The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)

Augmenting Markov Decision Processes with Advising

Loïs Vanhée,* **Laurent Jeanpierre,*** **Abdel-Illah Mouaddib***
GREYC, Université de Caen, France

Abstract

This paper introduces Advice-MDPs, an expansion of Markov Decision Processes for generating policies that take into consideration advising on the desirability, undesirability, and prohibition of certain states and actions. Advice-MDPs enable the design of designing semi-autonomous systems (systems that require operator support for at least handling certain situations) that can efficiently handle unexpected complex environments. Operators, through advising, can augment the planning model for covering unexpected real-world irregularities. This advising can swiftly augment the degree of autonomy of the system, so it can work without subsequent human intervention.

This paper details the Advice-MDP formalism, a fast Advice-MDP resolution algorithm, and its applicability for real-world tasks, via the design of a professional-class semi-autonomous robot system ready to be deployed in a wide range of unexpected environments and capable of efficiently integrating operator advising.

Introduction

Markov Decision Processes (MDPs) are one of the most classic model for planning under uncertainty. However, MDPs are difficult to deploy for many real-world problems (e.g. robots in public space, military operations, disaster-recovery). These problems introduce irregularities that are difficult to foresee at design time (e.g. path-finding hardened by wind, water, glass, acid, armed threats). These irregularities cause discrepancies between the planning model and the real-world, thus entailing the generation of irrelevant plans and failures.

Mixed-Initiative Planning Systems (MIPS) aim to overcome these issues by providing means for human operators to improve plan generation through the validation of actions and plans, and by providing information that the system cannot access or infer otherwise (e.g. presence of threat) (Zilberstein 2015). However, operator support should be requested with care, as it burdens the operators, causing Operator-Workload (OW), which is costly.

*Contact author: lois.vanhee@unicaen.fr. Funded by the Agence Nationale de la Recherche and the Direction Générale de l'Armement (ANR-14-ASTR-0018).

Available MIPS suffer from three major limitations: 1) *OW costs are linear to system use* (e.g. a check for each generated plan), 2) *no flexibility on OW management* (operator support must be performed last minute, causing time pressure, frustration, and stress (Hart and Staveland 1988)), and 3) the *system requires an expensive training phase*. Plan-validation based approaches suffer from limitations 1 and 2, while learning-based approaches suffer from all three limitations. This paper aims to answer the following research question: *how to decrease OW costs and enable flexible OW management, with limited training phase?* The key novelty brought by this paper lies in covering environmental irregularities by providing means for the user to *augment the planning model* using (prescriptive) advising. This way, operator support is required only once for multiple plan generations (thus lowering OW costs), can be performed both *a priori* and on-demand (before and when plans are to be generated), and no expensive training is required for the system to be operational.

This paper proposes a new formalism for overcoming these issues, which we call *Advice-MDPs* (Advice Markov Decision Processes). Advice-MDPs are Markov Decision Processes (MDPs (Puterman 1994)) expanded with advising, for defining situationally-forbidden actions and augmenting state definition with an external information on whether it is desired, undesired, or forbidden. The various types of advice are interpreted following an order, from the hardest (forbidden) to the softest (desired): we aim to generate advice-compliant policies that first avoid performing forbidden situational actions, then optimize the compromise between reaching goals and avoiding forbidden states, and then optimize the compromise between reaching desirable states and avoiding undesirable states.

In addition to the formalism, this paper proposes a fast approximate algorithm for solving Advice-MDPs. This algorithm is based on a lexicographic multi-criteria value-iteration algorithm using Ordered Weighted Regret (Sigaud and Buffet 2010)

Finally, this paper demonstrates the relevance of Advice-MDPs for solving real-world problems, by deploying them for a professional-class application. Our industry partner seeks to deploy mobile-robots in unexpected and complex environments, for contexts such as supporting the evacuation of civilians in harmful situations, disaster recovery (Ram-

churn et al. 2015), and security missions (e.g. terrorism, patrol in sensitive sites).

The system, which we call the "Anywhere Deployment" Robot System (ADRS), should be extremely flexible i.e. capable of adapting to unexpected irregularities that can affect robot orientation (e.g. wind, water, sand, acidity, glass, unstable ground, threats), which go beyond the possibilities of classic fully-autonomous systems. Furthermore, operator availability is highly limited and unpredictable: while operators can generally watch and support the system for a few minutes after deployment, they have little time for actively training it and, suddenly, they may become overloaded due to arising threats. Illustrations of the implementation of the ADRS are presented in Figure 1 and Figure 2.

Figure 1: Two NERVA robots we experimented with, in the realistic arena set by our industry partners.

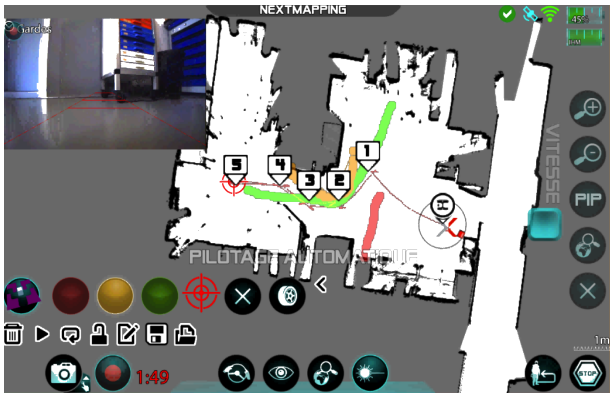


Figure 2: Operator interface: map, video stream. Lines on the map represent operator advising: desired (green), undesired (orange), and forbidden (red) advising. The target shape is the goal of the robot. Numbers are navigation waypoints.

Related Research

Mixed-Initiative Planning Systems (MIPSs): (Zilberstein 2015) define MIPSs as automated systems that collaboratively build plans with their operators. MIPS are acknowledged to be particularly useful when the planning model of the automated system is partial and when human judgment is needed to evaluate the plans generated based on the model (Zilberstein 2015).

Classic approaches for implementing MIPSs consists of an iterative loop where the system generates a plan and operators validate or refuse it, with a mechanism for justifying the rejection. If accepted, the plan is executed by the system. If refused, the system generates another plan, taking into consideration the reasons for the rejection of the first plan (e.g. the MAPGEN system, for planning Mars rover activities (Bresina et al. 2005)). The design of these systems is generally coupled with the design of interfaces for enabling operators to evaluate plans and argue on their rejection.

Advanced (MDP-based) MIPS approaches aim to reduce OW costs by reasoning about the operators' workload and capabilities and cost for requesting support from them. The HHP-MDP, HOP-POMDP, and OPOMDP formalisms integrate operators as a manageable (costly) resource of the planning domain, which can be used for acquiring observations and setting subgoals (Armstrong-Crews and Veloso 2007; Côté et al. 2012; Rosenthal and Veloso 2011). However, these approaches require from the system to be capable of evaluating the relevance of asking operator support, something which can be difficult to achieve.

MIPSs suffer from the limitations 1) and 2) presented in the introduction: OW costs are proportional to system use, as plans must be validated each time they are issued. Second, little OW flexibility is possible, as plans must be validated after goals are issued and before plans are executed. Operator support cannot be done *a priori* and delayed support generally entails neglect costs or even system failures (Crandall et al. 2005).

Advice-Sensitive Reinforcement Learning (ASRL): Advising has been applied for speeding up the learning process of Reinforcement Learning (Abel et al. 2017; Ng, Harada, and Russell 1999; Torrey et al. 2005). In ASRL, operators can advise learning agents by providing feedback on the desirability of (intended) agent actions, or by relating the task to similar ones.

ASRL faces the three limitations presented in the introduction (high OW costs, low OW management flexibility, expensive training phase). The training phase is particularly expensive in terms of OW costs, as operators are constantly involved during this period. Furthermore, reinforcement learning may be infeasible due to real-world constraints (e.g. the delay for being rewarded by a human is incompatible with a robot affected by world-dynamics such as sliding on ice). In addition, this form of advising only tweaks reward and transition functions, disregarding finer-grained advising that can introduce a richer semantic and behavioral indications for the system (e.g. a "bystanders may be encountered there" advice).

Background on Markov Decision Processes

Classic Markov Decision Processes: Markov Decision Processes (MDPs) are stochastic processes that are (partially) controlled by an agent (Sigaud and Buffet 2010). Formally, a MDP is represented by a tuple (S, A, T, R), where S is a set of states; A is a set of actions; $T : S \times A \times S \rightarrow$

$[0, 1]$ is a transition function, where $T(s, a, s')$ is the probability of reaching s' when playing a from s; and $R : S \to \mathbb{R}$ is the reward function, where $R(s)$ is the expected reward for reaching s.

Optimizing MDPs within a bounded horizon $h \in \mathbb{N}$ consists in finding a policy that maximizes the expected reward in using h actions. A policy $\pi : S \to A$ defines an action to be played for each state and Π is the set of all policies. The value of a policy π within a horizon h is $V_h^\pi : S \to \mathbb{R}$, where: $V_k^\pi(s) = R(s) + \sum_{s' \in S} T(s, \pi(s), s').V_{k-1}^\pi(s')$ if $k > 0$, else $V_0^\pi(s) = R(s)$. The set of optimal policies for the horizon h, represented by $\Pi_h^* \subseteq \Pi$ is the set of policies $\pi^* \in \Pi_h^*$ such that for all $s \in S$, $V_h^{\pi^*}(s) = \max_{\pi \in \Pi} V_h^\pi(s)$.

Multi-Objective MDPs: In Multi-Objective MDPs (MOMDPs) (Roijers, Vamplew, and Whiteson 2013), the agent controls a process that is related to not one, but a set of reward criteria (e.g. money, satisfaction, environment quality). The reward function of MOMDPs is represented by $R : S \to \mathbb{R}^k$. Optimizing MOMDPs is influenced by this multiplicity of criteria: the set of optimal policies is a Pareto-front of policies along $\mathbb{R}^k$ (e.g. two policies, one generating less money, but offering higher satisfaction, belong to the same Pareto-front). Often, MOMDPs are turned into MDPs using a formula for linearizing the MOMDP rewards (e.g. $R(s) = 0.5 \times R_{mon}(s) + 0.7 \times R_{sat}(s) + 2.3 \times R_{env}(s)$). However, doing so reduces the capability for expressing more elaborated reward profiles and constraints (e.g. the value of environment is minimal, as long as ten points are scored).

Formalizing Advice-MDPs

Advice-MDPs aim to enable operators to *augment the planning model with advice*, towards fitting the irregularities of the environment (like ASRL) with prescriptive *a priori* advising (like MIPSs). This way, Advice-MDPs cut down the OW costs of MIPS, by requiring operators to augment the planning model for including real-world irregularities once for all instead of once per use. Advice-MDPs offer higher OW management flexibility, by enabling advising *a priori* (once the environment is known, without needing goals to be set) instead of last-minute advising for MIPSs; Advice-MDPs require no training phase and advising can be achieved without executing plans, unlike ASRL approaches. Advice-MDPs enable for a large variety of advice types, which can introduce a helpful semantic context for advanced decision-making, unlike ASRL approaches that only enable to tweak mono-dimensional rewards.

Advice-MDPs expand classic MDPs, replacing reward function with advising. This paper considers the following five *advice types*: $\{\mathcal{G}, S_f, S_d, S_u, C_{\hat{\pi}}\}$, where $\mathcal{G} \subseteq S$ are goal states, to be reached by the system, $S_f \subseteq S$ are forbidden states, $S_d \subseteq S$ are desired states, $S_u \subseteq S$ are undesired states, and $C_{\hat{\pi}} \in f : S \to 2^A$ are forbidden actions to be avoided in certain states. In our ADRS application, S_d are safe and efficient zones, S_u are slow or inconvenient zones (e.g. sand, rock), and S_f are dangerous zones that should be avoided, unless inevitable (e.g. very poor floor, very close to enemy location), $C_{\hat{\pi}}$ are contextually forbidden actions (e.g. activating a visible sensor). The *relative importance* given to the various advice types follows a lexicographic order: first, match $C_{\hat{\pi}}$; then, best satisfy the tradeoff $\langle \mathcal{G}, S_f \rangle$; finally, best satisfy the trade-off $\langle S_d, S_u \rangle$. Formally, policies matching $C_{\hat{\pi}}$ are represented by:

$$\Pi_{C_{\hat{\pi}}} = \Pi \setminus \{\pi | \exists s \in S, \pi(s) \in C_{\hat{\pi}}(s)\}$$

Policies that best satisfy the tradeoff $\mathcal{G}, S_f$ belong to the Pareto-front of policies evaluated by $V^{\mathcal{G}, S_f, \pi} = (V^{\mathcal{G}, \pi}, -V^{S_f, \pi})$, where: $V^{S', \pi}(s) = \mathcal{N}_{vis}(S'|s, \pi)$, for which $\mathcal{N}_{vis}(S'|s, \pi)$ is the expected number of times S' is expected to be visited from s following π. Policies that best satisfy the tradeoff S_d, S_u belong to the Pareto-front of policies evaluated by $V^{S_d, S_u, \pi} = (V^{S_d, \pi}, -V^{S_u, \pi})$. Note that this representation for advising is selected for its simplicity, genericity, and ease to connect to user interfaces. However, more advice types can be integrated in the model.

Efficiently Solving Advice-MDPs

Algorithm 1: Fast Advice-MDP policy computer

Input : $S, A, T, \mathcal{G}, S_f, S_d, S_u, C_{\hat{\pi}}, H$
Output : MDP Value Function V'
$T \leftarrow T \setminus \{(s, a, s') | \forall s, s', a \in C_{\hat{\pi}}(s)\}$
$V^{\pi,0} \leftarrow 0$
$\forall s \in S, R'(s) \leftarrow \begin{cases} (1,0,0,0) & \text{if } s \in \mathcal{G} \\ (0,-1,0,0) & \text{if } s \in S_f \\ (0,0,1,0) & \text{if } s \in S_d \\ (0,0,0,-1) & \text{if } s \in S_u \end{cases}$
for $t = 0 \dots H-1$ **do**
  foreach $s \in S$ **do**
    foreach *criterion* $c \in \{\mathcal{G}, S_f, S_d, S_u\}$ **do**
      $V_c^{\pi_c^*, t+1}(s) \leftarrow R_c(s) + \max_{a \in A} \sum_{s' \in S} T(s, a, s').V_c^{\pi_c^*, t}(s')$
      $ideal_c \leftarrow \max_a R_c(s) + \sum_{s' \in S} T(s, a, s').V_c^{\pi, t}(s')$
      $antiIdeal_c \leftarrow \min_a R_c(s) + \sum_{s' \in S} T(s, a, s').V_c^{\pi, t}(s')$
      foreach $a \in A$ **do**
        $Rgt_c^t(s, a, \pi) \leftarrow \frac{||[R_c(s) + \sum_{s' \in S} T(s,a,s').V_c^{\pi,t}(s')] - V_c^{\pi_c^*, t+1}(s)||}{||ideal_c^{t,\pi}(s) - antiIdeal_c^{t,\pi}(s)||}$
    $optA \leftarrow argmin_a lexdiff(C_{\hat{\pi}}, Rgt_{\mathcal{G}}(s, a, \pi) + Rgt_{S_f}(s, a, \pi), Rgt_{S_d}(s, a, \pi) + Rgt_{S_u}(s, a, \pi))$
    $V^{\pi, t+1}(s) \leftarrow R(s) + \sum_{s' \in S} T(s, optA, s').V^{\pi, t}(s')$
  return V

We resolve Advice-MDPs by transforming them into MOMDPs. As a transformation, actions belonging to $C_{\hat{\pi}}$ are removed from the base T, and each advice-type is mapped to a criterion, which is rewarded when the system reaches an accordingly-advised state (e.g. $R_{S_d}(s)$ is 1 (else 0) if $s \in S_d$). Rewards are positive for desirable or goal states and negative otherwise. Since rewards are summed, the value of criterion c when following π is the number of expected visitations $\mathcal{N}_{vis,c}^\pi$ (e.g. "V(s)=(2,0,5,-7)" when following π for h steps means: "from s, following π is expected to lead

to the reach of 2 goals, 0 forbidden, 5 desirable, 7 undesirable states). Optimal solutions for this MOMDP are Pareto-optimal tradeoffs that maximize the number of visited goals and desired states, and minimize the number of forbidden and undesired states.

This transformation into MOMDPs (vs. a mono-dimentional MDP) offers multiple benefits, by avoiding boiling down the whole reward function in a single scale. First, MOMDPs enable for non-linear combination of factors (e.g. losing 3 wheels is much worse than losing one; crossing the first undesirable is very bad, but if inevitable, a rational balance should be found). Such combination can be achieved in mono-dimensional MDPs, by adding variables to the statespace, thus causing a combinatorial explosion. Else, if mono-dimensional reward functions introduce linearization artifacts, such as "trading three undesirable for one desirable", which are often not suitable for applications. Second, the MOMDP representation is simpler for operators, which lowers risks of errors. MOMDPs introduce a semantic context for applying advising and for easing system management (e.g. a slider for setting the balance between "following desirable" and "avoiding undesirable", a slider for balancing the importance between achieving goals and the tradeoff desirable/undesirable). This operation is less intuitive with linear MDPs, as it implies to reason about fiddling reward functions. Third, MOMDPs simplifies the introduction of a wider range of advice-types (e.g. a "can be pushed", "can annoy bystanders"), without having to redesign the reward function as a whole.

Our fast approximate algorithm for solving Advice-MDPs (Algorithm 1) is inspired by Algorithm 10.1 from (Sigaud and Buffet 2010, p.325). Exact MOMDP-resolution algorithms (Wakuta 2001) were left out due to their exponential complexity and our need for responsiveness. At each iteration, our algorithm backpropagates for each state the set of most promising expected multidimensional value for each action (and ideal and anti-ideal values for computing the relative value of each state, see next paragraph). Then, a *comparator function* selects the action leading to the most promising reward tuple, given the immediate choice (e.g. "(2,0,5,-7)" is better than "(2,-1,5,0)").

The MOMDP comparator is based on the Ordered Weighted Regret (OWR) function (Roijers, Vamplew, and Whiteson 2013), based on the LexDiff order (Sigaud and Buffet 2010), which enables maintaining the priority order of advice types, while introducing properties such as "undesirable states are to be avoided, but if crossing them is inevitable, then they should be best balanced with crossing desirable states". The OWR "calculates the regret of each [criterion] with respect to an estimated ideal reference point, sorts these into descending order, and calculates a weighted sum in which the weights are also in descending order" (Roijers, Vamplew, and Whiteson 2013). The regret for criterion c when playing action a in state s given a policy π for a horizon h is:

$$Rgt_c^h(s,a,\pi) = \frac{||V_c^{\pi,h}(s) - V_c^{\pi_c^*,h}(s)||}{||ideal_c^{h,\pi}(s) - antiIdeal_c^{h,\pi}(s)||}$$

where $V_c^{\pi,t}$ is expected total reward for c from following π for t steps, π_c^* is the "pure" policy that single-mindedly maximizes c, while discarding other criteria, $ideal_c^{t,\pi(s)}$ (respectively $antiIdeal_c^{t,\pi}(s)$) is the highest (respectively lowest) expected reward for c that can be acquired from s for all actions and then following π. $||ideal_c^{t,\pi}(s) - antiIdeal_c^{t,\pi}(s)||$ is the Chebyshev norm, which makes the regret relative to the relative grasp that the system has on c. If $ideal_c^{t,\pi}(s) = antiIdeal_c^{t,\pi}(s)$, then $Rgt_c(s,a,\pi) = 0$ (no regret when no grasp). The LexDiff strictly orders sets of criteria (in our case, $\{C_{\hat{\pi}}\} > \{\mathcal{G}, S_f\} > \{S_d, S_u\}$), lower-rank criteria are only considered when higher-rank criteria are equally satisfied. Altogether, LexDiff minimizes the regret of the most important advice types first and then the regret of lower-rank advice types. Note that this comparator function can be changed for altering the reaction of the system with regards to advising (e.g. allowing to trade the crossing of one forbidden state for avoiding many undesirable states).

In terms of solution quality, in our experiments, our algorithm always finds optimal MDP policies when undesirable and forbidden states can be avoided. When being forced to cross undesirable and forbidden states, our algorithm generates very satisfactory trajectories (avoiding as much as possible forbidden states, then a relatively conservative balance between avoiding undesirable states and reaching desirable states).

Professional-Grade Application

Application Context

Our industry partners aim to deploy highly flexible mobile robots for supporting operators for contexts such as disaster-recovery (Ramchurn et al. 2015), scouting in risky areas (e.g. wildlife monitoring, terrorism, illegal activities), and surveying in sensitive sites (e.g. chemical factory, nuclear plant).

Robots are given missions such as live observations (camera feed, establishing maps, monitoring threats) and situational actions (e.g. open a door, disarm traps). The robot we work with are NERVA robots illustrated in Figure 1, which are actually already being deployed by our partners, exclusively controlled through teleoperation. NERVAs are very robust (can be thrown over obstacles) and fast (up to 15km/h). They are equipped with four cameras and a wide array of specific sensors can be plugged in for capturing situational measurements (e.g. sound, gas, explosives, radioactivity).

As an application for intersecting both industrial and academic goals, we aim to make the robot capable of reaching as autonomously as possible set of locations that is required for completing sensing tasks. This goal revisits the very classical pathfinding problem, except that the many irregularities of varied, difficult, and unexpected real-world situations are added to the equation. These irregularities introduce two challenges making fully-autonomy currently unfeasible: first, available models fail to incorporate all these irregularities and the design of such an all-encompassing model would be very difficult and likely to be computationally too expensive; second, robot sensory capabilities are sometimes insufficient for recognizing such irregularity (e.g. the robot, only equipped with cameras, faces a puddle in a

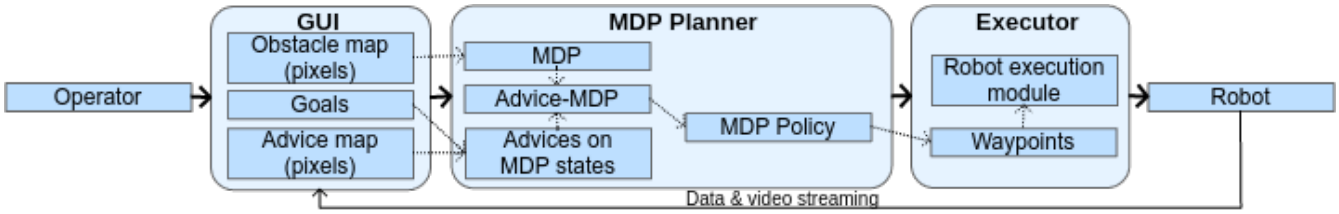


Figure 3: System loop: operator and sensor input are merged and displayed by the GUI. When goals are set, this input is turned into Advice-MDPs, given to the planner for generating MDP policies, turned into waypoints, smoothed, and executed by the robot.

chemical plant: can the puddle be recognized? Can the puddle be crossed? Is it deep? Is it a chemical that will destroy the robot? If so, how fast?).

We called our system the Anywhere Deployment Robot System (ADRS). An overview of the system loop presented in Figure 3. The ADRS generates a map of the surroundings via a Simultaneous Localization and Mapping (SLAM) using a laser. This map is then displayed with a Graphical User Interface (GUI), presented in Figure 2, on a tablet that operators wear on their wrist. This GUI enables operators to advise on goals, desirable, undesirable and forbidden areas, by simply clicking on an "advice-type" button and drawing on the map (points, lines, shapes). Then, the system generates an advice-compliant waypoint-based trajectory that minimizes the total distance crossed for reaching all goals. This trajectory is displayed back to the operator, who decides whether the robot should apply it (a "rush mode" can be turned on for skipping this check). At any time, before and during plan execution, the operator can update the advising, which then updates the waypoint trajectory. Likewise, robot trajectories are updated if the SLAM discovers new obstacles that jeopardize the current plan. Towards achieving high responsiveness and smooth interactions with operators, our fast Advice-MDP resolution algorithm enables *completing a whole sense-reason-act loop within two seconds* [1], which is satisfactory for operators. Special cases can be improved through engineering (e.g. having dodge reactive behaviors during replanning, so the robots does not become an easy target).

As an overall idea of the system behavior, and thus of the advising semantics, desirable areas (in green) mark zones that are interesting to be crossed by robots (safe ground, concealed from opposing threats), while still generally moving towards the goal. Undesirable areas (in orange), mark zones that are best avoided if possible, but that can be crossed for the sake of reaching the goal, or many more desirable areas (e.g. populated non-hostile areas, unsteady ground that slows the robot, slow acid or irradiated areas that will destroy the robot after the mission). Red areas mark zones that may harm the mission, to be avoided unless they there is no other means for reaching the goal (e.g. risk of being spotted, slippery ground that may cause crashes, unidentified puddle). These categories remain flexible and can change due to situational context (e.g. slow acid areas may be set to forbidden if losing the robot after its mission becomes an issue for follow-up actions).

Technically, the generated map is a 4096×4096 pixel map. Each pixel captures a 4cm^2 square. For keeping high responsiveness, this map is abstracted into a 400×400-tiles hexagonal grid (any tile containing at least one obstacle pixel is considered as an obstacle tile). This grid is then transformed into a canonical transition function for moving on grids (e.g. playing "East" on state/tile "(4,12)" leads to state/tile "(5,12)"). The optimal policy, π^*, provides a pathway, i.e. a sequence of sets of optimal tiles per iteration. As a straight transformation of this pathway creates highly inefficient step-by-step "stop-and-turn" move, a smoothing phase fastens robot moves by greedily searching for the waypoint path with the least number of steps that is totally covered by the optimal pathway. This step is illustrated in Figure 4a. Note that, for descriptive simplicity, this implementation relies on a deterministic MDP. However, Advice-MDPs can be applied on non-deterministic MDPs (e.g. uncertainty on the tile to be reached when playing move actions).

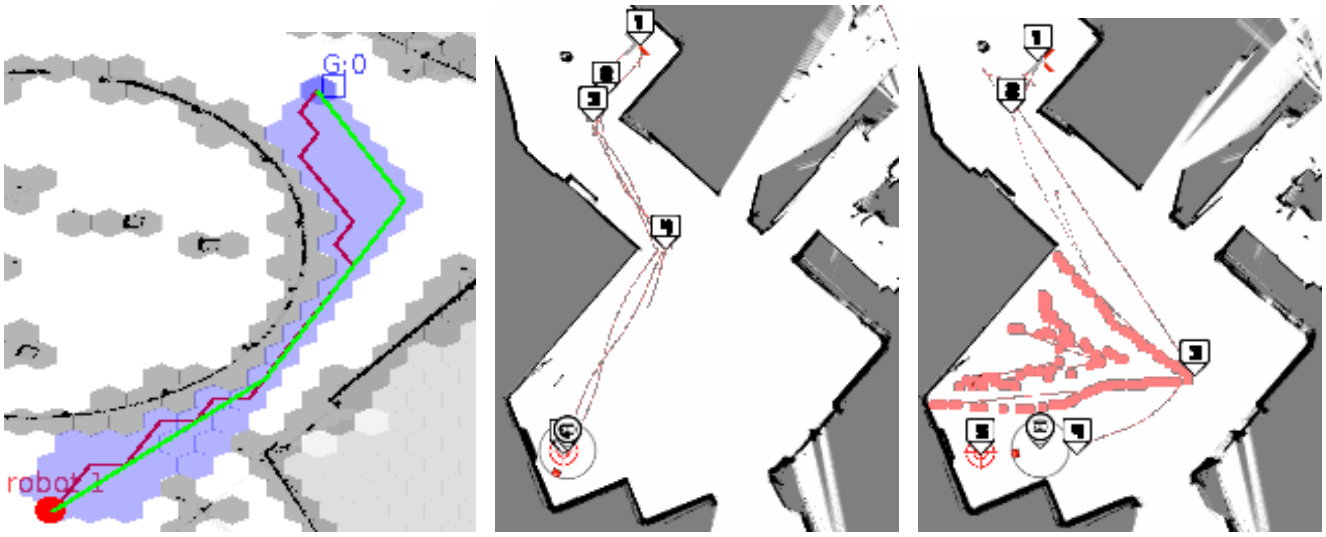
Empirical Evaluation

Varying The Degree of Autonomy

We compared Advice-MDPs against Fully-Autonomous Systems (FAS, based on classic advice-less MDPs) and Non-Autonomous systems (i.e. teleoperation), along the following criteria, partly drawn from (Hart and Staveland 1988; Steinfeld et al. 2006): 1) *flexibility* (range of situations that can be handled by the system); 2) *efficiency* (time for completing the mission); 3) *Operator Workload (OW)* (time required from operators for having the task completed, i.e. for teleoperation and advising); 4) *effectiveness* (percentage of the mission that is autonomously completed); 5) *OW flexibility* (flexibility left for the operator to decide when to operate), 6) *responsiveness* (time from task-allocation to action).

As a setup, we compared the performance of FAS, Advice-MDPs, and teleoperation in two scenarios of differing degrees of flexibility: in the corridor scenario, the robot is tasked to move to the other side of a building (Figure 4b); in the hole scenario, a (virtual) hole in the ground is to be avoided, else the robot risks breaking (Figure 4c). Each experiment was repeated 20 times. Each experimenter was trained to teleoperate and advise the robot for at least ten minutes.

[1] Demonstration videos are available at https://www.youtube.com/playlist?list=PLcULZgrd6hEgBYO6zJ74S7tH3DvX3Sf7g



(a) Optimal pathway towards the goal (blue tiles), considering obstacles (grey tiles). Red lines are the base waypoint path generated from the base MDP policy (many stop and turn operations). Green lines represented the smoothed path (minimizing the amount of waypoint, exploiting diagonals).

(b) Advice-MDP waypoint trajectories for the corridor scenario. Lines are former robot paths.

(c) Advice-MDP trajectories for the hole scenario. Red marks are "forbidden" advice, drawn by the operator.

Figure 4: Pathways on hexagonal and pixel maps

Experimental results (Table 1) detail the compromises between efficiency, flexibility, and OW costs. In terms of flexibility, FASs only succeed in the corridor scenario, as they risk getting stuck in the hole scenario; the Advice-MDPs systems handle both the corridor and hole missions; teleoperation handles all situations. Teleoperation remains more flexible than Advice-MDPs, as humans can make situational assumptions that fall out of the planning model (e.g. seeing empty crates as pushable objects rather than impassable obstacles).

In terms of OW costs, FASs are obviously the most effective, followed by Advice-MDPs, which only require one fast intervention for augmenting the planning model, and teleoperation last, which is much more expensive. Note that, once augmented, Advice-MDPs can operate with no subsequent OW costs, unless major changes in the environment (e.g. a zone catches fire).

In terms of efficiency, teleoperation is the most efficient; Advice-MDPs and FASs are equivalent (teleoperation allows for higher speed and avoiding ineffective stop-and-turn actions at waypoints).

In terms of effectiveness, Advice-MDPs achieve a score of 81.3% when augmenting the planning model is needed, which is very reasonable. In terms of OW flexibility, Advice-MDPs offers the highest OW-flexibility as advising can be done a priori, while teleoperation offers no OW-flexibility (must be done last minute, any delay lowers the efficiency of the system). Last, in terms of responsiveness, the software responds within two seconds for FASs and Advice-MDPs.

Comparing Against Related Approaches

This section compares Advice-MDPs against the two related approaches: MIPS and ASRL approaches. The experimental results are detailed in Table 1.

For the MIPS approaches, the operator is in charge of validating or rejecting the trajectories generated by the system. When rejecting a trajectory, operators indicate at which point the trajectory is incorrect. Then, the system removes the indicated state, replans and proposes an alternative trajectory.

In terms of performance, Advice-MDPs achieve slightly better efficiency for lower OW costs for the corridor experiment, as operators are asked to perform an unnecessary check for the MIPS (although, the cost is relatively low). For the hole experiment, Advice-MDPs achieve much better performance than MIPS, as rejecting plans until one becomes acceptable incurs low efficiency and high OW costs. Furthermore, after one planning phase, the Advice-MDPs system is ready to be operated without subsequent OW costs, while the whole process is to be repeated for the MIPS approach.

For the ASRL approach, a module has been implemented using the most relevant technique from (Abel et al. 2017) for this context: action pruning. During the training phase, the system asks before playing an action whether this action is the most relevant. If not, the system associates it with a low reward and proceeds with the new best action. Results highlight the that the Advice-MDP approach overcomes the ASRL approach. First, Advice-MDP avoids a very slow and OW-cost expensive learning phase, totally in 98 operator checks for the corridor scenario and 123 operator checks for the hole scenario. Second, the Advice-MDP approach

Table 1: Average autonomy effectiveness, efficiency, and operator workload costs, over 12 repetitions for the corridor and hole scenarios, depending on the system type. Lines marked by an asterisk note the OW cost during a training phase.

Scenario	System	Autonomy Effectiveness	Efficiency (s)	OW (s)
Corridor	Advice-MDPs	100%	33.7	0
Corridor	Teleop	0%	22.3	22.3
Corridor	Base MDPs	100%	34.5	0
Corridor	Action Pruning (ASRL)	NA	269.9	348.8*
Corridor	Plan Revision (MIPS)	95.1%	35.0	1.7
Hole	Advice-MDPs	81.3%	43.4	8.1
Hole	Teleop	0%	27.2	27.2
Hole	Base MDPs	100%	**Falls**	0
Hole	Action Pruning (ASRL)	NA	316.7	472.7*
Hole	Plan Revision (MIPS)	54.6%	75.2	34.9

enables for an effective prescription of all undesired states, while the ASRL approach only learns what is related to this specific trajectory. While some knowledge is re-used, the system is to be trained again for moving back to the initial position. Furthermore, during this learning phase, the robot can only rely on brute MDP-based plans, which imply operating on the level of "stop and turn" low-level actions, which dramatically slow down robot operations.

Conclusions and Perspectives

This paper introduces Advice-MDPs, a new formalism for designing systems capable of integrating operator support for augmenting planning models towards efficiently adapting the behavior of the system to real-world unexpected irregularities. Advice-MDPs expand the Markov Decision Process (MDP) formalism with the possibility to advise on the desirability, undesirability, or prohibition of states and actions. Furthermore, this paper introduces a fast algorithm for generating near-optimal advice-compliant policies, making possible to create reactive interaction-loops with operators for smoothing the task of supporting the robot.

We empirically demonstrate the benefits of Advice-MDPs against fully autonomous planning systems (Advice-MDPs trade reasonable operator workload costs for greater system flexibility), fully teleoperated systems (Advice-MDPs can dramatically decrease the operator workload costs), classic mixed-initiative planning systems (Advice-MDPs lower operators workload costs, the invested time from operator is better re-used over multiple planning operations, operators can better manage when to advise the system), and reinforcement-learning based planning systems (Advice-MDPs skip a long learning phase, thus decreasing operator workload costs and making the system much faster to set up). Conceptually, the core novelty and benefits offered by Advice-MDPs compared to classic approaches lies in enabling operators to *prescriptively augment the world-model*. This augmentation can then be used for multiple subsequent planning operations, be given early on (before planning is required), and does not require system training. Though, the benefits of advising depend on external factors, such as the advising quality (e.g. mistakes from operators) and unexpected environmental changes (e.g. new areas becoming forbidden or desirable), which can require additional efforts from operators (e.g. watching the environment and the robot, updating advising) and introduce risks of failure.

We demonstrate the applicability of these theoretical findings, by deploying Advice-MDPs in the design of a professional-grade robotic system, which we called the Anywhere Deployment Robot System (ADRS). The ADRS has been tested by its intended final users (our industry partners), who were highly satisfied by the flexibility features offered by Advice-MDPs and the relief the ADRS brings on the cognitive strain required for operating the system. The ADRS is being integrated within the robotic platform of our industry partners.

Regarding future work, we are expanding Advice-MDPs for integrating more advice types (e.g. a "passable" advice type for handling e.g. box-pushing behavior). Additionally, we are expanding our application to a multi-robot setting, considering new challenging issues in term of multi-agent planning and multi-robot coordination. Furthermore, we are investigating the combination between Advice-MDP and HOP-POMDP for combining the benefits of both approaches (Rosenthal and Veloso 2011).

Acknowledgements The authors wish to thank Melania Borit for her support improving the text and the reviewers for their constructive feedback.

References

Abel, D.; Salvatier, J.; Stuhlmüller, A.; and Evans, O. 2017. Agent-Agnostic Human-in-the-Loop Reinforcement Learning. *arXiv preprint arXiv:1701.04079*.

Armstrong-Crews, N., and Veloso, M. 2007. Oracular partially observable markov decision processes: A very special case. In *Robotics and Automation, 2007 IEEE International Conference on*, 2477–2482. IEEE.

Bresina, J.; Jonsson, A.; Morris, P.; and Rajan, K. 2005. Mixed- initiative activity planning for Mars rovers. In *Proceedings of the Int'l Joint Conference on Artificial Intelligence, 1709–1710.*, 1709–1710.

Côté, N.; Canu, A.; Bouzid, M.; and Mouaddib, A.-I. 2012. Humans-Robots Sliding Collaboration Control in Complex Environments with Adjustable Autonomy. In *Proceedings of Intelligent Agent Technology*.

Crandall, J. W.; Goodrich, M. A.; Olsen, D. R.; and Nielsen, C. W. 2005. Validating human-robot interaction schemes in multitasking environments. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 35(4):438–449.

Hart, S. G., and Staveland, L. E. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in psychology* 52:139–183.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, volume 10 of *Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics*. John Wiley & Sons, Inc.

Ramchurn, S. D.; Huynh, T. D.; Ikuno, Y.; Flann, J.; Wu, F.; Moreau, L.; Jennings, N. R.; Fischer, J. E.; Jiang, W.; Rodden, T.; Simpson, E.; Reece, S.; and Roberts, S. J. 2015. HAC-ER: A Disaster Response System Based on Human-Agent Collectives. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, 533–541. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Roijers, D.; Vamplew, P.; and Whiteson, S. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48:67–113.

Rosenthal, S., and Veloso, M. 2011. Modeling humans as observation providers using pomdps. 53–58. IEEE.

Sigaud, O., and Buffet, O. 2010. *Markov Decision Processes in Artificial Intelligence*. Wiley-IEEE Press.

Steinfeld, A.; Fong, T.; Kaber, D.; Lewis, M.; Scholtz, J.; Schultz, A.; and Goodrich, M. 2006. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 33–40. ACM.

Torrey, L.; Walker, T.; Shavlik, J.; and Maclin, R. 2005. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *ECML*, 412–424. Springer.

Wakuta, K. 2001. A multi-objective shortest path problem. *Mathematical Methods of Operations Research* 54(3):445–454.

Zilberstein, S. 2015. Building Strong Semi-Autonomous Systems. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence*, 4088–4092.