Markov Decision Processes with Functional Rewards

Olivier Spanjaard and Paul Weng [⋆]

LIP6, UPMC
firstname.lastname@lip6.fr

Abstract. Markov decision processes (MDP) have become one of the standard models for decision-theoretic planning problems under uncertainty. In its standard form, rewards are assumed to be numerical additive scalars. In this paper, we propose a generalization of this model allowing rewards to be functional. The value of a history is recursively computed by composing the reward functions. We show that several variants of MDPs presented in the literature can be instantiated in this setting. We then identify sufficient conditions on these reward functions for dynamic programming to be valid. In order to show the potential of our framework, we conclude the paper by presenting several illustrative examples.

1 Introduction

In sequential decision-making under uncertainty, Markov Decision Processes (MDPs) have received much attention as a natural framework both for modeling and solving complex structured decision problems [1–5]. In standard MDPs, scalar rewards – assumed to be additive – are granted along the process, and a policy is evaluated according to the probabilistic expectation of the cumulated rewards. Yet, it often happens that some of those assumptions have to be relaxed, as in the following cases already highlighted in the literature: rewards are not static, in the sense that they do not only depend on the current state and the action carried out: this is the case for instance in MDPs where rewards represent durations, and these durations vary over time (think about public transportation: the travel time depends on which bus you catch); rewards are not known with certainty: the determination of precise rewards remains a bottleneck in the specification of an MDP, and it is often advantageous to use imprecise or stochastic rewards; the evaluation of a policy is not performed via probabilistic expectation: this is the case for instance in high-stake planning situations, where one wishes to take the level of risk into consideration by using expected utility instead of probabilistic expectation. Each of these observations have encouraged researchers to provide different extensions of MDPs, together with dedicated solution procedures: Time-dependent MDPs [3], MDPs with imprecise rewards [5], MDPs with one-switch utility functions [4].

[⋆] Funded by the French National Research Agency under grant ANR-09-BLAN-0361.

In this article, we propose a general framework called *MDP with functional rewards* (abbreviated by FRMDP in the sequel) to handle most features of such variants, and we show how to recast them in our setting. Our approach is based on the use of functional rewards instead of scalar ones. This can be seen as a generalization of the work of Serafini [6], who searched for preferred paths in graphs with functional costs, and of the work of Liu and Koenig [4] who proposed functional value iteration in MDPs with *scalar* rewards. In FRMDPs, the value of a policy in a state can be defined in two ways: a *recursive definition* based on a functional version of the Bellman equations [7, 8] and an *explicit definition* based on the expectation of cumulated rewards. We first exhibit general conditions for which the backward induction procedure returns an optimal policy according to the recursive definition. We however emphasize that in some cases the optimal policy according to this recursive definition can differ from that according to the explicit definition. We then exhibit the condition under which both notions of optimality coincide. For the sake of simplicity, we only consider the finite horizon case in this paper, although the results could be extended in infinite horizon MDPs with goal states for instance.

2 Preliminaries and Motivating Example

A *Markov Decision Process* (MDP) can be defined as a collection of objects $\{S, A, p^a_{ss'}, r^a_s, g_s, h\}$ where S is a finite set of states, A is a finite set of actions, $p^a_{ss'}$ is the probability of reaching state s' after taking action a in state s, r^a_s is the immediate reward gained after taking action a in state s, g_s is the final reward received when the process stops at state s and $h \in \mathbb{N}$ is a finite horizon.

A *history* starting from state $s_h \in S$ corresponds to a sequence: $(s_h, a_h, s_{h-1}, a_{h-1}, \ldots, a_1, s_0)$ where $\forall i = 1, \ldots, h, (a_i, s_{i-1}) \in A \times S$. A *decision rule* $\delta : S \to A$ is a mapping from states to actions. By abuse of notation, transition probability $p^{\delta(s)}_{ss'}$ is denoted by $p^{\delta}_{ss'}$ and reward $r^{\delta(s)}_s$ by r^{δ}_s. The set of all decision rules is denoted by Δ. A *policy* π is a sequence of decision rules: $\pi = (\delta_h, \delta_{h-1}, \ldots, \delta_1)$ where $\forall t = 1, \ldots, h, \delta_t \in \Delta$ is the decision rule applied at the tth-to-last step.

The *value function* $v^\pi(s)$ of a policy $\pi = (\delta_h, \delta_h, \ldots, \delta_1)$ in a state s, defined as the expected cumulated reward obtained by executing policy π from state s, can be computed iteratively by using the *Bellman equations*:

$$\begin{aligned} v^\pi_0(s) &= 0 \\ v^\pi_t(s) &= r^{\delta_t}_s + \sum_{s' \in S} p^{\delta_t}_{ss'} v^\pi_{t-1}(s') \quad \forall t \le h \end{aligned} \tag{1}$$

A policy π is *preferred* to a policy π' if and only if $\forall s \in S, v^\pi(s) \ge v^{\pi'}(s)$. An *optimal* policy is a policy that is preferred to any other policy. The value function v^*_t of an optimal policy satisfies the *Bellman optimality equations*:

$$\begin{aligned} v^*_0(s) &= 0 \\ v^*_t(s) &= \max_{a \in A} r^a_s + \sum_{s' \in S} p^a_{ss'} v^*_{t-1}(s') \quad \forall t \le h \end{aligned} \tag{2}$$

```
1: $\forall s \in S, v^*_0(s) \leftarrow 0$; $t \leftarrow 0$
2: repeat
3:    $t \leftarrow t + 1$
4:    for all $s \in S$ do
5:       for all $a \in A$ do
6:          $q_t(s, a) \leftarrow r^a_s + \sum_{s' \in S} p^a_{ss'} v^*_{t-1}(s')$
7:       end for
8:       $v^*_t(s) \leftarrow \max_{a \in A} q_t(s, a)$
9:    end for
10: until $t = h$
```

Fig. 1. Backward Induction

This problem can be solved using the *backward induction* algorithm (Fig. 1).

A Coffee Robot Example. As a motivating example for our new framework, we present an adaptation of a shortest path problem in a deterministic graph, proposed by Serafini [6], to the coffee robot environment [9]. Consider a mobile robot whose task is to bring a coffee to a sleepy researcher. For simplicity, assume that the laboratory map can be seen as a 3x3 grid. Each cell of this grid corresponds to an area of the laboratory. The coffee machine is located at cell $(1, 1)$ and the researcher waits for her coffee at cell $(3, 3)$, where coordinates (n_r, n_c) stand for the row and the column of a cell. As the robot can bump into walls, furniture or people, it could spill the cup of coffee. Spilling the coffee has not the same cost according to the place where it happens: for instance, it is more damaging in offices (where there are carpets) than in corridors (where the floor is covered by lino). In Figure 2, the left (resp. middle) grid corresponds to the cost (resp. probability) of spilling the coffee according to the location of the robot. Four actions are available for the robot, which simply consists in deciding in which direction to go: (N)orth, (S)outh, (E)ast or (W)est. Due to faulty connections in its circuits, the robot has a tendency to drift to the left of the expected direction (once in ten): for instance, instruction "East" at cell $(2, 2)$ sometimes leads to cell $(1, 3)$ instead of $(2, 3)$. The robot stops as soon as the coffee is delivered or spilled. The aim is to determine a policy minimizing the expected cost of spilling the coffee.

The formalism of MDPs seems well-suited for this problem of sequential decision making under uncertainty. It can be formalized as follows: $S = \{(n_r, n_c) : n_r \in \{1, 2, 3\}, n_c \in \{1, 2, 3\}\}$; $A = \{N, S, E, W\}$ (the possible directions); $p^a_{ss'} = 0.1$if $s' = failure^a(s)$ or 0.9if $s' = success^a(s)$ where state s equal (n_r, n_c) and state $success^a(s)$ (resp. $failure^a(s)$) is the state reached if action a succeeds (resp. fails, drift to the left of the expected direction) in state s; when action a brings the robot to the wall, we assume that $success^a(s) = failure^a(s) = s$, meaning that the robot does not move; $g_s = 0, \forall s \in S$;

It now remains to define the rewards from the matrices of Figure 2. We assume that cost r_s (resp. probability P_s) of spilling the coffee when performing an action in state s is equal to the value indicated at cell s in the left (resp. middle) matrix. Note that in the general case, those costs and probabilities may depend on the action and the destination state. The subtlety here is that a cost is incurred only if a spilling occurs, which does not seem to naturally fit the

-1	-2	-1
-3	-3	-4
-1	-2	0

3%	5%	3%
2%	4%	2%
1%	5%	3%

S	E	S
S	E	S
E	E	

Fig. 2. The Coffee Example: costs (left) and probabilities (center) of spilling coffee, first decision rule (right) of π.

usual assumptions of MDPs. The most natural expression of v^π_t from v^π_{t-1} writes indeed, for any horizon h and $\pi = (\delta_h, \delta_{h-1}, \dots, \delta_1)$:

$$v^\pi_0(s) = 0$$
$$v^\pi_t(s) = P_s r_s + (1 - P_s) \sum_{s' \in S} p^{\delta_t}_{ss'} v^\pi_{t-1}(s') \quad \forall t \leq h$$

by explicitly taking into account the fact that once the coffee is spilled the robot stops (and therefore no more cost is incurred in this case). For instance, $v^\pi_t(2, 2) = 0.04(-3) + 0.96(0.9v^\pi_{t-1}(2, 3) + 0.1v^\pi_{t-1}(1, 3))$ for policy π whose first decision rule is indicated on the right of Figure 2. This is not the usual form of the Bellman equations due to P_s. In order to recover this usual form, one needs to redefine rewards as $c_s = P_s r_s$ and transition probabilities as $q^a_{ss'} = (1 - P_s)p^a_{ss'}$:

$$v^\pi_0(s) = 0$$
$$v^\pi_t(s) = c_s + \sum_{s' \in S} q^{\delta_t}_{ss'} v^\pi_{t-1}(s') \quad \forall t \leq h$$

We now show that MDPs with functional rewards provide a more intuitive setting for this problem, and encompass a wide class of MDP variants.

3 MDP with Functional Rewards

An *MDP with functional rewards* (FRMDP) is an MDP where each reward $r^a_{ss'}$ is replaced by a function $f^a_{ss'} : \mathcal{R} \to \mathcal{R}$ where $\mathcal{R}$ is a valuation space with a binary operator $\max^{\mathcal{R}}$ that defines a (not necessarily complete) order relation $\succeq_{\mathcal{R}}$: $\forall x, y \in \mathcal{R}, x \succeq_{\mathcal{R}} y \Leftrightarrow \exists z \in \mathcal{R}, x = \max^{\mathcal{R}}(y, z)$. Furthermore, each final reward $g_s \in \mathrm{I\!R}$ is replaced by a value in $\mathcal{R}$. Function $f^a_{ss'}(x)$ measures the value of executing action a in state s, moving to state s', and assuming that $x \in \mathcal{R}$ has already been received. Such functions are called *reward update functions*. By abuse of notation, for any decision rule δ, function $f^{\delta(s)}_{ss'}$ is denoted $f^\delta_{ss'}$.

In order to value a policy from an initial state, we assume that set $\mathcal{R}$ is endowed with a mixture operation m that assigns an element $m(p, x, y) = px + (1 - p)y$ in $\mathcal{R}$ to each p in $[0, 1]$ and each ordered pair (x, y) in $\mathcal{R} \times \mathcal{R}$. Pair $(\mathcal{R}, m)$ is assumed to be a *mixture set* [10], i.e., the following properties hold:

M1. $1x + 0x = x$,
M2. $p_1 x + (1 - p_1)y = (1 - p_1)y + p_1 x$,
M3. $p_1[p_2 x + (1 - p_2)y] + (1 - p_1)y = (p_1 p_2)x + (1 - p_1 p_2)y$,

for all x, y in $\mathcal{R}$ and p_1, p_2 in $[0, 1]$. Note that the mixture of more than two elements, i.e., $\sum_i p_i x_i$, is defined by inductive application of property M3. Moreover, we impose the following *independence* condition:

I. $x \succeq_{\mathcal{R}} y \Leftrightarrow px + (1-p)z \succeq_{\mathcal{R}} py + (1-p)z$

for all x, y, z in $\mathcal{R}$ and p in $[0, 1]$. This condition well-known in decision theory [11] states that when comparing two elements, only the differing parts are important in choosing the preferred ones.

As mentioned in the introduction, preferences over policies can be defined in two ways, as the value function of a policy can be given two definitions: a recursive one and an explicit one. We present those two definitions in the following two subsections. We show that the recursive definition allows the use of dynamic programming under some conditions that we will specify. We then reveal which property the reward update functions have to satisfy in order for the explicit definition and the recursive definition to be equivalent.

Recursive Definition. The value function of a policy $\pi = (\delta_h, \delta_{h-1}, \ldots, \delta_1)$ in a state s can be defined recursively with the functional version of the Bellman equations:

$$\begin{aligned} v_0^\pi(s) &= g_s \\ v_t^\pi(s) &= \sum_{s' \in S} p_{ss'}^{\delta_t} f_{ss'}^{\delta_t}\big(v_{t-1}^\pi(s')\big) \quad \forall t \le h \end{aligned} \tag{3}$$

A policy π is *preferred* to a policy π' in state s at step t, denoted by $\pi \succsim_{s,t} \pi'$, if and only if $v_t^\pi(s) \succeq_{\mathcal{R}} v_t^{\pi'}(s)$. Note that preference relation $\succsim_{s,t}$ may be partial as $\succeq_{\mathcal{R}}$ is not necessarily complete.

Usually, valuation space $\mathcal{R}$ is simply taken as the real line $\mathbb{R}$. The mixture operation (resp. $\max^{\mathcal{R}}$) is then the usual convex combination between reals (resp. the usual max operator). For instance, this is the case of standard MDPs, which are FRMDPs where $f_{ss'}^a$ is defined from $r_{ss'}^a$ by $f_{ss'}^a(x) = r_{ss'}^a + x$. The coffee robot example can also be casted in this framework, by setting $f_{ss'}^a(x) = P_s r_{ss'}^a + (1 - P_s)x$ and $g_s = 0$.

Besides, FRMDPs include many previously proposed variants of MDPs as special instances when $\mathcal{R} \neq \mathbb{R}$. We detail here two examples:

Time-dependent MDPs [3]. FRMDPs can model problems where the rewards depend on time. For instance, consider a navigation problem where one wants to minimize the arrival time. The duration $d_{ss'}^a(t)$ of a transition following action a from state s to s' depends on the departure time t at state s. The value of a history or a policy from a state s, interpreted here as the arrival time, is thus a function of the departure time. This situation can be modeled with $\mathcal{R} = \mathbb{R}^{\mathbb{R}}$, which denotes the set of real functions. For any $f \in \mathcal{R}$, if the value $f(t)$ represents the expected final arrival time with t being the departure time from some state s', then $f_{ss'}^a\big(f\big)(t) = f\big(d_{ss'}^a(t) + t\big)$ represents the expected final arrival time when executing action a at time t from state s and arriving in state s' (at time $d_{ss'}^a(t) + t$). Function g_s is set to the identity function Id. The choice for g_s will become clear when we present the solving method. For two functions $f, g \in \mathcal{R}$, $\max^{\mathcal{R}}(f, g) = h$ where $\forall x \in \mathbb{R}, h(x) = \max(f(x), g(x))$. Then, clearly, $f \succeq_{\mathcal{R}} g$ if and only if $\forall x \in \mathbb{R}$, $f(x) \ge g(x)$. Here, note that $\mathcal{R}$ is only partially ordered by $\succeq_{\mathcal{R}}$. The mixture operation is simply the convex combination of functions. Note that, contrarily to the dedicated framework provided by [3], FRMDPs cannot accommodate problems where the transition function also depends on the departure time.

Risk-sensitive MDPs [12, 4]: In standard MDPs, the decision criterion for comparing policies in a state is simply the mathematical expectation. However, it is insufficient when one wants to take into account risk aversion for instance. In that aim, one can use *expected utility* (EU) instead. For a discrete random variable X (total reward) whose possible realizations are $x_1, \ldots, x_k$ in $\mathbb{R}$, with probabilities $p_1, \ldots, p_k$ respectively, it is formulated as follows: $\sum_{i=1}^k p_i u(x_i)$, where u is the *utility function.* An MDP using EU as a decision criterion is an FRMDP with $\mathcal{R} = \mathbb{R}^{\mathbb{R}}$, $\forall f \in \mathcal{R}$, $\forall x \in \mathbb{R}$, $f^a_{ss'}\big(f\big)(x) = f\big(r^a_{ss'} + x\big)$ and $g_s = u$, where $r^a_{ss'}$ is the immediate reward of the initial MDP. Here, again, the value function v^π of policy π in a state s is a real function. Its interpretation is the following: $v^\pi(s)(x)$ represents the expected utility of the total reward obtained by following policy π from state s, assuming that cumulated reward x has already been received. At the last time step, clearly the value function of any policy in a state s is $g_s = u$, which can then be applied on the cumulated rewards received before reaching the final time step. Operator $\max^{\mathcal{R}}$, relation $\succeq_{\mathcal{R}}$ and the mixture operation are defined as in the previous example.

In a close framework with $\mathcal{R} = \mathbb{R}$, Kreps and Porteus [7] axiomatically justified a preference system defined recursively for taking into account preferences on temporal resolution of uncertainty. They identified under which conditions preferences over policies can be represented by functions $f^a_{ss'}$ and showed that $f^a_{ss'}$'s have to be strictly increasing functions. In their framework, Kreps and Porteus [8] showed that backward induction can be used to determine an optimal policy in finite horizon. Those general results can naturally and easily be extended to FRMDPs.

Proposition 1 *If Condition I holds and the functions $f^a_{ss'}$'s are strictly increasing, then an optimal value function v^*_h can be computed recursively with a functional version of the Bellman optimality equations:*

$$\begin{aligned} v^*_0(s) &= g_s \\ v^*_t(s) &= \max_{a\in A}{}^{\mathcal{R}} \sum_{s'\in S} p^a_{ss'} f^a_{ss'}\big(v^*_{t-1}(s')\big) \quad \forall t \le h \end{aligned} \tag{4}$$

Proof. First, note that Condition I implies:

$(x \succeq_{\mathcal{R}} y \text{ and } z \succeq_{\mathcal{R}} z') \Rightarrow px + (1-p)z \succeq_{\mathcal{R}} py + (1-p)z'$

for all x, y, z, z' in $\mathcal{R}$ and p in $[0, 1]$. This last condition can then be extended for mixing n elements by induction.

The proof of the proposition can then be done by induction. Obviously, $v^*_0(s) = g_s$ as it is true for any policy. Assume that v^*_{t-1} is the value function of an optimal policy. Then, for any policy π, we have $\forall s \in S$, $v^*_{t-1}(s) \succeq_{\mathcal{R}} v^\pi_{t-1}(s)$. As all reward update functions are strictly increasing, for any action a, we have $\forall s, s' \in S$, $f^a_{ss'}\big(v^*_t(s')\big) \succeq_{\mathcal{R}} f^a_{ss'}\big(v^\pi_t(s')\big)$. Compute $v^*_t(s)$ following Equation 4. By Condition I, we have then for any policy π, $\forall s \in S$, $v^*_t(s) \succeq_{\mathcal{R}} v^\pi_t(s)$. ■

Note the importance of the existence of the operator $\max^{\mathcal{R}}$ as it entails the unicity of the optimal value function, even if $\mathcal{R}$ may be partially ordered (as

```
1: $\forall s \in S, v_0^*(s) \leftarrow g_s$; $t \leftarrow 0$
2: repeat
3:    $t \leftarrow t+1$
4:    for all $s \in S$ do
5:       for all $a \in A$ do
6:          $q_t(s,a) \leftarrow \sum_{s' \in S} p_{ss'}^a f_{ss'}^a \left(v_{t-1}^*(s')\right)$
7:       end for
8:       $v_t^*(s) \leftarrow \max^{\mathcal{R}}_{a \in A} q_t(s,a)$
9:    end for
10: until $t = h$
```

Fig. 3. Functional Backward Induction

when $\mathcal{R} = \mathbb{R}^{\mathbb{R}}$ for instance). The conditions of Proposition 1 hold for all previous examples. When $\mathcal{R} = \mathbb{R}^{\mathbb{R}}$, it is obvious from the fact that $\succeq_{\mathcal{R}}$ is nothing but the pointwise dominance relation between functions. From now on, we will assume that the validity conditions of Proposition 1 hold. Then, Prop. 1 implies that the *functional backward induction* (Fig. 3) can be used to find an optimal policy.

Explicit Definition. Another natural way for constructing preferences over policies in FRMDPs is as follows. First, we define the value of a history $\gamma = (s_h, a_h, s_{h-1}, \ldots, a_1, s_0)$ by $r(\gamma) = f_{s_h s_{h-1}}^{a_h} \circ f_{s_{h-1} s_{h-2}}^{a_{h-1}} \circ \ldots \circ f_{s_1 s_0}^{a_1} \circ g_{s_0}$ where $\circ$ denotes the function composition operator. Note that $r(\gamma)$ is in $\mathcal{R}$. Then, as the application of a policy π in a state s induces a probability distribution, denoted P_s^π, over histories, the value of π in state s can be defined by:

$$\overline{v}_h^\pi(s) = \sum_\gamma P_s^\pi(\gamma) r(\gamma) \tag{5}$$

Using these value functions, we can compare policies. A policy π is preferred to a policy π' in state s at horizon h if and only if $\overline{v}_h^\pi(s) \succsim_{\mathcal{R}} \overline{v}_h^{\pi'}(s)$. Then, the value function of optimal policies in s at horizon h can be found as follows:

$$\overline{v}_h^*(s) = \max_\pi{}^{\mathcal{R}} \overline{v}_h^\pi(s) \tag{6}$$

Unfortunately, when one only assumes that $f_{ss'}^a$ is strictly increasing, (3) and (5) do not define the same functions in general, that is we do not have $v_h^\pi(s) = \overline{v}_h^\pi(s)$ for all policy π and all state s. Thus, optimal policies can be different in the two preference systems, which may be problematic. A linearity requirement on those reward update functions has to be enforced for the two preference systems to be equivalent. A function $f : \mathcal{R} \to \mathcal{R}$ is said to be *linear* (with respect to the mixture set) if and only if $\forall x, y \in \mathcal{R}, \forall p \in [0,1]$:

$$f(px + (1-p)y) = pf(x) + (1-p)f(y)$$

The following proposition formally states this result:

Proposition 2 *If all reward update functions are linear functions, then:*

(i) With (3), (5) can be computed recursively , i.e., $\overline{v}_t^\pi(s) = v_t^\pi(s) \quad \forall \pi, \forall s, \forall t$

If moreover Condition I holds and all reward update functions are strictly increasing, then:

(ii) With (4), (6) can be computed recursively, i.e., $\overline{v}_t^*(s) = v_t^*(s) \quad \forall s, \forall t$

Proof. Again, the proof of (i) can be done by induction to show that the value function defined by Equations 3 and that defined by Equations 5 are equal. Property (ii) is then a corollary of Proposition 1.

As a side note, one can also remark that $\mathcal{R}$ endowed with a mixture set, Condition I and the linearity requirement implies that FRMDPs are algebraic MDPs [13]. ■

For the previous examples, the linearity condition is satisfied. This is obvious for the coffee robot problem. When $\mathcal{R} = \mathbb{R}^{\mathbb{R}}$, for instance for risk-sensitive MDPs, we can check that $\forall f, g \in \mathcal{R}$, $\forall p \in [0,1]$, $\forall x \in \mathbb{R}$,

$$\begin{aligned} f^a_{ss'}\big(pf + (1-p)g\big)(x) &= \big(pf + (1-p)g\big)(r^a_{ss'} + x) \\ &= pf\big(r^a_{ss'} + x\big) + (1-p)g\big(r^a_{ss'} + x\big) \\ &= pf^a_{ss'}\big(f\big)(x) + (1-p)f^a_{ss'}\big(g\big)(x) \end{aligned}$$

Generalization When $\max^{\mathcal{R}}$ Does Not Exist. In this section, we relax the requirements imposed on $\mathcal{R}$. We do not assume anymore that there exists an operator $\max^{\mathcal{R}}$. The only assumption that we make on $\mathcal{R}$ is that it is partially ordered by an order relation $\succeq_{\mathcal{R}}$. For instance, in this setting, one can think of multicriteria problems with $\mathcal{R} = \mathbb{R}^k$ (where k is the number of criteria) and $\succeq_{\mathcal{R}}$ the Pareto dominance[1]. Clearly, in such an example, $\max^{\mathcal{R}}$ is not defined.

In this generalized framework, propositions similar to the previous ones can be proved. We first give a few notations. The set of maximal elements of a set with respect to an order relation $\succeq_{\mathcal{R}}$ is denoted $\forall Y \subseteq \mathcal{R}, \max^{\succeq_{\mathcal{R}}}(Y) = \{y \in Y : \forall z \in Y, \text{ not}(z \succ_{\mathcal{R}} y)\}$. Furthermore, we denote by $\mathcal{P}^*(\mathcal{R}, \succeq_{\mathcal{R}})$ the set $\{Y \subseteq \mathcal{R} : Y = \max^{\succeq_{\mathcal{R}}}(Y)\}$. $\max^{\succeq_{\mathcal{R}}}$ can be seen as a binary operator, i.e. $\forall X, Y \in \mathcal{P}^*(\mathcal{R}, \succeq_{\mathcal{R}})$, $\max^{\succeq_{\mathcal{R}}}(X, Y) = \max^{\succeq_{\mathcal{R}}}(X \cup Y)$. Besides, for any function $f : \mathcal{R} \to \mathcal{R}$, for any $Y \subseteq \mathcal{R}$, $f(Y) = \{f(y) : y \in Y\} \subseteq \mathcal{R}$.

All the propositions that we have presented previously can now be written by replacing $\mathcal{R}$ by space $\mathcal{P}^*(\mathcal{R}, \succeq_{\mathcal{R}})$ which is endowed with $\max^{\succeq_{\mathcal{R}}}$ seen as a binary operator. We just write the equations for finding the maximal value functions, which are elements of the following sets:

$$\begin{aligned} &V^*_0(s) = \{g_s\} \\ &V^*_t(s) = \max\nolimits_{a \in A}{}^{\succeq_{\mathcal{R}}} \{\textstyle\sum_{s' \in S} p^a_{ss'} f^a_{ss'}\big(v(s')\big) : v \in V^*_{t-1}\} \end{aligned}$$

where $V^*_{t-1} = \{v \in \mathcal{R}^S : \forall s \in S, v(s) \in V^*_{t-1}(s)\}$. In this more general setting, one can then formulate a *generalized functional backward induction* (Fig. 4)

Discussion. As noted before, FRMDPs may be reformulated as MDPs by state augmentation. However, this may not be a natural way for representing preferences in the sequential decision-making problem at hand. We argue that the general FRMDP framework is more suitable when preferences become a bit sophisticated. First, it allows more flexibility into the modeling of preferences by allowing functional rewards. Moreover, it clearly uncouples, when modeling a problem, the dynamic of the system and the preference structure.

In practice, for being able to apply efficiently the functional backward induction or the functional value iteration, one exploits specific properties of set $\mathcal{R}$,

[1] $(x_1, \ldots, x_k) \succeq_{\mathcal{R}} (y_1, \ldots, y_k) \Leftrightarrow \forall i = 1, \ldots, k, x_i \geq y_i$.

```
1: $\forall s \in S, V_0^*(s) \leftarrow \{g_s\}$; $t \leftarrow 0$
2: **repeat**
3:     $t \leftarrow t + 1$
4:     **for all** $s \in S$ **do**
5:         **for all** $a \in A$ **do**
6:             $Q_t(s,a) \leftarrow \max_{v \in V_{t-1}^*}^{\succeq_{\mathcal{R}}} \sum_{s' \in S} p_{ss'}^a f_{ss'}^a\Big(v(s')\Big)$
7:         **end for**
8:         $V_t^*(s) \leftarrow \max_{a \in A}^{\succeq_{\mathcal{R}}} Q_t(s,a)$
9:     **end for**
10: **until** $t = h$
```

Fig. 4. Generalized Functional Backward Induction

together with the nature of operator $\max^{\mathcal{R}}$ (or relation $\succeq_{\mathcal{R}}$). For instance, if $\mathcal{R}$ is the set of piecewise-linear real functions and $\max^{\mathcal{R}}$ is the pointwise maximum operation, Boyan and Littman [3] (in the setting of time-dependent MDPs) discuss the restrictions needed for exact computations, and suggest efficient data structures to represent and manipulate the subsequent (piecewise-linear) value functions. More generally, if $\mathcal{R}$ is the set of real functions, these real functions may be approximated by piecewise-linear functions, and algorithms dedicated to this latter type of functions can be applied to find approximate solutions by adapting the work of Liu and Koenig [14].

Interestingly, note that the class of piecewise linear functions is not the only one for which efficient algorithms can be designed. Consider a certain class $\mathcal{C}$ of real functions such that $\mathcal{C}$ is a real space vector. A real function f is called *piecewise-$\mathcal{C}$* if the real line can be partitioned into intervals such that f restricted on each interval is in $\mathcal{C}$, i.e., $\exists n \in \mathbb{N}^*, \exists f_1, \ldots, f_n \in \mathcal{C}, \exists w_0, \ldots, w_n \in \mathbb{R} \cup \{-\infty, +\infty\}, w_0 < \ldots < w_n, \forall i = 1, \ldots, n, \forall x \in]w_{i-1}, w_i], f(x) = f_i(x)$. Obviously, any function in $\mathcal{C}$ is piecewise-$\mathcal{C}$. Let $PW\mathcal{C}$ denote the set of real functions that are piecewise-$\mathcal{C}$.

If we take $\mathcal{R} = PW\mathcal{C}$, value functions are elements of $PW\mathcal{C}^S$, the set of functions from S to $PW\mathcal{C}$. We define operator L from $PW\mathcal{C}^S$ to $PW\mathcal{C}^S$ by:

$$(Lv)(s) = \max_{a \in A}{}^{\mathcal{R}} \sum_{s' \in S} p_{ss'}^a f_{ss'}^a(v)$$

where $v \in PW\mathcal{C}^S$ and $s \in S$. With this operator, (4) can simply be rewritten:

$$v_0^*(s) = g_s$$
$$v_t^*(s) = L(v_{t-1}^*)(s) \quad \forall t \leq h$$

It is then easy to see that the real space vector $PW\mathcal{C}^S$ is closed under operator L, i.e., $Lv \in PW\mathcal{C}^S$ for any $v \in PW\mathcal{C}^S$.

This framework is a generalization of piecewise-linear functions. Other classes of functions may be convenient for efficient computations, e.g., the class of piecewise-linex function[2] [14]. Another interesting class is $\mathcal{C} = P_2$, the set of polynomials of degree 2, as the application of operator L on functions in PWP_2 can be computed efficiently. We detail how to proceed in the next first example.

[2] A linex function is the sum of a linear function and an exponential function.

4 Illustrative Examples

Our framework is very general as it encompasses many previously proposed models. We now present three new examples to illustrate the usefulness of our results. In standard MDPs, one wants to find a policy that maximizes the expected cumulated rewards. However, in high-stake planning situations, it is sensible to take also into account the variability of the total reward received from the initial state. A way to tackle this issue is to compute a policy optimizing the *expected utility* (EU) of the total reward. Expected utility is very popular in decision theory, as it enables to simply model risk aversion by concavity of the utility function ($\lambda u(x) + (1-\lambda)u(x') \leq u(\lambda x + (1-\lambda)x')$ for $\lambda \in [0, 1]$). If utility function u is quadratic, i.e., $u(x) = bx^2 + cx + d$, risk aversion is therefore modeled by setting coefficient b to a negative value.

Deterministic Rewards. In this first example, we show how one can optimize EU with a quadratic utility function and deterministic rewards. Let $P_2 = \{f(x) = bx^2 + cx + d : b, c, d \in \mathbb{R}\}$ be the set of polynomials of degree at most 2. Let PWP_2 be the set of functions that are piecewise-P_2.

In order to operationally implement operator L, three primitives are required: one primitive implementing function $f^a_{s,s'}$, one for the linear combination of functions in PWP_2, and one for the pointwise maximum of functions in PWP_2.

For any states s, s' and any action a, function $f^a_{ss'}$ is defined here by $\forall v \in PWP_2, \forall x \in \mathbb{R}$, $f^a_{ss'}\big(v\big)(x) = v(r^a_{ss'} + x)$. By definition, any piecewise-P_2 function v is defined by a sequence (w_i, b_i, c_i, d_i) for $i = 1, \ldots, n$ where $\forall x \in \mathbb{R}$, $\exists j = 1, \ldots, n$, $x \in]w_{j-1}, w_j]$, $v(x) = b_j x^2 + c_j x + d_j$. Function $f^a_{ss'}(v)$ is given by the following sequence for $i = 1, \ldots, n$, $(w_i - r^a_{ss'}, b_i, 2b_i r^a_{ss'} + c_i, b_i {r^a_{ss'}}^2 + c_i r^a_{ss'} + d_i)$.

To obtain linear combinations of piecewise-quadratic functions, one first finds an interval partition of the real line such that on each interval, each PWP_2 function is in P_2, then computes linear combinations of coefficients b_i, c_i, d_i.

We now present how to compute the pointwise max of two functions in PWP_2. The case of more than two functions is obtained by induction. We present the computation in the general case, but this computation could be slightly simplified by using the fact that the functions are all increasing since they represent utility functions. Let $v, v' \in PWP_2$. We can assume that both functions are defined on the same interval partition, otherwise just take a finer interval partition. On each interval $]w, w']$, four cases can occur:

1. $v - v'$ has no root in $]w, w'[$, then the max on $]w, w']$ is v if $v - v'$ is positive in $]w, w'[$ and v' otherwise.
2. $v - v'$ has one root r in $]w, w'[$, then we obtain two P_2 functions (possibly identical), one defined on $]w, r]$, the other on $]r, w']$, equal to v or v' depending on the sign of $v - v'$ on these two intervals.
3. $v - v'$ has two roots $r_1 < r_2$ in $]w, w'[$, then we obtain three P_2 functions on the following three intervals $]w, r_1]$, $]r_1, r_2]$ and $]r_2, w']$ equal to v or v' depending on the sign of $v - v'$ in those intervals.
4. $v - v' = 0$, then the max on $]w, w']$ is simply v.

Note that the computation of the roots is of course very simple since it amounts to solve quadratic equations by using discriminants.

For this setting, we implemented the functional value iteration with piecewise-quadratic functions. On the classic problem of navigation of an autonomous agent in a grid, our experiments show that the functional value iteration is between 10% to 50% slower than standard value iteration, depending on the size of the problem and the quadratic utility function. This experimental observation indicates that functional value iteration could be reasonably exploited when a suitable class of functions is chosen.

Random Rewards. The second problem we present is a generalization of an optimal path problem in graphs with stochastic weights investigated by Loui [15]. The difference here is that the consequences of actions are stochastic. More specifically, this problem corresponds to an MDP where rewards are not known precisely. For illustration, consider a stochastic shortest problem where the duration of an action is not known with certainty. In this setting, one could model the durations with independent random variables. If we were optimizing the expected duration time, one could replace the random variables by their mean and solve the problem as a standard MDP. However, if instead, we optimize the expected utility of the total duration time, in order to take into account risk attitude, the problem does not boil down to a standard MDP anymore.

Such a problem can be formalized in the setting of FRMDP as follows: $\mathcal{R}$ is the set of real random variables; $f^a_{ss'}(X) = R^a_{ss'} + X$ where $R^a_{ss'} \in \mathcal{R}$; $g_s = 0$ where 0 is the null random variable.

Without any assumption on the random variables, the determination of a policy optimizing EU in this setting can be hard in the general case. We assume from now on that all rewards are independent Gaussian random variables. This property has nice consequences from the computational viewpoint: the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is completely characterized by its mean μ and its variance σ^2, and furthermore $\lambda_1 X_1 + \lambda_2 X_2 \sim \mathcal{N}(\lambda_1\mu_1 + \lambda_2\mu_2, \lambda_1^2\sigma_1^2 + \lambda_2^2\sigma_2^2)$ for two independent random variables $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$. Consequently, the sum of rewards obtained along a history and therefore the total reward obtained by applying a given policy are both Gaussian random variables. As underlined by [15], the expected utility of a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is equal to $d + c\mu + b\mu^2 + b\sigma^2$ if the utility function u is quadratic, i.e., $u(x) = bx^2 + cx + d$. As u is a utility function, we restrict its domain of definition to an interval on which it is increasing. Then, on that interval, the function $u' : \mathrm{I\!R}^2 \to \mathrm{I\!R}$ defined by $u'(x, y) = d + cx + bx^2 + by$ is increasing in its first argument. For risk aversion, b is negative, which means that u' is decreasing on its second argument. Consequently, for two Gaussian defined respectively by (μ, σ^2) and (μ', σ'^2), whenever $\mu \geq \mu'$ and $\sigma^2 \leq \sigma'^2$, one can conclude that the Gaussian defined by (μ, σ^2) is preferred. This observation suggests to use the following approach. The sequential decision-making problem can be formalized in the setting of FRMDPs as follows: $\mathcal{R} = \{(\mu, \sigma^2) : \mu \in \mathrm{I\!R}, \sigma^2 \in \mathrm{I\!R}^*_+\}$; $f^a_{ss'}(\mu, \sigma^2) = (\mu^a_{ss'} + \mu, {\sigma^a_{ss'}}^2 + \sigma^2)$ where $R^a_{ss'} \sim \mathcal{N}(\mu^a_{ss'}, {\sigma^a_{ss'}}^2)$; $g_s = (0, 0)$. We define $\succeq_{\mathcal{R}}$ as follows $(\mu, \sigma^2) \succeq_{\mathcal{R}} (\mu', \sigma'^2) \Leftrightarrow (\mu \geq \mu'$ and $\sigma^2 \leq \sigma'^2)$. Note that this order relation is partial. The mixture set on $\mathcal{R}$ is defined as follows: $p(\mu, \sigma^2) + (1-p)(\mu', \sigma'^2) = (p\mu + (1-p)\mu', p^2\sigma^2 + (1-p)^2\sigma'^2)$. Condition I holds in this context and one can easily check that functions $f^a_{ss'}$'s are linear and strictly increasing. Since order relation $\succsim_{\mathcal{R}}$ is partial, applying the Bellman optimality equations yields a set of maximal value functions. Finally, the value function that is optimal with respect to EU can be found by scanning this set.

Time-Dependent Rewards. As a final example, we simply note that various features of different previously proposed extensions of MDPs can be mixed and take into account simultaneously in FRMDPS. For instance, the optimization of expected utility in problems where rewards are time-dependent can naturally be expressed in our framework. However, to the best of our knowledge, no previous framework can model this kind of setting.

5 Conclusion

In this paper, we have proposed FRMDPs as a new general framework for modeling sequential decision-making problems under uncertainty when preferences are sophisticated. It generalizes many previously known propositions and encompasses new problems. We made explicit the conditions that allow the use of a functional backward induction algorithm. We showed in this paper that in some situations an FRMDP could be solved directly and efficiently. To illustrate our proposition, we showed its exploitation on three new problems that previous frameworks could not tackle.

References

1. Dean, T., Kaelbling, L., Kirman, J., Nicholson, A.: Planning with deadlines in stochastic domains. In: AAAI. Volume 11. (1993) 574–579
2. Littman, M.L., Dean, T.L., Kaelbling, L.P.: On the complexity of solving markov decision problems. In: UAI. (1995) 394–402
3. Boyan, J., Littman, M.: Exact solutions to time-dependent MDPs. In: NIPS. (2000) 1026–1032
4. Liu, Y., Koenig, S.: Risk-sensitive planning with one-switch utility functions: Value iteration. In: AAAI. (2005) 993–999
5. Regan, K., Boutilier, C.: Regret based reward elicitation for Markov decision processes. In: UAI. (2009) 444–451
6. Serafini, P.: Dynamic programming and minimum risk paths. European Journal of Operational Research **175** (2006) 224–237
7. Kreps, D., Porteus, E.: Temporal resolution of uncertainty and dynamic choice theory. Econometrica **46** (1978) 185–200
8. Kreps, D., Porteus, E.: Dynamic choice theory and dynamic programming. Econometrica **47** (1979) 91–100
9. Boutilier, C., Dearden, R., Goldszmidt, M.: Stochastic dynamic programming with factored representations. Artificial Intelligence **121** (2000) 49–107
10. Hernstein, I., Milnor, J.: An axiomatic approach to measurable utility. Econometrica **21** (1953) 291–297
11. Fishburn, P.: Utility theory for decision making. Wiley (1970)
12. Denardo, E., Rothblum, U.: Optimal stopping, exponential utility and linear programming. Mathematical Programming **16** (1979) 228–244
13. Perny, P., Spanjaard, O., Weng, P.: Algebraic Markov decision processes. In: IJCAI. Volume 19. (2005) 1372–1377
14. Liu, Y., Koenig, S.: Functional value iteration for decision-theoretic planning with general utility functions. In: AAAI. (2006) 1186–1193
15. Loui, R.: Optimal paths in graphs with stochastic or multidimensional weights. Communications of the ACM **26** (1983) 670–676