

Reconciling Rationality and Stochasticity: Rich Behavioral Models in Two-Player Games

Mickael Randour^{1,*}

Computer Science Department, Université Libre de Bruxelles (ULB), Belgium

Abstract. Two traditional paradigms are often used to describe the behavior of agents in multi-agent complex systems. In the first one, agents are considered to be fully rational and systems are seen as multi-player games. In the second one, agents are considered to be fully stochastic processes and the system itself is seen as a large stochastic process. From the standpoint of a particular agent — having to choose a strategy, the choice of the paradigm is crucial: the most adequate strategy depends on the assumptions made on the other agents.

In this paper, we focus on two-player games and their application to the automated synthesis of reliable controllers for reactive systems — a field at the crossroads between computer science and mathematics. In this setting, the reactive system to control is a player, and its environment is its opponent, usually assumed to be fully antagonistic or fully stochastic. We illustrate several recent developments aiming to breach this narrow taxonomy by providing formal concepts and mathematical frameworks to reason about richer behavioral models.

The interest of such models is not limited to reactive system synthesis but extends to other application fields of game theory. The goal of our contribution is to give a high-level presentation of key concepts and applications, aimed at a broad audience. To achieve this goal, we illustrate those rich behavioral models on a classical challenge of the everyday life: planning a journey in an uncertain environment.

1 Introduction

Rationality. Ever since the seminal book of von Neumann and Morgenstern [37], game theory has grown to be a prominent mathematical framework providing general and powerful concepts and methods to reason about interactions between cooperating and/or competing agents in complex systems. Applications of game theory are common in diverse fields such as computer science, mathematics, economics, biology, etc.

One of the core underlying concepts of game theory is the *rationality hypothesis* [24]: roughly speaking, agents are modeled as players who have clear personal objectives, are aware of their alternatives, form sound expectations about any unknowns, and choose their actions coherently (regarding some notion of optimality with respect to the situation and the objective). In some sense, players are often seen as selfish beings who place their personal benefit above the common good. This indeed led to a whole area of research where mechanisms are sought to enforce that personal benefits coincide with the common good, thus giving a natural incentive to players to behave in a not-so-selfish way. In the particular setting of zero-sum games, this hypothesis generally results in antagonistic interactions between the players.

Rationality in computer science. While the very notion of agent rationality is often debated in application fields such as social sciences and economics, there are areas where it is without any doubt a natural hypothesis. One of such areas is our core research field of computer science. Indeed, computer processes for example are fully rational agents: they behave according to a preset set of rules in order to achieve a well-formalized objective, and interact in a usually predictable fashion. Similarly, it is quite common to model their interaction through zero-sum games: e.g., processes competing for access to a shared resource. A very promising application of game theory linked to computer science is the emerging field of *reactive system synthesis*. In a nutshell, computer scientists are envisioning ways to automatically build provably-correct and efficient controllers for so-called reactive systems (i.e., computer systems continuously interacting with an uncontrollable environment). This building process, called *synthesis*, is based on strong mathematical

* The author is an F.R.S.-FNRS Postdoctoral Researcher

grounds and aims to guarantee the correctness of the synthesized controller: this is notably of tremendous importance for safety-critical systems (power plants, ABS for cars, airplane and railway traffic). At a very high level, the game-theoretic formulation of the synthesis problem would be to consider a two-player zero-sum game between the reactive system to control and the uncontrollable environment. The environment is thus considered to be antagonistic and the goal is to find a strategy for the reactive system that guarantees a correct behavior whatever the actions of the environment. If such a strategy exists, it provides a formal model of the controller to implement. We describe this area in more details and with an illustrative example in Sect. 2.

Stochasticity. While game theory is usually focused on rational players, other models have been proposed to model large systems composed of autonomous agents. For example, considering agents as *stochastic processes* is often a sufficient abstraction to reason about macroscopic properties of a complex system with adequate accuracy. When every agent is assumed to be stochastic, the resulting system can itself be described by a stochastic process, for example as a Markov chain [26]. A particularly interesting setting for this paper is the one where a rational agent faces a stochastic one: this can be described as a Markov decision process, combining both controllable and stochastic behaviors [26]. The gap between two-player games (i.e., two rational agents) and Markov decision processes (i.e., one rational agent and one stochastic agent) is filled in stochastic games which combine two rational agents and a stochastic one. Those are also called competitive Markov decision processes in a seminal book by Filar and Vrieze [18].

The paradigm defines the objective hence the adequate strategy. Let us take a very simple example to illustrate this. Assume you have to choose a way to get from your home to your workplace: you could take your car, catch a bus, go to the railway station, ride your bike, etc. Maybe you could also take different routes. In this example, you are the rational agent. Now, there is another agent: the environment. Essentially, this agent represents everything that can impact your journey: bad weather, bad traffic, train delay, etc. What assumptions should we make on this agent? If we are considering the average time-to-work, for example when deciding whether to buy a car or a yearly train pass, the natural option would be to see the environment as fully stochastic (following a stochastic model that we obtain from real data for example) and to reason about the *expected time-to-work* as this is what we will observe in the long run. But if we are considering which transportation to take on a particular day, for example the morning before a very important meeting, then the situation is different: in this case, it seems reasonable to choose the option that will *guarantee* that we will not be late, even if this option takes a little more time on average (e.g., traffic can be stopped for a long time because of an accident even if the car is the best option on average). Such a situation is best modeled through a two-player zero-sum game against the environment.

New paradigms. Those two paradigms are the prominent ones in the literature, especially for applications in computer science. But they are not sufficient to reason about more complex situations, where richer behavioral models have to be considered. For example, *one may want to combine a good expected time-to-work with acceptable guarantees on the worst-case reaching-time*. To do so, we need appropriate mathematical frameworks, algorithms and tools to decide the existence of adequate strategies and to exploit them.

Establishing such meaningful frameworks, both expressive enough to tackle the complexity of real-life applications and abstract enough to allow efficient analysis, is a challenge which is gaining attention in the research community. This paper tries to illustrate, at a very high level and for a large audience, some advances in this area that we have obtained through different collaborations. It is based on a series of papers on the subject [28,10,9,30,31,7], most in computer science conferences and journals.

Outline. As stated before, the aim of this contribution is to give a broad picture of several new solution concepts for strategies, mixing assumptions of rationality and stochasticity. The paper only contains the minimal amount of technical details necessary to ensure sound definitions of the different models and we point in each section to the corresponding full papers containing detailed techniques. In Sect. 2, we give a comprehensive discussion of the controller synthesis problem and how it can be answered using game theory. To illustrate the approach, a toy example of a lawnmower controller is used as a case study. In Sect. 3, we present the core of the paper: novel solution concepts for rich behavioral models, illustrated through an everyday life application — the planning of a journey in an uncertain environment. We end the paper with a short conclusion in Sect. 4.

2 Context: controller synthesis for reactive systems

As mentioned in Sect. 1, our core research field is the automated synthesis of reliable and efficient controllers for reactive systems through game theory. This section is based on [28]: its goal is to illustrate the cornerstones of such an approach. It is organized as follows: Sect. 2.1 presents the general context in more details; Sect. 2.2 describes a motivating toy example — a robotized lawnmower — in an informal way; Sect. 2.3 shows how its interaction with its environment can be formalized through game theory; and Sect. 2.4 discusses the synthesis process and its outcome on the example.

2.1 Automated controller synthesis through game theory

Nowadays, more and more aspects of our society depend on *critical reactive computer systems*, i.e., systems that continuously interact with their uncontrollable environment. Think about control programs of power plants, ABS for cars or airplane and railway traffic managing. Therefore, we are in dire need of systems capable of sustaining a safe behavior despite the nefarious effects of their environment.

Good developers know that testing do not capture the whole picture: never will it *proves* that no bug or flaw is present in the considered system. So for critical systems, it is useful to apply **formal verification**. That means using *mathematical tools* to prove that the system follows a given specification which models desired behaviors. Among those tools are *model checking* techniques, introduced in the 80s [16,35,1] based on logic, automata and games (seminal works by Büchi, Rabin, etc). While verification applies *a posteriori*, checking that the formal model of a system satisfies the needed specification, it is most of the time desirable to start *from* the specification and automatically build a system from it, in such a way that desired properties are proved to be maintained in the process. This *a priori* process is the more ambitious and harder **synthesis** problem [15,27,25], on which an important part of the research community is currently focusing its effort.

The mathematical framework we use is **game theory** [37,24]. Roughly speaking, we consider a reactive system controller as a player (player 1), and its uncontrollable environment as its adversary (player 2). We model their interactions in a game on a graph [20], where vertices model *states* of the system and its environment, and edges model their possible *actions*. Players alternatively decide how to move a pebble on this graph. They choose how to move the pebble according to *strategies*, creating an infinite sequence of states called *play* that represents the *behavior of the system*. The goal of the controller is to enforce a given specification, encoded through a *winning objective* (a play is winning if it belongs to a predefined set of acceptable plays). The goal of the environment, considered antagonistic, is to prevent the controller from enforcing its specification. In our context, establishing *winning strategies* (i.e., strategies that guarantee victory whatever the strategy of the environment) corresponds to synthesizing implementable models of provably correct controllers.

The synthesis process is depicted in Fig. 1. In this section, we do not address the full theoretical deepness of such an approach but rather try to motivate and illustrate its usefulness toward an audience who may not be familiar with it, by discussing a motivating toy example.

A wide variety of games (and thus system models) have been studied recently, with diverse enforceable behaviors (e.g., [20,11,4,17,38,36,13,12]). In our illustration, we focus on systems that must satisfy *qualitative behaviors* (e.g., always eventually granting requests, never reaching a deadlock) along with multiple *quantitative requirements* (e.g., maintaining a bound on the mean response time, never running out of energy). Such settings can be handled using techniques developed by Chatterjee et al. in [13].

2.2 A toy example: the lawnmower

Consider the following running example. We want to synthesize a controller for a robotized lawnmower. This lawnmower is automatically operated, without any human intervention. We present its informal specification, as well as the effects the environment can have on its operation.

- In this partial, simplified specification, the gardener does not ask for the lawnmower to satisfy any bound on the frequency of grass-cuttings. However, as he wishes that the grass does not grow boundlessly, the

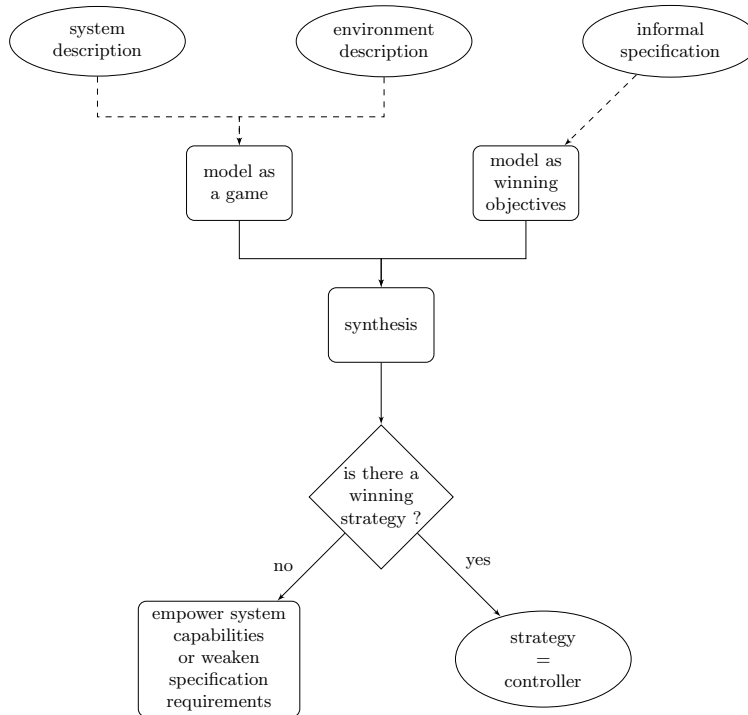


Fig. 1. Controller synthesis through game theory: process.

lawnmower should cut the grass infinitely often in the future (as if it stops someday, the grass will not stop growing from then on).

- The lawnmower has an electric battery that can be recharged under sunshine thanks to solar panels, and a fuel tank that can only be filled when the lawnmower is back on its base. Both are considered unbounded to keep things simple.
- The weather can be cloudy or sunny.
- The lawnmower can refuel (2 fuel units) at its base under both weather conditions, but can only recharge its battery (2 battery units) when it is sunny. Resting at the base takes 20 time units.
- When cloudy, it can operate either under battery (1 battery unit) or using fuel (2 fuel units), both according to the same speed (5 time units). When sunny, the lawnmower may either cut the grass slowly, which always succeeds and consumes no energy (as the sun recharges the battery along the way), but takes 10 time units. Or it may cut the grass fast, which consumes both 1 unit of fuel and 1 unit of battery, but only takes 2 time units.
- When operating fast, the lawnmower makes considerably much noise, which may wake up the cat that resides in the garden and prompt it to attack the lawnmower. In that case, the grass-cutting is interrupted and the lawnmower goes back to its base, losing 40 time units as repair is needed. The cat does not go out if the weather is bad.
- As the gardener cannot benefit from his garden while the lawnmower is operating, he wishes that the mean time required by actions of the lawnmower is less than 10 time units.

While simple, this toy example already involves qualitative requirements (the grass should be mown infinitely often), along with quantitative ones. There are indeed three quantities that have to be taken into account: battery and fuel are energy quantities, which should never be exhausted, and time is a quantity which mean over an infinite operating of the lawnmower should be less than a given bound.

Given this informal description of the capabilities of the system and its environment, as well as the specification the system should enforce, we need to build a system controller that guarantees satisfaction of the specification.

2.3 Modeling the lawnmower example as a two-player game

Game. We model the states and the interactions of the couple system/environment as a graph game where the system (here, the lawnmower) is player 1 and the environment is its adversary player 2. Formally, a *game* is a tuple $G = (S_1, S_2, s_{\text{init}}, E, w)$ where (i) S_1 and S_2 resp. denote the finite sets of *states* belonging to player 1 and player 2, with $S_1 \cap S_2 = \emptyset$; (ii) $s_{\text{init}} \in S = S_1 \cup S_2$ is the initial state; (iii) $E \subseteq S \times S$ is the set of *edges* such that for all $s \in S$, there exists $s' \in S$ such that $(s, s') \in E$; and (iv) $w: E \rightarrow \mathbb{Z}^d$ is the d -dimensional weight labeling function.

The game starts at the initial state s_{init} , and if the current state is a player 1 (resp. player 2) state, then player 1 (resp. player 2) chooses an outgoing *edge*. This choice is made according to a *strategy* of the player: given the sequence of visited states, a strategy chooses an outgoing edge. For this illustration, we only consider strategies that operate this choice deterministically. This process of choosing edges is repeated forever, and gives rise to an outcome of the game, called a *play*, that consists in the infinite sequence of states that are visited. Formally, a *play* in game G is an infinite sequence of states $\pi = s_0 s_1 s_2 \dots$ such that $s_0 = s_{\text{init}}$ and for all $i \geq 0$, we have $(s_i, s_{i+1}) \in E$.

Applying this formalism, we represent the lawnmower problem as the game depicted on Fig. 2. Edges correspond to choices of the system or its environment and taking an edge implies a change on the three considered quantities, as denoted by the edge label. The *grass-cutting* state is special as the specification requires that it should be visited infinitely often by a suitable controller.

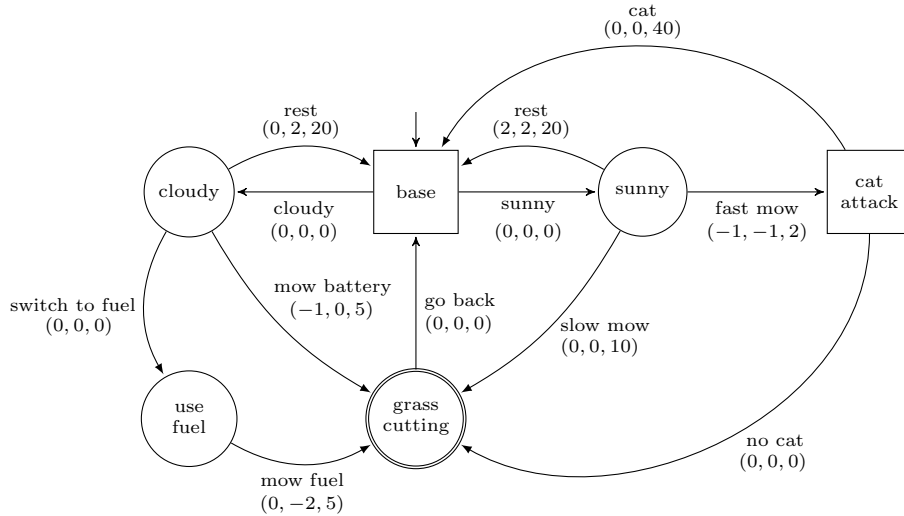


Fig. 2. Lawnmower game. Edges are fitted with tuples denoting changes in battery, fuel and time respectively.

Strategies. Formally, a *strategy* for player i , $i \in \{1, 2\}$, in G is a function $\sigma_i: S^* S_i \rightarrow S$ defined over all valid prefixes of plays (w.r.t. the underlying game graph) and such that for all prefix $\eta = s_0 s_1 \dots s_n$ with $s_n \in S_i$, we have $(s_n, \sigma_i(\eta)) \in E$. The history of a play (i.e., the previously visited states and their order of appearance) may thus in general be used by a strategy to prescribe its choice. In full generality, strategies may require infinite memory (as they need to recall histories of unbounded length). A strategy σ_i for player i has *finite memory* if the history it needs to remember can be bounded. In that case, the strategy can be encoded

by a special kind of finite automaton, called deterministic Moore machine. As discussed earlier, a strategy of player 1 (the lawnmower) provides a complete description of a controller for the system, prescribing the actions to take in response to any situation. Therefore, our task is to build a strategy that satisfies the specification.

Objectives. To devise such a strategy, it is needed to formalize the specification as objectives for player 1 in the game. The conjunction of objectives yields a set of winning plays that endorse the specification. A strategy of player 1 is thus said to be *winning* if, *against every possible strategy of the adversary*, the play induced by following this strategy belongs to the winning set of plays.

The informal specification developed in Sect. 2.2 is encoded as the following objectives. We omit technical details for the sake of this illustration.

- *Battery and fuel.* Both constitute energy types which quantities are never allowed to drop below zero. A play is thus winning for the *energy objective* if the running sum of the weights encountered along it (i.e., changes induced by the taken edges) never drops below zero on any of the first two dimensions.
- *Mean action time.* The specification asks that the lawnmower spends no more than 10 time units per action on average in the long run. That is, it is allowed to take more than 10 time units on some actions, but the long-run mean should be below this threshold. Therefore, the *mean-payoff objective* requires that the *limit* of the mean of the third-dimension weights over prefixes of a play is lower than 10.
- *Infinitely frequent grass-cutting.* To satisfy this part of the specification, a strategy of player 1 must ensure that the grass-cutting state is visited infinitely often along the induced play. This is encoded as a *Büchi objective*.

2.4 Synthesis of a correct and efficient controller

Process. Since in this example, our desire is to build a practical real-world controller, we are only interested in strategies that require *finite* memory. From a theoretical standpoint, there exist classes of games where infinite memory may help to achieve better results (see for example [36,29]), but infinite-memory strategies are ill-suited for practical use, as implementing a controller with infinite-memory capabilities is ruled out.

The core of the synthesis process depicted on Fig. 1 is thus to construct, if possible, a finite-memory strategy that ensures satisfaction of the previously defined objectives, as well as a corresponding initial value of the energy parameters, commonly referred to as *initial credit*. That is because for the energy objectives, it is allowed to start the game with some finite quantity in stock, before taking any action. Think about starting a race with some fuel in your tank.

While of importance for the analysis of systems with both qualitative and quantitative requirements, the synthesis problem for the class of games that is used to model the lawnmower problem, i.e., games with parity and multi energy or mean-payoff objectives, has only been considered recently [13]. Applying the corresponding techniques, we are able to synthesize a winning strategy for player 1, hence a suitable controller for the lawnmower.

Lawnmower controller. To conclude our illustration, we exhibit a synthesized controller that enforces the desired specification. Notice that there may exist other acceptable controllers. The one we present here is quite simple but already asks for some memory (in the form of bookkeeping of battery and fuel levels). The controller implements the following strategy.

- Start the game with empty battery and fuel levels.
- If the weather is sunny, mow slowly.
- If the weather is cloudy,
 - if there is at least one unit of battery, mow on battery,
 - otherwise, if there is at least two units of fuel, mow on fuel,
 - otherwise, rest at the base.

Notice that this strategy guarantees never running out of energy (which satisfies the energy objectives), induces infinitely frequent grass-cuttings (which satisfies the Büchi objective), and produces a play on which the mean time per action is less than 10 against any strategy of the adversary (which satisfies the mean-payoff objective). In this sample controller, the lawnmower never uses the “fast mow” action as the adversary could very well play “cat” and prevent visit of the grass-cutting state.

2.5 Typical questions and challenges

The lawnmower illustration is only a toy example, too simple to be of real practical use. Nonetheless, the underlying synthesis techniques aim at dealing with real-world problems, and some of them already proved their interest in industrial contexts. The typical challenges faced in the field are the following.

First, in order to accurately represent the complexity of practical applications, our game models must be *sufficiently expressive*. The needed expressiveness may be in terms of complex winning objectives (e.g., [12,6]), of the capacity to analyze trade-offs between different objectives (e.g., [13,30]), of rich interaction between the players [7], and so on. In Sect. 3, we focus on recent advances toward richer behavioral models [9,10,31].

Second, modeling real applications usually results in huge games, while allowing our techniques to describe complex behavior results in increased computational complexity. For example, it was shown in [13] that synthesizing correct controllers for specifications similar to the lawnmower one requires exponential time in the size of the game. Moreover, optimal controllers for such specifications also require exponential memory. In practice, we favor simpler controllers whenever possible: they are easier to conceive, and cheaper to produce and maintain. Therefore, to preserve the practical interest of synthesis techniques, it is crucial to consider the trade-off between expressiveness and *tractability* of the resulting techniques.

Similarly, it is important that the developed theoretical frameworks and algorithms are then implemented efficiently in *software tools* supporting the synthesis process. This point yields its own challenges, and a lot of effort is put in devising efficient representations and techniques to implement game-theoretic concepts. For example, see [5] for implementation of techniques linked to the lawnmower case.

3 Rich behavioral models

As argued in Sect. 1, for a rational agent having to adopt a strategy in a multi-agent complex system, the best choice greatly depends on the assumptions made on the other agents. This is especially true in two-player games such as the ones used for synthesis, as presented in Sect. 2. Here, we develop the practical case sketched in the introduction: *the dilemma of an employee having to choose a way to get to work*. We show that, depending on the situation, the natural objective of the employee and the assumptions he should make on his environment vary. Then, we illustrate novel concepts that go beyond the two traditional paradigms of fully antagonistic and fully stochastic environments.

This section is based on a more technical survey co-authored with Raskin and Sankur [31]. The goal of this section is to illustrate the different concepts intuitively, abstracting most technical details to focus on the philosophy of each solution concept. For each one, we point the interested reader to corresponding publications where the models are formally defined and adequate synthesis techniques are developed. This section is structured as follows:

- Sect. 3.1 presents the practical case and defines the necessary notions of the model;
- Sect. 3.2 presents a classical solution consisting in minimizing the expected time-to-work, assuming the environment to be fully stochastic;
- Sect. 3.3 addresses the case of risk-averse strategies;
- Sect. 3.4 shows how to guarantee an acceptable worst-case reaching-time by considering the environment as fully antagonistic;
- Sect. 3.5 presents a novel framework allowing the employee to adopt a strategy ensuring both worst-case guarantees and good expected time-to-work.
- Finally, Sect. 3.6 sketches new concepts that permit reasoning on trade-offs between different aspects: e.g., the employee wants to minimize the risk of being late but also to travel at an acceptable cost.

3.1 Planning a journey in an uncertain environment

An everyday life challenge. Let us consider the dilemma of an employee choosing a mean of transport and an itinerary to get to work. This kind of choice is one that we all face at times, some sort of *shortest path problem*: we want to find the quickest way to go from one place to another. The usual way to solve this in the real life is to resort to a GPS navigation device or our favorite web mapping service. Behind the scenes, such a problem is often modeled as finding a path of minimal length in a weighted graph (which may be directed or not). See for example Fig. 3: vertices represent different locations and edges $L \rightarrow L'$ are labeled with a weight that models the time to go from L to L' . The problem is then to find a path from the origin (A) to the destination (D) that minimizes the sum of weights (this path is depicted in thick red in the figure). This is a well-known algorithmic problem with elegant solutions (e.g., Dijkstra and Bellman-Ford algorithms [14]). But, is it really always that easy? It would be if we lived in a *fully deterministic* world, where going from A to B *always* take the same precise amount of time. But we all know that uncontrollable events can impact a journey: bad traffic can slow our car, trains can be delayed, etc. So, what can we do?

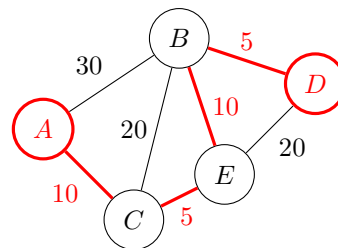


Fig. 3. In a fully deterministic world, traveling would be as easy as finding a shortest path in a graph.

Modeling an uncertain environment. One traditional answer to this observation is to go from a model where the employee is the only agent with choices (a graph) to a model where a second agent represents the environment and interacts with the employee. As discussed earlier, two classical assumptions can be made on the environment: either we see it as a rational agent (yielding a two-player game) or as a stochastic agent (yielding a Markov decision process). In our case, the environment represents traffic, weather, etc. While it may sometimes *seem* like other people are taking the road only to slow us down, it is reasonable to assume that the environment is not a rational agent per se. On the contrary, a stochastic agent is a good model of such phenomena: one can obtain data estimating the probability that a specific part of town is crowded at specific times of the day, the probability that a bus on a given line is delayed, and so on. Hence for the moment, we consider the environment to be *fully stochastic* and the problem to solve becomes a *stochastic shortest path problem* [26]. Still, the very notion of *rationality should not be ruled out* for the environment: we will see that even if the environment is inherently stochastic, the employee sometimes has to see it as a rational, antagonistic agent in order to choose adequate strategies, achieving complex winning objectives (see Sect. 3.4 and Sect. 3.5).

Toy example. A simple example of a *Markov decision process* modeling the rational employee faced to a stochastic environment is depicted in Fig. 4. Vertices model situations in which the employee usually has several choices, represented by outgoing edges. For example, at home, he has the possibility to go to the train station, to take his car or to take his bicycle. His goal is to reach work. Each action has a name, an integer label representing the time taken by the action, and may result in different outcomes according to a discrete probability distribution, modeling the stochastic environment. For example, if the employee decides to take his car (1 minute), then the journey depends on traffic conditions and may result—with respective probability 0.2, 0.7 and 0.1—in a 20-minute, 30-minute or 70-minute drive. When he decides to bike, he reaches his office in 45 minutes: this action is deterministic, not impacted by the environment. Finally, the employee can also try to catch a train, which takes 35 minutes to reach work. But trains can be delayed (potentially multiple times): in that case, the employee decides if he waits or if he goes back home (and then takes his car or his bike).

We give the formal definition of Markov decision processes and related notions in the following. Informally, Markov decision processes can be seen as two-player games against a stochastic adversary (i.e., the opponent adopts a fixed randomized strategy known to the rational agent). Indeed, they are sometimes called $1\frac{1}{2}$ -player games (the half-player being the stochastic one).

Markov decision processes. A *Markov decision process* (MDP) is a tuple $D = (S, s_{\text{init}}, A, \delta, w)$ where (i) S is a finite set of *states*; (ii) $s_{\text{init}} \in S$ is the initial state; (iii) A is a finite set of *actions*; (iv) $\delta: S \times A \rightarrow \mathcal{D}(S)$

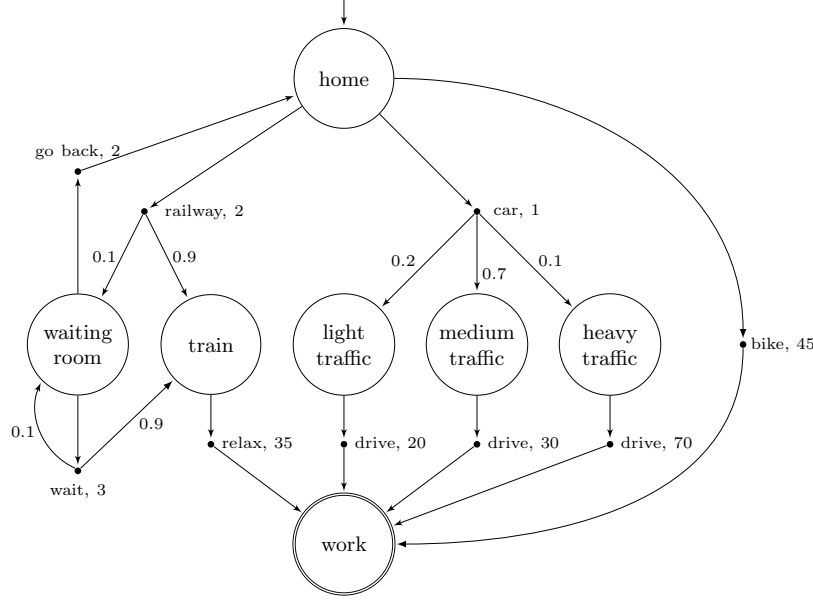


Fig. 4. An everyday life application of stochastic shortest path problems: choosing a mean of transport to go from home to work. Actions (black dots) are labeled with durations in minutes, and stochastic transitions are labeled with their probability.

is a partial function called the *probabilistic transition function*, where $\mathcal{D}(S)$ denotes the set of rational probability distributions over S ; and $(v) w: A \rightarrow \mathbb{Z}^d$ is a d -dimensional weight function. The set of actions that are available in a state $s \in S$ is denoted by $A(s)$. For any dimension $i \in \{1, \dots, d\}$, we denote by $w_i: A \rightarrow \mathbb{Z}$ the projection of w to the i -th dimension, i is omitted when there is only one dimension.

We define a *run* ρ of D as an infinite sequence $\rho = s_1 a_1 \dots a_{n-1} s_n \dots$ of states and actions such that $\delta(s_i, a_i)(s_{i+1}) > 0$ for all $i \geq 1$. Essentially, runs in MDPs are analogous to plays in games.

Strategies. As for games, strategies are defined over all valid prefixes of runs (w.r.t. the underlying graph). Two differences here: first there is only one player in an MDP, the rational agent; second, we allow strategies to use *randomness* in addition to memory. Therefore, a *strategy* σ is a function $(SA)^*S \rightarrow \mathcal{D}(A)$ such that for all prefix $\eta = s_1 a_1 \dots a_{n-1} s_n$, we have $\text{Supp}(\sigma(\eta)) \subseteq A(s_n)$, where Supp denotes the support of the probability distribution. As for games, we are interested in the complexity of strategies: whether we need no memory, finite memory or infinite memory; whether we need to use randomness or not.

In order to properly define the different solution concepts proposed in the following, we need to introduce two additional notions: induced Markov chains and the truncated sum payoff function.

Markov chains induced by a strategy. When fixing the strategy σ of the rational agent in an MDP, the resulting process is fully stochastic and is called a *Markov chain* (MC). We omit technical details here (the interested reader will find exhaustive discussion in [26,18,1]). Markov chains are essentially MDPs where for all $s \in S$, we have that $|A(s)| = 1$. Given an MDP D and a strategy σ , let D^σ be the induced MC. Assuming σ uses only finite memory, then D^σ is also finite.

In this MC D^σ , an *event* is a measurable set of runs \mathcal{E} . Every event has a uniquely defined probability [34]: we denote by $\mathbb{P}_D^\sigma(\mathcal{E})$ the probability that a run of the MDP D belongs to \mathcal{E} when the player follows strategy σ . Given a measurable function f mapping runs to the numerical domain, we denote by $\mathbb{E}_D^\sigma(f)$ the *expected value* or *expectation* of f over runs in D induced by strategy σ .

Truncated sum payoff. Let $D = (S, s_{\text{init}}, A, \delta, w)$ be an MDP with a *single-dimensional* weight function $w: A \rightarrow \mathbb{N}_0$ that assigns to each action $a \in A$ a strictly positive integer. Let $T \subseteq S$ be a set of target states. Given a run $\rho = s_1 s_2 \dots s_i \dots$, with $s_1 = s_{\text{init}}$, in the MDP, we define its *truncated sum* up to T to be $\text{TS}^T(\rho) = \sum_{j=1}^{n-1} w(a_j)$ if s_n is the first visit of a state in $T \subseteq S$ within ρ ; otherwise if T is never reached,

then we set $\text{TS}^T(\rho) = \infty$. The function TS^T is measurable, and so this function has an expected value in an MC and sets of runs defined from TS^T are measurable.

Intuitively, we consider graphs where all actions induce strictly positive costs and the truncated sum function computes the sum of the weights up to the first visit of a designated target T : this is exactly *what we need to model the shortest path problem*. In the following, we survey several solution concepts of interest for this problem.

3.2 Solution concept \mathcal{S}_1 : minimizing the expected time-to-work

As mentioned in Sect. 1, the most traditional approach to decision-making in such a stochastic environment is to reason about the *expected payoff*, thus in our case (Fig. 4), to choose the mean of transport that minimizes the expected time-to-work. Such a problem can be formalized as follows.

Definition 1 (Problem \mathcal{S}_1). *Given a single-dimensional weighted MDP $D = (S, s_{\text{init}}, A, \delta, w)$ and a threshold $\ell \in \mathbb{N}$, decide if there exists σ such that $\mathbb{E}_D^\sigma(\text{TS}^T) \leq \ell$.*

This problem is well-studied in the literature, and one can prove that deciding the existence of a strategy achieving an expected time-to-work less than a given threshold can be done in polynomial time, and that such a strategy is simple: it is *pure* (i.e., does not require randomness) and *memoryless* (i.e., it depends only on the current state). In layman’s terms, this solution concept is easy to handle and produced strategies are very simple. Formal discussion of this problem can be found in [3].

Theorem 1 ([3]). *Problem \mathcal{S}_1 can be decided in polynomial time. Optimal pure memoryless strategies always exist and can be constructed in polynomial time.*

Solution \mathcal{S}_1

σ : take the car.

$\mathbb{E}_D^\sigma(\text{TS}^T) = 33$ minutes.

In our practical case, it turns out that taking *the car is the strategy that minimizes the expected time-to-work*: this choice gives an expectation equal to 33 minutes. Adopting such a strategy makes sense if one thinks about regular travels, for example when deciding whether to buy a car or a yearly train pass. Indeed, in the long run, the *observed average* time-to-work will be quite close to expectation.

3.3 Solution concept \mathcal{S}_2 : traveling without taking too many risks

Observe that taking the car presents some risk: if the traffic is heavy, then work is only reached after 71 minutes. This can be unacceptable for the employee’s boss if it happens too frequently. So if the employee is *risk-averse*, optimizing the expectation may not be the best choice. For example, the employee may want to reach work within 40 minutes with high probability, say 95%. In this case, we need to solve a different problem, called a *percentile problem*.

Definition 2 (Problem \mathcal{S}_2). *Given a single-dimensional weighted MDP $D = (S, s_{\text{init}}, A, \delta, w)$, value $\ell \in \mathbb{N}$, and probability threshold $\alpha \in [0, 1] \cap \mathbb{Q}$, decide if there exists a strategy σ such that $\mathbb{P}_D^\sigma[\{\rho \mid \text{TS}^T(\rho) \leq \ell\}] \geq \alpha$.*

This problem is also well-studied in the literature [23,32,33,21,30]. Unfortunately, it is more complex to solve than problem \mathcal{S}_1 : it can be proved that computing an adequate strategy this objective requires *pseudo-polynomial* time, i.e., exponential time in the length of the binary encoding of the weights. Furthermore, in general the corresponding strategies also require exponential memory.

Theorem 2. *Problem \mathcal{S}_2 can be decided in pseudo-polynomial time, and it is PSPACE-hard. Optimal pure strategies with exponential memory always exist and can be constructed in exponential time.*

Solution \mathcal{S}_2 σ : take the train and always wait.

$$\mathbb{P}_D^\sigma [\{\rho \mid \text{TS}^T(\rho) \leq \ell\}] = 0.99.$$

First, observe that taking the train ensures to reach work within 40 minutes in 99% of the runs. Indeed, if the train is not delayed, we reach work with 37 minutes, and this happens with probability 9/10. Now, if the train is late and the employee decides to wait, the train arrives in the next 3 minutes with probability 9/10: in that case, the employee arrives at work within 40 minutes. So, the strategy consisting in going to the railway station and waiting for the train (as long as needed) gives us a probability 99/100 to reach work within 40 minutes, fulfilling our objective.

Second, it is easy to see that both bicycle and car are excluded in order to satisfy the problem \mathcal{S}_2 . With the bicycle we reach work in 45 minutes with probability one, and with the car we reach work in 71 minutes with probability 1/10, hence we miss the constraint of 40 minutes too often.

3.4 Solution concept \mathcal{S}_3 : strict worst-case guarantees

Assume now that the employee wants a strategy to go from home to work such that work is *guaranteed* to be reached within 60 minutes (e.g., to avoid missing an important meeting with his boss). It is clear that both previously defined optimal (w.r.t. problems \mathcal{S}_1 and \mathcal{S}_2 respectively) strategies are excluded: there is the possibility of heavy traffic with the car (and a journey of 71 minutes), and trains can be delayed indefinitely in the *worst case*. While the probability of such an event is actually zero, the mere existence of the corresponding run is still not acceptable when reasoning about worst-case behaviors. Moreover, whatever the threshold fixed on the acceptable time-to-work, the train option will exceed it with positive—yet very small—probability.

To ensure a strict upper bound on the reaching-time, an adequate model is to bring back *rationality* to the environment and to consider it, not anymore as a stochastic agent, but as an *antagonistic opponent*. The result is a two-player zero-sum game as in Sect. 2, where the uncertainty becomes adversarial: when there is some uncertainty about the outcome of an action, we do not consider a probabilistic model but we let an *adversary* decide the outcome of the action. The problem is defined as follows.

Definition 3 (Problem \mathcal{S}_3). *Given single-dimensional weighted MDP $D = (S, s_{\text{init}}, A, \delta, w)$, set of target states $T \subseteq S$, and value $\ell \in \mathbb{N}$, decide if there exists a strategy σ such that for all run ρ , we have that $\text{TS}^T(\rho) \leq \ell$.*

In this case, we again find an easy setting: polynomial time suffices to decide if an adequate strategy exists, and such strategies require neither memory nor randomness. More details can be found in [8,19,12].

Theorem 3 ([22]). *Problem \mathcal{S}_3 can be decided in polynomial time. Optimal pure memoryless strategies always exist and can be constructed in polynomial time.*

Solution \mathcal{S}_3 σ : take the bicycle.

Worst-case reaching-time: 45 minutes.

If we apply this technique to our running example, we obtain that taking the bicycle is a safe option to ensure the strict 60 minutes upper bound. Indeed, its worst-case reaching-time is 45 minutes. However, the expected time-to-work when following this strategy is also 45 minutes, which is far from the optimum of 33 minutes that can be obtained when we neglect the worst-case constraint (solution \mathcal{S}_1). The goal of the next solution concept is exactly to find a strategy allying the best aspects of solutions \mathcal{S}_1 and \mathcal{S}_3 .

3.5 Solution concept \mathcal{S}_4 : minimizing the expected time under strict worst-case guarantees

In answer to the previous situation, the employee may be interested in synthesizing a strategy that *minimizes the expected time-to-work under the constraint that work is reached within 60 minutes in the worst case*. This is a complex objective which cannot be expressed in the traditional frameworks. Intuitively, it combines two different views of the environment: for the expected value, the environment is seen as a *stochastic agent*, while for the worst case, it is seen as a *rational antagonistic agent*. In that sense, it constitutes an example of what we call *rich behavioral models*.

In recent joint work [9,10], we developed sound formal grounds for the analysis of such complex objectives. We called the corresponding problem the *beyond worst-case synthesis problem*. Its goal is to look for strategies that ensure, *simultaneously*, a worst-case threshold (when probabilities are replaced by adversarial choices), and a good expectation (when probabilities are taken into account). It is formally defined as follows.

Definition 4 (Problem \mathcal{S}_4). *Given a single-dimensional weighted MDP $D = (S, s_{\text{init}}, A, \delta, w)$, a set of target states $T \subseteq S$, and two values $\ell_1, \ell_2 \in \mathbb{N}$, decide if there exists a strategy σ such that:*

1. For all induced run ρ , $\text{TS}^T(\rho) \leq \ell_1$,
2. $\mathbb{E}_D^\sigma(\text{TS}^T) \leq \ell_2$.

This problem can be seen as a generalization subsuming both problems \mathcal{S}_1 and \mathcal{S}_3 . While those problems are both solvable in polynomial time and pure memoryless strategies suffice in both cases, problem \mathcal{S}_4 proves to be inherently harder. We showed that, similarly to problem \mathcal{S}_2 , pseudo-polynomial time is necessary to construct winning strategies and those strategies require memory (but no randomness). Extensive discussion of this novel framework and ad-hoc techniques can be found in [9,10].

Theorem 4 ([10]). *Problem \mathcal{S}_4 can be decided in pseudo-polynomial time and is NP-hard. Pseudo-polynomial memory is always sufficient and in general necessary, and satisfying strategies can be constructed in pseudo-polynomial time.*

Solution \mathcal{S}_4

σ : try to take the train, if the train is delayed three times consecutively, then go back home and take the bicycle.

$\mathbb{E}_D^\sigma(\text{TS}^T) \approx 37.34$ minutes.

Worst-case reaching-time: 58 minutes.

For our employee, the optimal strategy w.r.t. problem \mathcal{S}_4 is the following: try to take the train, if the train is delayed three times consecutively, then go back home and take the bicycle. This strategy is safe as it always reaches work within 58 minutes and its expectation is ≈ 37.34 minutes (much better than taking directly the bicycle). Hence, this a perfectly suitable choice for the employee. However, it comes at a cost in terms of strategy complexity: observe that this strategy requires finite memory (because the employee must be able to recall how many times the train has been delayed already), in contrast to solutions \mathcal{S}_1 and \mathcal{S}_3 .

3.6 Solution concept \mathcal{S}_5 : trade-offs between multiple objectives

The last solution concept we describe will be illustrated on a different example. The MDP on Fig. 5 represents a simplified choice model for commuting, but introduces two-dimensional weights: each action is labeled with a duration, in minutes, and a cost, in dollars. Multi-dimensional MDPs are useful to analyze systems with *multiple objectives* that are potentially conflicting and make necessary the analysis of trade-offs. For instance, we may want a choice of transportation that gives us high probability to reach work in due time but also limits the risk of an expensive journey. Since faster options are often more expensive, *trade-offs* have to be considered.

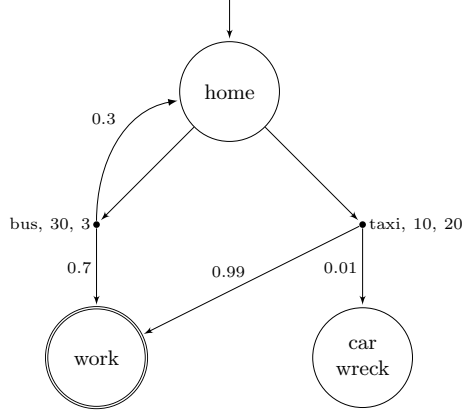


Fig. 5. Multi-constraint percentile queries can help when actions both impact the duration of the journey (first dimension) and its cost (second dimension): trade-offs have to be considered.

Recall problem \mathcal{S}_2 : it asks to decide the existence of strategies satisfying a *single percentile constraint*. As all models discussed up to now, this problem can only be applied to single-dimensional MDPs. For example, one may look for a strategy that ensures that 80% of the runs take at most 40 minutes (constraint C1), or that 50% of them cost at most 10 dollars (C2). A good strategy for C1 would be to take the taxi, which guarantees that work is reached within 10 minutes with probability $0.99 > 0.8$. For C2, taking the bus is a good option, because already 70% of the runs will reach work for only 3 dollars. Note that taking the taxi does not satisfy C2, nor does taking the bus satisfy C1.

In practice, a desirable strategy should be able to satisfy *both* C1 and C2. This is the goal of our model of *multi-constraint percentile queries*, introduced in [30]. It can be formalized as follows.

Definition 5 (Problem \mathcal{S}_5). Given a d -dimensional weighted MDP $D = (S, s_{\text{init}}, A, \delta, w)$, and $q \in \mathbb{N}$ percentile constraints described by sets of target states $T_i \subseteq S$, dimensions $k_i \in \{1, \dots, d\}$, value thresholds $\ell_i \in \mathbb{N}$ and probability thresholds $\alpha_i \in [0, 1] \cap \mathbb{Q}$, where $i \in \{1, \dots, q\}$, decide if there exists a strategy σ such that

$$\forall i \in \{1, \dots, q\}, \mathbb{P}_D^\sigma [\{\rho \mid \text{TS}_{k_i}^{T_i}(\rho) \leq \ell_i\}] \geq \alpha_i$$

where $\text{TS}_{k_i}^{T_i}$ denotes the truncated sum on dimension k_i and w.r.t. target set T_i .

Our technique is able to solve the problem for queries with multiple constraints, potentially related to different dimensions of the weight function and to different target sets: this offers great flexibility which is useful in modeling practical applications. In terms of complexity, two things should be noted. First, while significantly more expressive than problem \mathcal{S}_2 , this problem does not induce a blow-up in complexity: the algorithmic complexity and the memory needs for strategies are similar. Second, among the solution concepts presented in this paper, problem \mathcal{S}_5 is the only one that actually requires strategies to use randomness in full generality.

Theorem 5 ([30]). Problem \mathcal{S}_5 can be decided in exponential time in general, and pseudo-polynomial time for single-dimension single-target multi-constraint queries. The problem is PSPACE-hard even for single-constraint queries. Randomized exponential-memory strategies are always sufficient and in general necessary, and satisfying strategies can be constructed in exponential time.

Solution \mathcal{S}_5

σ : try the bus once, then take the taxi if the bus does not depart.

$$\mathbb{P}_D^\sigma [\{\rho \mid \text{TS}_1^T(\rho) \leq 40\}] > 0.99.$$

$$\mathbb{P}_D^\sigma [\{\rho \mid \text{TS}_2^T(\rho) \leq 10\}] = 0.7.$$

Let us see what can be an appropriate strategy for the conjunction ($C1 \wedge C2$) in our example. One is to try the bus once, and then take the taxi if the bus does not depart. Indeed, this strategy ensures that work is reached within 40 minutes with probability larger than 0.99 thanks to runs home-bus-work (probability 0.7 and duration 30) and home-bus-home-taxi-work (probability 0.297 and duration 40). Furthermore, it also ensures that more than half the time, the total cost to target is at most 10 dollars, thanks to run home-bus-work which has probability 0.7 and cost 3. Observe that this strategy requires *memory*.

Solution \mathcal{S}_5

σ' : take the bus (resp. the taxi) with probability 3/5 (resp. 2/5).

$$\mathbb{P}_D^\sigma[\{\rho \mid \text{TS}_1^T(\rho) \leq 40\}] > 0.81.$$

$$\mathbb{P}_D^\sigma[\{\rho \mid \text{TS}_2^T(\rho) \leq 10\}] > 0.5.$$

In this particular example, it is possible to build another acceptable strategy which is memoryless but requires *randomness*. Consider the strategy that flips an unfair coin in home to decide if we take the bus or the taxi, with probabilities 3/5 and 2/5 respectively. Constraint C1 is ensured thanks to runs home-bus-work (probability 0.42) and home-taxi-work (probability 0.396). Constraint C2 is ensured thanks to runs (home-bus)^{*n*}-work with $n = 1, 2, 3$: they have probabilities 0.42, ≥ 0.07 and ≥ 0.01 respectively, totaling to ≥ 0.5 , while they all have cost at most $3 \cdot 3 = 9 < 10$.

4 Conclusion

Through this contribution, we discussed two traditional paradigms used for describing the behavior of agents in complex systems: *rationality* and *stochasticity*. Both have proved to be powerful abstraction mechanisms in many contexts. The main point of this paper is that it is sometimes necessary to bring together these two paradigms in *rich behavioral models*.

Our core research field is *controller synthesis* for reactive systems, hence the models we develop often arise from problems linked to computer systems. In Sect. 2, we gave an overview of the application of game theory for controller synthesis, highlighting the crux of the approach and the main challenges.

In Sect. 3, we surveyed five solution concepts, illustrated on an everyday life example: how to plan a journey in an uncertain environment. We presented three classical approaches, based on the traditional dichotomy between rational and stochastic environments, and then discussed recent advances breaching this dichotomy. The simple examples presented in the paper are already sufficient to witness that such rich behavioral models do have great practical interest.

Our presentation was at a very high level, aimed at a broad audience, but the reader interested in actual mathematical models, algorithms and tools will hopefully find more information in the corresponding cited papers. It is also worth noting that all concepts presented here for the shortest path are indeed applicable in various, more general frameworks (e.g., many other payoff functions have been studied).

While our work is motivated by applications to controller synthesis, we are led to believe that such models can also prove interesting in other areas. For example, related models have been used in economics [2] to model investor profiles as strategies in games with stochastic aspects. With that in mind, beyond worst-case strategies (solution concept \mathcal{S}_4) that ensure both sufficient risk-avoidance and profitable expected return can be of interest, as well as related models of value-at-risk [30,31].

References

1. C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008.
2. N. Berger, N. Kapur, L.J. Schulman, and V.V. Vazirani. Solvency games. In *Proc. of FSTTCS, LIPIcs 2*, pages 61–72. Schloss Dagstuhl - LZI, 2008.
3. D.P. Bertsekas and J.N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

4. R. Bloem, K. Chatterjee, T.A. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *Proc. of CAV*, LNCS 5643, pages 140–156. Springer, 2009.
5. A. Bohy, V. Bruyère, E. Filiot, and J.-F. Raskin. Synthesis from LTL specifications with mean-payoff objectives. In *Proc. of TACAS*, LNCS 7795, pages 169–184. Springer, 2013.
6. P. Bouyer, N. Markey, M. Randour, K.G. Larsen, and S. Laursen. Average-energy games. In *Proc. of GandALF*, EPTCS 193, pages 1–15, 2015.
7. R. Brenguier, L. Clemente, P. Hunter, G.A. Pérez, M. Randour, J.-F. Raskin, O. Sankur, and M. Sassolas. Non-zero sum games for reactive synthesis. In *Proc. of LATA*, LNCS 9618, pages 3–23. Springer, 2016.
8. T. Brihaye, G. Geeraerts, A. Haddad, and B. Monmege. To reach or not to reach? Efficient algorithms for total-payoff games. In *Proc. of CONCUR*, LIPIcs 42, pages 297–310. Schloss Dagstuhl - LZI, 2015.
9. V. Bruyère, E. Filiot, M. Randour, and J.-F. Raskin. Expectations or guarantees? I want it all! A crossroad between games and MDPs. In *Proc. of SR*, EPTCS 146, pages 1–8, 2014.
10. V. Bruyère, E. Filiot, M. Randour, and J.-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In *Proc. of STACS*, LIPIcs 25, pages 199–213. Schloss Dagstuhl - LZI, 2014.
11. A. Chakrabarti, L. de Alfaro, T.A. Henzinger, and M. Stoelinga. Resource interfaces. In *Proc. of EMSOFT*, LNCS 2855, pages 117–133. Springer, 2003.
12. K. Chatterjee, L. Doyen, M. Randour, and J.-F. Raskin. Looking at mean-payoff and total-payoff through windows. *Information and Computation*, 242:25–52, 2015.
13. K. Chatterjee, M. Randour, and J.-F. Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3-4):129–163, 2014.
14. B.V. Cherkassky, A.V. Goldberg, and T. Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Math. programming*, 73(2):129–174, 1996.
15. A. Church. Applications of recursive arithmetic to the problem of circuit synthesis. *Summaries of the Summer Institute of Symbolic Logic*, 1:3–50, 1957.
16. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, LNCS 131, pages 52–71. Springer, 1981.
17. A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.
18. J. Filar and K. Vrieze. *Competitive Markov decision processes*. Springer, 1997.
19. E. Filiot, R. Gentilini, and J.-F. Raskin. Quantitative languages defined by functional automata. In *Proc. of CONCUR*, LNCS 7454, pages 132–146. Springer, 2012.
20. E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, LNCS 2500. Springer, 2002.
21. C. Haase and S. Kiefer. The odds of staying on budget. In *Proc. of ICALP*, LNCS 9135, pages 234–246. Springer, 2015.
22. L. Khachiyan, E. Boros, K. Borys, K.M. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. pages 204–233, 2008.
23. Y. Ohtsubo. Optimal threshold probability in undiscounted markov decision processes with a target set. *Applied Math. and Computation*, 149(2):519 – 532, 2004.
24. M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
25. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of POPL*, pages 179–190. ACM Press, 1989.
26. M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.
27. P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1):206–230, 1987.
28. M. Randour. Automated synthesis of reliable and efficient systems through game theory: A case study. In *Proceedings of the European Conference on Complex Systems 2012*, Springer Proceedings in Complexity XVII, pages 731–738. Springer, 2013.
29. M. Randour. *Synthesis in Multi-Criteria Quantitative Games*. PhD thesis, UMONS, Université de Mons, Belgium, 2014.
30. M. Randour, J.-F. Raskin, and O. Sankur. Percentile queries in multi-dimensional Markov decision processes. In *Proc. of CAV*, LNCS 9206, pages 123–139. Springer, 2015.
31. M. Randour, J.-F. Raskin, and O. Sankur. Variations on the stochastic shortest path problem. In *Proc. of VMCAI*, LNCS 8931, pages 1–18. Springer, 2015.
32. M. Sakaguchi and Y. Ohtsubo. Markov decision processes associated with two threshold probability criteria. *Journal of Control Theory and Applications*, 11(4):548–557, 2013.

33. M. Ummels and C. Baier. Computing quantiles in Markov reward models. In *Proc. of FOSSACS*, LNCS 7794, pages 353–368. Springer, 2013.
34. M.Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proc. of FOCS*, pages 327–338. IEEE Computer Society, 1985.
35. M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. of LICS*, pages 332–344. IEEE Computer Society, 1986.
36. Y. Velner, K. Chatterjee, L. Doyen, T.A. Henzinger, A.M. Rabinovich, and J.-F. Raskin. The complexity of multi-mean-payoff and multi-energy games. *Information and Computation*, 241:177–196, 2015.
37. J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.
38. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.