A Survey on Probabilistic Planning and Temporal Scheduling with Safety Guarantees

Jan Vermaelen and **Hoang Tung Dinh** and **Tom Holvoet**
KU Leuven, Dept. of Computer Science, imec-DistriNet, B-3001 Leuven, Belgium
{jan.vermaelen, hoangtung.dinh, tom.holvoet}@cs.kuleuven.be

Abstract

Autonomous robotic systems are often safety-critical since unsafe behavior can have disastrous consequences. In this paper, we survey existing frameworks that can incorporate safety guarantees or constraints in the design of an autonomous system. Rather than verifying the guarantees during simulation or testing, such frameworks can enforce them upfront. Furthermore, in a physical setting, the effects of an agent's actions can not be considered deterministic. The frameworks have to take into account the uncertain or probabilistic effects of actions. Different frameworks provide different types of guarantees. Yet no comprehensive overview of such frameworks and the guarantees they offer exists today. This survey tries to answer the need for an overview.

Probabilistic planning is often tackled using Markov Decision Processes (MDPs). Many extensions of MDPs exist, some of which can provide explicit safety guarantees. In the existing research, constraints on objective functions, reachable states, and execution paths of the system are obtained. For scheduling, Simple Temporal Networks (STNs) are addressed. STNs inherently incorporate temporal constraints, enforcing temporal relations. As an extension, probabilistic constraints on failure can be imposed as well.

Introduction

Autonomous robotic systems are becoming popular in both industry and domestic applications. An autonomous system makes decisions autonomously to, ideally, achieve or maximize its objective. The input relevant for the decisions often originates from (noisy) sensor values. For safety-critical applications, explicit safety guarantees are desirable since unsafe behavior can have disastrous consequences. An attentive and thoughtful planning strategy regarding the agent's actions is required.

When a system is deployed in a physical environment, the effects of the system's actions can not be considered fully deterministic. This uncertainty has to be taken into account during planning. For example, the effects on the agent and its environment could be modeled probabilistically. The strength of the obtainable guarantees will suffer from this uncertainty as well, leading to probabilistic claims.

Traditionally, autonomous systems can be verified once they are designed. Properties are checked **after** planning, for example, using a model checking tool (Kwiatkowska, Norman, and Parker 2002). Making use of model checking as a building block, a plan that provides concrete guarantees can be attained via a non-trivial iteration of planning and verification. In each iteration, the plan has to be adapted in such a way that the guarantees become more likely, until the verification step yields satisfactory results. It becomes interesting to investigate frameworks that can take constraints into account in one single planning step.

In the literature, different frameworks exist, taking into account different kinds of uncertainty, and providing different kinds of guarantees. However, a systematic overview that supports understanding which frameworks are relevant for which needs is lacking. Such an overview is relevant, both for researchers and practitioners, to identify the most appropriate instrument for a particular problem situation. This paper aims to provide such an overview, focusing on the guarantees that can be provided by the different frameworks.

As an illustrative example, consider the deployment of Unmanned Aerial Vehicles (UAVs) for inspection tasks on industrial sites, alongside people and potentially hazardous equipment. In such a setting, it is highly desirable to obtain safety guarantees. From a planning perspective, safe decision making is essential. For example, a safety guarantee can state that the behavior of the UAV will not cause it to crash into other agents. We do not consider external hazardous events such as hardware failure. Furthermore, spinning the rotors of the UAV for a certain amount of time at a certain speed does not yield a deterministic translation of the UAV in the physical world. Many external factors, such as the wind and the UAV's payload, influence the movement of the UAV. When discussing the different frameworks, we use this UAV example to clarify their possibilities and strengths.

In the remainder of this section, we outline the methodology used in this survey and elaborate on related surveys. In the next section, we elaborate on the different frameworks that emerge in our study. We investigate the practical contributions of the different frameworks regarding their abilities to provide guarantees. In the final section, we summarize the overall conclusions and discuss future work. The inability to provide hard guarantees for practical applications of systems in uncertain environments will emerge.

Methodology

The methodology followed to obtain the overview of frameworks present in this study is based on the approach described in the *Guidelines for performing Systematic Literature Reviews in Software Engineering* (Kitchenham and Charters 2007). However, in an effort to cover most of the relevant and interesting research, we rely on *snowballing* as well (Wohlin 2014).

The scope of this survey is delimited to planning for autonomous systems that interact with the physical world. In other words, we are primarily interested in planning for autonomous robotic systems, which can cope with uncertainty regarding the effects of their actions. We formulate the research question: "*What kinds of guarantees related to safety can be provided on the behavior of a physical, autonomous system, using different probabilistic planning approaches?*" The corresponding search criteria are "*probabilistic planning*", together with the term "*safe*" and the term "*guarantee*" or "*constraint*". Variants on those terms are taken into account as well.

The initial search is limited to the first ten pages on *Google Scholar* (on February 12, 2020). Many papers are considered out of scope or irrelevant, to focus on the relevant core of the existing research. Papers that are not related, such as most medical papers, are ignored. For the papers that are related, we focus on the frameworks they address. We only take into account papers that describe (robust) planning frameworks with explicit guarantees or frameworks that provide an essential basis for this requirement. We do not look into the specific algorithms that support these frameworks. For the reader interested in the solution strategies, we refer to the corresponding literature.

A number of the obtained papers relate to safety in the exploration phase of *Reinforcement Learning* approaches. In this survey, we do not look into such frameworks for unknown models. For safety-critical systems, a naive exploration approach would lead to disastrous effects. However, we are aware that much recent research is tackling this problem.

The obtained set of papers form a good basis for snowballing, leading to additional relevant research. This survey summarizes the available research, and as such, provides an overview and a point of reference for further research. For the relevant frameworks, we indicate what kinds of explicit guarantees they can provide.

Related Work

For safety-critical systems, safety guarantees are of utmost importance. To the best of our knowledge, there is currently no survey investigating different probabilistic planning or scheduling frameworks, focusing on the guarantees they can provide.

A related survey on motion planning addresses uncertainties that can arise in practical applications (Dadkhah and Mettler 2012). The autonomous guidance of UAVs is considered, and the different sources of uncertainty that can occur are distinguished. The survey discusses relevant techniques for the planning of a system in the presence of uncertainties. Contributions from both the *Artificial Intelligence and Robotics* and the *Dynamical Systems and Controls* community are included.

Another survey, regarding decision making and optimization, looks deeper into the uncertainty representations (Keith and Ahner 2019). Many different strategies are compared. The use of uncertainty sets seems to remain a popular technique. In both surveys, Markov Decision Processes and relevant extensions thereon, make an appearance. In our survey, we majorly focus on frameworks that additionally provide guarantees on the achieved plan.

Furthermore, *Probabilistic Model Checking* is conceptually related to our focus of interest. Probabilistic guarantees can be achieved on the model of a system (Katoen 2016; Kwiatkowska, Norman, and Parker 2002). In contrast to our focus, the guarantees are obtained *after* designing the system. A complete model of the system, including its plan, is required to be able to apply model checking or verification in general. We are interested in probabilistic planning or scheduling frameworks that can inherently provide the desired guarantees.

In general, we observe that the most popular frameworks for probabilistic planning are based on *Markov Decision Processes* (MDPs). Note that MDPs also provide an appropriate basis for model checking (Baier, Hermanns, and Katoen 2019; Kwiatkowska, Norman, and Parker 2002). From a scheduling point of view, *Simple Temporal Networks* (STNs) are applicable. As we focus on explicit safety guarantees, traditional MDPs, and to some extent, traditional STNs, will not suffice.

Frameworks

In this section, we take a look at the different frameworks that are present in the literature and what guarantees they can provide. As we do not look into the specific algorithms to obtain solutions for each of the frameworks, we want to make a general remark upfront. For some frameworks, different algorithms are available to solve the problem at hand, possibly resulting in different solutions. For more elaborate and fine-grained problems, obtaining an exact solution, using sound and complete algorithms, becomes infeasible. Faster algorithms, which return approximate solutions, might be preferred. The precision and correctness of the obtained guarantees will depend on the properties of the used algorithm.

The different frameworks are illustrated using a practical example. We look into the modeling of an application of a UAV. The UAV has to operate on an industrial plant, executing surveillance and visual inspection of the infrastructure and the operation of other agents. In the real, physical world, the effects of the UAV executing actions can not be predicted precisely. Different frameworks will address this uncertainty in different ways. An illustrative safety property could state that the autonomous UAV should *never* fly into other agents. More feasibly, one could demand that the autonomous UAV only makes contact with other agents with a probability below some small threshold.

Markov Decision Process

The most popular framework used for probabilistic planning is a *Markov Decision Process* (MDP) (Puterman 2014). MDPs are being addressed in numerous fields (White 1985; 1988; 1993). A traditional MDP is, however, not able to provide explicit safety guarantees. We elaborate on MDPs first, since several other frameworks are based on them.

In an MDP, actions and states of the system have associated rewards or costs. Solving an MDP corresponds to finding a policy[1] that maximizes the expected cumulative reward. This cumulative reward is calculated as the discounted sum of rewards (positive) and costs (negative).

Formally, an MDP is defined as a 6-tuple $\langle H, \gamma, S, A_s, T_{a,s}, R_{a,s} \rangle$, where H represents the possibly infinite decision horizon; $\gamma \in [0, 1]$ denotes the discount factor; S is the set of all possible states; A_s holds the available actions for each state s; $T_{a,s}$ holds the transition probabilities to the succeeding states, for executing action a in state s; and $R_{a,s}$ holds the corresponding rewards. Algorithms to solve an MDP are usually based on *Value Iteration* or *Policy Iteration*.

We describe a traditional MDP for the UAV application at hand.[2] For simplicity reasons, S and A_s are often assumed to be discrete and finite. However, continuous domains are often more appropriate, even for a simple example like this UAV. Numerous research has been dealing with this, as well. A number of properties contribute to the state of the system. Examples of these properties are the UAV's location L, its height H, and its battery level B. The domain of each of those properties can be represented as a set. For example, if the battery level is expressed using a percentage, the domain is defined as $B = \{0, 1, \ldots, 100\}$. The entire state space is represented as a set $S = L \times H \times B \times \ldots$, containing every possible state of the UAV.

For a UAV, the set of actions A can be defined as $\{Lift, Lower, Hover, MoveNorth, \ldots\}$. We assume that all actions are applicable in all states. It is the responsibility of the autopilot (and hence in the end, of the planner) to make sure that no unsafe actions are executed.

The decision horizon H has to be set sufficiently large so that a complete flight path of the UAV can be taken into account. If a recurrent inspection is desired and the planning includes a recharging procedure for the UAV's battery, a much larger, perhaps infinite horizon might be preferred.

The discount factor γ is set close to 1. A lower discount factor stimulates the UAV to reach its objective faster (since a later reward gets discounted more) and could cause the UAV to take more risks.

The transition probabilities in $T_{a,s}$ describe the effects of the UAV executing its actions. For example, a $Lift$ action could make the UAV increase its height by one step[3] with a probability of 0.8, while the height may remain constant with a probability of 0.2. Further, the effects are state-dependent. Executing a $Lower$ action when the UAV is hovering at a certain height, will likely cause this height to decrease. Executing a $Lower$ action when the UAV is landed on the ground, should decrease its height only with a probability of 0.

Finally, a simple reward function $R_{a,s}$ assigns a high, positive reward to states or actions that achieve a (partial) goal, such as acquiring some visual inspection. Unsafe or undesired states and actions get a large, negative reward to stimulate the UAV to avoid them.

As mentioned, planning with a traditional MDP does not provide explicit guarantees. MDPs only try to maximize the reward or minimize the cost. However, many other frameworks are based on this traditional MDP.

Partially Observable MDP

A first extension of MDPs is related to the agent's inability to observe the current state. If the system is determined by a known MDP, but the agent can not always observe the system's underlying state, the agent should instead maintain a probability distribution over the possible states in which the system could be. This distribution is referred to as the agent's *belief*. Such an extension of an MDP is called a *Partially Observable MDP* (POMDP).

Formally, a POMDP is defined similarly to an MDP, extended with the notions of observations and beliefs. For this purpose, the original 6-tuple is extended with Ω, being the set of possible observations, and the probabilistic observation function O, which correlates the observations with actual states. Finally, the initial belief b_0 can be added to the tuple as well. The solution of a POMDP consists of a policy that provides a mapping from beliefs to actions.

Different approaches and algorithms related to POMDPs were already investigated decades ago (Monahan 1982). Since then, much further research has become available (Lovejoy 1991; White 1991; Cassandra 1998b; Aberdeen 2003). Similar to MDPs, POMDPs are being used in many different fields (Cassandra 1998a).

The precise state of the UAV is not known at each moment in time. The observations mainly depend on sensor data. For properties such as the $Location$ and the $Height$, the system relies on data from cameras and GPS sensors. Such data does not always yield correct or complete information. For example, if the UAV explores an environment under low light conditions or in the presence of too few GPS connections, information may lack. A POMDP addresses this inability of an agent to precisely determine its current state. The current belief is determined by the previous belief and the executed action, along with the current observation and the transition probabilities (which, to some extent, represent the theoretical, physical behavior of the UAV).

Analogous to MDPs, POMDPs can not provide explicit safety guarantees. However, POMDPs take into account uncertainty regarding state observation. This feature is highly relevant for the planning of physical systems. Numerous other frameworks are based on POMDPs.

[1] For an MDP, a policy contains a mapping from every possible state of the system to the action that has to be executed by the system in that state.

[2] We do not provide a complete and detailed MDP. The application is simplified in order to provide a comprehensive example.

[3] As we did not specify how the domain of height is modeled, "one step" could refer to one unit, one level, ...

Robust (PO)MDP

When using (PO)MDPs for planning the behavior of physical agents, the transition probabilities are usually estimated from data gathered during previous executions. This estimation inherently introduces some level of uncertainty on these parameters. The obtained policy and the expected total reward can be very sensitive to small changes in the transition probabilities. *Robust (PO)MDPs* take such uncertainty into account (Nilim and El Ghaoui 2005; Xu and Mannor 2010; Wiesemann, Kuhn, and Rustem 2013; Tamar, Mannor, and Xu 2014; Osogami 2015).

Formally, a Robust MDP is defined similarly to a traditional MDP, except for the transition probabilities $T_{a,s}$. An uncertainty set $\tau_{a,s}$ can be used to denote the uncertain transitions, rather than using known, fixed probabilities. Furthermore, probabilistic information regarding the unknown parameters can be considered, for example, using confidence regions. The resulting policy has to attain the highest worst-case performance over that confidence region. Uncertainty regarding the observation parameters can be taken into account as well.

The example we provided earlier, in which the UAV's $Lift$ action has a probability of 0.8 to succeed (and 0.2 to fail), is purely illustrative. Practically, such probabilities are indeed estimated from data gathered over previous flights. After sufficient flights, the achieved data becomes representative, but some level of uncertainty remains. Ideally, a Robust (PO)MDP also takes the quantity of, or confidence in, the gathered flight data into account.

Robust (PO)MDPs do not achieve explicit safety guarantees. However, we did opt to include them in this survey, as they contribute to handling uncertainty. In practical, physical applications, Robust (PO)MDPs can be more useful as they take into account the reality that transition probabilities may not be known exactly. We did not want to withhold this insight from the reader.

Constrained (PO)MDP

In our research question, we emphasize the need for safety guarantees. The fundamental MDP and POMDP frameworks only maximize the reward. They do not take into account additional constraints. From here on, we look into MDP-based extensions that do.

A first step is to take into account additional constraints on expected costs or resource usage (Altman 1999). This extension turns the traditional (PO)MDP into a *Constrained (PO)MDP* (C(PO)MDP).

Formally, a C(PO)MDP corresponds to a traditional (PO)MDP, extended with $\kappa_{a,s}$, an n-dimensional cost vector, for each action a in each state s. Apart from the traditional reward that is to be maximized, n other objectives are expressed, related to the n respective costs. On these secondary objectives, constraints are applicable. Such constraints can limit expected costs or utilization of certain resources to a maximum threshold. An illustrative example of such a resource is the available time. Each action requires some time to execute. The corresponding constraint then represents the maximum expected duration of the agent's execution.

This extension turns the problem at hand into a constrained optimization problem. The resulting policy is still a policy that maximizes the overall reward, yet obeys the posed constraints. The book "*Constrained Markov Decision Processes*" by *Eitan Altman, 1999* provides a thorough overview of CMDPs, examples of applications, and approaches for obtaining solutions (Altman 1999). CMDPs are addressed for strategic mission planning with constraints (Ding, Pinto, and Surana 2013) and form an appropriate basis for reinforcement learning problems as well (Maeda et al. 2019).

Several algorithms to solve CPOMDPs have been proposed. Initially, an exact dynamic programming update was presented (Isom, Meyn, and Braatz 2008). Subsequently, more efficient online and approximate approaches were introduced (Undurti and How 2010; Lee et al. 2018; Kim et al. 2011; Poupart et al. 2015; Walraven and Spaan 2018).

Intuitively, one could argue that constraints can, to some extent, be dealt with using the normal cost function of a traditional MDP. For illustrative purposes, assume a scenario in which constraints are present on individual states of the system. By assigning an arbitrarily high penalty to undesired states, the planner tries to avoid these states. This approach, however, does not necessarily lead to proper solutions. If an undesired state has a very high cost assigned to it, while (under normal operation) this state could unintentionally get reached with a relatively small probability, the planner might behave too conservatively. The system will make sure to avoid such states, even if this behavior leads to zero progress. Lowering the penalty does not solve the problem either. The planner can switch from being too conservative to too risky (Undurti and How 2010).

A property relevant for a UAV is that it should not run out of battery power during flight. Simply assigning a high cost to states that correspond with a low battery level, provides no guarantees. However, since each action has a specific power consumption, we can define an objective that keeps track of the total power consumption. A CMDP can then be addressed to add a constraint to limit the expected power usage, preferably, to a threshold smaller than the capacity of the battery.

Chance Constrained (PO)MDP

Besides constraints on costs and the use of resources, probabilistic constraints are relevant as well. This concept is addressed by *Chance Constrained (PO)MDPs* (CC-(PO)MDPs). They still maximize the expected cumulative reward, yet bound the probability ("chance") of certain events occurring during execution. *Chance Constraints* can be used to put bounds on the probabilities of violating safety constraints or of failure.

Formally, a CC-POMDP can correspond to a traditional POMDP extended with a set C, containing constraints defined over the states of the system; and a vector Δ, containing thresholds on the probabilities of violating constraints defined by C (Santana, Thiébaux, and Williams 2016). Alternatively, Δ can denote a joint risk budget, that is, a threshold on the probability of visiting any state of a predefined

set of risky or unsafe states (Khonji, Jasour, and Williams 2019). The risk can also be bounded as a function of the reward (Ayton and Williams 2018). Constraints regarding the state probability distribution have been subject to research as well (El Chamie et al. 2018).

For the UAV application, a set of unsafe states can be considered in which the UAV will likely experience failure. Straightforward examples consist of states in which the UAV runs out of battery power during flight and states in which the UAV flies into other agents. Both examples lead to scenarios in which the UAV would crash. A CC-(PO)MDP can be addressed to constrain the probability of visiting such undesired states. For the UAV at hand, as for any safety-critical system, this kind of guarantee is highly relevant. We can demand that the expected probability of the UAV showing such unsafe behavior remains below some arbitrarily low threshold.

Different sorts of unsafe behavior could be given different thresholds. Operation under low battery conditions might be considered less unsafe, as a low battery indicator can still signal the UAV to return home, before draining the battery completely. A higher threshold might be adequate. The threshold for approaching other agents too closely, however, should be set sufficiently low. Finally, states violating *both* constraints (that is, the UAV approaches other agents too closely while experiencing a low battery level) might be considered more unsafe than those violating the individual constraints. The UAV might have a delayed reaction time to critical observations due to reduced power availability.

It is important to remark that an accurate model of the environment of the UAV and the effects of its actions is required to obtain precise guarantees. Parameters like the transition probabilities, which might not be straightforward to estimate in the first place, have to be known precisely.

Finally, interesting to note is the relation between CC-POMDPs and CPOMDPs. It has been observed that the latter is more general than the former (Khonji, Jasour, and Williams 2019). As a result, any CC-POMDP problem can be reduced to a CPOMDP problem. It follows that an algorithm to solve CPOMDPs can also be used to solve CC-POMDPs.

Path Constrained MDP

Constraints on consecutive states of the system have been studied as well. *Path Constrained MDPs* (PC-MDPs) were proposed to combine a probabilistic model checking aspect with a planning approach (Teichteil-Königsbuch 2012). The resulting policy is obtained in a single design pass. PC-MDPs allow the user to express constraints on the execution traces of the system, in Probabilistic (Real Time) Computation Tree Logic (PCTL). Execution of the obtained policy satisfies the posed probabilistic path constraints.

Formally, a PC-MDP corresponds to a traditional MDP, extended with a set Λ, containing a number of PCTL path constraints; and P denoting the set of probabilities corresponding to these constraints. The constraints are expressed using *strong until* temporal operators. $\lambda_i = fU_{\diamond p_i}g$ indicates that there is at least ($\diamond \in \{\geq, >\}$), at most ($\diamond \in \{\leq, <\}$), or exactly ($\diamond \in \{=\}$) a probability of p_i that eventually g will become true, and that f is and remains true until then. The formulas f and g are boolean state functions that map each state to a boolean value.

For the UAV application, an entire execution path can be interpreted as a complete flight scenario, from take-off to landing. An illustrative constraint expresses that in each possible path, no landing procedure should be initiated (for example, by considerably decreasing the thrust) before all environment checks have been run successfully. A PC-MDP can take such constraints into account. The constraint can be written as $(\neg l)Uc$, where l holds in a state if the landing procedure is initiated and c denotes whether the environment checks have been executed successfully. More expressive and realistic, using a probabilistic variant thereon, $(\neg l)U_{\geq 0.99}c$ states that the temporal formula must hold with a probability of at least 0.99, for a randomly sampled flight. Exceptions could, for example, be tolerated when uncontrollable factors cause an emergency landing to obtain priority to avoid more disastrous hazards. Note that constraints of the form $(True)U_{<p}h$ can be used to ensure that a state in which h holds is only visited with a probability below p.

If the probabilistic expressiveness is not required ($p_i = 0$ *or* 1), a more efficient approach can be addressed (Sprauel, Kolobov, and Teichteil-Königsbuch 2014). The *strong until* operators then express more traditional state constraints. For example, $(\neg f)U_{=0}g$ ensures that a state where g holds should not be visited before a state is visited where f holds. The constraint $(True)U_{=0}g$ ensures that no state should ever get reached where g holds. Similarly, $(True)U_{=1}g$ ensures that a mandatory goal g gets reached eventually.

POMDPs with Safe-Reachability Objectives

The goal of *POMDPs with safe-reachability objectives* is to satisfy a safe-reachability objective in all possible execution paths of the system, that is, including worst-case scenarios (Wang, Chaudhuri, and Kavraki 2018a).

A safe-reachability objective can ensure that a goal state is eventually reached with a probability above a certain threshold. Simultaneously, the probability of visiting an unsafe state is *always* kept below some threshold. It follows that POMDPs with safe-reachability objectives contrast with C(PO)MDPs and CC-(PO)MDPs, as the latter obtain a policy that maximizes the agent's expected cumulative reward while bounding an *expected* cost or risk. Furthermore, to stimulate reaching the goal, C-POMDPs and CC-POMDPs typically hold large reward values for goal states. This approach is unable to provide any hard guarantees. POMDPs with safe-reachability objectives, on the other hand, inherently try to make the agent reach its goal state with a probability above a certain threshold. It follows that POMDPs with safe-reachability objectives can provide stronger guarantees.

Formally, a traditional POMDP gets supplemented with a safe-reachability objective. This safe-reachability objective essentially defines a tuple $\langle Safe, Goal \rangle$, where $Safe$ and $Goal$ are sets of *safe* and *goal* states, respectively. Further, a minimum threshold (ζ_g) on the probability of reaching a goal state ($\in Goal$) and a maximum threshold (ζ_s) on

the probability of reaching unsafe states ($S \setminus Safe$) are included. Concretely, the resulting policy makes the agent reach a belief for which the probability of being in a *goal* state exceeds the threshold ζ_g. Each belief encountered until this goal was reached, corresponds to an unsafe state only with a probability below the threshold ζ_s.

Offline policy synthesis methods to solve POMDPs with safe-reachability objectives have been proposed in the literature (Wang, Chaudhuri, and Kavraki 2018a; 2019). Further, an online approach (Wang, Chaudhuri, and Kavraki 2018b) and a sampling-based approach (Newaz, Chaudhuri, and Kavraki 2019) address efficiency and scalability.

Applied to the UAV example, a double guarantee regarding the behavior of the UAV and its mission can be enforced. We can formulate an objective claiming that the inspection flight goal of the UAV gets achieved with a minimum probability, while the probability of crashing is bounded. The flight goal is achieved in any state in which the required inspection footage has been obtained, and the UAV is landed safely back at its home base. The set of unsafe states contains states in which the UAV will be likely to crash, as discussed before. This safe-reachability objective will be satisfied in all executions traces that result from the obtained policy.

Other MDP Extensions

Probabilistic Goal MDPs acquire a policy that maximizes the probability of achieving a predetermined minimum reward (Xu and Mannor 2011). This approach is inherently different from a traditional MDP, which maximizes the expected reward itself. No additional guarantees are acquired. However, a variant making use of a CMDP can put a constraint on this probability, rather than maximizing it (Xu and Mannor 2011). This *Chance-Constrained* variant does obtain strong guarantees. Furthermore, ***Risk Sensitive POMDPs*** (RS-POMDPs) yield a policy that maximizes the probability of the cumulative cost being below a certain predefined threshold (Hou, Yeoh, and Varakantham 2016). Again, note the difference with minimizing the cost itself, as a traditional (PO)MDP does. RS-POMDPs do not provide additional guarantees.

Less interesting from a safety point of view is the concept of maximizing the probability of the agent achieving its goal (Lacerda, Parker, and Hawes 2015; Lacerda et al. 2019; Steinmetz, Hoffmann, and Buffet 2016), or better, putting a constraint on that probability (Steinmetz, Hoffmann, and Buffet 2016; Lacaze-Labadie, Lourdeaux, and Sallak 2017). From a liveness point of view, however, such guarantees are desirable. Note, when unsafe behavior is only considered to correspond to reaching dead-ends[4], the achievement of a goal can provide an indirect additional safety value. When the agent reaches a goal, no unsafe state has been visited before doing so.

[4]Dead-ends are considered states from which no recovery is possible, such as a fatal crash.

Simple Temporal Network

Another kind of framework we include in this survey, next to MDPs and their extensions, are *Simple Temporal Networks*[5] (STNs) (Dechter, Meiri, and Pearl 1991). The problem tackled by an STN is by nature different from that tackled by an MDP. Whereas MDPs support planning, at the level of states and actions with probabilistic effects, STNs are used as a temporal scheduling tool. Non-deterministic influences on the agent and its behavior will affect the duration of an action (as we elaborate upon in the next section), rather than the state reached *after* the action. Compared to MDPs, this can be interpreted as planning on a higher abstraction level. Efficient algorithms for solving STNs have been proposed (Xu and Choueiry 2003; Planken, de Weerdt, and van der Krogt 2008).

For an STN, a set of time-point variables is considered, along with a set of constraints on these variables. From a planning point of view, a time-point variable can be used to indicate the start point of an action or the occurrence of any relevant event in general. The constraints limit the amount of time that elapses between such events. If two events represent the start and end of an action, a constraint can express the duration possible for this action. As a result, depending on the application at hand, STNs can be used as a planning tool, taking a temporal approach.

Formally, an STN is defined as a tuple $\langle T, C \rangle$, where T represents the set of time-point variables $\{t_0, t_1, t_2, \ldots, t_n\}$, and C is the set of temporal constraints of the form $t_i - t_j \leq c_{ij}$. The solution of this STN consists of a value assignment for each of the variables $\{t_1 = s_1, t_2 = s_2, \ldots, t_n = s_n\}, s_i \in \mathbb{R}$, obeying all constraints. Further, t_0 is used to represent a fixed, arbitrarily chosen reference point in time.

Since an STN is inherently modeled using temporal difference constraints, safety constraints of this format can be added without additional effort. Stating that event a and b should occur within the same n time frame can be expressed as $|t_a - t_b| \leq n$. Stating that event c should occur before event d, corresponds to $t_d - t_c > 0$.

The UAV application at hand is approached as a temporal scheduling problem, rather than a planning problem. We obtain a temporal plan, describing which action to execute, at which point in time, instead of a state-based policy. Whereas for an MDP, the state of the UAV and the influence of the actions thereon form essential input for the planning problem, we now only have temporal events and constraints at our disposal. We have to represent the actions of the UAV using temporal events. Each action a has a duration, which falls within a corresponding interval $T_a = [p, q]$. The starting point $t_{a,begin}$ and endpoint $t_{a,end}$ of the action are expressed as temporal variables. The duration is expressed as a constraint $p \leq t_{a,end} - t_{a,begin} \leq q$. For example, the action *TakeOff* can take between 1 and 10 seconds to execute:

$$1s \leq t_{TakeOff,end} - t_{TakeOff,begin} \leq 10s.$$

[5]In the literature, the terms "Simple Temporal Problem" (STP) and "Simple Temporal Network" (STN), that is, the graph representation of the former, are used interchangeably. We will, consistent with the majority of the literature, commit to the latter.

Navigating between two locations can take more time:

$$120s \leq t_{MoveFromAtoB,end} - t_{MoveFromAtoB,begin} \leq 600s.$$

The time interval in which inspection is desired can be constrained as well. For example, the time for inspection can be determined by the presence of a moving entity:

$$10{:}10AM \leq t_{inspection,begin} \leq 10{:}15AM.$$

After defining a reference time point t_0, for example, the time of deployment of the UAV, the STN is defined completely. A solution of an STN instantiates all the temporal variables. This solution provides information regarding *when* the UAV has to execute the different actions, by providing the timestamp at which the actions start and end. A temporal plan is achieved.

Any safety constraint that was taken into account in the model will be satisfied as well. For example, the constraint

$$t_{TakeOff,begin} - t_{PreFlightCheck,end} \geq 0$$

ensures that the $PreFlightCheck$ is always completed before $TakeOff$ is executed.

Finally, an STN can be extended with observation nodes (Tsamardinos, Vidal, and Pollack 2003) and decision nodes (Cairo et al. 2017) to add conditional functionality and decisions to the framework, resulting in Conditional STNs (with Decisions) (CSTN(D)s). Depending on the concrete decisions at hand, a CSTND could provide additional guarantees. Such guarantees are, however, entirely application dependent. We do not look further into them.

STN with Uncertainty

In practice, the duration of actions and the time intervals between related events are often not controllable by the agent. An *STN with Uncertainty* (STNU) elaborates on this reality (Vidal and Fargier 1999; Morris and Muscettola 2000; Cimatti et al. 2014). The durations are still said to be bounded by a set interval, yet not controllable. A new kind of constraint is added, called a contingent constraint. Contingent constraints take into account that the exact duration of an action, or more generally, the time between two events, is beyond the control of the planning entity.

Formally, an STNU can be defined as a tuple $\langle T, C, U \rangle$, where $\langle T, C \rangle$ is interpreted as a traditional STN, and U denotes the set of contingent constraints. A contingent constraint has the same form as a traditional constraint but depends on an uncontrollable variable.

Using a traditional STN, we seemingly were able to obtain a proper solution for the scheduling problem of the UAV. However, an STN indeed assumes that the duration of the actions is controllable. The scheduler is able to set both the start and end timestamps of the actions. For example, the UAV's travel time between two locations can be set to a well-chosen value in order to meet the other constraints. In reality, this travel time is determined by external effects such as wind and the payload of the UAV. The exact duration of the action is not controllable by the autopilot. An STNU takes this uncontrollable nature of the duration into account. The provided intervals no longer describe a duration range from which the scheduler can choose, but rather provide bounds for the uncontrollable duration.

A relevant property of STNUs is their *controllability* (Vidal and Fargier 1999; Morris and Muscettola 2000; 2005; Morris 2006; Cimatti et al. 2014). An STNU is said to be *weakly controllable* if a solution exists that satisfies all constraints, given upfront how the uncontrollable durations turn out. More useful in practical applications, an STNU is said to be *strongly controllable* if a solution exists that satisfies all constraints, regardless of how the uncontrollable durations could turn out. In dynamic planning applications, the STNU can be said to be *dynamically controllable* if a solution exists that satisfies all (future) constraints, given how the past durations turned out. If such solutions are addressed, we consider the resulting temporal plan to be safe. Otherwise, at least one of the temporal constraints could get violated. The UAV might fail to execute the flight correctly and end up in an unsafe state.

STNUs, compared to STNs, achieve more realistic modeling of the system's behavior, while they satisfy the same kinds of temporal constraints.

Conditions and decisions can be added to STNUs as well (Hunsberger, Posenato, and Combi 2012; Combi, Hunsberger, and Posenato 2013; Zavatteri and Viganò 2019). Furthermore, Conditional STNUs can be extended to take into account runtime resource constraints (Combi et al. 2019).

Probabilistic STN

When applying an STNU, it can be troublesome to model a fixed, non-trivial interval for the time required by an agent to execute an action. For example, a UAV traveling between two locations could take more time than expected. There is always a non-zero chance for the UAV to require more time to complete the action, as a result of external influences. Furthermore, some values in the interval might be more likely than others. In a *Probabilistic STN* (PSTN), the uncontrollable events are modeled with probability distributions (Tsamardinos 2002), rather than relying on finite, uniform intervals for the durations. Note, as durations are no longer bound, compliance with the temporal difference constraints can not be guaranteed. A PSTN can only maximize the probability of such correct execution.

Formally, a PSTN is defined similarly to an STNU, except for the set of contingent constraints, which have to make room for a set of conditional probability density functions. The time required by the UAV to travel between two locations can now be modeled more correctly. The timestamp $t_{MoveFromAtoB,end}$ is modeled using a normal distribution, conditional to $t_{MoveFromAtoB,begin}$. As a result, the duration of the travel action is normally distributed. Sufficient flight data is required to estimate an accurate probability distribution.

A combination of the two frameworks that address uncertainty (STNUs and PSTNs), introduced as PSTNUs, has been investigated as well (Santana et al. 2016).

Chance Constrained PSTN

Lastly, we elaborate on *Chance Constrained PSTNs* (CC-PSTNs), which build further on PSTNs. A CC-PSTN bounds the risk of temporal inconsistencies while a utility objective is maximized (Fang, Yu, and Williams 2014). Hence, as their name implies, Chance Constrained PSTNs are semantically related to Chance Constrained (PO)MDPs, which bound the probability of exhibiting unsafe behavior while maximizing the expected reward. CC-PSTNs were proposed to address the over-conservatism that occurs in methods that solely try to minimize the risk, such as traditional PSTNs.

Formally, CC-PSTNs are defined similarly to PSTNs, except for the uncontrollable durations. Rather than having uncontrollable time points, conditioned on activating time points, the uncontrollable durations are modeled explicitly. As a result, the user can also define constraints between uncontrollable events, resulting in a more expressive framework. Furthermore, an upper bound $\delta \in [0, 1]$ on the risk of failure, that is, the probability of violating at least one constraint, is added, as well as an objective function V, which is a function of the assigned time-point variables.

A computationally more efficient variant exists, which deals with multiple chance constraints defined over subsets of the entire plan (Wang and Williams 2015). This variant does not maximize an objective function. The main goal is to satisfy all posed constraints.

A straightforward application of a utility objective is planning the UAV to finish its inspection task as fast as possible. The objective function, which is to be minimized, is naturally set to the timestamp corresponding to the completion of the task $V = t_{finish}$. At the same time, the chance of experiencing temporal inconsistencies is kept below some predefined threshold. Note, this constraint can, to some extent, be interpreted as a bound on the probability of the UAV showing unsafe behavior. If during execution, no inconsistencies occur, the UAV behaves concordant the intended model, which we assume to be safe. However, when inconsistencies do occur, not all constraints are satisfied, and the execution might deviate from the intended model. Safe behavior can not be guaranteed.

Conclusion

Most frameworks present in the literature are based on Markov Decision Processes, or to a smaller extent, on Simple Temporal Networks. Despite our followed methodology being quite extensive, we want to emphasize that our summary might not be complete. Nonetheless, the overview indicates the popularity of using MDPs for probabilistic planning and STNs for temporal scheduling.

Making use of MDP based frameworks, guarantees can be provided regarding: the use of resources (*C(PO)MDP*); probabilities on visited states (*CC-(PO)MDP*), if desired simultaneously for reachability and safety (*safe-reachability objectives*); and probabilities on entire execution paths (*PC-MDP*) of the system at hand. Using STNs, guarantees are either related to temporal constraints (*STN*) or the chance of failure (*CC-PSTN*). An overview is present in Table 1. Note that the accuracy and correctness of the achieved guarantees stands or falls with the accuracy and correctness of the used algorithms and data.

Future research should look into the extremes of what can be guaranteed in practical applications. In our study, we came across different MDP extensions that deal with constraints or provide explicit guarantees. Further, we encountered Robust (PO)MDPs, frameworks addressing the uncertainty regarding transition probabilities, which arises in practical applications. Future research should focus on a combination of both. Being able to provide hard guarantees for practical systems dealing with this uncertainty would be desirable. From a scheduling point of view, (CC-)PSTNs already deal with such uncertainties to some extent, better than STNUs.

Table 1: A high-level overview of the covered approaches, and the concepts regarding which they provide guarantees

Framework	**Provided Guarantees**
(PO)MDP & Robust (PO)MDP	∅
C(PO)MDP	Resource usage and costs
CC-(PO)MDP	Probabilities on events
PC-MDP	Probabilities on paths
POMDP with safe-reachability objectives	Probabilities on reaching unsafe & goal states
STN & STNU	Temporal differences
PSTN	∅
CC-PSTN	Probabilities on failure

Acknowledgments

This research is partially funded by the Research Fund KU Leuven. We thank the anonymous reviewers for the insightful pointers they provided.

References

Aberdeen, D. 2003. A (revised) survey of approximate methods for solving partially observable markov decision processes. Technical report, National ICT Australia.

Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC Press.

Ayton, B. J., and Williams, B. C. 2018. Vulcan: A monte carlo algorithm for large chance constrained mdps with risk bounding functions.

Baier, C.; Hermanns, H.; and Katoen, J.-P. 2019. The 10,000 facets of mdp model checking. In *Computing and Software Science*, 420–451.

Cairo, M.; Combi, C.; Comin, C.; Hunsberger, L.; Posenato, R.; Rizzi, R.; and Zavatteri, M. 2017. Incorporating decision nodes into conditional simple temporal networks. In *24th International Symposium on Temporal Representation and Reasoning (TIME)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Cassandra, A. R. 1998a. A survey of pomdp applications. In *Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes*, 17–24.

Cassandra, A. R. 1998b. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. Ph.D. Dissertation, Brown University.

Cimatti, A.; Hunsberger, L.; Micheli, A.; and Roveri, M. 2014. Using timed game automata to synthesize execution strategies for simple temporal networks with uncertainty. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, 2242–2249.

Combi, C.; Posenato, R.; Viganò, L.; and Zavatteri, M. 2019. Conditional simple temporal networks with uncertainty and resources. *Journal of Artificial Intelligence Research* 64(1):931–985.

Combi, C.; Hunsberger, L.; and Posenato, R. 2013. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. *Evaluation* 1:1.

Dadkhah, N., and Mettler, B. 2012. Survey of motion planning literature in the presence of uncertainty: Considerations for uav guidance. *Journal of Intelligent & Robotic Systems* 65(1):233–246.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial intelligence* 49(1-3):61–95.

Ding, X. C.; Pinto, A.; and Surana, A. 2013. Strategic planning under uncertainties via constrained markov decision processes. In *IEEE International Conference on Robotics and Automation*, 4568–4575.

El Chamie, M.; Yu, Y.; Açıkmeşe, B.; and Ono, M. 2018. Controlled markov processes with safety state constraints. *IEEE Transactions on Automatic Control* 64(3):1003–1018.

Fang, C.; Yu, P.; and Williams, B. C. 2014. Chance-constrained probabilistic simple temporal problems. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, 2264–2270.

Hou, P.; Yeoh, W.; and Varakantham, P. 2016. Solving risk-sensitive pomdps with and without cost observations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 3138–3144.

Hunsberger, L.; Posenato, R.; and Combi, C. 2012. The dynamic controllability of conditional stns with uncertainty. *arXiv preprint arXiv:1212.2005*.

Isom, J. D.; Meyn, S. P.; and Braatz, R. D. 2008. Piecewise linear dynamic programming for constrained pomdps. In *Proceedings of the 23rd national conference on Artificial intelligence*, volume 1, 291–296.

Katoen, J.-P. 2016. The probabilistic model checking landscape. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, 31–45. Association for Computing Machinery.

Keith, A. J., and Ahner, D. K. 2019. A survey of decision making and optimization under uncertainty. *Annals of Operations Research* 1–35.

Khonji, M.; Jasour, A.; and Williams, B. 2019. Approximability of constant-horizon constrained pomdp. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, volume 7 of *IJCAI'19*, 5583–5590.

Kim, D.; Lee, J.; Kim, K.-E.; and Poupart, P. 2011. Point-based value iteration for constrained pomdps. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, IJCAI'11, 1968–1974.

Kitchenham, B., and Charters, S. 2007. Guidelines for performing systematic literature reviews in software engineering.

Kwiatkowska, M.; Norman, G.; and Parker, D. 2002. Prism: Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 200–204.

Lacaze-Labadie, R.; Lourdeaux, D.; and Sallak, M. 2017. Heuristic approach to guarantee safe solutions in probabilistic planning. In *IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 579–585.

Lacerda, B.; Faruq, F.; Parker, D.; and Hawes, N. 2019. Probabilistic planning with formal performance guarantees for mobile service robots. *The International Journal of Robotics Research* 38(9):1098–1123.

Lacerda, B.; Parker, D.; and Hawes, N. 2015. Optimal policy generation for partially satisfiable co-safe ltl specifications. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, 1587–1593.

Lee, J.; Kim, G.-H.; Poupart, P.; and Kim, K.-E. 2018. Monte-carlo tree search for constrained pomdps. In *Advances in Neural Information Processing Systems*, 7923–7932.

Lovejoy, W. S. 1991. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research* 28(1):47–65.

Maeda, S.-i.; Watahiki, H.; Okada, S.; and Koyama, M. 2019. Reconnaissance and planning algorithm for constrained mdp. *ArXiv* 1909.09540.

Monahan, G. E. 1982. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management science* 28(1):1–16.

Morris, P., and Muscettola, N. 2000. Execution of temporal plans with uncertainty. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth conference on Innovative Applications of Artificial Intelligence*, 491–496.

Morris, P. H., and Muscettola, N. 2005. Temporal dynamic controllability revisited. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, 1193–1198.

Morris, P. 2006. A structural characterization of temporal dynamic controllability. In *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming*, CP'06, 375–389.

Newaz, A. A. R.; Chaudhuri, S.; and Kavraki, L. E. 2019. Monte-carlo policy synthesis in pomdps with quantitative and qualitative objectives. In *RSS 2019*.

Nilim, A., and El Ghaoui, L. 2005. Robust control of markov decision processes with uncertain transition matrices. *Operations Research* 53(5):780–798.

Osogami, T. 2015. Robust partially observable markov decision process. In *International Conference on Machine Learning*, 106–115.

Planken, L.; de Weerdt, M.; and van der Krogt, R. 2008. P3c: A new algorithm for the simple temporal problem. In *Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'08, 256–263.

Poupart, P.; Malhotra, A.; Pei, P.; Kim, K.-E.; Goh, B.; and Bowling, M. 2015. Approximate linear programming for constrained partially observable markov decision processes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 3342–3348.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Santana, P.; Vaquero, T.; Toledo, C.; Wang, A.; Fang, C.; and Williams, B. 2016. Paris: A polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling*, ICAPS'16, 267–275.

Santana, P.; Thiébaux, S.; and Williams, B. 2016. Rao*: An algorithm for chance-constrained pomdp's. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 3308–3314.

Sprauel, J.; Kolobov, A.; and Teichteil-Königsbuch, F. 2014. Saturated path-constrained mdp: Planning under uncertainty and deterministic model-checking constraints. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, 2367–2373.

Steinmetz, M.; Hoffmann, J.; and Buffet, O. 2016. Goal probability analysis in probabilistic planning: Exploring and enhancing the state of the art. *Journal of Artificial Intelligence Research* 57:229–271.

Tamar, A.; Mannor, S.; and Xu, H. 2014. Scaling up robust mdps using function approximation. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, ICML'14, II–181–II–189.

Teichteil-Königsbuch, F. 2012. Path-constrained markov decision processes: Bridging the gap between probabilistic model-checking and decision-theoretic planning. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12, 744–749.

Tsamardinos, I.; Vidal, T.; and Pollack, M. E. 2003. Ctp: A new constraint-based formalism for conditional, temporal planning. *Constraints* 8(4):365–388.

Tsamardinos, I. 2002. A probabilistic approach to robust execution of temporal plans with uncertainty. In *Hellenic Conference on Artificial Intelligence*, 97–108.

Undurti, A., and How, J. P. 2010. An online algorithm for constrained pomdps. In *IEEE International Conference on Robotics and Automation*, 3966–3973.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence* 11(1):23–45.

Walraven, E., and Spaan, M. 2018. Column generation algorithms for constrained pomdps. *Journal of artificial intelligence research* 62:489–533.

Wang, A. J., and Williams, B. C. 2015. Chance-constrained scheduling via conflict-directed risk allocation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Wang, Y.; Chaudhuri, S.; and Kavraki, L. E. 2018a. Bounded policy synthesis for pomdps with safe-reachability objectives. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, 238–246. International Foundation for Autonomous Agents and Multiagent Systems.

Wang, Y.; Chaudhuri, S.; and Kavraki, L. E. 2018b. Online partial conditional plan synthesis for pomdps with safe-reachability objectives.

Wang, Y.; Chaudhuri, S.; and Kavraki, L. E. 2019. Point-based policy synthesis for pomdps with boolean and quantitative objectives. *IEEE Robotics and Automation Letters* 4(2):1860–1867.

White, D. J. 1985. Real applications of markov decision processes. *Interfaces* 15(6):73–83.

White, D. J. 1988. Further real applications of markov decision processes. *Interfaces* 18(5):55–61.

White, C. C. 1991. A survey of solution techniques for the partially observed markov decision process. *Annals of Operations Research* 32(1):215–230.

White, D. J. 1993. A survey of applications of markov decision processes. *Journal of the operational research society* 44(11):1073–1096.

Wiesemann, W.; Kuhn, D.; and Rustem, B. 2013. Robust markov decision processes. *Mathematics of Operations Research* 38(1):153–183.

Wohlin, C. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 1–10.

Xu, L., and Choueiry, B. Y. 2003. A new efficient algorithm for solving the simple temporal problem. *10th International Symposium on Temporal Representation and Reasoning, and Fourth International Conference on Temporal Logic*. 210–220.

Xu, H., and Mannor, S. 2010. Distributionally robust markov decision processes. In Lafferty, J. D.; Williams, C. K. I.; Shawe-Taylor, J.; Zemel, R. S.; and Culotta, A., eds., *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc. 2505–2513.

Xu, H., and Mannor, S. 2011. Probabilistic goal markov decision processes. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2046–2052.

Zavatteri, M., and Viganò, L. 2019. Conditional simple temporal networks with uncertainty and decisions. *Theoretical Computer Science* 797:77–101. Temporal Representation and Reasoning (TIME 2017).