A Case for Robust AI in Robotics

Shashank Pathak[1], Luca Pulina[2], and Armando Tacchella[3]

[1] iCub Facility — Istituto Italiano di Tecnologia
shashank.pathak@iit.it
[2] POLCOMING — Universitá degli Studi di Sassari
lpulina@uniss.it
[3] DIBRIS — Università degli Studi di Genova
armando.tacchella@unige.it

Abstract. Researchers envision a world wherein robots are free to interact with the external environment, thereby including human beings, other living creatures, robots and a variety of inanimate objects. It is always tacitly assumed that interactions will be smooth, i.e., they will fulfill several desirable properties ranging from safety to appropriateness. We posit that a reasonable mathematical model to frame such vision is that of Markov decision processes, and that ensuring smooth interactions amounts to endow robots with control policies that are provably compliant with side conditions expressed in probabilistic temporal logic.

1 Context, Motivation, Objectives

A *(stationary, discrete time) Markov decision process* is a collection of objects $\mathcal{M} = \{T, S, A_s, p(\cdot|s,a), r(s,a)\}$ where $T \subseteq \mathbb{N}^+$ is a set of time points over an *infinite horizon*; S is a finite, time-independent set of *states* which the system can occupy at each $t \in T$; A_s is a finite time-independent set of *actions* allowable in some state $s \in S$ where $A = \bigcup_{s \in S} A_s$ collects all actions; $p(j|s,a)$ is a (non-negative) *transition probability function* such that $\sum_{j \in S} p(j|s,a) = 1$, denoting the probability that the system is in state j at time $t+1$ when action $a \in A_s$ is accomplished in state s at time t; $r(s,a)$ is a function $|r(s,a)| \leq M$ for some finite $M \in \mathbb{R}$; r is defined for all states and actions, and it denotes the value of the *reward* received when executing an action in a state. A *labelling function* $L : S \to AP$ can be used to express additional properties of states by labelling them with atomic propositions from a set AP.

A *(stationary) policy* (also *scheduler* or *controller*) specifies a procedure for action selection in each state of a Markov decision process. A policy π is *deterministic* if it is a function $\pi : S \to A_s$, and is *stochastic* if it is a function $\pi : S \times A \to [0,1]$ with $\pi(s,a) = 0$ for all $a \notin A_s$ and $s \in S$, and $\sum_{a \in A_s} \pi(s,a) = 1$ for all $s \in S$. The well known *Markov decision problem (MDP)* amounts to compute a policy which fulfils some optimally criterion within a Markow decision process $\mathcal{M}$. Usually the criterion is linked to the rewards that can be accrued when acting in $\mathcal{M}$ according to a policy π starting from some state s, i.e., the *value* $V_\pi(s)$ of state s under policy π. A policy π is better than

or equal to π' ($\pi \geq \pi'$) exactly when $V_\pi(s) \geq V_{\pi'}(s)$ for all $s \in S$. Given some reasonable definition of value $V.(s)$ for all $s \in S$, solving an MDP amounts to find π^* such that $\pi^* \geq \pi$ for all possible π — more about MDPs and related decision problems can be found in [1].

From a modeling point of view, Markov decision processes and associated optimization problems, capture a broad set of approaches to the analysis and synthesis of intelligent behavior for autonomous agents which can be put to good use in the field of Robotics. For instance, in [2] it is argued that many problems of AI planning under uncertainty can be modeled as MDPs. In the learning community — see [3] for a recent persective — reinforcement learning (RL) is viewed as one of the key techniques to synthesize intelligent behavior for interactive agents, and the mathematical underpinning of RL is also given by Markov decision processes. Even closer to field robotics, the area of optimal control has a long tradition of leveraging MDPs for problems involving sequential decision making under uncertainty — see, e.g., [4]. Given the widespread adoption of MDPs in AI and related (sub)fields, proposing techniques to achieve robustness of autonomous agents based on Markov decision processes is bound to have a broad impact. We believe that Robotics might benefit the most from robust AI techniques, since robots ought to be functional but also dependable, and the trade-off between these two aspects need to be fully understood and explored.

Our key proposition is to extend the modeling framework of MDPs to one that includes explicit side conditions expressed in *probabilistic computation tree logic (PCTL)*. The syntax of PCTL is defined considering the set Σ of *state formulas*, and the set Π of *path formulas*. Given a set of *atomic propositions* AP, Σ is defined inductively as: (i) if $\varphi \in AP$ then $\varphi \in \Sigma$; (ii) $\top \in \Sigma$; if $\alpha, \beta \in \Sigma$ then also $\alpha \wedge \beta \in \Sigma$ and $\neg\alpha \in \Sigma$; and (iii) $\mathcal{P}_{\bowtie p}[\psi] \in \Sigma$ where $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0, 1]$ and $\psi \in \Pi$, where $\mathcal{P}_{\bowtie p}[\psi]$ is the *probabilistic path operator*. The set Π contains exactly the expressions of type $X\alpha$ (*next*), $\alpha \mathcal{U}^{\leq k} \beta$ (*bounded until*) and $\alpha \mathcal{U} \beta$ (*until*) where $\alpha, \beta \in \Sigma$ and $k \in \mathbb{N}$ — more on PCTL can be found in [5]. Given an MDP $\mathcal{M}$, a definition of value $V.(s)$ for all states of $\mathcal{M}$, and a PCTL formula φ, the *Markov decision problem with probabilistic side condition* (MDPP) can be defined as the problem of finding π^* such that $\pi* \geq \pi$ for all policies π and $\mathcal{D}_{\mathcal{M},\pi^*} \models \varphi$, i.e., φ is always satisfied in the *discrete-time Markov chain (DTMC)* $\mathcal{D}_{\mathcal{M},\pi^*}$ corresponding to the combination of $\mathcal{M}$ and π^* — see [5] for details about DTMCS and PCTL semantics, and [6] for details about combining MDPs with policies to yield DTMCs.

2 State of the art

We can distinguish current approaches to MDPP into two broad categories, the first one oblivious of formal techniques, and the second one deeply rooted in formal verification and reasoning. In the former category we can list *multi-objective reinforcement learning* [7] — with Geibel and Wysotski's approach as a special case [8]. The main idea of these approaches is to encode the requirement expressed by φ in the value function $V.(s)$. Both approaches do not require

knowledge of $p(\cdot|s,a)$ because the relevant information is learned by interacting with (a physical realisation of) $\mathcal{M}$. In this way, solving the Markov decision problem yields a policy that most probably also satisfies φ, although formal guarantees are not provided. Still in the first category, we can consider Gillula and Tomlin's *safe online learning* [9] which, albeit restricted to safety properties, provides a mathematically precise way to combine side conditions to the online solution of an MDP via reinforcement learning (RL) [10]. *Decision-theoretic planning* [2] — a.k.a. *indirect RL* [3] *model-based RL* [10] or *controller synthesis on MDPs* [11] — is another approach in which MDPP can be formalised by taking into account both the elements related optimisation of V_π and the side conditions expressed by φ. In this case the solution is precise, but it requires the knowledge of $p(\cdot|s,a)$ in advance and the policy synthesised is always deterministic. Yet another way to incorporate side conditions is to consider the overall model as a *constrained MDP* [12], where one type of cost is to be optimised while keeping the other types of costs within a bound. As before, the approach requires knowledge of the model. All the methods listed above do not cover the cases in which both $p(\cdot|s,a)$ is unknown *and* the optimal policy is stochastic. Also, the learning-based ones have an unclear relation between the logic specification of φ and functional specification of rewards.

Another set of approaches to MDPP is the one based on formal methods which can be used to solve parts of the MDPP problem. For a known model $\mathcal{M}$ and a given policy π, *probabilistic model checking* supported by efficient tools like `PRISM` [13] or `MRMC` [14] can be applied to check whether the side condition φ is satisfied by the controller policy π. If a policy π does not satisfy the PCTL property φ, *model repair* [15] can be applied to modify the policy such that φ becomes true. First the DTMC model resulting from the MDP model under the given policy π is parameterised, using linear combinations of real-valued parameters in the transition probabilities, where the parameter domains define the allowed areas for the repair. Additionally, a *cost-function* over the parameters can be given. Now model repair can be applied to find (if it exists) a parameter valuation within the parameter domains which on the one hand induces the satisfaction of the property φ and on the other hand minimises the value of the cost-function, i.e., changing the transition probabilities and thereby *repairing* the DTMC with minimal costs. Unfortunately, this approach needs non-linear optimisation and therefore it does not scale for larger models. An approach that we recently proposed with other researchers uses a *greedy repair* algorithm [6]. Instead of global optimisation, it uses local repair steps iteratively. Though it needs to iteratively invoke probabilistic model checking, this approach scales well also for large models. However, it can incorporate rewards and values $V_\pi(s)$ only heuristically in a quite restricted manner.

3 Towards Robust AI in Robotics: a Challenge

Potentially, an AI agent embodied in a robot may face a wide variety of scenarios each characterized by different safety constraints and learning objectives. To

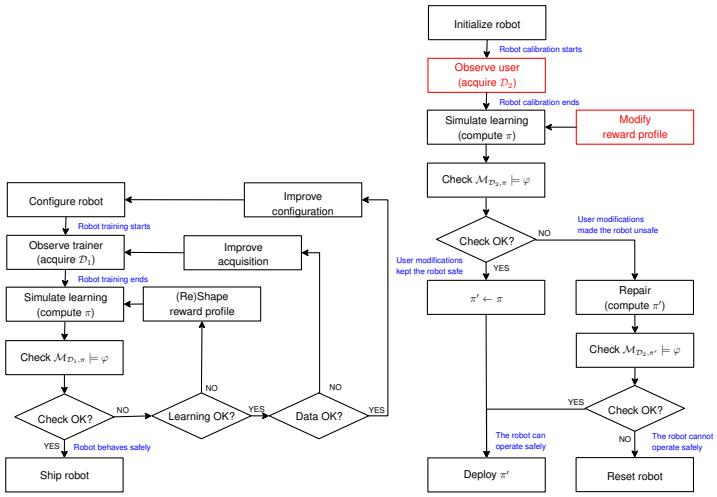

Fig. 1. A challenging MDPP setting. The boxes in red denote activities in which the end user can calibrate the robot or modify its behavior.

obtain a quantitative assessment about our capability to attack the MDPP problem it is useful to focus on a specific scenario which contains all the basic ingredients found in more complex ones, yet it is significant and amenable of a relatively simple implementation. In particular, the case of a single robot interacting with a single human across a common workspace is considered. It is assumed that the robot observes the human while she is accomplishing a given task, which at some point, requires the robot to chip in and, e.g., finalize the task alone or help the human to do so. The task must be learned by the robot, but RL is run offline in a simulator to avoid risk of injuries to the human during the trial-and-error process which characterizes RL. As shown in Figure 1 two different flows of activities are considered. The first one — Figure 1 (left) — is thought to happen at the end of the production stage (factory), where the robot is configured, trained and checked by experts to accomplish a given task. The second one — Figure 1 (right) — is thought to happen during the deployment stage (e.g., household), where the user is allowed to (*i*) *calibrate* the robot, i.e., adapt its behavior to the contingencies of the environment to be found at the user's place, and to (*ii*) *modify* the robot's behavior, i.e., customize the robot according to specific preferences.

We believe that the current state of the art is unable to solve the MDPP problem in a totally satisfactorily way in cases like the one exemplified in Figure 1. However there are strong potentials in *combining* efficient but potentially imprecise engineering approaches with precise but potentially inefficient formal methods. For example, RL-based methods are well established for MDP controller synthesis, where the optimality criteria are encoded by rewards and the value function $V_\pi(s)$. To assure that the controller learned by RL is safe, during RL learning we could use probabilistic model checking. If the current (not

yet necessarily optimal) controller turns out to be unsafe, we could repair the controller. Additionally, it might also be necessary to modify the rewards and/or the value function to direct RL towards safe solutions. This could be done, for example, based on probabilistic counterexamples [16]. In contrast with reward-shaping approaches that guarantee invariance of the optimal policy learned [17], such a reward or value function repair aims to obtain sub-optimal but safe policy.

References

1. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons (1994)
2. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research **11**(1) (1999) 94
3. Wiering, M., Van Otterlo, M.: Reinforcement learning. In: Adaptation, Learning, and Optimization. Volume 12. Springer (2012)
4. Bertsekas, D.P., Bertsekas, D.P., Bertsekas, D.P., Bertsekas, D.P.: Dynamic programming and optimal control. Athena Scientific Belmont, MA (1995)
5. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
6. Pathak, S., Ábrahám, E., Jansen, N., Tacchella, A., Katoen, J.: A greedy approach for the efficient repair of stochastic models. In: Proc. of NFM'15. Volume 9058 of LNCS, Springer (2015) 295–309
7. Natarajan, S., Tadepalli, P.: Dynamic preferences in multi-criteria reinforcement learning. In: Proc. of ICML'05, ACM (2005) 601–608
8. Geibel, P., Wysotzki, F.: Risk-Sensitive Reinforcement Learning Applied to Control under Constraints. Journal of Artificial Intelligence Research **24** (2005) 81–108
9. Gillula, J.H., Tomlin, C.J.: Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor. In: Proc. of ICRA'12, IEEE (2012) 2723–2730
10. Sutton, R., Barto, A.: Reinforcement Learning – An Introduction. MIT Press (1998)
11. Dräger, K., Forejt, V., Kwiatkowska, M., Parker, D., Ujma, M.: Permissive controller synthesis for probabilistic systems. In: Proc. of TACAS'14. Springer (2014) 531–546
12. Altman, E.: Constrained Markov decision processes. Volume 7. CRC Press (1999)
13. Kwiatkowska, M.Z., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Proc. of CAV. Volume 6806 of LNCS, Springer (2011) 585–591
14. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. Performance Evaluation **68**(2) (2011) 90–104
15. Bartocci, E., Grosu, R., Katsaros, P., Ramakrishnan, C., Smolka, S.A.: Model repair for probabilistic systems. In: Proc. of TACAS. Volume 6605 of LNCS, Springer (2011) 326–340
16. Ábrahám, E., Becker, B., Dehnert, C., Jansen, N., Katoen, J., Wimmer, R.: Counterexample generation for discrete-time Markov models: An introductory survey. In: Proc. of SFM. Volume 8483 of LNCS, Springer (2014) 65–121
17. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: ICML. Volume 99 (1999) 278–287