Decision-Making with Non-Markovian Rewards: Guiding search via automata-based reward shaping*

Alberto Camacho†, Oscar Chen‡†, Scott Sanner⋆, Sheila A. McIlraith†
†Department of Computer Science, University of Toronto
⋆Department of Mechanical & Industrial Engineering, University of Toronto
‡Department of Engineering, University of Cambridge
†{acamacho,sheila}@cs.toronto.edu, ‡ozhc2@cam.ac.uk, ⋆ssanner@mie.utoronto.ca

Abstract

Markov Decision Processes (MDPs) are a popular framework for decision making under uncertainty. In an MDP, the agent attempts to maximize the expected cumulative reward under the assumption that rewards and environment dynamics satisfy the so-called *Markovian* property – namely, that they depend on the agent's current state and not on the past history. Our concern in this paper is with both the specification and effective exploitation of Non-Markovian reward in the context of Markov Decision Processes (NMRDPs). Previous approaches for NMRDPs transfom the problem into a standard MDP that can be solved by standard search-based MDP planners. While MDP planners based on heuristic-search and UCT-based techniques now dominate the state of the art, these planners struggle with non-Markovian rewards which often provide little guidance to the relatively myopic lookahead of these solvers. Here we explore the use of *reward shaping* to automatically transform Non-Markovian rewards – specified in Linear Temporal Logic – into automata with reshaped rewards. Once in this form, automata-based reward transformations can be exploited by off-the-shelf MDP planners to guide search, while crucially preserving policy optimality guarantees. We augment benchmark domains from the International Probabilistic Planning Competition with non-Markovian rewards and evaluate our technique using state-of-the-art MDP solvers. 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.

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. For example, an autonomous electric vehicle may acquire reward for always recharging its battery after a trip. Similarly, a personal robot may acquire reward by opening the refrigerator, removing a prescribed item, and closing the refrigerator immediately thereafter. 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 (Kocsis and Szepesvári 2006). 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 (Keller and Eyerich 2012), 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 (Ng, Harada, and Russell 1999) 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.

*This technical report extends work presented at SoCS17 and RLDM17. The reference for this work is: *Camacho, O. Chen, S. Sanner, and S. A. McIlraith. Non-Markovian Rewards Expressed in LTL: Guiding search via reward shaping. 2017. At SoCS.To appear.*

†Work performed while the author was at University of Toronto.

2 Background

2.1 Decision-Theoretic Planning

Markov Decision Processes: *Markov Decision Processes* (MDPs) (Puterman 1994) are popular models for decision-theoretic planning problems (Boutilier, Dean, and Hanks 1999). 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 (Sanner 2010) is a modelling language that allows for a lifted, compact representation of factored MDPs. States are given by assignments to all ground state fluents of the RDDL specification, and transition dynamics are described by conditional probabilities that can be expressed as a stochastic form of *successor state axioms* (Reiter 1991). Intuitively, the updates on the truth of each ground fluent are described with respect to what holds in the current state and the actions that are performed. Modern algorithms for MDPs are typically based on UCT-search techniques that sacrifice optimality in favor of scalability. The current state-of-the-art solution method for MDPs is PROST (Keller and Eyerich 2012), a Monte-Carlo sampling algorithm based on UCT and heuristic search for finite-horizon MDPs. Whereas PROST generates good-quality solutions to moderate-sized MDPs, its performance suffers in large problems that require a significant look-ahead. In such cases, the Monte-Carlo roll-outs cannot sufficiently capture the structures inherent in the problem, which leads to greedy/myopic search behavior.

Non-Markovian Reward Decision Processes: Bacchus, Boutilier, and Grove (1996) generalized the MDP model by allowing for reward functions to take into account the trajectory of visited-states at time step t. Formally, a *Non-Markovian Reward Decision Process* (NMRDP) is a tuple $M = \langle S, A, P, R, T, \gamma, s_0 \rangle$, where S, A, P, T, γ , and s_0 are as defined in MDPs. Unlike MDPs, the domain of the reward function R is the set of finite sequences of states over S, which we denote as $S^\star$. More precisely, the reward associated to a finite state trajectory $\Gamma = s_0, s_1, \ldots, s_n$ is $R(\Gamma)$. The optimality criteria of solutions remains the same as in MDPs, with the exception of the history-dependent reward function.

Thiébaux et al. (2006) is, to the best of our knowledge, the most recent translation-based approach to NMRDPs. Their approach translated NMRDPs into MDPs that can be solved (possibly sub-optimally) using anytime state-based heuristic search methods (Thiébaux, Kabanza, and Slaney 2002). In addition to their translation, the authors contributed with NMRDPP (Gretton et al. 2003), a framework for solving NMRDPs that made it possible to have experimental results for the first time, and a comparison of their method with previous translation-based approaches (Gretton, Price, and Thiébaux 2003). Previous translation-based methods to solving NMRDPs suffered from state-space explosion when keeping track of the history trajectory (Bacchus, Boutilier, and Grove 1996) or were only amenable to structural policy construction methods (Bacchus, Boutilier, and Grove 1997).

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 found, 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 (Ng, Harada, and Russell 1999), if $F(s, a, s')$ is chosen from a restricted class of potential-based reward 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 state-of-the-art heuristic search approximate methods.

Theorem 1 ((Ng, Harada, and Russell 1999))**.** *Potential-based MDP reward shaping preserve optimal, and near-optimal solutions.*

2.2 Linear Temporal Logic and Automata

Linear Temporal Logic (LTL) is a compelling language for expressing temporal properties over (infinite) sequences of states. First developed for program verification in (Pnueli 1977), the syntax of LTL includes the logical connectives ($\wedge$, $\vee$, $\neg$), unary modal operators *next* ($\bigcirc$), and binary modal operator *until* ($\mathcal{U}$). Other operators, such as *eventually* ($\Diamond\alpha \equiv \top\ \mathcal{U}\ \alpha$) and *always* ($\Box\alpha \equiv \neg\Diamond\neg\alpha$) are defined using these basic operators. The truth of an LTL formula φ is evaluated over *infinite* sequences of states $\pi = s_1, s_2, \ldots$. We say that π satisfies φ ($\pi \models \varphi$, for short) iff $\pi, 1 \models \varphi$, where for every natural number $i \geq 1$:

- $\pi, i \models p$, for a propositional variable p, iff $s_i \models p$
- $\pi, i \models \neg\psi$ iff it is not the case that $\pi, i \models \psi$
- $\pi, i \models (\psi \wedge \chi)$ iff $\pi, i \models \psi$ and $\pi, i \models \chi$
- $\pi, i \models \bigcirc\varphi$ iff $i < |\pi|$ and $\pi, i + 1 \models \varphi$
- $\pi, i \models (\varphi_1\ \mathcal{U}\ \varphi_2)$ iff for some j in $\{i, \ldots, |\pi|\}$, it holds that $\pi, j \models \varphi_2$ and for all $k \in \{i, \ldots, j - 1\}$, $\pi, k \models \varphi_1$

LTL has been widely used by the planning community to express temporally extended goals (e.g., (Baier and McIlraith 2006)), preferences (e.g., (Baier, Bacchus, and McIlraith 2009; Li et al. 2015)), and non-Markovian rewards (e.g., (Bacchus and Kabanza 1998)). In the context of this paper, LTL formulae are evaluated over sequences of propositional states that are execution traces of an NMRDP or MDP. LTL allows compact description of temporal properties of state trajectories, such as "always have money" ($\Box(have(money))$).

LTL Interpreted Over Finite Traces The use of LTL interpreted over *finite* traces for planning goes back at least to 1996 with the work of Bacchus et al. (Bacchus, Boutilier, and Grove 1996). In their work, they exploited *Past* LTL (PLTL) — a version of LTL whose semantics interpret the truth of a formula in a given state with respect of its past history — using it to express non-Markovian rewards in NMRDPs (Bacchus, Boutilier, and Grove 1996; 1997). PLTL has also been used to describe preferred explanations in the context of dynamical diagnosis (Sohrabi, Baier, and McIlraith 2011).

Other works have exploited finite trace interpretations with the more common subset of LTL that uses future modalities. Exploitation of this variant of LTL again dates back at least as far as Bacchus and Kabanza (1998) who used LTL to describe domain control knowledge to control search in automated plan generation. This resulted in a planning tool, TLPlan, later extended to exploit heuristic search (Baier and McIlraith 2006) and preferences (Baier, Bacchus, and McIlraith 2007). Thiébaux et al. (2006) introduced \$FLTL— a variant of LTL augmented with a predicate \$ to indicate receipt of reward — and an approach to NMRDPs based on progression of \$FLTL formulae.

In this work, we use the variant of future LTL interpreted over finite traces formalized by De Giacomo and Vardi (2013), and named LTL_f. LTL_f is used in different applications, including fair and unfair plan synthesis (e.g. (Camacho et al. 2017; De Giacomo and Vardi 2016)), business process modeling (e.g. (De Giacomo and Vardi 2013)), and diagnosis (e.g. Bienvenu, Fritz, and McIlraith (2006)). LTL_f extends the synthax of LTL with the modality **final** $\equiv \neg\bigcirc\top$ and operator *weak next* ($\bullet$), where $\bullet\alpha \equiv \bigcirc\alpha \vee$ **final** indicates that either α holds in the next state, or there is no next state. The main difference between LTL_f and PLTL is that the former semantics evaluate a finite state trajectory $\Gamma = s_0, s_1, \ldots, s_m$ by looking into the future from s_0, whereas the semantics of the latter evaluate Γ by looking into the past from s_m.

LTL_f and Deterministic Finite-State Automata A *Deterministic Finite-State Automaton* (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 a DFA 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 $q_{i+1} = \delta(q_i, s_i)$. The *guard* of a transition from q to q', denoted by $guard(q \to q')$, is a propositional formula so that $q' = \delta(q, s)$ iff $s \models guard(q \to q')$. We say that A accepts w if $q_n \in Q_{Fin}$. We say that M accepts w if $q_{n+1} \in Q_{Fin}$.

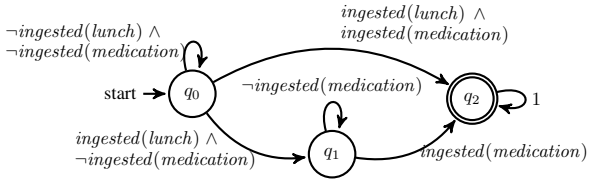

Figure 1: DFA corresponding to LTL_f formula $(\Diamond ingested(medication)) \wedge (\neg ingested(medication)\ \mathcal{U}\ ingested(lunch))$.

Given LTL_f formula φ, one can construct a corresponding DFA A_φ that accepts a word π iff it satisfies φ (e.g. (De Giacomo and Vardi 2015)). Other versions of LTL can be transformed into DFA. In particular, Sohrabi, Baier, and McIlraith (2011) showed a method to transform φ into an NBA (that can also be determinized), and all basic temporal operators in PDDL3 have automaton representations (cf. (Gerevini et al. 2009)).

Running example: Figure 1 depicts a DFA corresponding to LTL_f formula $\varphi := \varphi_1 \wedge \varphi_2$, where $\varphi_1 := \Diamond ingested(medication)$ indicates that the medication is taken at some point, and $\varphi_2 := (\neg ingested(medication)\ \mathcal{U}\ ingested(lunch)$ indicates that the medication should not be taken before having had lunch. Automaton states are represented by nodes, and transitions are represented by arcs. Transition labels describe the guards. Finally, accepting states are depicted by double-ringed nodes. The word $\pi = \{s_0, s_1, s_2\}$ with $s_1 \models \{\neg ingested(lunch) \wedge \neg ingested(medication)\}$, $s_2 \models \{ingested(lunch) \wedge \neg ingested(medication)\}$, and $s_3 \models \{ingested(lunch) \wedge ingested(medication)\}$ 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 LTL_f formula.

3 Problem: NMRDPs with temporally-extended reward formulae

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, Boutilier, and Grove (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 here in 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. Formally, a TERF R is defined by $R(\Gamma) = \Sigma_{i=1}^{m} R_i(\Gamma)$, where $R_i(\Gamma) = r_i$ if $\Gamma \models \varphi_i$, and $R_i(\Gamma) = 0$ otherwise. For simplicity, we sometimes abuse notation and confuse the TERF R with the set $\{R_1, \ldots, R_m\}$.

Definition 1. *(adapted from (Bacchus, Boutilier, and Grove 1996)) A TERF for an NMRDP is described by a finite set*

of mappings $\varphi_i : r_i$, where here φ_i are $\mathsf{LTL_f}$ formulae, and $r_i \in \mathbb{R}$. A TERF R is evaluated over a finite state trajectory Γ by $R(\Gamma) = \Sigma_{i=1}^{m} R_i(\Gamma)$, where $R_i(\Gamma) = r_i$ if $\Gamma \models \varphi_i$, and $R_i(\Gamma) = 0$ otherwise.

Previous approaches to NMRDPs used PLTL (Bacchus, Boutilier, and Grove 1996; 1997) and \$FLTL (Thiébaux et al. 2006) to describe reward formulae. Throghout the paper, we express reward formulae in $\mathsf{LTL_f}$, although our approach is extendable to any language that can be compiled to DFA – in particular, PLTL and the temporally extended goals compliant with the PDDL3 standard (Gerevini et al. 2009).

Running Example (cont.): Returning to our previous example, we can define a TERF with the mapping $\varphi : 100$, that gives a positive reward of 100 to agent behavior $\varphi := (\Diamond ingest(medication)) \wedge (\neg ingest(medication)\, \mathcal{U}\, ingest(lunch))$. Note that $\varphi : 100$ rewards all sequences of states for which there is a prefix that satisfies φ. 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: $(\neg ingest(medication)\, \mathcal{U}\, (ingest(medication) \wedge \neg\bigcirc\top)) \wedge (\neg ingest(medication)\, \mathcal{U}\, ingest(lunch))$.

4 Approach: Automata-based compilations from NMRDPs to MDPs

To solve an NMRDP M with TERF R, we compile the problem into an MDP M' with a Markovian reward R' that can be solved with a conventional off-the-shelf MDP solver. Our approach is realized in three steps: (i) for each $\varphi : r$ in R, 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 straightforward 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.

4.1 Compiling TERFs Away

Elaborating on step (ii), M' augments M with extra fluents and actions that integrate the dynamics of the DFAs within the MDP, making it possible for the reward function to be Markovian. 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. Table 1 contains technical details of the compilation. We provide a detailed description of the dynamics of the compilation below.

M' has the same fluents as M, plus the auxiliary fluents described below. The fluents $\mathbf{world}, \mathbf{sync}, \mathbf{reward}$ control the dynamics of the problem, forcing an alternation between three different modes: **world** mode, **sync** mode, and **reward** mode. For each automaton A_φ, and for each automaton state $q \in A_\varphi$, M' has fluents f_q. Intuitively, each fluent f_q simulates an automaton state q. The set of actions in M' contains the set of actions A in M. For simplicity, and without loss of generality, we assume A contains an action *no-op* – denoting that the agent performs no explicit action – and an action o_{end} that the agent is required to execute at the end of the search horizon.

In **world** mode, execution of actions from A emulate the stochastic transition model of the actions in the original NMRDP. After an action is applied in **world** mode, the dynamics of the MDP switches to **sync** mode. In **sync** mode, the truth of the automaton state fluents f_q are updated to simulate the state transition of each automaton A_ϕ with respect to the current state of the MDP. After an action is applied in **sync** mode, the dynamics of the MDP switches to **reward** mode. In **reward** mode, the agent collects reward upon satisfaction of each of the $\mathsf{LTL_f}$ reward formulae in the simulated NMRDP. More precisely, for each mapping $\varphi : r$ in the TERF, a reward r is given to the agent in state s when f_q holds in s for some accepting automaton state $q \in A_\varphi$. The resulting reward is Markovian, and is formalized by R' in Table 1. Without loss of generality, we assume that the agent is forced to perform o_{end} in the last **reward** mode (that is, in the last turn) before reaching the search horizon, T'.

The horizon in M', T', is three times the horizon T in the original NMRDP M. This is because each step in M has three counterparts in M', corresponding to the **world**, **sync**, and **reward** modes. Likewise, the discount factor of M' is $\gamma' = \gamma^{1/3}$. Finally, the initial state s'_0 of M' has all the fluents in the initial state s_0 of M, plus fluents f_q for the initial automaton state q of each automaton A_φ, and fluent $\mathbf{sync}$ that forces the agent to start in **sync** mode.

The compilation described above transforms a finite-horizon NMRDP M into a finite-horizon MDP M' that preserves optimal, and near-optimal solutions. A key aspect to understand this property is to realize that the stochastic transition model in M' in **world** mode simulates the transitions in M, whereas automata transitions simulated in **sync** mode are deterministic. As such, there exists a correspondence between finite state-action trajectories in M and M'. For a state-action trajectory $\Gamma = s_0, a_1, s_1, a_2, \ldots, s_n$ in M one can construct a state-action trajectory in M' by interleaving synchronization and reward state-actions between each a_i. Conversely, for each state-action trajectory Γ' in M' one can construct Γ by removing synchronization and reward state-actions from Γ'. The association described above defines a correspondence between policies Π in M and policies Π' in M'. It is straightforward to see that $V_{\Pi,0}(s_0) = V_{\Pi',0}(s'_0) - R(s_0)$. In consequence, the compilation from NMRDP into MDPs preserves optimal, and near-optimal solutions.

Theorem 2. *The automata-based compilation from NMRDPs into MDPs preserves optimal and near-optimal solutions.*

	Original NMRDP	Compiled MDP
Initial state	s_0	$s'_0 = s_0 \cup \bigcup_{(\varphi:r)\in R}\{f_q \mid q \text{ initial state of } A_\varphi\} \cup \{\mathbf{sync}\}$
Successor state axioms	$p' \leftrightarrow (\phi_p^+) \vee (p \wedge \neg\phi_p^-)$	$p' \leftrightarrow (\phi_p^+ \wedge \mathbf{world}) \vee (p \wedge (\neg\mathbf{world} \vee (\neg\phi_p^- \wedge \mathbf{world}))$ $\mathbf{world}' \leftrightarrow \mathbf{reward}$ $\mathbf{sync}' \leftrightarrow \mathbf{world}$ $\mathbf{reward}' \leftrightarrow \mathbf{sync}$ $f_q \leftrightarrow (\mathbf{sync} \wedge \phi_q^+) \vee (f_q \wedge \neg\mathbf{sync})$
Reward function	$R = \{(\varphi_i : r_i)\}_{i=1..m}$	$\forall s$ such that $s \models \mathbf{reward}$, $R'(s) = \sum_{(\varphi:r)\in R} \sum_{q\in A_\varphi,\ q \text{ accepting}} r_i \mathbb{1}_{f_q}(s)$
Discound factor	γ	$\gamma' = \gamma^{1/3}$
Horizon	T	$T' = 3T$

Table 1: Dynamics of the compiled MDP in terms of the dynamics of the original NMRDP. Here, $p \in F$ are propositional variables and $q \in A_\varphi$ are automaton states for each pair $(\varphi : r)$ in R. The successor state axioms for a predicate fluent p are described in the form $p' \leftrightarrow \phi$, where ϕ are the conditions that make p true at time $t+1$ as a function of the values of fluents at time t. The function ϕ_p^+ (resp. ϕ_p^-) is a propositional formula that describes the conditions under which p is made true (resp. false). Likewise, $\phi_q^+ = \bigvee_{q'\in A_\varphi} guard(q' \to q) \wedge F_{q'}$ describes the conditions under which f_q is made true. In the definition of R', $\mathbb{1}_{f_q}(s)$ is the indicator function that evaluates to 1 when f_q holds in s, and 0 otherwise.

Proof sketch. Follows from the observation above, establishing the correspondence between state-action sequences in M and M'. □

Running Example (cont.): Returning to our assisting 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.

The following results establish the size of the compiled MDP is polynomially bounded in the size of the DFAs from the TERF. The compilation of PLTL and $\mathsf{LTL_f}$ formulae into automata is worst-case double-exponential in the size of the formulae.

Theorem 3. *The size of the compiled MDP is polynomially larger in the size of the DFAs, and worst-case double-exponential in the size of the* $\mathsf{LTL_f}$ *formulae.*

Proof sketch. The set of ground fluents in the compiled MDP is augmented with automaton state fluents f_q and f_q^c, one for each automaton state. It is well-known that the size of a DFA is worst-case double-exponential in the size of the PLTL and $\mathsf{LTL_f}$ formula. Other fluents and parametrized actions in the compiled MDP are bounded in size. □

5 Improving Performance via Reward Shaping

The above approach to solving NMRDPs preserves optimality (cf. Theorem 2). 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.

In particular, state-of-the-art MDP planner PROST (Keller and Eyerich 2012) suffers from this myopic issue. PROST expands a search tree by favoring the branches that achieve the most immediate reward. An estimation of the expected reward is computed in a leaf node by either (i) throwing a Monte-Carlo (MC) roll-out, (ii) running iterative deepening search (IDS) with a bounded horizon, or (iii) running depth-first search (DFS) with a bounded horizon. When the rewards are sparse, neither of the methods below provide good guidance. MC roll-outs make random moves that can hardly collect any non-Markovian reward. IDS does not collect reward most of the time, and it does not scale with long look ahead horizon. Finally, DFS is blindly guided most of the time.

5.1 Automata-based Potentials

Reward shaping can be applied to the compiled MDP, as with regular MDPs, with the aim to provide better guidance to MDP solvers. The particular structure of the compiled MDPs, where automata fluents capture relevant historical information, suggest that we can exploit automata to design a class of shaping rewards that provide effective guidance by exploiting automata. In this section, we introduce a class of automata-based shaping reward functions that is, by construction, potential based. In Section 6, we evaluate preliminary tests to test the guidance of this class of potentials.

We augment the construction of the MDP with additional fluents, f_q^c, one for each automaton fluent f_q. These fluents

keep track of the previous configuration of the automata, and its value is updated in **sync** mode according to the successor state exioms:

$$f_q^{c\prime} \leftrightarrow (\mathbf{sync} \wedge f_q) \vee (f_q^c \wedge \neg \mathbf{sync})$$

Formally, our shaping reward function is defined over states s in **reward** mode (i.e. $s \models \mathbf{reward}$) as follows:

$$F(s,a) = \gamma \sum_{f_q} \phi(f_q) - \sum_{f_q^c} \phi(f_q^c)$$

for all actions $a \in A \setminus \{o_{end}\}$, and f_q and f_q^c that hold in s. Unlike Ng, Harada, and Russell (1999)'s method, our shaping function F does not depend on two consecutive states in the MDP, but on the two states visited *in the last two* **world** *modes*. Intuitively, F emulates a potential-based reward transformation of the form $F(s,a,s') = \gamma\phi(s') - \phi(s)$ (i.e., as defined by Ng, Harada, and Russell) in the original NMRDP. The important thing to notice here is that, in the compiled MDP, the application of F is *delayed* to the **reward** mode. This is because many off-the-shelf MDP planners employ reward functions of the form $R(s,a)$, rather than the more general $R(s,a,s')$.

The potential function $\phi(s)$ decomposes into a sum of potential functions evaluated on the automaton states that hold true in state s. More precisely, the potential function has the form $\phi(s) = \sum_{f_q \in s} \phi(f_q)$. Automaton state copies f_q^c make it possible to evaluate the potential in the previous **world** mode state. In order to preserve optimality (and near-optimality) of solutions to finite-horizon NMRDPs, we need to substract the shaping rewards at the end of each finite execution trace. This is performed upon application of o_{end} action in **reward** mode, by extending the domain of F to states $s \models \mathbf{reward}$ as follows:

$$F(s, o_{end}) = -\sum_{f_q^c} \phi(f_q^c)$$

for all f_q^c that hold in s.

Theorem 4. *Automata-based reward shaping preserves optimal, and near-optimal solutions.*

Running Example (cont.): 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.

In what follows, we introduce different criteria to engineer potential functions. It is not our purpose to give an exhaustive list, nor to study theoretical guarantees. Rather, our purpose is to inspire the reader about the number of ways that reward shaping can be used to improve guidance in NMRDPs. In Section 6, we prove empirically that even naive potential functions can successfully guide search and result in significant improvements in terms of the quality of the solutions found by approximate methods.

Liveness Preservation: It is possible to incentive exploration of those states for which satisfaction of the reward formula φ is, in principle, still reachable. This is the case of non-Markovian liveness behaviors. In this case, we can assign $\phi(f_q) = 0$ to non-accepting sink automaton states $q \in A_\varphi$, and $\phi(f_q) = c$ otherwise, where $c \in \mathbb{R}^+$ is a constant

Heuristic Guidance: In order to incentive the search process towards satisfication of the TERFs — and, therefore, collect reward —, we can distribute the potentials $\phi(f_q)$ in a way that they increase monotonically w.r.t. the inverse of *a distance measure* between q and the set of accepting states in A_φ. For example, the potentials can be distributed proportionally according to the graph distance in the directed graph representation of A_φ.

Attenuation Factor: Similar to the discount factor in infinite-horizon MDPS, an attenuation factor can be applied to the potentials in order to incentive early achievement of rewards. Unlike the discount factor, the attenuation factor does not need to be uniform over all behaviours in the TERF. For example, we can attenuate potentials by a factor $T - k/T$, where k is the steps counter in the current state and T is the horizon limit. Whereas the potentials with this technique are not constant over time, it has been shown that optimality, and near-optimality guarantees are equally preserved in dynamic reward shaping (Devlin and Kudenko 2012).

Given the linearity of the shaping function, linear combinations of the methods presented above preserve optimality guarantees. Similarly, the technique to attenuate the potentials is orthogonal to the liveness preservation and heuristic guidance techniques presented above.

6 Empirical Evaluation

We conducted experiments with the purpose of evaluating the impact of different reward shaping techniques in the quality of the solutions obtained to MDP compilations of NMRDPs. We conducted our experiments in a selection of MDP problems described in RDDL from previous International Probabilistic Planning Competitions (IPPCs), in which we replaced the Markovian rewards by TERFs. We used different configurations of PROST as the MDP planner. Namely, the current state-of-the-art UCT*, the configurations that won the IPPC 2011 and IPPC 2014, and a configuration of PROST that behaves like the basic UCT algorithm by Kocsis and Szepesvári (2006). For UCT*, we test two different heuristic evaluation functions based in iterative deepening search (IDS), and depth-first search (DFS).

6.1 Guiding Towards Long-term TERFs

In our first set of experiments, we evaluated the impact of reward shaping to guide search for long-term non-Markovian rewards. We conducted our tests in a modification of the *academic-advising* domain. In these problems, the agent can take courses. Courses have prerequisites — e.g. second-year courses have as prerequisite all first-year courses, and so on —, that affect the probability to pass a course. The agent is given a (non-Markovian) reward upon completion of all courses, if these were taken in an order that satisfies all prerequisites. In $\mathsf{LTL_f}$, the reward formula is the conjunc-

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 2: Number of runs (over 30 trials) that achieved the non-Markovian reward in the *academic-advising* problems. Different MDP planners were used to solve the compiled MDP problem, with and without reward shaping (RS).

tion of two families of subfomulae. The first family asks the agent to pass all courses, and is captured by the formulae $\Diamond(passed(c))$, one for each course c. The second family asks the agent to take courses in an order that satisfies the prerequisites, that is, if c' is a prerequisite of c, then c should not be taken until c' is passed. This is captured by the formulae $\Box(\bigwedge_{c'}((taken(c) \wedge prereq(c', c)) \rightarrow passed(c'))$.

Table 2 shows the summary of the results of our tests on different *academic-advising* problems. Each problem p_Y_C has the TERF described above, where Y is the number of academic years, and C is the number of courses per academic year. We compiled each NMRDP problem into an MDP, with and without reward shaping. Finally, we evaluated the quality of the solutions to our compilations using different configurations of PROST.

As predicted, state-of-the-art anytime planners are shortsighted and struggle to find good long-term strategies in MDP compilations to NMRDPs. To improve the look ahead, we modified the MDP compilation to have only one mode, *world* mode, in which the automata is also progressed. The results of these tuned experiments are reported in Table 2 as *MDP compilation*. We distinguish a very abrupt decrease in performance in larger problem instances. This is because the look ahead performed by PROST in presence of sparse non-Markovian rewards is highly uninformed (i.e. nearly blind search), and for sufficiently larger instances the look ahead cannot provide any information. If this occurs, PROST makes early random moves that prevent the agent from collecting any long-term reward. On the other hand, when the MDP compilations are complemented with reward shaping, the search performance improves significantly. In our tests with reward shaping, the potentials are distributed uniformly with the number of courses passed — i.e. inversely proportional to the distance to the accepting automaton states as described in Section 5.1 —, and go down to zero when the prerequisites are violated. State-of-the-art configurations of PROST achieved the TERF in all trials. Remarkably, reward shaping boosted the performance of the basic UCT algorithm from zero to achieving the non-Markovian reward in almost all trials.

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

Table 3: Average reward achieved (over 30 trials) in a *wildfire* problem with desired behaviours, for each cell c: *never have fire in c for more than two turns in a row*. Different MDP planners were used to solve the compiled MDP problem, with and without reward shaping (RS).

6.2 Guiding Towards Tradeoff Liveness

In our second set of experiments, we evaluated the practicality of our approach in problems where the stochasticity of the domain may make it infeasible to accomplish all the desired behaviors. We used a modification of the *wildfire* domain. In the *wildfire* domain, some places in a grid field are originally burning. Fire can propagate to neighboring cells with certain probability. The agent can attempt to extinguish such fire, one cell at a time. We consider a *wildfire* problem in a 3×3 grid with desired behaviours, one for each cell c: *never have fire in c for more than two turns in a row*. We assign each one reward $r = 100$ if, at the end of an execution trace of length 10, the behaviour is satisfied. We experimented with different configurations of PROST, UCT, and a naive planner that takes no action. The results are summarized in Table 3. In these experiments, we notice that reward shaping is beneficial in the quality of the plans. Moreover, PROST easily runs out of memory (512MB) if no reward shaping is applied. This is because the sparsity of rewards forces it to expand a large search tree. Limiting the memory usage in PROST resulted in policies of lower quality than those obtained with reward shaping. On the other hand, naive reward shaping ($\phi(F_q) = r$ in accepting states, zero otherwise) successfully guides search, with a significant reduction in memory, and demonstrating improved scalability.

7 Summary and Discussion

NMRDPs provide a powerful framework for modelling decision-making problems with behaviour-based rewards. In this paper we use $\mathsf{LTL_f}$ 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 $\mathsf{LTL_f}$ 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 $\mathsf{LTL_f}$, the proposed approach will work for rewards specified in any formal language for which there is a corresponding automata representation (e.g., (Baier et al. 2008)).

References

Bacchus, F., and Kabanza, F. 1998. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence* 22(1-2):5–27.

Bacchus, F.; Boutilier, C.; and Grove, A. J. 1996. Rewarding behaviors. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAI)*, 1160–1167.

Bacchus, F.; Boutilier, C.; and Grove, A. J. 1997. Structured solution methods for non-markovian decision processes. In *Proc. of the 14th National Conference on Artificial Intelligence (AAAI)*, 112–117.

Baier, J. A., and McIlraith, S. A. 2006. Planning with temporally extended goals using heuristic search. In *Proc. of the 16th Intl. Conference on Automated Planning and Scheduling (ICAPS)*, 342–345.

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2007. A heuristic search approach to planning with temporally extended preferences. In *Proc. of the 20th Intl. Joint Conference on Artificial Intelligence (IJCAI)*, 1808–1815.

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

Baier, J. A.; Fritz, C.; Bienvenu, M.; and McIlraith, S. 2008. Beyond classical planning: Procedural control knowledge and preferences in state-of-the-art planners. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI), Nectar Track*, 1509–1512.

Bienvenu, M.; Fritz, C.; and McIlraith, S. A. 2006. Planning with qualitative temporal preferences. In *Proc. of the 10th Intl. Conference on Knowledge Representation and Reasoning (KR)*, 134–144.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research (JAIR)* 11:1–94.

Camacho, A.; Triantafillou, E.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2017. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *Proc. of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 3716–3724.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *Proc. of the 23rd Intl. Joint Conference on Artificial Intelligence (IJCAI)*, 854–860.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In *Proc. of the 24th Intl. Joint Conference on Artificial Intelligence (IJCAI)*, 1558–1564.

De Giacomo, G., and Vardi, M. Y. 2016. LTL_f and LDL_f synthesis under partial observability. In *Proc. of the 25th Intl. Joint Conference on Artificial Intelligence (IJCAI)*, 1044–1050.

Devlin, S., and Kudenko, D. 2012. Dynamic potential-based reward shaping. In *Proc. of the 11th Intl. Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, 433–440.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173(5-6):619–668.

Gretton, C.; Price, D.; Thiébaux, S.; et al. 2003. NMRDPP: a system for decision-theoretic planning with non-Markovian rewards. In *Proceedings of the Workshop on Planning under Uncertainty and Incomplete Information at ICAPS*, 48–56.

Gretton, C.; Price, D.; and Thiébaux, S. 2003. Implementation and comparison of solution methods for decision processes with non-Markovian rewards. In *Proc. of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 289–296.

Keller, T., and Eyerich, P. 2012. PROST: probabilistic planning based on UCT. In *Proc. of the 22nd Intl. Conference on Automated Planning and Scheduling (ICAPS)*.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *Proc. of the 17th European Conference on Machine Learning (ECML)*, 282–293.

Li, M.; She, Z.; Turrini, A.; and Zhang, L. 2015. Preference planning for markov decision processes. In *Proc. of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 3313–3319.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations : Theory and application to reward shaping. In *Proc. of the 16th Intl. Conference on Machine Learning (ICML)*, volume 3, 278–287.

Pnueli, A. 1977. The temporal logic of programs. In *Proc. of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, 46–57.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc.

Reiter, R. 1991. *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression*. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. San Diego, CA: Academic Press. 359–380.

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

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *Proc. of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, 261–267.

Thiébaux, S.; Gretton, C.; Slaney, J. K.; Price, D.; Kabanza, F.; et al. 2006. Decision-theoretic planning with non-markovian rewards. *Journal of Artificial Intelligence Research (JAIR)* 25:17–74.

Thiébaux, S.; Kabanza, F.; and Slaney, J. K. 2002. Anytime state-based solution methods for decision processes with non-Markovian rewards. In *Proc. of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 501–510.