exponential, the algorithm performs well in practice and no class of games is known for which an exponential number of iterations is required.

Finally, there also exists a randomised subexponential-time algorithm to solve the problem. For simple stochastic games, it was introduced in [59] and can be extended to general stochastic games as shown in [12]. Intuitively, the algorithm randomly tries to guess the optimal transition in a chosen Player 1 state of the game and verifies whether the best strategy with the chosen transition is optimal. If not, it removes the transition and proceeds.

For the special case of almost-sure reachability properties  $\mathbb{P}_{\geq 1}[\mathbb{F} a]$ , the strategy synthesis problem can be solved in quadratic time as follows. The problem is first reduced to an equivalent strategy synthesis problem for a 2-player, non-stochastic game with a reachability property as shown in [26], which is then solved using a simple graph algorithm, see, e.g., [22] and references therein.

For step-bounded numerical reachability queries  $\mathbb{P}_{\min} = ?[\mathbb{F}^{\leq k} a]$  and  $\mathbb{P}_{\max} = ?[\mathbb{F}^{\leq k} a]$ , pure finite-memory strategies need to be considered. Intuitively, longer paths with higher probability of reaching a state labelled with proposition  $a$  may be preferred when enough time remains until the deadline  $k$ , whereas shorter paths with lower probability might need to be considered when the deadline is approaching. The optimal values and a Player 1 strategy for a game  $\mathcal{G}$  can be computed by performing a fixed number  $k$  of iterations of the value iteration algorithm on the game  $\mathcal{G}^{\leq k}$  that is an extension of the game  $\mathcal{G}$  to keep track of the number of steps performed. Formally,  $\mathcal{G}^{\leq k}$  has states of the form  $(s, i)$ , where  $s \in S$  and  $i \in \{0, 1, \dots, k\}$ , and the transition function is

$$\Delta^{\leq k}((s, i), (s', i+1)) = \Delta^{\leq k}((s, k), (s', k)) = \Delta(s, s')$$

for  $s, s' \in S$  and  $i < k$ . It is easy to see that, indeed,  $\Pr_{\mathcal{G},s}^{\max}(\mathbb{F}^{\leq k} a) = v_k^*((s, 0))$ .

$$v_n^*(s) = \begin{cases} 0 & \text{if } n = 0, \\ \max_{s' \in S} \{r(s) + \Delta(s, s') \cdot v_{n-1}^*(s')\} & \text{if } n > 0 \text{ and } s \in S_1, \\ \min_{s' \in S} \{r(s) + \Delta(s, s') \cdot v_{n-1}^*(s')\} & \text{if } n > 0 \text{ and } s \in S_2, \\ r(s) + \sum_{s' \in S} \Delta(s, s') \cdot v_{n-1}^*(s') & \text{if } n > 0 \text{ and } s \in S_p. \end{cases}$$

Fig. 4. Value iteration algorithm for the total reward numerical query  $R_{\max}^r = ?[C]$ .

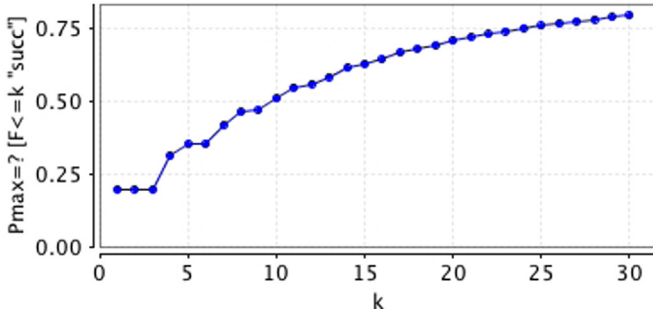


Fig. 3. Optimal values obtained for step-bounded numerical query  $P_{\max} = ?[F \leq k \text{ succ}]$  for the game introduced in Example 1.

We conclude this section by commenting on a reduction from reachability to reward properties. The strategy synthesis problem for a game  $\mathcal{G}$ , state  $s \in S$  and probabilistic reachability property  $P_{\text{exp}}[F a]$  can be reduced to strategy synthesis for a game  $\mathcal{G}'$ ,  $s$  and a total reward property  $R_{\text{exp}}^r[F^c a]$ , where the game  $\mathcal{G}'$  is  $\mathcal{G}$  with a new probabilistic terminal state  $s_f$ . The transitions of all states  $s \in S$  labelled with proposition  $a$  are defined as  $\Delta(s, s_f) = 1$  and  $\Delta(s, s') = 0$  otherwise. The reward structure is defined as  $r(s) = 1$  for all  $s \in S$  labelled with  $a$  and  $r(s) = 0$  otherwise.

**Example 2.** Recall the autonomous car example introduced in Example 1, modelled with the stochastic game  $\mathcal{G}$  shown in Fig. 1 (a) with starting state  $s_0 \in S_p$ . Consider the numerical query  $P_{\max} = ?[F \text{ succ}]$  to determine the maximal probability of reaching a state labelled with proposition *succ* indicating that the car successfully finished the route. The maximal probability is approximately 0.87 and the corresponding pure memoryless optimal strategy is to brake for both hazards. Next, consider the step-bounded numerical query  $P_{\max} = ?[F \leq k \text{ succ}]$ . We computed results for step bounds  $1 \leq k \leq 30$ . The optimal values can be observed in Fig. 3. For  $k \leq 3$ , the only way for the car to successfully finish the route in at most 3 steps starting from  $s_0$  is not to encounter any hazards, and thus the maximum probability is 0.2. For  $k=4$  and  $k=5$ , the maximum probability is approximately 0.32 and 0.35, respectively, and there exists a pure memoryless optimal strategy that always brakes in a traffic jam and honks when approaching a pedestrian. Finally, for  $k \geq 6$ , the maximum probability is gradually increasing with  $k$  and, here, optimal strategies are pure, but require finite memory. To be specific, an optimal strategy is to always brake in a traffic jam, and when approaching a pedestrian react as follows. If at most 4 or at least 6 steps remain until the bound  $k$ , then honk. If exactly 5 steps remain to successfully finish the route, then brake.

### 3.3. Probabilistic LTL

The standard approach to solving numerical queries of the form  $P_{\max} = ?[\psi]$  with an arbitrary LTL formula  $\psi$  is to translate the formula  $\psi$  into a deterministic Rabin automaton [13] of the size up to doubly exponential in the size of the formula. Since LTL formulas considered in control are typically small, the size of the corresponding automaton is manageable. The synchronous product of the game  $\mathcal{G}$  and the automaton is a Rabin stochastic game. Such games can be solved by combining the value iteration algorithm

for reachability queries with any algorithm for Rabin non-stochastic games as presented in [27]. Alternatively, the deterministic Rabin automaton can be translated to a different type of automaton called a parity automaton [9] and the product parity stochastic game can be solved using the strategy iteration or randomised subexponential algorithm presented in [28].

Similarly as for the reachability properties, the problem of solving a stochastic game with respect to an almost-sure LTL property  $P_{\geq 1}[\psi]$  can be solved by reducing the corresponding almost-sure Rabin stochastic game to an equivalent Rabin 2-player, non-stochastic game [26]. An overview of existing algorithms, exponential and deterministic subexponential, for non-stochastic Rabin games can be found in [22].

**Example 3.** Consider the safety numerical query  $P_{\max} = ?[G \neg \text{acc}]$  for the autonomous car example from Example 1. The query aims to compute the maximum probability with which an accident can be avoided. The maximum probability is approximately 0.92, which is higher than the maximum probability of successfully finishing the route computed in Example 2. The reason is that violation of road rules is not considered an accident. There exists a pure memoryless optimal strategy, namely, to change lane in a traffic jam and to brake when approaching a pedestrian.

### 3.4. Total reward properties

To solve the numerical query  $R_{\max}^r = ?[\rho]$ , techniques similar to those for probabilistic reachability as described in Section 3.2 can be applied.

First, let  $\rho = C$  and assume that the considered game is stopping, all rewards are non-negative  $r : S \rightarrow \mathbb{R}_{\geq 0}$  and terminal states have reward 0. This means that the expected total reward is always finite. The optimal values for Player 1 states  $s \in S_1$  are defined as

$$\mathbb{E}_{\mathcal{G}, s}^{\max}(\text{rew}(r, C)) = v^*(s) = \lim_{n \rightarrow \infty} v_n^*(s), \quad (4)$$

where  $v_n^*(s)$  is iteratively computed as indicated in Fig. 4. Similarly as for the value iteration in Section 3.2, the limit is not guaranteed to converge in finite time, but using a precision threshold the limit values can be computed in a number of iterations that is at most exponential in the size of the game [25]. Unlike in the reachability case, since the game is assumed to be stopping, there is only one necessary and sufficient condition for a (pure memoryless) optimal Player 1 strategy  $\pi^*$ . Namely, for every state  $s \in S_1$ , it must hold that

$$\pi^*(s) \in \arg \max_{s' \in S} \Delta(s, s') \cdot v^*(s'). \quad (5)$$

An optimal strategy can thus be constructed from the optimal values in linear time using the above equations. Maximisation queries with non-positive rewards  $r : S \rightarrow \mathbb{R}_{\leq 0}$ , as well as minimisation queries, can be solved in an analogous way, and the value iteration can be used for a fixed number of iterations to approximate the optimal value and strategy to solve properties of the type  $\phi = R_{\text{max}}^r[C]$ .

For non-stopping games, the set of states that receive infinite total reward can be computed by solving the game with respect to a parity condition [73]. After removing these states, value iteration

algorithm can be applied to compute the (bounded) optimal values for the remaining states.

Next, let  $\rho = \mathbb{F}^* a$ . For  $* = c$ , the query can be reduced to the case above, with  $\rho = c$ , by adding a new terminal state  $s_f$  with  $r(s_f) = 0$  and altered transitions for states  $s$  such that  $a \in L(s)$  by letting  $\Delta(s, s_f) = 1$  and  $\Delta(s, s') = 0$  otherwise. For  $* = \infty$ , we proceed in a similar fashion. However, while value iteration in Fig. 4 computes the least fixed point, in this case we need to compute the greatest fixed point as zero reward paths that do not reach a state labelled with proposition  $a$  need to be identified. This can be done using computation in Fig. 4 for an altered game, where all zero rewards are changed to an arbitrary  $\varepsilon > 0$  [35]. Finally, for  $* = 0$ , the optimal strategy may depend on the rewards accumulated so far and pure finite memory strategies suffice for Player 1 to win. The computation combines value iteration algorithms from Figs. 2 and 4, see [35] for details.

Step-bounded numerical queries with  $\rho = c \leq k$  can be solved using a fixed number  $k$  of iterations of the value iteration algorithm on the game  $\mathcal{G}^{\leq k}$  defined in Section 3.2 and optimal strategies might thus require memory.

**Example 4.** For the autonomous car in Example 1, we compute the minimum expected total energy and time demands before successful route completion,  $R_{\min}^{\text{energy}} = ?[\mathbb{F}^c \text{succ}]$  and  $R_{\min}^{\text{time}} = ?[\mathbb{F}^c \text{succ}]$ . For energy, the minimum value is approximately 3.33 with the pure memoryless optimal strategy to honk for both hazards. For time, the minimum value is approximately 2.71 with the pure memoryless optimal strategy to change lane for both hazards. Note that changing lane in a traffic jam might violate traffic rules, resulting in entering the grey terminal state in Fig. 1(a). While in such a case the route cannot be successfully completed any more, the time cost drops to 0 for all the following steps, and thus changing lane, while potentially violating road rules, results in lower expected total time. In comparison, if the time cost assigned to the terminal state corresponding to traffic rules violation was 1, the optimal strategy would be to brake in a traffic jam and change lane when approaching a pedestrian with the expected total time approximately 3.46.

### 3.5. Discounted reward properties

To solve numerical queries of the form  $R_{\max}^r = ?[D^\beta]$ , we present their reduction to probabilistic reachability queries [2], as well as to total reward queries [34].

Let  $\mathcal{G}$  be a stochastic game with a reward structure  $r$ . First, we describe the reduction to probabilistic reachability. Without loss of generality, assume that all rewards take values in the interval  $[0, 1]$ . Construct a game  $\mathcal{G}' = (S', (S_1, S_2, S'_p), \Delta')$  defined as follows. First, add two terminal probabilistic states  $s_0, s_1 \in S'_p$ . Next, for every Player 1 or Player 2 state  $s \in S_1 \cup S_2$  and every  $s' \in S$  such that  $\Delta(s, s') = 1$ , we add a new probabilistic state  $t_{s,s'} \in S'_p$  and define  $\Delta'(s, t_{s,s'}) = 1, \Delta'(s, s') = 0$ . For probabilistic states  $t_{s,s'}$ , we let

$$\Delta'(t_{s,s'}, t) = \begin{cases} \beta(1 - r(s)) & \text{if } t = s_0, \\ \beta r(s) & \text{if } t = s_1, \\ 1 - \beta & \text{if } t = s', \\ 0 & \text{otherwise.} \end{cases}$$

Finally, for probabilistic states  $s \in S_p$  we define

$$\Delta'(s, s') = \begin{cases} \beta(1 - r(s)) & \text{if } s' = s_0, \\ \beta r(s) & \text{if } s' = s_1, \\ (1 - \beta)\Delta(s, s') & \text{otherwise.} \end{cases}$$

Consider  $AP = \{a\}$  and the labelling function defined as  $L(s_1) = \{a\}$  and  $L(s) = \emptyset$  otherwise. It holds that every optimal strategy for  $\mathcal{G}'$  with respect to the numerical query  $R_{\max} = ?[\mathbb{F} a]$  is also an optimal strategy for  $\mathcal{G}$  with respect to the numerical query  $R_{\max}^r = ?[D^\beta]$ .

Next, we present a reduction to total reward queries that builds on the same principles. Construct a game  $\mathcal{G}' = (S', (S_1, S_2, S'_p), \Delta')$  and a reward structure  $r'$  defined as follows. First, for every  $s \in S_1 \cup S_2$  add a new probabilistic state  $t_s \in S'_p$  and add a new terminal probabilistic state  $s_f \in S'_p$ . For all states  $s, s' \in S$  such that  $\Delta(s, s') > 0$  define  $\Delta'(s, t_s) = \Delta(s, s'), \Delta'(s, s') = 0$ . For probabilistic states  $t_s$ , we let

$$\Delta'(t_s, t) = \begin{cases} \beta & \text{if } t = s_f, \\ 1 - \beta & \text{if } t = s, \\ 0 & \text{otherwise.} \end{cases}$$

The reward structure  $r'$  is such that  $r'(s) = r(s)$  for  $s \in S$  and the reward is 0 otherwise. It holds that every optimal strategy for  $\mathcal{G}'$  with respect to numerical query  $R_{\max}^{r'} = ?[C]$  is also an optimal strategy for  $\mathcal{G}$  with respect to the numerical query  $R_{\max}^r = ?[D^\beta]$ .

### 3.6. Average reward properties

Unlike other infinite horizon properties such as total reward, expected average reward disregards all transient behaviour. Nevertheless, it was proven in [50,58] that pure memoryless strategies still suffice for both players to win. In [58], the authors show that, for a game  $\mathcal{G}$  with an average reward numerical query  $R_{\max}^r = ?[S]$ , there exists a discount factor  $\beta$  such that any strategy optimal in  $\mathcal{G}$  with respect to discounted reward numerical query  $R_{\max}^r = ?[D^\beta]$  is also optimal with respect to the average reward query. Moreover, for a strategy to be optimal for the latter, it suffices if it is optimal for the former for every  $\beta$  sufficiently close to 1. The authors in [2] then compute a concrete value of the discount factor and prove that a solution to numerical queries of the form  $R_{\max}^r = ?[S]$  can be found as a solution to numerical query  $R_{\max}^r = ?[D^\beta]$  for any  $\beta \in [\beta^*, 1)$ , where

$$\beta^* = 1 - \left( (n!)^2 \cdot 2^{2n+3} \cdot r_{\max}^{2n^2} \right)^{-1},$$

$$n = |S|,$$

$$r_{\max} = \max_{s \in S} r(s).$$

Alternatively, it has been shown in [10] that, unlike for the general case, for stochastic games that are ergodic, i.e., the optimal average reward is independent of an initial state of the game, optimal strategies can be constructed using locally optimal moves. The algorithm reduces the game, using a potential transformation, to a canonical form in which the locally optimal moves are also globally optimal. The algorithm is pseudo-polynomial if the game has a constant number of probabilistic states, and otherwise it can be up to exponential in the number of probabilistic states.

Finally, for a related property called an almost-sure average reward property, where the aim is to achieve a certain average reward with probability 1 (as opposed to the expected average), one can use the approach presented in [6]. As the algorithm was primarily designed to solve games with respect to a conjunction of such properties, we discuss the property and the approach in more detail in Section 4.7.

### 3.7. Branching-time properties

The definition of properties in Definition 6 allows one to reason about probabilistic and expected reward properties of games over linear time. Below, we extend the definition to branching time by combining properties in a PCTL\*-like fashion. The resulting logic has been introduced and studied in [35,73].

**Definition 8** (Branching-time property). A branching-time property is a formula  $\phi$  in the following grammar:

$$\phi ::= \text{true} | a | \neg \phi | \phi \wedge \phi | \mathbb{P}_{\bowtie p}[\psi] | R_{\bowtie}^r[\mathbb{F}^* \phi],$$

$$\psi ::= \phi | \text{true} | a | \neg \psi | \psi \wedge \psi | \mathbb{X} \psi | \psi \cup \psi,$$



where  $a \in AP$  is an atomic proposition,  $\bowtie \in \{\leq, <, \geq, >\}$ ,  $p \in [0, 1]$ ,  $ris$  a reward structure,  $x \in \mathbb{R}$ , and  $*$   $\in \{0, \infty, c\}$ .

Note that, unlike in Definition 6, here we only allow  $\rho = \mathbb{F}^* a$  for the reward operator  $R_{\bowtie x}^r[\rho]$ . The semantics for branching-time properties is as shown in Table 1, with formulas  $\phi$  being interpreted over states  $s \in S$  of the game  $\mathcal{G}$ , and formulas  $\psi$  being interpreted over paths  $\lambda \in \text{Path}_{\mathcal{G}}$  of the game. For completeness, the definitions not shown in Table 1 are given as follows:

$$\begin{aligned} s \models \text{true} & \quad \text{always,} \\ s \models a & \quad \iff a \in L(s), \\ s \models \neg\phi & \quad \iff s \not\models \phi, \\ s \models \phi_1 \wedge \phi_2 & \iff s \models \phi_1 \text{ and } s \models \phi_2, \\ \lambda \models \phi & \quad \iff \lambda_0 \models \phi. \end{aligned}$$

The problem of verification for branching-time properties is formulated in a way analogous to Problem 1. Given a game  $\mathcal{G}$ , its state  $s$  and a branching-time property  $\phi$ , the verification problem can be solved similar to PCTL\* model checking for MDPs [8]. Intuitively, the solution is achieved by traversing the parse tree of  $\phi$  in a bottom-up fashion. Iteratively, the innermost subformulas, which are either probabilistic LTL properties or total reward properties, are solved using techniques discussed in Sections 3.2–3.4, and replaced by new atomic propositions such that a state is labelled with the new proposition if and only if the answer to Problem 1 for the corresponding property is ‘yes’. For full description, see [73].

On the other hand, the formulation of the strategy synthesis problem in Problem 2 does not extend to branching-time properties in a straightforward way. For example, for a formula  $\phi = P_{\bowtie p_1}[\psi_1] \wedge P_{\bowtie p_2}[\psi_2]$ , the semantics implies that  $s \models \phi$  if there exists a Player 1 strategy  $\pi_1 \in \Pi$  such that, for all Player 2 strategies  $\sigma \in \Sigma$ , it holds  $\Pr_{\mathcal{G},s}^{\pi_1,\sigma}(\psi_1) \bowtie p_1$ , and, at the same time, there exists a Player 1 strategy  $\pi_2 \in \Pi$  (possibly different than  $\pi_1$ ) such that, for all Player 2 strategies  $\sigma \in \Sigma$ , it holds  $\Pr_{\mathcal{G},s}^{\pi_2,\sigma}(\psi_2) \bowtie p_2$ . This means that, even if the satisfaction  $s \models \phi$  holds, there may not exist a single Player 1 strategy that is a witness to it. While the problem of strategy synthesis cannot be formulated in this way for the full logic from Definition 8, there exist branching-time properties for which the problem can be formulated and is indeed interesting. For example, consider a formula  $\phi = P_{\bowtie p}[\mathbb{F} R_{\bowtie x}[\mathbb{F}^c a]]$  which states that, starting from a state  $s \in S$ , there exists a Player 1 strategy  $\pi_1$  such that, with probability that satisfies the bound  $p$  and under any Player 2 strategy, the game reaches a state  $s' \in S$  from which there exists a (possibly different) Player 1 strategy  $\pi_2$  that guarantees that the expected total reward before reaching a state labelled with proposition  $a$  satisfies the bound  $x$ . Note that, if indeed  $s \models \phi$ , a witness to this satisfaction is the strategy  $\pi_1$  and the strategy  $\pi_2$  does not need to be constructed. In [73], the author discussed the strategy synthesis problem for a fragment of branching-time properties in Definition 8. To be specific, an algorithm is described to synthesise a winning Player 1 strategy for branching-time properties of the form  $P_{\bowtie p}[\psi]$  and  $R_{\bowtie x}[\mathbb{F}^* \phi]$ . Similar to the verification case above, the algorithm is based on traversing the parse tree of the formula and constructing strategies for probabilistic LTL and total reward properties using techniques from Sections 3.2 to 3.4.

**Remark 1.** The semantics of branching-time properties can also be defined in a different way as follows. First, let the properties in Definition 6 be interpreted over Markov chains rather than over states of a game. More formally, given a game  $\mathcal{G}$ , its state  $s \in S$ , a Player 1 strategy  $\pi \in \Pi$ , and a Player 2 strategy  $\sigma \in \Sigma$ , we let  $\mathcal{G}, s, \pi, \sigma \models P_{\bowtie p}[\psi]$  if and only if  $\Pr_{\mathcal{G},s}^{\pi,\sigma}(\psi) \bowtie p$ , and the semantics for the reward operator  $R_{\bowtie x}^r[\rho]$  is defined in an analogous way. Given a state  $s \in S$  and a property  $\phi = P_{\bowtie p}[\psi]$  or  $\phi = R_{\bowtie x}^r[\rho]$ , the verification

problem then asks for existence of a Player 1 strategy  $\pi$  such that, for all Player 2 strategies  $\sigma$ , it holds  $\mathcal{G}, s, \pi, \sigma \models \phi$ , and, likewise, the strategy synthesis problem aims to construct such a Player 1 strategy. This semantics of properties can be straightforwardly extended to branching-time properties in Definition 8. For example, for a formula  $\phi = \phi_1 \wedge \phi_2$ , it holds that  $\mathcal{G}, s, \pi, \sigma \models \phi$  if and only if  $\mathcal{G}, s, \pi, \sigma \models \phi_1$  and  $\mathcal{G}, s, \pi, \sigma \models \phi_2$ . Especially, note the difference between this semantics of a conjunction and the semantics given earlier in this section, after Definition 8. Both formulations of the verification and strategy synthesis problems can now be directly extended from simple linear-time and reward properties to branching-time properties. The problems are, however, very intricate. It has been shown in [16] that, already for PCTL, a fragment of the above branching-time properties with the probabilistic operator, the games are generally not determined. That means that there might exist states of the games from which neither of the two players has a winning strategy. Moreover, winning strategies may require (possibly infinite) memory and/or randomisation. Therefore, it makes sense to formulate the verification and strategy synthesis problems for specific subclasses of strategies, e.g., pure finite-memory or randomised infinite-memory strategies. In [16], the authors prove several complexity results for the verification problem with restricted classes of properties and strategies, including an undecidability result for a simple fragment of PCTL and finite-memory strategies.

### 3.8. Stochastic games with multiple players

The definition of a stochastic game in Definition 1 can be extended to a multi-player stochastic game as follows.

**Definition 9 (Multi-player stochastic game).** A multiplayer stochastic game is a tuple  $\mathcal{G} = (S, (S_1, \dots, S_n, S_p), \Delta)$ , where  $S$  is a finite set of states partitioned into a set of probabilistic states  $S_p$  and sets  $S_1, \dots, S_n$  of states of Players 1 to  $n$ , respectively. Probabilistic transition function  $\Delta : S \times S \rightarrow [0, 1]$  is such that for all states  $s \in \bigcup_{1 \leq i \leq n} S_i$  it holds that  $\Delta((s, s')) \in \{0, 1\}$  for every  $s' \in S$ , and for probabilistic states  $s \in S_p$  we have  $\sum_{s' \in S} \Delta((s, s')) = 1$ .

A strategy for Player  $i \in \{1, \dots, n\}$  is defined as in Definition 4. A strategy for a coalition of players  $C \subseteq \{1, \dots, n\}$  consists of a set of strategies for the players in the coalition, one for each player. Definitions of linear- and branching-time properties in Definitions 6 and 8 can be extended to consider coalitions of players  $C \subseteq \{1, \dots, n\}$  using syntax  $\langle\langle C \rangle\rangle \phi$ . For branching-time properties, the resulting logic is called rPATL\*; for detailed definition of the semantics, see [35,73]. Intuitively, multi-player verification and strategy synthesis problems ask whether and how players of the coalition  $C$  can cooperatively guarantee satisfaction of the property  $\phi$ . These problems reduce to the corresponding Problems 1 and 2 for the stochastic game with two players, where Player 1 represents the collective behaviour of the coalition  $C$  and Player 2 represents the remaining players  $\{1, \dots, n\} \setminus C$ .

## 4. Multi-objective game solving

In this section, we discuss the problem of strategy synthesis, where the goal is to simultaneously satisfy a certain combination of properties of the form in Definition 6.

$$\begin{aligned}
V_n^*(s) &= \begin{cases} \{\mathbf{x} \in \mathbb{R}_{\geq 0}^m \mid \mathbf{x} \leq \mathbf{r}(s)\} & \text{if } n = 0, \\ \text{dwc}(\mathbf{r}(s) + \text{conv}(\bigcup_{\Delta(s,s')=1} V_{n-1}^*(s'))) & \text{if } n > 0 \text{ and } s \in S_1, \\ \text{dwc}(\mathbf{r}(s) + \bigcap_{\Delta(s,s')=1} V_{n-1}^*(s')) & \text{if } n > 0 \text{ and } s \in S_2, \\ \text{dwc}(\mathbf{r}(s) + \sum_{\Delta(s,s')>0} \Delta(s,s') \cdot V_{n-1}^*(s')) & \text{if } n > 0 \text{ and } s \in S_p, \end{cases} \\
x \cdot X &= \{x \cdot \mathbf{x} \mid \mathbf{x} \in X\}, \\
\mathbf{x} + X &= \{\mathbf{x} + \mathbf{x}' \mid \mathbf{x}' \in X\}, \\
\text{dwc}(X) &= \{\mathbf{y} \mid \exists \mathbf{x} \in X : \mathbf{y} \leq \mathbf{x}\}, \\
\text{conv}(X) &= \{\mathbf{y} \mid \exists \mathbf{x}, \mathbf{x}' \in X, \alpha \in [0, 1] : \mathbf{y} = \alpha \mathbf{x} + (1 - \alpha) \mathbf{x}'\}.
\end{aligned}$$

**Fig. 5.** Iterative computation of an  $\varepsilon$ -approximation of the Pareto set for a multi-objective total reward property. Here,  $x \in \mathbb{R}_{\geq 0}$  is a real number,  $\mathbf{x} \in \mathbb{R}_{\geq 0}^m$  is a vector,  $X \subseteq \mathbb{R}_{\geq 0}^m$  is a set, and  $\leq$  is the componentwise partial order on  $\mathbb{R}_{\geq 0}^m$ . Given a stopping game  $\mathcal{G}$  with multiple reward structures  $\mathbf{r}$  and a multi-objective total reward property  $\Phi(\mathbf{x})$ , the approximation is computed for every state  $s \in S$  in  $k = |S| + \lceil |S| \cdot \frac{\ln(\varepsilon \cdot (n \cdot M)^{m+1})}{\ln(1 - \delta)} \rceil$  iterations, where  $M = |S| \cdot \frac{\max_{s \in S} \mathbf{r}(s)}{\delta}$ ,  $\delta = \Delta_{\min}^{|S|}$ , and  $\Delta_{\min}$  is the smallest positive probability in  $\mathcal{G}$ .

#### 4.1. Problem formulation

**Definition 10** (*Multi-objective property*). A multi-objective property  $\Phi$  is a conjunction of properties of the form  $\mathbb{P}_{\geq p}[\psi]$  and  $\mathbb{R}_{\geq x}^r[\rho]$ .

The semantics of a multi-objective property involving  $n$  probabilistic properties and  $m$  reward properties, i.e.,

$$\Phi = \bigwedge_{i=1}^n \mathbb{P}_{\geq p_i}[\psi_i] \wedge \bigwedge_{j=1}^m \mathbb{R}_{\geq x_j}^r[\rho_j]$$

is defined over states  $s \in S$  of a game  $\mathcal{G}$  as follows. It holds that  $s \models \Phi$  if and only if there exists a Player 1 strategy  $\pi \in \Pi$  such that, for all Player 2 strategies and  $1 \leq i \leq n$ , it holds  $\Pr_{\mathcal{G},s}^{\pi,\sigma}(\psi_i) \geq p_i$ , and similarly, for all Player 2 strategies and  $1 \leq j \leq m$ , it holds  $\mathbb{E}_{\mathcal{G},s}^{\pi,\sigma}(\text{rew}(r_j, \rho_j)) \geq r_j$ . Note that, while such a conjunction is syntactically a branching-time property according to Definition 8, its semantics is different. In fact, the semantics of a multi-objective property is in line with the alternative semantics of branching-time properties discussed in Remark 1.

Every property of the form  $\mathbb{R}_{\leq x}^r[\rho]$  is equivalent to property  $\mathbb{R}_{\geq -x}^r[\rho]$  and, similarly, every property of the form  $\mathbb{P}_{\leq p}[\psi]$  is equivalent to  $\mathbb{P}_{\geq 1-p}[\neg\psi]$ . Thus, the above definition of a multi-objective property covers conjunctions of all properties from Definition 6.

While in Definition 10 we define multi-objective properties only as conjunctions (unlike, e.g., in [32], where any positive Boolean combinations are allowed), we address more complex combinations for some classes of properties further in this section.

Let  $\Phi$  be a multi-objective property involving  $n$  probabilistic properties and  $m$  reward properties. For simplicity, we use  $\mathbf{r} = (r_1, \dots, r_m)$  to denote the vector of reward structures and  $\mathbf{r}(s) = (r_1(s), \dots, r_m(s))$ , for every  $s \in S$ . Similarly,  $\mathbf{p} = (p_1, \dots, p_n)$  and  $\mathbf{x} = (x_1, \dots, x_m)$  denote the vectors of probability and reward bounds. Instead of  $\Phi$ , we sometimes write  $\Phi(\mathbf{p}, \mathbf{x})$  to emphasise the corresponding bounds. We say that  $\Phi(\mathbf{p}, \mathbf{x})$ , or the vector of bounds  $(\mathbf{p}, \mathbf{x})$  for  $\Phi$ , is *achievable* if and only if there exists a winning strategy for Player 1 that guarantees all properties in  $\Phi$  with bounds  $\mathbf{p}, \mathbf{x}$ . The optimal achievable vectors of bounds are called Pareto vectors.

**Definition 11** (*Pareto set*). Let  $\Phi$  be a multi-objective property involving  $n$  probabilistic and  $m$  reward properties. A vector  $(\mathbf{p}, \mathbf{x}) \in \mathbb{R}^{n+m}$  is called a Pareto vector if the property  $\Phi(\mathbf{p} - \varepsilon, \mathbf{x} - \varepsilon)$  is achievable for every  $\varepsilon > 0$  and  $\Phi(\mathbf{p} + \varepsilon, \mathbf{x} + \varepsilon)$  is not achievable for any  $\varepsilon > 0$ . Pareto set  $P$  is the set of all Pareto vectors for  $\Phi$ .

The problems of multi-objective verification and strategy synthesis are formulated analogously to the single-objective case stated in Problems 1 and 2. Unlike in the single-objective case, optimal strategies might not exist. This is already true, for

example, for *precise-value games*, where the objective is to achieve a precise value (related to probability or reward). Such a property can be expressed as a conjunction of two single-objective properties, and it has been shown in [34] that, in these games, a winning strategy may not exist for either of the two players.

In this section, we discuss existing solutions to multi-objective strategy synthesis depending on what type of properties are being combined. The solutions compute  $\varepsilon$ -approximations of Pareto sets and the corresponding  $\varepsilon$ -optimal strategies.

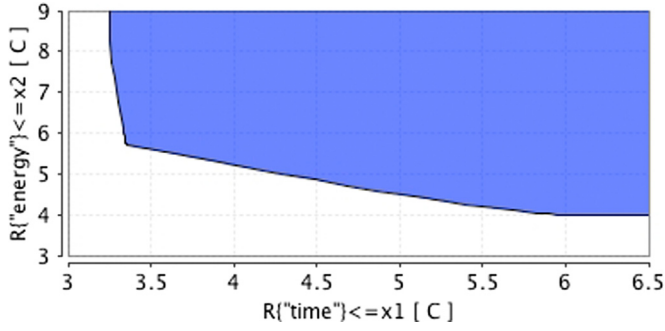
**Definition 12** (*Pareto set approximation*). For  $\varepsilon > 0$ , an  $\varepsilon$ -approximation of the Pareto set is a set of vectors  $Q$  such that for every  $(\mathbf{q}, \mathbf{y}) \in Q$  there exists a Pareto vector  $(\mathbf{p}, \mathbf{x}) \in P$  with  $\|(\mathbf{q}, \mathbf{y}) - (\mathbf{p}, \mathbf{x})\| \leq \varepsilon$ , and vice versa, for every Pareto vector  $(\mathbf{p}, \mathbf{x}) \in P$  there exists a vector  $(\mathbf{q}, \mathbf{y}) \in Q$  with  $\|(\mathbf{q}, \mathbf{y}) - (\mathbf{p}, \mathbf{x})\| \leq \varepsilon$ , where  $\|\cdot\|$  is the Manhattan distance defined as the sum of componentwise differences.

#### 4.2. Multi-objective total reward properties

Here we discuss the strategy synthesis problem for multi-objective properties that only involve total reward properties of the form  $\mathbb{R}_{\geq x}^r[C]$ . The problem has been recently investigated in [32,33]. As discussed in Section 4.1, there exist games in which neither Player 1 nor Player 2 have winning strategies. Moreover, for stopping games with precise-value objectives randomised exponential memory strategies may be needed for Player 1 to win [34], and, for stopping games with general total reward objectives, randomised infinite memory strategies may be required [32]. The problem of whether there exists a pure winning strategy for Player 1, in stopping games, is undecidable [32].

The  $\varepsilon$ -approximation of the Pareto set for a stopping game  $\mathcal{G}$  and a multi-objective property  $\Phi(\mathbf{x})$  can be computed using the iteration algorithm in Fig. 5. Intuitively, the set  $V_n^*(s)$  for a state  $s \in S$  computed in the  $n$ th iteration of the algorithm is the downward closure of vectors of bounds achievable by Player 1, from  $s$ , in the finite time horizon of up to  $n$  steps. As Player 1 can randomise between successors of his/her states, the set  $V_n^*(s)$  for  $s \in S_1$  is computed as a downward, convex closure of the union of  $V_{n-1}^*(s')$ , for all  $s'$  such that  $\Delta(s, s') = 1$ . For  $s \in S_2$ , the bounds must be achievable for all successor states and, hence, we take the intersection. Finally, for probabilistic states  $s \in S_p$ , we consider the sum weighted by the corresponding probabilistic distribution.

Given an  $\varepsilon$ -approximation of the Pareto set, the corresponding  $\varepsilon$ -optimal Player 1 strategy can be constructed as described in [33]. To succinctly represent such strategies using a finite set of memory elements, the authors extend the definition of a strategy from Definition 4 to allow *stochastic memory update*, i.e., the memory update function is of type  $\pi_u : M \times S \rightarrow \mathcal{D}(M)$  and the initial memory element function is  $\pi_{\text{init}} : S \rightarrow \mathcal{D}(M)$ . In the construction, the vertices of approximation sets  $V_n^*(s), s \in S$ , act as memory



**Fig. 6.** An  $\varepsilon$ -approximation of the Pareto set for multi-objective total reward property  $R_{\leq x_1}^{\text{time}}[C] \wedge R_{\leq x_2}^{\text{energy}}[C]$  for the game introduced in [Example 1](#).

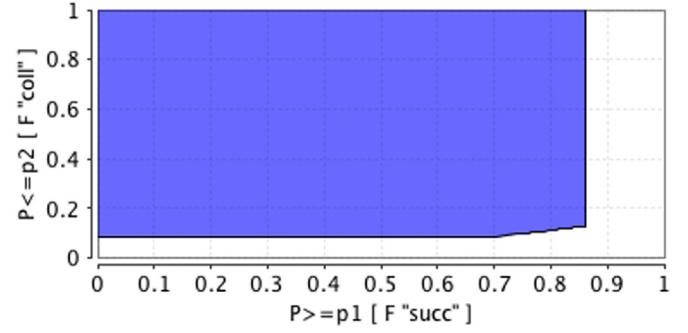
elements and represent the vector of reward bounds that the strategy currently aims to achieve. The distributions in functions  $\pi_u$  and  $\pi_{\text{init}}$  are constructed so that the expected value of the next memory element is an  $\varepsilon$ -approximation of the target reward bounds  $\mathbf{x}$ . Comparing to deterministic update strategies from [Definition 4](#), with stochastic memory update strategies the memory required to win reduces from up to exponential to linear for stopping games with precise-value objectives [\[34\]](#) and from up to infinite to finite for stopping games with general total reward objectives [\[32\]](#).

Besides conjunctions of total reward properties, the authors in [\[32\]](#) discuss multi-objective properties constructed as a disjunction of total reward properties. It is shown that there exists a strategy achieving the disjunction if and only if there exists a strategy achieving a certain single-objective total reward property and thus pure memoryless strategies suffice to achieve a disjunction of total reward properties. Moreover, an algorithm for computing an  $\varepsilon$ -approximation of the Pareto sets for stopping games is presented. By combining the two algorithms for conjunctions and disjunctions, we obtain a solution for any positive Boolean combination of total reward properties for stopping stochastic games through first rewriting the combination into conjunctive normal form.

**Example 5.** For the autonomous car from [Example 1](#), we consider the conjunction of total reward properties  $R_{\leq x_1}^{\text{time}}[C] \wedge R_{\leq x_2}^{\text{energy}}[C]$  that aims to compute a strategy that, simultaneously, guarantees that the expected total time is at most  $x_1$ , and the expected total energy is at most  $x_2$ . An  $\varepsilon$ -approximation of the Pareto set computed using the algorithm in [Fig. 5](#) for  $\varepsilon = 0.1$  is shown in [Fig. 6](#). For example, for  $x_1 = 3.4, x_2 = 5.7$ , the winning strategy generated by PRISM-games for the property is a stochastic memory update strategy with 155 memory elements that, in a non-trivial way, probabilistically switches between changing lane and honking for both hazards. The strategy can be viewed at [\[52\]](#).

#### 4.3. Multi-objective probabilistic reachability properties

Using the reduction of probabilistic reachability properties to total reward properties described in [Section 3.2](#), the iterative algorithm in [Fig. 5](#) can be adapted to compute  $\varepsilon$ -approximations of Pareto sets for any stopping stochastic game with a multi-objective property that involves only probabilistic reachability properties. Disjunctions of probabilistic reachability properties can be addressed in a similar manner, using the reduction from [Section 3.2](#) with the algorithm for disjunctions of total reward properties from [\[32\]](#). Hence, stopping games with any positive Boolean combination of probabilistic reachability properties can be handled.



**Fig. 7.** An  $\varepsilon$ -approximation of the Pareto set for multi-objective probabilistic LTL property  $P \geq p_1[F \text{succ}] \wedge P \geq p_2[G \neg \text{acc}]$  for the game introduced in [Example 1](#).

#### 4.4. Multi-objective probabilistic LTL properties

For stopping stochastic games, the strategy synthesis problem for multi-objective properties involving only probabilistic LTL properties  $P \geq p_i[\psi_i], 1 \leq i \leq n$ , has been discussed in [\[33\]](#). In this case, the solution is to construct a deterministic Rabin automaton for each  $\psi_i$  and then build a synchronous product of all the automata and the original game  $\mathcal{G}$ , with a new terminal state which is entered after  $\mathcal{G}$  enters any of its terminal states. Since  $\mathcal{G}$  is stopping, it indeed suffices to analyse satisfaction of formulas upon reaching a terminal state. The problem then reduces to solving the product game with respect to a multi-objective reachability property. Finally, since the resulting product game is again stopping, we can apply the approach from [Section 4.3](#). It follows that, in fact, we can solve any positive Boolean combination of probabilistic LTL properties for stopping stochastic games.

For general stochastic games, the strategy synthesis problem for multi-objective probabilistic LTL properties remains open.

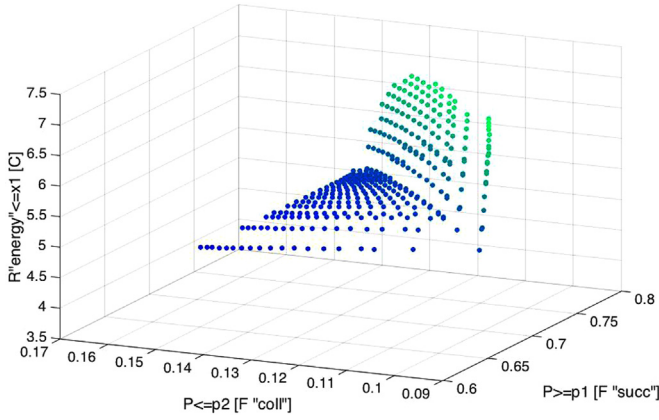
**Example 6.** Consider the conjunction  $P \geq p_1[F \text{succ}] \wedge P \geq p_2[G \neg \text{acc}]$  of the reachability and safety LTL properties for the autonomous car example. An  $\varepsilon$ -approximation of the Pareto set computed using the algorithm in [Fig. 5](#) for  $\varepsilon = 0.001$  is shown in [Fig. 7](#). For example, for  $p_1 = 0.7, p_2 = 0.2$ , there exists a pure memoryless winning strategy (in PRISM-games generated as a stochastic memory update strategy with 25 memory elements) that brakes in a traffic jam and honks when approaching a pedestrian. The strategy can be viewed at [\[52\]](#).

#### 4.5. Mixed multi-objective properties and compositional strategy synthesis

From the sections above it follows that, for stopping stochastic games, we can  $\varepsilon$ -approximate Pareto sets for any positive Boolean combination of total reward, probabilistic reachability and probabilistic LTL properties.

In [\[7\]](#), the authors also discuss a different approach to multi-objective strategy synthesis through composition. First, a composition of stochastic games is defined, in a way that preserves the identity of Player 1. Here, the component stochastic games  $\mathcal{G}_i, i \in I = \{1, \dots, N\}$ , that are being composed are considered to have labels on Player 1 and Player 2 transitions referred to as actions and, in the composed game  $\mathcal{G} = \parallel_{i \in I} \mathcal{G}_i$ , component games synchronise on actions. Properties of the component games, as well as the composed game, are then defined over sequences of actions called traces, rather than over paths as in [Definition 6](#). Under the assumption that the component games are compatible, i.e., all actions of Player 1 in each composite game are enabled and fully controlled by Player 1, the Player 1 strategy  $\pi = \parallel_{i \in I} \pi_i$  for  $\mathcal{G}$  that is a composition of Player 1 strategies  $\pi_i$  for component games  $\mathcal{G}_i$  preserves all properties. More precisely, if strategies  $\pi_i$  guarantee a





**Fig. 8.** An  $\varepsilon$ -approximation of the Pareto set for mixed multi-objective property  $P \geq p_1 [F \text{ succ}] \wedge P \geq p_2 [G \neg \text{acc}] \wedge R_{\leq x_1}^{\text{energy}} [C]$  for the game introduced in Example 1.

(possibly multi-objective) property  $\Phi_i$  in component games  $\mathcal{G}_i$ , then the composed strategy  $\pi$  guarantees property  $\Phi$  in  $\mathcal{G}$ , where  $\Phi$  is any property for the composed game that can be derived from  $\Phi_i$  using, for example, assume-guarantee rules in [53]. In particular, Player 1 of different component games can cooperate to achieve a common goal: if in one component game Player 1 guarantees a property  $\Phi_2$  under some assumption  $\Phi_1$  on the environment, i.e.,  $\Phi_1 \Rightarrow \Phi_2$ , and Player 1 in a different component game ensures  $\Phi_1$ , then the composition satisfies property  $\Phi_2$ .

The framework for compositional strategy synthesis presented in [7] first computes an  $\varepsilon$ -approximation  $Q$  of the Pareto set for  $\Phi$  based on  $\varepsilon$ -approximations  $Q_i$  of Pareto sets for  $\Phi_i$ . For a chosen achievable vector of bounds  $(\mathbf{p}, \mathbf{x})$  for  $\Phi$ , Player 1 strategies  $\pi_i$  are synthesised for component games  $\mathcal{G}_i$  that achieve  $\Phi_i(\mathbf{p}_i, \mathbf{x}_i)$ , where  $(\mathbf{p}_i, \mathbf{x}_i)$  are the bounds obtained by projecting  $(\mathbf{p}, \mathbf{x})$  from  $Q$  to  $Q_i$ . The composed strategy  $\pi = \parallel_{i \in I} \pi_i$  then achieves  $\Phi(\mathbf{p}, \mathbf{x})$ . Note that in order to take full advantage of assume-guarantee rules, we would need to be able to synthesise strategies for arbitrary Boolean combinations of properties.

**Example 7.** An example illustrating the compositional approach to multi-objective game solving can be found in [78]. Here, we present results obtained using reductions to total reward properties as discussed earlier in this section. We combine various properties for the autonomous car from Example 1 in the following conjunction:

$$P \geq p_1 [F \text{ succ}] \wedge P \geq p_2 [G \neg \text{acc}] \wedge R_{\leq x_1}^{\text{energy}} [C].$$

An  $\varepsilon$ -approximation of the Pareto set computed using the algorithm in Fig. 5 for  $\varepsilon = 0.01$  is shown in Fig. 8. For example, for  $p_1 = 0.7, p_2 = 0.13, x_1 = 5.7$ , the winning strategy for the property generated by PRISM-games is a stochastic memory update strategy with 775 memory elements that, in a non-trivial way, probabilistically chooses between all three reactions for a traffic jam and between honking and braking for a pedestrian. The strategy can be viewed at [52].

#### 4.6. Multi-objective discounted reward properties

To the best of our knowledge, properties that combine multiple discounted reward properties have only been addressed for the subclass of stochastic games with one player and probabilistic states, i.e., MDPs [29]. Note that the reduction from discounted reward to total reward properties discussed in Section 3.5 alters the transition probabilities of the game depending on the discount factor  $\beta \in (0, 1)$ . It follows that, using this reduction, the iterative algorithm in Fig. 5 can be applied to compute  $\varepsilon$ -approximations of

Pareto sets for any stochastic game with a Boolean combination of discounted reward properties with the same discount factor.

#### 4.7. Multi-objective average reward properties

For multi-objective synthesis with multiple average reward properties, we cannot apply the approach presented in Section 4.2. The reason is that the algorithm in Fig. 5 approximates the Pareto set in a finite number of iterations by combining the achievable values of successive states. However, infinite horizon properties such as the expected average reward disregard all transient behaviour. Preliminary results for multi-objective average reward synthesis have been presented in [6], where the authors consider conjunctions of a special case of the (single-objective) expected average reward properties, almost sure average reward properties  $R_{=1, \geq x}^r [S]$ , that require that the average reward achieved over a path is above a given bound with probability 1. Formally, given a game  $\mathcal{G}$ , its state  $s \in S$ , a reward structure  $r$  and Player 1 and Player 2 strategies  $\pi, \sigma$ , respectively, the relation  $\mathcal{G}, s, \pi, \sigma \models R_{=1, \geq x}^r [S]$  holds true if and only if

$$\Pr_{\mathcal{G}, s}^{\pi, \sigma} (\{\lambda \in \text{Path}_s | \text{rew}(r, S)(\lambda) \geq x\}) = 1.$$

Note that this implies  $\mathcal{G}, s, \pi, \sigma \models R_{\geq x}^r [S]$ , but the reverse implication is not necessarily true, see [6] for an example.

The authors show that synthesis for multi-objective properties of this type reduces to synthesis for multi-objective expected energy properties. Intuitively, given a reward structure  $r$  possibly assigning both positive and negative values to states, the expected energy property requires that, for every state  $s \in S$  of the game, there exists a bound  $x$  such that the expected total reward obtained starting from  $s$  in  $k$  steps is at least  $x$  for all  $k \geq 0$ . Only finite-memory (possibly stochastic memory update) strategies are considered and it holds that every Player 1 strategy that satisfies the expected energy property also satisfies the almost sure average reward property over the same reward structure with bound 0, and hence the same applies for  $\varepsilon$ -optimal strategies. As  $R_{=1, \geq x}^r [S]$  is equivalent to  $R_{=1, \geq 0}^{r-x} [S]$ , the above property can be adapted for any almost sure average reward property.

Given a game  $\mathcal{G}$ , multiple reward structures  $\mathbf{r}$  (allowing both positive and negative reward values), a vector of bounds  $\mathbf{x}$  for the almost sure average reward properties and  $\varepsilon > 0$ , the authors design an algorithm that terminates with a finite-memory stochastic update  $\varepsilon$ -optimal strategy if the vector  $\mathbf{x}$  is achievable. The algorithm uses value iteration to compute  $\varepsilon$ -optimal strategies for the corresponding multi-objective expected energy property.

Finally, the authors generalise the almost sure average reward property to a ratio reward property  $R_{=1, \geq x}^{r/c} [S]$ , where, given a game  $\mathcal{G}$ , its state  $s \in S$ , two reward structures  $r, c$  and Player 1 and Player 2 strategies  $\pi, \sigma$ , respectively, the relation  $\mathcal{G}, s, \pi, \sigma \models R_{=1, \geq x}^{r/c} [S]$  holds true if and only if

$$\Pr_{\mathcal{G}, s}^{\pi, \sigma} (\{\lambda \in \text{Path}_s | \text{rew}(r/c, S)(\lambda) \geq x\}) = 1, \text{ where}$$

$$\text{rew}(r/c, S)(\lambda) := \liminf_{k \rightarrow \infty} \frac{\text{rew}(r, C^{\leq k})(\lambda)}{\text{rew}(c, C^{\leq k})(\lambda) + 1}.$$

Note that property  $R_{=1, \geq x}^r [S]$  is equivalent to property  $R_{=1, \geq x}^{r/c} [S]$  for a reward structure  $c(s) = 1, s \in S$ . To solve the strategy synthesis problem for a ratio reward property  $R_{=1, \geq x}^{r/c} [S]$ , it suffices to solve the problem for the almost sure average reward property  $R_{=1, \geq 0}^{r-xc} [S]$ , and this can be straightforwardly extended to multi-objective properties using vectors.



## 5. Implementation

Software tools for analysis of games include the following. Among the tools that focus on subclasses of stochastic games, QUASY [30] offers synthesis of strategies for MDPs and non-stochastic games with mean-payoff objectives. Methods for expected ratio reward objectives are implemented in [44]. Multi-Gain [15] solves MDPs with multi-objective mean-payoff properties. PRISM [56] performs verification for MDPs with single- and multi-objective properties, namely probabilistic LTL and expected total reward. MOCHA [1] is a tool for verification and strategy synthesis for non-stochastic games with alternating-time temporal logic (ATL) specifications, as well as for automatic checking of assume-guarantee queries.

For stochastic games, GIST [31] offers support for qualitative verification, *i.e.*, probability 1 or non-zero probability, of stochastic games with  $\omega$ -regular properties. GAVS+ [37] includes implementation of value and policy iteration for stochastic games with reachability properties.

Finally, for various extensions of stochastic games briefly discussed in Section 7 below, EAGLE [75] and PRALINE [17] analyse Nash equilibria for non-stochastic games. Uppaal Stratego [42] performs strategy synthesis for real-time systems against quantitative properties. The TuLiP toolbox [79] provides synthesis for linear (continuous) systems with GR(1) specification.

In comparison, most of the algorithmic solutions presented in this paper including single- and multi-objective, as well as compositional strategy synthesis problems for stochastic games and games with multiple players, have been implemented in the open-source tool called PRISM-games [36,55], which can be downloaded from [65]. PRISM-games can be used to model, verify, solve and simulate stochastic games with complex properties. It has been developed as an extension of the probabilistic model checker PRISM [56] and takes advantage of PRISM's modelling and specification language, as well as the existing user interface and simulator.

The original version, PRISM-games 1.0 [36], allows one to model multi-player stochastic games as introduced in Definition 9 using modules with synchronising actions. The recently released version, PRISM-games 2.0 [55], adds a compositional modelling approach to facilitate the compositional strategy synthesis discussed in Section 4.5.

The specification language is based on rPATL [35], a fragment of the branching-time logic in Definition 8 that also allows specification of properties for coalitions of players as discussed in Section 3.8. In particular, rPATL subsumes single-objective probabilistic reachability, a restricted class of probabilistic LTL properties, and total reward properties with  $\rho = F^* a$ , and their Boolean combinations. In the first version, PRISM-games support rPATL formulas, numerical queries and precise-value operators  $\mathbb{P} = p$ ,  $R^T = x$  [34]. The new version adds several single-objective properties, namely total reward properties with  $\rho = c$  for stopping games, average reward and ratio properties for a special class of games called controllable multichain games (for details, see [78]), and almost sure average reward and ratio properties. Besides single-objective properties, PRISM-games 2.0 allows multi-objective properties expressed as Boolean combinations of the same type of reward properties, except for the almost sure average and ratio reward properties for which only conjunctions are supported.

From the implementation point of view, PRISM-games builds on the Java-based engine of PRISM and handles games in an explicit-state fashion. In the multi-objective strategy synthesis, a feature introduced in the new version of the tool, the computation relies on the Parma Polyhedra Library [5] for symbolic manipulation of convex sets during  $\varepsilon$ -approximate computation of Pareto sets.

## 6. Case studies

Stochastic games have been used to model and analyse various control and networked systems. Here we list a set of examples that have been evaluated using PRISM-games, and offer their intuitive description. As mentioned in Section 5, tools such as GIST and GAVS+ provide partial support for stochastic games, but have only been used with small, illustrative examples. For more information, we refer the interested reader to the indicated publications and references therein. Experimental evaluation of some of the examples can also be found in [36,55]. A more exhaustive list of examples is maintained in the publications section of the PRISM-games website [65] and in the database of PRISM and PRISM-games case studies [66].

### *Microgrid demand-side management [73]*

The example models a decentralised energy management algorithm for smart grids. The system consists of a set of households that generate loads of various duration. Each household follows a simple algorithm to execute a load if the current energy cost is below a pre-agreed limit, otherwise it only executes the load with a pre-agreed probability. The energy cost to execute a load for a single time unit is the number of loads currently being executed in the grid. The algorithm is analysed with respect to the expected load per cost unit for a household, formulated as a single-objective total reward property.

### *Collective decision making for sensor networks [73]*

Sensor networks comprise of a set of low-power, autonomous devices that often must collaborate to achieve a goal. Here, a set of sensors is considered with the goal to agree on a target with the highest quality using a decentralised decision algorithm. In the algorithm, a sensor can probabilistically change its preferred target either based on its own exploration of available targets or based on communication with other sensors. The proposed decision procedure is analysed with respect to the speed of convergence, formulated as a total reward property, and robustness, *i.e.*, the ability to recover from a bad decision to a good one, formulated as a branching-time property with nested probabilistic operators.

### *Reputation protocol for user-centric networks [73]*

User-centric networks are designed to encourage users to cooperate in sharing resources and services in order to, for example, provide connectivity in a mobile ad hoc network. The case study presents a general model consisting of providers offering services to requesters. A requester chooses a provider and submits a request. The provider decides whether to accept the request based on a trust level towards the requester that is dependent on his/her reputation across all users in the network. If the request is accepted, the cost of the service is negotiated. After service delivery, the requester chooses whether to pay the cost or not, thus increasing or decreasing his/her trustworthiness, respectively. Using expected total reward properties, the maximum number of unpaid services that the requester can obtain is computed as well as the minimum price at which the requester can buy a particular number of services. Strategy synthesis is used to uncover possibly undesirable optimal behaviour of the requester in the latter case, and an adjustment to the protocol is suggested to improve it.

### Futures market investor [66]

An investor in a futures market decides when to invest in shares of a specific company. The decision can be made on the first day of any month collecting the payoff one month later. The market value of shares changes probabilistically over time within a bounded range and the distribution changes based on the current value. Moreover, the market can temporarily decide to bar the investor from making the investment. The corresponding stochastic game is analysed to compute the maximum expected payoff that the investor can guarantee for various initial share values and the optimal strategies are discussed.

### Human-in-the-loop UAV mission planning [46]

An unmanned aerial vehicle (UAV) is performing road network surveillance, reacting to inputs from a human operator. The UAV acts autonomously in fulfilling most of the piloting functions, such as selecting most of the waypoints that comprise the route, and flying the route. The operator primarily performs sensor tasks at waypoints but may also pick a road for the UAV at waypoints. The optimal UAV piloting strategy depends on mission objectives, e.g., safety, reachability, coverage, and operator characteristics, i.e., workload, proficiency, and fatigue. For the stochastic game modelling the situation, the minimum expected time of completing the temporal mission of covering a set of waypoints is computed. Moreover, a multi-objective property is considered to analyse the trade-off between the completion time and the number of visits to restricted operating zones.

### Autonomous urban driving [33]

An autonomous car is considered that drives through an urban environment and reacts to hazards such as pedestrians, obstacles, and traffic jams. Note that this case study serves as a motivation for our illustrative example presented in Example 1. Here, the car does not only decide on the reactions to hazards, but also chooses the roads to take in order to reach a target location. The presence of hazards, as well as the effects of reactions, may differ between roads. Through multi-objective strategy synthesis, strategies with optimal trade-off between the probability of reaching the target location, the probability of avoiding accidents and the overall quality of roads on the route are identified.

### Aircraft power distribution [6]

An aircraft electrical power network is considered, where power is to be routed from generators to buses through controllable switches. The generators can exhibit failures and switches have delays. The system consists of several components, each containing buses and generators, and the components can deliver power to each other. The network is modelled as a composition of stochastic games, one for each component. Compositional strategy synthesis is applied to find strategies with good trade-off between uptime of buses and failure rate. The property is modelled as a conjunction of ratio reward properties.

### Self-adaptive software architectures [51,38,39,19]

Software systems dealing with distributed applications in changing environments normally require human supervision to continue operation in all conditions. Self-adaptive software architecture is a response to these demands, where the system automatically adapts its structure and behaviour according to changes in real time. Both single- and multi-objective verification of multi-player stochastic games is applied to analyse three

self-adaptive software architectures, namely, the impact of communication topology for collections of fully cooperative systems defending against an external attack, the infrastructure for a news website, and an adaptive industrial middleware used to monitor and manage sensor networks in renewable energy production plants.

### DNS bandwidth amplification attack [43]

The Domain Name System (DNS) is an Internet-wide hierarchical naming system for assigning IP addresses to domain names, and any disruption of the service can lead to serious consequences. A notable threat to DNS, namely the bandwidth amplification attack, where an attacker attempts to flood a victim DNS server with malicious traffic, is modelled as a stochastic game. Verification and strategy synthesis is used to analyse and generate countermeasures to defend against the attack.

## 7. Conclusion

In this work, we have overviewed the existing body of knowledge and algorithmic solutions to the verification and strategy synthesis problems for stochastic games. We addressed a large class of properties, from probabilistic linear-time through various expected reward properties, to their branching-time and multi-objective combinations. As demonstrated through the case studies, the techniques can be used to analyse various control systems, for example, in network management, autonomous and human-in-the-loop planning, and security attack countermeasures. Evaluation of such systems can be achieved using the practical implementation of the algorithms in PRISM-games. Though several of the algorithms have high computational complexity, the range of case studies that have been tackled using stochastic games is encouraging, and we anticipate that by adapting implementation techniques that have been successful in probabilistic verification, for example symbolic methods and Monte carlo sampling will allow us to broaden the applicability even further.

While some of the open questions have already been identified in the previous sections, the following extensions of games pose further challenges.

*Concurrent games*, where players choose their moves concurrently rather than in turns, comprise the original games with probability introduced by Shapley [71], and they are a natural extension of stochastic games discussed here. For an overview of existing techniques, see, e.g., [47,61,22].

*Partial-observation games*, where the current state of the game is only partially observed (by one or both of the players), represent another widely studied model of games [21,70]. Recent results include [20,24]. Besides concurrent and stochastic games, they subsume models such as partially observable Markov decision processes (POMDPs) [63] and probabilistic automata [69].

There exist several extensions of games to infinite state spaces. For example, *uncertain* or *bounded-parameter MDPs* [62,57,80,67], *pushdown games* [45], and a large class of *timed games* [11,49,14,68].

Finally, one can consider *nonzero-sum games*, or *games with equilibria*, where the objectives of players are not necessarily dual. For an overview of results, see, e.g., [77,76] and references therein.

## Acknowledgements

We would like to thank Vojtěch Forejt, Clemens Wiltsche, Dave Parker and Nicolas Basset for valuable comments and discussions.

## References

- [1] R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, S. Tasiran, MOCHA: modularity in model checking, in: *Proceedings of Computer Aided Verification CAV*, 1998, pp. 521–525.
- [2] D. Andersson, P.B. Miltersen, The complexity of solving stochastic games on graphs, in: *Algorithms and Computation, Series, Lecture Notes in Computer Science*, vol. 5878, 2009, pp. 112–121.
- [3] R. Ash, C. Doléans-Dade, *Probability and Measure Theory*, Harcourt, Academic Press, Burlington, MA, USA, 2000.
- [4] Z. Aslanyan, F. Nielson, D. Parker, Quantitative Verification and Synthesis of Attack-Defence Scenarios, in: *Proceedings of Computer Security Foundations Symposium CSF*, 2016, In press.
- [5] R. Bagnara, P.M. Hill, E. Zaffanella, *The Parma Polyhedra Library: toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems*, *Sci. Comput. Program.* 72 (1–2) (2008) 3–21.
- [6] N. Basset, M.Z. Kwiatkowska, U. Topcu, C. Wilsche, Strategy synthesis for stochastic games with multiple long-run objectives, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, 2015, pp. 256–271.
- [7] N. Basset, M.Z. Kwiatkowska, C. Wilsche, Compositional controller synthesis for stochastic games, in: *Proceedings of Concurrency Theory CONCUR*, 2014, pp. 173–187.
- [8] A. Bianco, L. de Alfaro, Model checking of probabilistic and nondeterministic systems, in: *Proceedings of Foundations of Software Technology and Theoretical Computer Science FSTTCS*, Series, Lecture Notes in Computer Science, vol. 1026, 1995, pp. 499–513.
- [9] U. Boker, O. Kupferman, A. Steinitz, Parityizing Rabin and Streett, in: *Proceedings of Foundations of Software Technology and Theoretical Computer Science FSTTCS*, 2010, pp. 412–423.
- [10] E. Boros, K.M. Elbassioni, V. Gurvich, K. Makino, A pumping algorithm for ergodic stochastic mean payoff games with perfect information, in: *Proceedings of Integer Programming and Combinatorial Optimization IPCO*, 2010, pp. 341–354.
- [11] P. Bouyer, V. Forejt, Reachability in stochastic timed games, in: *Proceedings of International Colloquium on Automata, Languages and Programming ICALP*, Series, Lecture Notes in Computer Science, 2009, pp. 103–114.
- [12] H. Björklund, S. Sandberg, and S. Vorobyov, On Combinatorial Structure and Algorithms for Parity Games, Department of Information Technology, Uppsala University, Technical Report 2003–002, 2003.
- [13] F. Blahoudek, M. Křetínský, J. Strejček, Comparison of LTL to deterministic Rabin automata translators, in: *Proceedings of Logic for Programming, Artificial Intelligence, and Reasoning LPAR*, Series, Lecture Notes in Computer Science, vol. 8312, 2013, pp. 164–172.
- [14] T. Brázdil, V. Forejt, J. Krčál, J. Křetínský, A. Kučera, Continuous-time stochastic games with time-bounded reachability, *Inf. Comput.* 224 (2013) 46–70.
- [15] T. Brázdil, K. Chatterjee, V. Forejt, A. Kucera, MultiGain: a controller synthesis tool for mdps with multiple mean-payoff objectives, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, 2015, pp. 181–187.
- [16] T. Brázdil, V. Brozek, V. Forejt, A. Kucera, Stochastic games with branching-time winning objectives, in: *Proceedings of Logic in Computer Science LICS*, 2006, pp. 349–358.
- [17] R. Brenguier, PRALINE: a tool for computing Nash equilibria in concurrent games, in: *Proceedings of Computer Aided Verification CAV*, Series, Lecture Notes in Computer Science, vol. 8044, 2013, pp. 890–895.
- [18] G. Brown, M. Carlyle, J. Salmerón, K. Wood, Defending critical infrastructure, *Interfaces* 36 (6) (2006) 530–544.
- [19] J. Cámara, G.A. Moreno, D. Garlan, Stochastic game analysis and latency awareness for proactive self-adaptation, in: *Proceedings of Software Engineering for Adaptive and Self-Managing Systems SEAMS*, 2014, pp. 155–164.
- [20] K. Chatterjee, L. Doyen, Partial-observation stochastic games: how to Win when belief fails, *Trans. Comput. Logic* 15 (2) (2014) 16.
- [21] K. Chatterjee, L. Doyen, T.A. Henzinger, A survey of partial-observation stochastic parity games, *Formal Methods Syst. Des.* 43 (2) (2013) 268–284.
- [22] K. Chatterjee, T.A. Henzinger, A survey of stochastic omega-regular games, *J. Comput. Syst. Sci.* 78 (2) (2012) 394–413.
- [23] K. Chatterjee, M. Jurdzinski, T.A. Henzinger, Quantitative stochastic parity games, in: *Proceedings of Symposium on Discrete Algorithms SODA*, 2004, pp. 121–130.
- [24] K. Chatterjee, L. Doyen, S. Nain, M.Y. Vardi, The complexity of partial-observation stochastic parity games with finite-memory strategies, in: *Proceedings of Foundations of Software Science and Computation Structures FOSSACS*, 2014, pp. 242–257.
- [25] K. Chatterjee, T.A. Henzinger, Value iteration, in: *25 Years of Model Checking—History, Achievements, Perspectives*, 2008, pp. 107–138.
- [26] K. Chatterjee, L. de Alfaro, T.A. Henzinger, The complexity of stochastic Rabin and Streett games, in: *Proceedings of International Colloquium on Automata, Languages and Programming ICALP*, 2005, pp. 878–890.
- [27] K. Chatterjee, T.A. Henzinger, Strategy Improvement for Stochastic Rabin and Streett Games, in: *Proceedings of Concurrency Theory CONCUR*, 2006, pp. 375–389.
- [28] K. Chatterjee, T.A. Henzinger, Strategy improvement and randomized sub-exponential algorithms for stochastic parity games, in: *Proceedings of Symposium on Theoretical Aspects of Computer Science STACS*, 2006, pp. 512–523.
- [29] K. Chatterjee, V. Forejt, D. Wojtczak, Multi-objective discounted reward verification in graphs and MDPs, in: *Proceedings of Logic for Programming, Artificial Intelligence, and Reasoning LPAR*, Series, Lecture Notes in Computer Science, vol. 8312, 2013, pp. 228–242.
- [30] K. Chatterjee, T.A. Henzinger, B. Jobstmann, R. Singh, QUASY: quantitative synthesis tool, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, 2011, pp. 267–271.
- [31] K. Chatterjee, T. Henzinger, B. Jobstmann, A. Radhakrishna, Gist: a solver for probabilistic games, in: *Proceedings of Computer Aided Verification CAV*, Series, Lecture Notes in Computer Science, vol. 6174, 2010, pp. 665–669.
- [32] T. Chen, V. Forejt, M.Z. Kwiatkowska, A. Simaitis, C. Wilsche, On stochastic games with multiple objectives, in: *Proceedings of Mathematical Foundations of Computer Science MFCS*, 2013, pp. 266–277.
- [33] T. Chen, M.Z. Kwiatkowska, A. Simaitis, C. Wilsche, Synthesis for multi-objective stochastic games: an application to autonomous urban driving, in: *Proceedings of Quantitative Evaluation of Systems QUEST*, 2013, pp. 322–337.
- [34] T. Chen, V. Forejt, M. Kwiatkowska, A. Simaitis, A. Trivedi, M. Ummels, Playing stochastic games precisely, in: *Proceedings of Concurrency Theory CONCUR*, Series, Lecture Notes in Computer Science, vol. 7454, 2012, pp. 348–363.
- [35] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, A. Simaitis, Automatic verification of competitive stochastic systems, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, Series, Lecture Notes in Computer Science, vol. 7214, 2012, pp. 315–330.
- [36] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, A. Simaitis, PRISM-games: a model checker for stochastic multi-player games, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, Series, Lecture Notes in Computer Science, vol. 7795, 2013, pp. 185–191.
- [37] C. Cheng, A. Knoll, M. Luttenberger, C. Buckl, GAVS+: an open platform for the research of algorithmic game solving, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, 2011, pp. 258–261.
- [38] J. Cámara, D. Garlan, B. Schmerl, A. Pandey, Optimal planning for architecture-based self-adaptation via model checking of stochastic games, in: *Proceedings of Symposium on Applied Computing SAC*, 2015, pp. 428–435.
- [39] J. Cámara, G.A. Moreno, D. Garlan, Reasoning about human participation in self-adaptive systems, in: *Proceedings of Software Engineering for Adaptive and Self-Managing Systems SEAMS*, 2015, pp. 146–156.
- [40] A. Condon, The complexity of stochastic games, *Inf. Comput.* 96 (1992) 203–224.
- [41] A. Condon, On algorithms for simple stochastic games, *Adv. Comput. Complex. Theory* 13 (1993) 51–73.
- [42] A. David, P. Jensen, K. Larsen, M. Mikucionis, J. Taankvist, Uppaal Stratego, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, Series, Lecture Notes in Computer Science, vol. 9035, 2015, pp. 206–211.
- [43] T. Deshpande, P. Katsaros, S. Smolka, S. Stoller, Stochastic game-based analysis of the dns bandwidth amplification attack using probabilistic model checking, in: *Proceedings of European Dependable Computing Conference EDCC*, 2014, pp. 226–237.
- [44] C. Essen, D. Giannakopoulou, Analyzing the Next Generation Airborne Collision Avoidance System, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, 2014, pp. 620–635.
- [45] K. Etesami, M. Yannakakis, Recursive Concurrent Stochastic Games, *CoRR*, vol. abs/0810.3581, 2008.
- [46] L. Feng, C. Wilsche, L. Humphrey, U. Topcu, Controller synthesis for autonomous systems interacting with human operators, in: *Proceedings of International Conference on Cyber-Physical Systems ICCPS*, 2015, pp. 70–79.
- [47] J. Filar, K. Vrieze, *Competitive Markov Decision Processes*, Springer-Verlag New York Inc., New York, NY, USA, 1996.
- [48] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, Automated verification techniques for probabilistic systems, in: *Proceedings of Formal Methods for Eternal Networked Software System SFM*, Series, Lecture Notes in Computer Science, vol. 6659, 2011, pp. 53–113.
- [49] V. Forejt, M. Kwiatkowska, G. Norman, A. Trivedi, Expected reachability-time games, in: *Proceedings of Formal Modelling and Analysis of Timed Systems FORMATS*, Series, Lecture Notes in Computer Science, 2010, pp. 122–136.
- [50] D. Gillette, Stochastic games with zero stop probabilities, in: *Contributions to the Theory of Games*, vol. 39, 1957, pp. 179–187.
- [51] T. Glazier, J. Cámara, B. Schmerl, D. Garlan, Analyzing resilience properties of different topologies of collective adaptive systems, in: *Proceedings of Self-Adaptive and Self-Organizing Systems Workshops SASOW*, 2015, pp. 55–60.
- [52] [Online]. Available: (<http://www.prismmodelchecker.org/files/ecc16/>).
- [53] M. Kwiatkowska, G. Norman, D. Parker, H. Qu, Compositional probabilistic verification through multi-objective model checking, *Inf. Comput.* 232 (2013) 38–65.
- [54] M. Kwiatkowska, D. Parker, Automated verification and strategy synthesis for probabilistic systems, in: *Proceedings of Automated Technology for Verification and Analysis ATVA*, Series, Lecture Notes in Computer Science, vol. 8172, 2013, pp. 5–22.
- [55] M. Kwiatkowska, D. Parker, C. Wilsche, PRISM-games 2.0: A Tool for Multi-Objective Strategy Synthesis for Stochastic Games, in: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, 2016, In press.
- [56] M. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of Probabilistic Real-time Systems, in: *Proceedings of Computer Aided Verification CAV*, Series, Lecture Notes in Computer Science, vol. 6806, 2011, pp. 585–591.
- [57] M. Lahijanian, S.B. Andersson, C. Belta, Formal verification and synthesis for discrete-time stochastic systems, *Trans. Autom. Control* 60 (8) (2015) 2031–2045.

- [58] T.M. Liggett, S.A. Lippman, Stochastic games with perfect information and time average payoff, *SIAM Rev.* 11 (4) (1969) 604–607.
- [59] W. Ludwig, A subexponential randomized algorithm for the simple stochastic game problem, *Inf. Comput.* 117 (1) (1995) 151–155.
- [60] D.A. Martin, The determinacy of blackwell games, *J. Symbol. Logic* 63 (4) (1998) 1565–1581.
- [61] A. Neyman, S. Sorin, NATO SA Division, Stochastic Games and Applications, Series, NATO Science Series: Mathematical and Physical Sciences. Springer, Netherlands, 2003.
- [62] A. Nilim, L. El Ghaoui, Robust control of Markov decision processes with uncertain transition matrices, *Oper. Res.* 53 (5) (2005) 780–798.
- [63] C. Papadimitriou, J.N. Tsitsiklis, The complexity of Markov decision processes, *Math. Oper. Res.* 12 (3) (1987) 441–450.
- [64] A. Pnueli, The temporal logic of programs, in: Proceedings of Foundations of Computer Science, 1977, pp. 46–57.
- [65] PRISM-games Website. [Online]. Available: <http://www.prismmodelchecker.org/games/>.
- [66] PRISM and PRISM-Games Case Studies. [Online]. Available: <http://www.prismmodelchecker.org/casestudies/>.
- [67] A. Puggelli, W. Li, A.L. Sangiovanni-Vincentelli, S.A. Seshia, Polynomial-time verification of PCTL properties of MDPs with convex uncertainties, in: Proceedings of Computer Aided Verification CAV, 2013, pp. 527–542.
- [68] M.N. Rabe, S. Schewe, Optimal time-abstract schedulers for CTMDPs and continuous-time Markov games, *Theoret. Comput. Sci.* 467 (2013) 53–67.
- [69] M.O. Rabin, Probabilistic automata, *Inf. Control* 6 (3) (1963) 230–245.
- [70] D. Rosenberg, E. Solan, N. Vieille, Stochastic games with imperfect monitoring, in: Advances in Dynamic Games, Series, Annals of the International Society of Dynamic Games, vol. 8, 2006, pp. 3–22.
- [71] L.S. Shapley, Stochastic games, in: National Academy of Sciences, 1953, pp. 1095–1100.
- [72] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, G. Meyer, PROTECT: a deployed game theoretic system to protect the ports of the United States, in: Proceedings of Conference on Autonomous Agents and Multiagent Systems AAMAS, 2012, pp. 13–20.
- [73] A. Simaitis, Automatic verification of competitive stochastic systems (Ph.D. dissertation), Department of Computer Science, University of Oxford, 2014.
- [74] K. Thompson, Retrograde analysis of certain endgames, *Int. Comput. Chess Assoc.* 9 (3) (1986) 131–139.
- [75] A. Toumi, J. Gutierrez, M. Wooldridge, A tool for the automated verification of nash equilibria in concurrent games, in: Proceedings of International Conference on Theoretical Aspects of Computing ICTAC, Series, Lecture Notes in Computer Science, vol. 9399, 2015, pp. 583–594.
- [76] M. Ummels, Stochastic multiplayer games: theory and algorithms (Ph.D. dissertation), RWTH Aachen University, 2010.
- [77] M. Ummels, D. Wojtczak, The complexity of nash equilibria in stochastic multiplayer games, *Log. Methods Comput. Sci.* 7 (3) (2011).
- [78] C. Wiltsche, Assume-guarantee strategy synthesis for stochastic games (Ph.D. dissertation), Department of Computer Science, University of Oxford, 2015.
- [79] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, R.M. Murray, TuLiP: a software toolbox for receding horizon temporal logic planning, in: Proceedings of Conference on Hybrid Systems: Computation and Control HSCC, 2011, pp. 313–314.
- [80] E.M. Wolff, U. Topcu, R.M. Murray, Robust control of uncertain Markov decision processes with temporal logic specifications, in: Proceedings of Conference on Decision and Control CDC, 2012, pp. 3372–3379.
- [81] T. Wongpiromsarn, E. Frazzoli, Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications, in: Proceedings of Conference on Decision and Control CDC, 2012, pp. 7644–7651.
- [82] Z. Yin, A.X. Jiang, M. Tambe, C. Kiekintveld, K. Leyton-Brown, T. Sandholm, J. P. Sullivan, TRUSTS: scheduling randomized patrols for fare inspection in transit systems using game theory, *AI Mag.* 33 (4) (2012) 59–72.