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
{acamachosheilas}@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 transform 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 look-ahead 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 approach using state-of-the-art MDP planners.

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 $\Pi$ in state $s$ at step $k$, $V_{\Pi,k}(s)$, is the expected discounted cumulative reward over the horizon $T - k$ following $\Pi$. Formally, $V_{\Pi,k}(s) = \mathbb{E}_\Pi[\sum_{i=k}^{T-1} \gamma^i R_i | s]$, where $R_i$ denotes the immediate reward obtained at step $i$ if the agent follows policy $\Pi$ from $s$. An optimal policy sequence $\Pi^*$ for an MDP over horizon $T$ with initial state $s_0$ satisfies $\Pi^* = \arg\max_{\Pi} \mathbb{E}_{\Pi}[V_{\Pi,0}(s_0)]$.

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 $S, A, P, T, \gamma$, 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 $S^*$. 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 $\phi$), 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 ($\land$, $\lor$, $\neg$), unary modal operators $\text{next}$ ($\Diamond$), and binary modal operator $\text{until}$ ($\mathcal{U}$). Other operators, such as eventually ($\text{eventually } \Diamond \alpha \equiv \mathcal{U} \alpha$) and always ($\text{always } \Box \alpha \equiv \neg \Diamond \neg \alpha$) are defined using these basic operators. The truth of an LTL formula $\varphi$ is evaluated over infinite sequences of states $\pi = s_1, s_2, \ldots$. We say that $\pi$ satisfies $\varphi$ ($\pi \models \varphi$, 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 $\pi, i \models \psi$
- $\pi, i \models (\psi \land \chi)$ iff $\pi, i \models \psi$ and $\pi, i \models \chi$
- $\pi, i \models \Box \varphi$ 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” (∇(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). Thiebaux et al. (2006) introduced $\text{FLTL}$ — a variant of LTL augmented with a predicate $\$ to indicate receipt of reward — and an approach to NMRDPs based on progression of $\text{FLTL}$ formulae.

In this work, we use the variant of future LTL interpreted over finite traces formalized by De Giacomo and Vardi (2013), and named LTLf. LTLf is used in different applications, including fair and unfair plan synthesis (e.g. (Ca macho 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)). LTLf extends the syntax of LTL with the modality $\text{final} = \neg O \top$ and operator weak next $\otimes$ (where $\otimes \equiv \neg O \otimes \forall \text{ final}$ indicates that either $\alpha$ holds in the next state, or there is no next state. The main difference between LTLf 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 $\Gamma$ by looking into the past from $s_m$.

**LTLf and Deterministic Finite-State Automata** A Deterministic Finite-State Automaton (DFA) is a tuple $\langle Q, \Sigma, \delta, q_0, Q_{\text{Fin}} \rangle$, where $Q$ is a finite set of states, $\Sigma$ is the alphabet of the automaton, $\delta : Q \times \Sigma \rightarrow Q$ is a transition function, $q_0 \in Q$ is the initial state, and $Q_{\text{Fin}} \subseteq Q$ 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 $\Sigma$. In the scope of this paper, $\Sigma$ 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

![Figure 1: DFA corresponding to LTLf formula (\(\text{final}(\text{ingested(medication)}) \land \neg\text{ingested(medication)} \lor \text{ingested(lunch)})\).](image)

**Running example:** Figure 1 depicts a DFA corresponding to LTLf formula $\varphi := \varphi_1 \land \varphi_2$, where $\varphi_1 := \diamond \neg\text{ingested(medication)}$ indicates that the medication is taken at some point, and $\varphi_2 := \neg\text{ingested(medication)} \lor \text{ingested(lunch)}$ 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 $s_1 = \{\neg\text{ingested(lunch)} \land \neg\text{ingested(medication)}\}$, $s_2 = \{\text{ingested(lunch)} \land \neg\text{ingested(medication)}\}$, and $s_3 = \{\neg\text{ingested(lunch)} \land \neg\text{ingested(medication)}\}$ 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 $\pi$ satisfies the LTLf 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, $\varphi_i$, specified here in LTLf, together with a set of mappings to rewards $r_i$, denoted $\varphi_i : r_i$. Reward $r_i$ is realized upon satisfaction of $\varphi_i$. Formally, a TERF $R$ is defined by $R(\Gamma) = \sum_{i=1}^{m} R_i(\Gamma)$, 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 \( \varphi_i \) are LTL\(_i\) formulae, and \( r_i \in \mathbb{R} \). A TERF \( R \) is evaluated over a finite state trajectory \( \Gamma \) by \( R(\Gamma) = \sum_{i=1}^{m} R_i(\Gamma) \), 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 SFLTL (Thiébaux et al. 2006) to describe reward formulae. Throughout the paper, we express reward formulae in LTL\(_i\), 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 \( \varphi := (\Diamond \text{ingest(medication)}) \land (\neg \text{ingest(medication)} \cup \text{ingest(lunch)}) \). Note that \( \varphi : 100 \) rewards all sequences of states for which there is a prefix that satisfies \( \varphi \). To only reward the first occurrence of the behaviour within a sequence of states, one could modify the above LTL\(_i\) formula as follows: \((\neg \text{ingest(medication)} \cup (\text{ingest(medication)} \land \neg \Diamond \top)) \land (\neg \text{ingest(medication)} \cup \text{ingest(lunch)})\).

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 \( R \), \( \varphi \) is transformed into a corresponding DFA \( A_\varphi \); (ii) an MDP \( M' \) is constructed from \( M \) by augmenting state variables and transitions to reflect the state and progress of each \( A_\varphi \) towards its accepting condition. The Markovian reward function \( R' \) is associated with being in the accepting conditions of each \( A_\varphi \), denoting satisfaction of reward-worthy behaviour \( \varphi \); 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 expedition 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 LTL\(_i\) 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 world, sync, reward control the dynamics of the problem, forcing an alternation between three different modes: world mode, sync mode, and reward mode. For each automaton \( A_\varphi \), 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 \( e_{\text{end}} \) 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 \( A_\varphi \) 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 LTL\(_i\) 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 \( e_{\text{end}} \) 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 \( A_\varphi \), and fluent 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 \( \Gamma' \) in \( M' \) one can construct \( \Gamma \) by removing synchronization and reward state-actions from \( \Gamma' \). The association described above defines a correspondence between policies \( \Pi \) in \( M \) and policies \( \Pi' \) 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.
Table 1: Dynamics of the compiled MDP in terms of the dynamics of the original NMRDP. Here, $p \in F$ are propositional variables and $q \in A_p$ 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 $\phi$ are the conditions that make $p$ true at time $t + 1$ as a function of the values of fluents at time $t$. The function $\phi_p^+$ (resp. $\phi_p^-$) is a propositional formula that describes the conditions under which $p$ is made true (resp. false). Likewise, $\phi_q^+ = \bigvee_{q' \in A_p} \text{guard}(q' \rightarrow q) \land F_{q'}$ describes the conditions under which $f_q$ is made true. In the definition of $R'$, $\mathbb{I}_{f_q}(s)$ 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): 
\[ \pi = \{ \neg \text{ingest(lunch)}, \neg \text{medication}, \text{ingest(lunch)}, \text{medication}, \text{ingest(lunch)} \}. \]

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 $\{ \neg \text{medication}, \text{ingest(lunch)} \}$ holds. The following sync mode synchronizes the automaton state to $q_1$, and so on until reaching world state $s_2$, where $\{ \text{medication}, \text{ingest(lunch)} \}$ 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 LTLf 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 LTLf formulae.

Proof sketch. The set of ground fluents in the compiled MDP is augmented with automaton state fluents $f_q$ and $f_q^*$, 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 LTLf 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 exploits 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, $f_q^*$, 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:

\[ f_q' \leftrightarrow (\text{sync} \land f_q) \lor (f_q^c \land \neg \text{sync}) \]

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

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

for all actions \( a \in A \setminus \{ o_{\text{end}} \} \), 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 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 \( \phi(s) = \sum_{f_q} \phi(f_q) \). 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 \( o_{\text{end}} \) action in reward mode, by extending the domain of \( F \) to states \( s \models \text{reward} \) as follows:

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

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 \( \varphi \) 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 satisfaction 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 \( A_\varphi \). For example, the potentials can be distributed proportionally according to the graph distance in the directed graph representation of \( A_\varphi \).

**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 LTL formulas, the reward formula is the conjunc-
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., (Baier et al. 2008)).
References


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


