Decision-Making with Non-Markovian Rewards: From LTL to automata-based reward shaping

Alberto Camacho
Department of Computer Science
University of Toronto
Toronto, Canada
acamacho@cs.toronto.edu

Oscar Chen*
Department of Engineering
University of Cambridge
Cambridge, UK
ozhc2@cam.ac.uk

Scott Sanner
Department of Mechanical & Industrial Engineering
University of Toronto
Toronto, Canada
ssanner@mie.utoronto.ca

Sheila A. McIlraith
Department of Computer Science
University of Toronto
Toronto, Canada
sheila@cs.toronto.edu

Abstract

In many decision-making settings, reward is acquired in response to some complex behaviour that an agent realizes over time. An autonomous taxi may receive reward for picking up a passenger and subsequently delivering them to their destination. An assistive robot may receive reward for ensuring a person in their care takes their medication once daily soon after eating. Such reward is acquired by an agent in response to following a path – a sequence of states that collectively capture the reward-worthy behaviour. Reward of this sort is referred to as non-Markovian reward because it is predicated on state history rather than current state. Our concern in this paper is with both the specification and effective exploitation of non-Markovian reward in the context of Markov Decision Processes (MDPs). State-of-the-art UCT-based planners struggle with non-Markovian rewards because of their weak guidance and relatively myopic lookahead. Here we specify non-Markovian reward-worthy behaviour in Linear Temporal Logic. We translate these behaviours to corresponding deterministic finite state automata whose accepting conditions signify satisfaction of the reward-worthy behaviour. These automata accepting conditions form the basis of Markovian rewards that can be solved by off-the-shelf MDP planners, while crucially preserving policy optimality guarantees. We then explore the use of reward shaping to automatically transform these automata-based rewards into reshaped rewards that better guide search. We augmented benchmark MDP domains with non-Markovian rewards and evaluated our technique using PROST, a state-of-the-art heuristic and UCT-based MDP planner. Our experiments demonstrate significantly improved performance achieved by the exploitation of our techniques. The work presented here reflects the use of Linear Temporal Logic to specify non-Markovian reward, but our approach will work for any formal language for which there is a corresponding automata representation.

Keywords: Markov Decision Processes, Non-Markovian Rewards, LTL, Monte Carlo Search, Reward Shaping

Acknowledgements

The authors gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC).

*This work was performed while the second author was affiliated with the University of Toronto.

1 Introduction

In Markov Decision Processes agents typically receive positive or negative reward in response to their current state. Nevertheless, agents may also realize reward in response to more complex behaviour that is reflected over a sequence of states. An autonomous taxi may receive reward for picking up a passenger and subsequently delivering them to their destination. Similarly, a personal robot getting ice cubes from the freezer is rewarded for opening the freezer, removing the ice cubes, *and* closing the freezer soon after. Such reward is commonly referred to as non-Markovian reward because it is predicated on the state history rather than solely on the current state. Our concern in this paper is with both the specification and effective exploitation of non-Markovian reward in Markov Decision Processes (MDPs). Here we use Linear Temporal Logic to specify non-Markovian rewards. Notwithstanding, our approach is applicable to other formal languages for which there exist corresponding automata representations.

Current state-of-the-art MDP planners are based on heuristic search and variants of UCT techniques [11]. UCT policies tend to make greedy and myopic decisions. As such, these planners struggle with non-Markovian rewards since there is little guidance for their relatively myopic lookahead. The impact of this myopic guidance can be seen in state-of-the-art MDP planner PROST [10], a UCT-based planner that generates high-quality solutions for moderately sized MDPs, but whose performance suffers in large problems that require significant lookahead.

In this paper we explore transformation of the reward function through reward shaping [12] as a means of mitigating for the myopic lookahead of UCT-based methods. To this end, we propose an approach to solving non-Markovian Reward Decision Problems (NMRDPs) by transforming our reward-worthy non-Markovian behaviour into corresponding deterministic finite state automata. The accepting conditions of these automata signify satisfaction of the reward-inducing behaviour in a manner that is solvable with off-the-shelf MDP planners, crucially preserving optimality guarantees. Moreover, we use reward shaping with these automata-based reward encodings in order to induce non-sparse, myopic-friendly rewards. This helps guide the accrual of non-Markovian reward. We evaluate our approach to solving NMRDPs via experimentation with off-the-shelf state-of-the-art heuristic and UCT-based MDP planners. Experiments with a set of International Probabilistic Planning Competition (IPPC) domains augmented with non-Markovian rewards show significantly improved performance using our automata representation together with reward shaping.

2 Background

2.1 Model-based Decision Making

Markov Decision Processes: *Markov Decision Processes* (MDPs) [14] are popular models for decision-theoretic planning problems [5]. An MDP is a tuple $M = \langle S, A, P, R, T, \gamma, s_0 \rangle$, where: S is a finite set of states; A is a finite set of actions; $P_a(s, s')$ is the probability of reaching the state $s' \in S$ after applying action a in state $s \in S$; $R : S \times A \times S \to \mathbb{R}$ is the reward function (sometimes $R : S \times A \to \mathbb{R}$); $T \in \mathbb{N}$ is the horizon; $\gamma \in (0, 1]$ is the discount factor; and $s_0 \in S$ is the *initial state* of the MDP. Solutions to an MDP are a sequence of step-dependent *policies* $\Pi = (\pi_0, \ldots, \pi_{T-1})$ that map states $s \in S$ at step k ($0 \leq k < T$) to actions $\pi_k(s) \in A$. The *value* of a policy Π in state s at step k, $V_{\Pi,k}(s)$, is the expected discounted cumulative reward over the horizon $T - k$ following Π. Formally, $V_{\Pi,k}(s) = \mathbb{E}_\Pi\{\Sigma_{i=k}^{T-1} \gamma^i R_i\}$, where R_t denotes the immediate reward obtained at step i if the agent follows policy Π from s. An optimal policy sequence Π^* for an MDP over horizon T with initial state s_0 satisfies $\Pi^* = \text{argmax}_\Pi V_{\Pi,0}(s_0)$.

MDPs are commonly described using factored representations of the states and dynamics. In particular, RDDL [15] is a modelling language that allows for a lifted, compact representation of factored MDPs. The current state-of-the-art solution method for MDPs specified in RDDL is PROST [10], a Monte Carlo sampling algorithm based on UCT and heuristic search. Whereas PROST generates good-quality solutions for moderate-sized MDPs, its performance suffers in large problems that require significant look-ahead. In such cases, the Monte-Carlo roll-outs cannot capture the structures inherent in the problem, leading to myopic search behavior.

Non-Markovian Reward Decision Processes: NMRDPs (e.g., [1, 2, 16]) generalize the MDP model by allowing reward functions to range over the history of visited states. In contrast to MDPs, the domain of the reward function R ranges over the set of finite state sequences drawn from S, denoted $S^\star$. As in conventional MDPs, optimal solutions maximize the expected discounted cumulative reward.

2.2 Finite Linear Temporal Logic and Deterministic Finite-State Automata

Linear Temporal Logic (LTL) is a compelling language for expressing temporal properties over (infinite) sequences of states, balancing expressiveness with ease of use. LTL was initially developed to express safety and liveness properties for program verification [13] and has been used for a myriad of applications including the specification of temporally extended goals and preferences in planning (e.g., [4]). Here we specify non-Markovian rewards using LTL_f, a variant of LTL interpreted over finite traces. Our specification language is similar to \$FLTL, a finitely interpreted future LTL used by

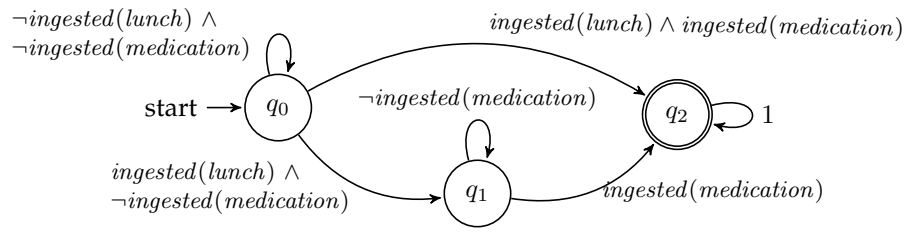

Figure 1: DFA corresponding to $\mathsf{LTL_f}$ formula $(\Diamond ingested(medication)) \wedge (\neg ingested(medication)\, \mathcal{U}\, ingested(lunch))$.

Thiébaux et al. (2006) to specify non-Markovian rewards. Bacchus et al. (1996, 1997) employed a finitely interpreted Past LTL (PLTL) to specify non-Markovian rewards.

The syntax of $\mathsf{LTL_f}$ includes the logical connectives ($\wedge$, $\vee$, $\neg$), unary modal operators *next* ($\bigcirc$), *weak next* ($\bullet$), and binary modal operator *until* ($\mathcal{U}$). Other operators, such as *always* ($\Box$) and *eventually* ($\Diamond$), can be defined in terms of these basic operators. The truth of an $\mathsf{LTL_f}$ formula φ is evaluated over *finite* sequences of states which – in the context of this paper – are the propositional states of an NMRDP or MDP (see [8] for details). We write $\pi \models \varphi$ when a sequence of states $\pi = s_0, \ldots, s_n$ satisfies φ.

By way of illustration, an assistive robot might accumulate reward by ensuring that its ward takes their medication daily and that they do so after eating lunch. Such behaviour might be expressed by $\mathsf{LTL_f}$ formula $(\Diamond ingested(medication)) \wedge (\neg ingested(medication)\, \mathcal{U}\, ingested(lunch))$.

2.2.1 $\mathsf{LTL_f}$ and Deterministic Finite-State Automata

Given $\mathsf{LTL_f}$ formula φ, one can construct a corresponding *Deterministic Finite-State Automaton* (DFA) A_φ that accepts a word π iff it satisfies φ (e.g. [9]). A DFA is a tuple $\langle Q, \Sigma, \delta, q_0, Q_{Fin} \rangle$, where Q is a finite set of states, Σ is the *alphabet* of the automaton, $\delta : Q \times \Sigma \to Q$ is a transition function, $q_0 \in Q$ is the initial state, and $Q_{Fin} \subseteq Q$ is a set of accepting states. The transition dynamics of an automaton is defined over *finite words*, or sequences $w = s_0, s_1, \ldots, s_n$ of elements in Σ. In the scope of this paper, Σ are the states of an MDP. At every stage i, the automaton makes a deterministic transition from state q_i to state $q_{i+1} = \delta(q_i, s_i)$. We say that M accepts w if $q_{n+1} \in Q_{Fin}$.

Fig. 1 shows a DFA corresponding to $\mathsf{LTL_f}$ formula $(\Diamond ingested(medication)) \wedge (\neg ingested(medication)\, \mathcal{U}\, ingested(lunch))$. Automaton states are represented by nodes, and transitions are represented by arcs. Transition labels describe the conditions that need to hold in a state to allow a particular transition. These are called *guards*. Finally, accepting states are depicted by double-ringed nodes. The word $\pi = \{\neg ingested(lunch), \neg ingested(medication); ingested(lunch), \neg ingested(medication); ingested(lunch), ingested(medication)\}$ induces one and only one run in the automaton, $\{q_0, q_0, q_1, q_2\}$. As this run finishes in an accepting state, it follows that π satisfies the $\mathsf{LTL_f}$ formula.

3 Problem: Solving NMRDPs with $\mathsf{LTL_f}$ rewards

As noted in Section 1, state-of-the-art MDP planners based on heuristic search and UCT struggle with non-Markovian rewards. In the rest of this paper we propose a novel method to effectively address this shortcoming. Following Bacchus et al. (1996), we specify rewards in an NMRDP as a *temporally extended reward function* (TERF). This TERF is realized by a set of reward behaviours, φ_i, specified in here in $\mathsf{LTL_f}$, together with a set of mappings to rewards r_i, denoted $\varphi_i : r_i$. Reward r_i is realized upon satisfaction of φ_i.

Returning to our previous example, we can define a TERF that gives a positive reward of 100 to agent behavior φ, satisfying $(\Diamond ingested(medication)) \wedge (\neg ingested(medication)\, \mathcal{U}\, ingested(lunch))$, written $\varphi : 100$. To only reward the first occurrence of the behaviour within a sequence of states, one could modify the above $\mathsf{LTL_f}$ formula as follows: $(\neg ingested(medication)\, \mathcal{U}\, (ingested(medication) \wedge \neg\bigcirc\top)) \wedge (\neg ingested(medication)\, \mathcal{U}\, ingested(lunch))$.

4 Approach: From $\mathsf{LTL_f}$ to automata-based reward shaping

To solve an NMRDP, M, we compile M with TERF R into an MDP M' with a Markovian reward R' that can be solved with a conventional off-the-shelf MDP planner. Our approach is realized in three steps: (i) for each $\varphi : r$ in the TERF, the $\mathsf{LTL_f}$ formula φ is transformed into a corresponding DFA A_φ; (ii) an MDP M' is constructed from M by augmenting state variables and transitions to reflect the state and progress of each A_φ towards its accepting condition. The Markovian reward function R' is associated with being in the accepting conditions of each A_φ, denoting satisfaction of reward-worthy behaviour φ; and (iii) M' is solved using an off-the-shelf MDP planner, thus obtaining a solution that can be

converted, in a straightforward manner, into a solution to M. For the purposes of this paper, we limit our explication to finite-horizon NMRDPs. Notwithstanding, our approach can be extended to infinite-horizon NMRDPs.

Elaborating on step (ii), M' augments M with extra fluents and actions that integrate the dynamics of the DFAs within the MDP. The dynamics of M' expand *each* time step into three modes: **world**, **sync**, and **reward**. In **world** mode, an action from the NMRDP is applied. In **sync** mode, the automata states are synchronized according to the observed state. Intuitively, the automata states simulate the runs of the automata given the observed **world** state trajectories. The assignment of reward is delayed to **reward** mode and is performed upon satisfaction of each of the $\mathsf{LTL_f}$ reward formulae in the TERF . This is detected when an automaton reaches an accepting state. Solutions to the original NMRDP– that is, mappings from state trajectories into actions – can be obtained from solutions to the compiled MDP M' by simulating the state trajectories in M'. Our compilation preserves optimal solutions, as there is a bijection between state trajectories in M and M' that preserves the accumulated reward. The interested reader can find the technical details in [6, 7].

Theorem 1. *The automata-based compilation from NMRDPs into MDPs preserves optimal solutions.*

Returning to our assistive robot example with TERF defined by $\varphi : 100$, suppose the agent performs actions $ingest(lunch)$ followed by $ingest(medication)$, which induce the state trajectory (only relevant subset of state shown): $\pi = \{\neg ingested(lunch), \neg ingested(medication); ingested(lunch), \neg ingested(medication); ingested(lunch), ingested(medication)\}$. The dynamics in the compiled MDP start by processing the initial state, and self-transitioning from the automaton state q_0 to itself. In **reward** mode, no reward is given. Then, in **world** mode the action $ingest(lunch)$ is performed, leading to a state s_1 in which $\{\neg ingested(medication), ingested(lunch)\}$ holds. The following **sync** mode synchronizes the automaton state to q_1, and so on until reaching world state s_2, where $\{ingested(medication), ingested(lunch)\}$ holds. At this point, the automaton synchronizes to state q_3, that is accepting. In **reward** mode, a reward of 100 is given.

4.1 Improving Performance via Reward Shaping

The above approach to solving NMRDPs preserves optimality (cf. Theorem 1). Here we augment our approach with reward shaping in an effort to mitigate for the sparse reward inherent in our non-Markovian rewards, that aggravates the weak guidance and lookahead of state-of-the-art UCT-based MDP planners.

Reward shaping is a common technique in MDPs which aims to improve search by transforming the reward function. Such reward transformations have the form $R'(s, a, s') = R(s, a, s') + F(s, a, s')$, where R is the original reward function and F is a *shaping reward function*. The intuition behind reward shaping is that by increasing (resp. decreasing) the reward in states that lead to other high-value states or trajectories (resp. low-value states or trajectories), we can increase the effectiveness of search and the quality of solutions, while reducing search memory and run times. Unfortunately, reward shaping with an arbitrary $F(s, a, s')$ may lead to an optimal policy that is suboptimal w.r.t. the original unshaped reward. However, as noted by [12], if $F(s, a, s')$ is chosen from a restricted class of *potential-based* shaping functions defined as $F(s, a, s') = \gamma\phi(s') - \phi(s)$ (for some real-valued function ϕ), then this guarantees preservation of optimal and near-optimal policies with respect to the original unshaped MDP. Preservation of near-optimality is desirable since it provides guarantees for suboptimal solutions obtained by approximate methods.

Here we introduce a reward shaping technique that is, by construction, potential-based, and thus preserves optimal and near-optimal solutions. Given a particular automaton configuration in s, the idea is to decompose the potential $\phi(s)$ into sums of potentials for each of the automaton state variables, f_q, that together describe the current automaton configuration. Since many off-the-shelf MDP planners employ reward functions of the form $R(s, a)$, rather than the more general $R(s, a, s')$, we record the previous automaton configuration in corresponding state variables f_q^c, and delay attribution of reward to **reward** mode. Our corresponding shaping reward function is $F(s) = \gamma \sum_{f_q} \phi(f_q) - \sum_{f_q^c} \phi(f_q^c)$, for all f_q and f_q^c that hold in s. To preserve optimality in finite-horizon NMRDPs, partial rewards given to state trajectories that do not finish in accepting states of the TERF must be subtracted; full details are in [7].

Theorem 2. *Automata-based reward shaping preserves optimal, and near-optimal solutions.*

In the assistive robot example, we may want to provide some guidance by assigning potentials $\phi(q_0) = 0$, $\phi(q_1) = 50$, and $\phi(q_2) = 100$. Intuitively, these potentials assign positive reward for transitioning from q_0 to q_1, with the rationale that state trajectories that yield such transitions make progress towards achievement of an accepting state.

5 Empirical Evaluation

We performed preliminary experiments to assess the benefits of our automata-based reward transformations at guiding search for high-quality solutions. For this task, we selected two different benchmarks from the previous IPPC: *academic-advising*, and *wildfire*. We replaced the original rewards with non-Markovian rewards, and compiled the resulting NMRDPs into MDPs as described above. All problems were described in RDDL [15], and we used PROST [10] as the MDP planner. In *academic-advising* problems, p_m_n, the agent is rewarded for passing $m \cdot n$ courses, each one having m course

MDP Planner	Compilation	p_3_3	p_4_2	p_4_3	p_4_4
PROST UCT*(IDS)	MDP	30	30	0	0
PROST UCT*(DFS)	MDP	30	30	0	0
PROST IPPC-2014	MDP	2	30	0	0
PROST IPPC-2011	MDP	27	30	2	0
UCT	MDP	0	0	0	0
PROST UCT*(IDS)	MDP + RS	30	30	30	30
PROST UCT*(DFS)	MDP + RS	30	30	30	30
PROST IPPC-2014	MDP + RS	30	30	30	30
PROST IPPC-2011	MDP + RS	30	30	30	30
UCT (3 steps look ahead)	MDP + RS	29	30	29	30

Table 1: Number of runs (over 30 trials) that achieved the non-Markovian reward in the *academic-advising* problems. Compilations tested with and without reward shaping (RS).

MDP Planner	No RS	RS
PROST UCT*(IDS)	MLE	617
PROST UCT*(DFS)	MLE	627
PROST IPPC-2014	MLE	620
PROST IPPC-2011	MLE	637
UCT (3 steps look ahead)	423	527
no actions taken	263	263

Table 2: Average reward achieved (over 30 trials) in the MDP compilations of the *wildfire* problem, with and without reward shaping (RS). MLE indicates that PROST exceeded the memory limit of 512 MB.

prerequisites. In the *wildfire* problems, the agent is given reward to extinguish fire in a 3×3 grid, if performed no later than two time steps from its origination. The potentials used for reward shaping are naive, and distributed uniformly in the automaton states according to the distance to an accepting state. We observed that reward shaping can be an effective technique to provide guidance. In some of our tests, the quality of solutions improved drastically from zero average reward, to optimal policies even with simple UCT search (see Table 1). The *wildfire* problems are more probabilistically complex, and PROST easily ran out of memory (512 MB) in problems without reward shaping (see Table 2). On the other side, the guidance provided by reward shaping reduced the amount of memory needed by PROST, which found policies of increased quality than those obtained without reward shaping – even when we limited the memory usage of PROST.

6 Summary and Discussion

NMRDPs provide a powerful framework for modelling decision-making problems with behaviour-based rewards. In this paper we use LTL_f to specify rich non-Markovian rewards and present a technique for solving NMRDPs through a compilation to MDPs that can be solved with off-the-shelf MDP planners. Our approach integrates automata representations of the LTL_f formulae into the compiled MDP. We leverage reward shaping to help guide search, mitigating for the sparseness of non-Markovian rewards and the poor lookahead of some state-of-the-art UCT-based methods. Our experiments demonstrate that automata-based reward shaping is an effective method to enhance search and obtain solutions of superior quality. While non-Markovian rewards were specified here in LTL_f, the proposed approach will work for rewards specified in any formal language for which there is a corresponding automata representation (e.g., [3]).

References

[1] F. Bacchus, C. Boutilier, and A. J. Grove. Rewarding behaviors. In *AAAI*, pages 1160–1167, 1996.

[2] F. Bacchus, C. Boutilier, and A. J. Grove. Structured solution methods for non-markovian decision processes. In *AAAI*, pages 112–117, 1997.

[3] J. A. Baier, C. Fritz, M. Bienvenu, and S. McIlraith. Beyond classical planning: Procedural control knowledge and preferences in state-of-the-art planners. In *AAAI*, pages 1509–1512, 2008.

[4] J. A. Baier, F. Bacchus, and S. A. McIlraith. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, 173(5-6):593–618, 2009.

[5] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94, 1999.

[6] A. Camacho, O. Chen, S. Sanner, and S. A. McIlraith. Non-Markovian rewards expressed in LTL: Guiding search via reward shaping. In *SoCS*, 2017. To appear.

[7] A. Camacho, O. Chen, S. Sanner, and S. A. McIlraith. Decision-making with non-markovian rewards: Guiding search via automata-based reward shaping. Technical Report CSRG-632, Department of Computer Science, University of Toronto, 2017.

[8] G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860, 2013.

[9] G. De Giacomo and M. Y. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, pages 1558–1564, 2015.

[10] T. Keller and P. Eyerich. PROST: probabilistic planning based on UCT. In *ICAPS*, 2012.

[11] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *In: ECML-06. Number 4212 in LNCS*, pages 282–293. Springer, 2006.

[12] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations : Theory and application to reward shaping. In *ICML*, volume 3, pages 278–287, 1999. doi: 10.1.1.48.345.

[13] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[14] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994. ISBN 0471619779. doi: 10.1080/00401706.1995.10484354.

[15] S. Sanner. Relational dynamic influence diagram language (RDDL): Language description. http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf, 2010.

[16] S. Thiébaux, C. Gretton, J. K. Slaney, D. Price, F. Kabanza, et al. Decision-theoretic planning with non-markovian rewards. *JAIR*, 25:17–74, 2006.