

# Multi-Criteria Approaches to Markov Decision Processes with Uncertain Transition Parameters

Dimitri Scheftelowitsch, Peter Buchholz  
Informatik IV, TU Dortmund, Germany  
{dimitri.scheftelowitsch,peter.buchholz}@cs.tu-dortmund.de

## Abstract

Markov decision processes (MDPs) are a well established model for planing under uncertainty. In most situations the MDP parameters are estimates from real observations such that their values are not known precisely. Different types of MDPs with uncertain, imprecise or bounded transition rates or probabilities and rewards exist in the literature. Commonly the resulting processes are optimized with respect to the most robust policy which means that the goal is to generate a policy with the best worst case behavior. However, implementing such a policy could lead to potential losses of reward in most situations. In general, one is interested in policies which *behave well* in all situations which results in a multi-objective view of decision making.

In this paper we consider policies for the discounted reward of MDPs with uncertain parameters. In particular, the approach is defined for bounded-parameter Markov decision processes (BMDPs) [GLD00]. In this setting the worst, best and average case performance of a policy is analyzed. The paper presents some theoretical results and algorithms to compute or approximate the convex hull of *pure* Pareto optimal policies in the value vector space.

## 1 Introduction

Markov decision processes are a common tool to describe decision situations in many different contexts, such as economy, artificial intelligence and planning [Put94]. The general idea is to specify a system by means of different *states* in which it can be, *actions* which a decision maker can perform to affect the (probabilistic) future behavior, and *rewards* or *costs* that depend on the state and decision. After an action has been chosen, the system changes its state depending on the action and the current state but not on the past behavior; transitions are, in general, randomized and defined by the system's properties.

However, modeling a (physical or biological or artificial) system suffers from several limitations, the most important of which are the inherent loss of precision that is introduced by measurement errors and discretization artifacts which necessarily happen due to objective limitations of measuring equipment or due to incomplete knowledge about the system behavior. Thus, the real probability distribution for transitions is in most cases an uncertain value which is given by either external parameters or confidence intervals. For the latter, the Markov

decision process model has been extended to bounded-parameter Markov decision processes (BMDPs) [GLD00] or the slightly more general classes of MDPs with incomplete or uncertain transition rates [SL73, WE94]. In these classes of MDPs, best-case and worst-case policies for the expected discounted reward measure have been considered.

Apart from the upper and lower bounds on the value vector, which follow from the uncertainty in the transition probabilities or rewards, also expected values can be of interest if a probability distribution for the parameters of the MDP is available. This reasoning is motivated by the requirement to infer not only “best-case” and “worst-case” performance but also to consider the performance on average and give the user a possibility to pick a policy that suits her or his preference for the best-case, worst-case and average-case performance simultaneously. By doing so, we depart into the realm of multi-objective optimization where more than one solution may be optimal, as solutions can be incomparable.

## 1.1 Related work

For a general introduction on MDP theory and solution methods, we recommend the book of Puterman [Put94]. The extension to bounded-parameter Markov decision processes is introduced and widely discussed in the works of Givan et al. [GLD00]. BMDPs are a specific subclass of MDPs with uncertain or imprecise transition rates proposed by Satia and Lave [SL73] and White and El-Deib [WE94]. MDPs with uncertainty have been extended even further, our results also apply to convex uncertainty sets discussed in [PLSVS13]. Until today several aspects of MDPs with imprecise parameters are considered in the literature [DSdB11, WKR13]. However, in almost all cases, the goal is to compute some robust policy which assures the best possible behavior in the worst case.

We analyze MDPs with parameter uncertainty in the context of multi-objective optimization which allows us to simultaneously optimize the worst, best and average case behavior. Solution methods for multi-objective Markov decision processes are discussed in [WdJ07, Whi82, CMH06, BN08, PW10]. Furthermore, in this work, we use results for stochastic games; their properties are widely discussed in the textbook [FV96]; the extension to multi-objective stochastic games is introduced in [CFK<sup>+</sup>13].

## 1.2 Our Contribution

In this paper we consider MDPs with uncertain parameters and in particular BMDPs as multi-objective optimization problems. The classes are slightly extended by adding expected parameter values in addition to bounds. In this setting we introduce methods to compute or approximate the set of Pareto optimal pure policies. We show the relation between this set and the set of all Pareto optimal policies. In contrast to general multi-objective optimization of MDPs, the multi-objective goal function of a BMDP has an additional structure since the value function has ordered components which can be exploited when computing optimal policies.

### 1.3 Structure of the Paper

The paper is structured as follows. In the next section we introduce the basic definitions and some first results. Section 3 describes optimization of the multi-objective goal function relative to a weight vector. Afterwards the multi-objective approach is introduced under different assumptions. Section 5 introduces algorithms to compute (sub)sets of Pareto optimal policies. We propose three algorithms computing different subsets of the set of Pareto optimal policies. Afterwards a small example is presented. The paper ends with the conclusions and a summary of future work for multi-objective BMDPs.

## 2 Definitions and first steps

First we introduce common notation and formalisms used in this paper.

**Notation** For a matrix  $M$ , we denote by  $m_{i,j}$  the entry in row  $i$  and column  $j$ . Vector identifiers, such as  $\mathbf{v}$ , appear in bold script, to distinguish them from scalars and matrices. For a natural number  $n$ , by  $[n]$  we designate the set  $\{1, 2, \dots, n\}$ . For multi-dimensional identifiers, such as matrices or vectors, the order relations  $\leq$  or  $\geq$  mean “less than or equal in all components” or, respectively, “greater than or equal in all components”.

We begin with the definition of Markov reward processes which are the basic stochastic process resulting from a MDP and a fixed policy.

**Definition 1** (Markov reward process). A *Markov reward process* (MRP) is a tuple  $(S, P, \mathbf{r})$ , where  $S = [n]$  is the (finite) state space,  $P \in \mathbb{R}^{n \times n}$  ( $P \geq 0$ ,  $P\mathbf{1} = \mathbf{1}$ ) is a stochastic *transition matrix* and  $\mathbf{r} \in \mathbb{R}_{\geq 0}^{n,1}$  is a non-negative reward vector.

For an initial state  $s_1 \in S$ , a MRP defines a sequence of random variables  $(X_t)_{t \in \mathbb{N}}$  where  $X_1 = s_1$ , and  $X_{i+1}$  for  $i \in \mathbb{N}$  is selected subject to the probability distribution  $\Pr[X_{i+1} = s \mid X_i = s'] = p_{s,s'}$ . Furthermore, it defines a sequence of random variables  $R_i$  ( $i \in \mathbb{N}$ ) where  $R_i = r_s$  for  $X_i = s$ .  $X_i$  is the  $i$ -th state and  $R_i$  the  $i$ -th reward. We consider here only MRPs and also MDPs with finite state spaces.

A Markov decision process defines in principle a set of MRPs where a decision maker can select a suitable MRP.

**Definition 2.** A *Markov decision process* is a tuple  $(S, A, T, R)$  where  $S = [n]$  is a (finite) set of states,  $A = [m]$  is finite set of *actions*,  $R: S \times A \rightarrow \mathbb{R}_{\geq 0}$  is a set of  $m$  reward vectors, and  $T = \{P^1, \dots, P^m\} \subset \mathbb{R}^{n \times n}$  is a set of  $m$  transition matrices with  $P^i \geq 0$  and  $P^i\mathbf{1} = \mathbf{1}$ .

For a sequence of actions  $(a_t)_{t \in \mathbb{N}} \in A$ , a MDP defines sequences of random variables  $(X_t)_{t \in \mathbb{N}}$  and  $(R_t)_{t \in \mathbb{N}}$  where  $X_1 = s_1$  is the initial state, and  $X_{i+1}$  for  $i \in \mathbb{N}$  is chosen subject to the probability distribution  $\Pr[X_{i+1} = s \mid X_i = s', a_i] = p_{s,s'}^{a_i}$ ; furthermore,  $R_t$  is defined as  $R_t = R(X_i, a_i)$ .

In the following discussion, we need individual rows of the transition matrix and define for a transition matrix  $P$  and state  $s$ ,  $\mathbf{p}_s$  as the  $s$ -th row of  $P$ .

To optimize the performance of a MDP, a *decision rule* or *policy* is needed. Formally, a policy is a function  $f: S^{\mathbb{N}} \times A \rightarrow \mathbb{R}$  that maps (finite) *histories* of states to probability distributions on the action space. We call a policy  $f$

- *stationary* if it depends only on the current state,
- *deterministic* if it always maps a history to a Dirac distribution, i.e.,  $f(\cdot, a) \in \{0, 1\}$ ,
- *pure* if it is stationary and deterministic,
- *mixed* if it is stationary, but not pure.

It is easy to see that a stationary policy  $\pi$  induces a Markov reward process with transition matrix  $P^{(\pi)}$  and reward vector  $\mathbf{r}^{(\pi)}$ , as the transition probabilities under  $\pi$  depend only on the current state. We denote by  $\mathbf{p}_s^\pi$  row  $s$  of  $P^\pi$  and by  $r_s^\pi$  the reward of state  $s$  under policy  $\pi$ .

States of MRPs or MDPs often describe an aggregated view of the real system such that the Markov property (i.e., the homogeneous and memoryless transition probabilities) is only approximately correct and transition probabilities and rewards are estimates resulting from measurements or the opinion of some experts. This implies that there is always uncertainty about the parameters of the model and also about the behavior of the real system according to some policy that has been derived from the MDP. The class of bounded-parameter MDPs (BMDPs) introduced in [GLD00] includes this uncertainty by considering intervals rather than point estimates for the parameters of MDPs. BMDPs contain implicitly a definition of a bounded-parameter MRP (BMRP).

**Definition 3** (Bounded-parameter Markov decision/reward process [GLD00]). A *bounded-parameter Markov reward process* (BMRP) is a tuple  $(S, P_\downarrow, \mathbf{r}_\downarrow)$  where  $S = [n]$  is a (finite) set of states,  $P_\downarrow = (P_\downarrow, P_\uparrow)$ ,  $0 \leq P_\downarrow \leq P_\uparrow$ ,  $P_\downarrow \mathbf{1} \leq \mathbf{1} \leq P_\uparrow \mathbf{1}$ , and  $\mathbf{r} = (\mathbf{r}_\downarrow, \mathbf{r}_\uparrow)$ ,  $0 \leq \mathbf{r}_\downarrow \leq \mathbf{r}_\uparrow$ . A BMRP defines a set of MRPs  $((S, P, \mathbf{r}) | P_\downarrow \leq P \leq P_\uparrow, P \mathbf{1} = \mathbf{1}, \mathbf{r}_\downarrow \leq \mathbf{r} \leq \mathbf{r}_\uparrow)$ .

Analogously, a *bounded-parameter Markov decision process* (BMDP) is a tuple  $(S, A, T_\downarrow, R_\downarrow)$  where  $S = [n]$  is a (finite) set of states,  $A = [m]$  is (finite) set of actions,  $R_\downarrow = ((\mathbf{r}_\downarrow^1, \mathbf{r}_\uparrow^1), \dots, (\mathbf{r}_\downarrow^m, \mathbf{r}_\uparrow^m))$  is a set of  $m$  reward vector pairs, and  $T_\downarrow = ((P_\downarrow^1, P_\uparrow^1), \dots, (P_\downarrow^m, P_\uparrow^m))$  is a set of  $m$  matrix pairs. For each  $a \in A$ ,  $(S, P_\downarrow^a, \mathbf{r}_\downarrow^a)$  is a BMRP.

We denote by  $\mathbf{p}_s^a \in \mathbf{p}_{s\downarrow}^a$  all vectors  $\mathbf{p}_s^a$  such that  $p_{s,s'\downarrow}^a \leq p_{s,s'}^a \leq p_{s,s'\uparrow}^a$  for all  $s' \in [n]$  and  $\sum_{s'=1}^n p_{s,s'}^a = 1$ . Similarly  $\mathbf{r}^a \in \mathbf{r}_\downarrow^a$  specifies all vector  $\mathbf{r}^a$  such that  $r_{s\downarrow}^a \leq r_s^a \leq r_{s\uparrow}^a$ . BMDPs define upper and lower bounds for the transition probabilities and rewards and allow one to analyze the worst and best case behavior. Additionally, we wish to take the “average performance” into consideration and extend the formalism with a probability measure on the possible transition matrices and reward vectors.

**Definition 4** (Stochastic bounded-parameter Markov decision/reward process). For a bounded-parameter Markov reward process  $(S, P_\downarrow, \mathbf{r}_\downarrow)$  and a probability density function  $Pr$  on  $P_\downarrow, \mathbf{r}_\downarrow$ , a *stochastic bounded-parameter Markov reward process* (SMRP) is the tuple  $(S, P_\downarrow, \mathbf{r}_\downarrow, Pr)$ . By adding a probability measure on the transition matrices for each action and the rewards, a *stochastic bounded-parameter Markov decision process* (SBMDP) is defined as  $(S, A, T_\downarrow, R_\downarrow, Pr)$ .

The *expected discounted reward* for an infinite horizon is taken as performance metric. For an arbitrary sequence of rewards  $(r_t)_{t \in \mathbb{N}}$  and a *discount factor*  $\gamma \in [0, 1)$ , the sum  $\rho = \sum_{t \in \mathbb{N}} \gamma^{t-1} r_t$  defines the discounted reward. Since the rewards generated by a given Markov decision process are bounded and the discount factor is smaller than 1, the sum converges to a finite value. For a MDP and a state  $s \in S$ , a given policy  $f$  defines a sequence of transition matrices  $(P_t^{(f)})_{t \in \mathbb{N}}$  and reward vectors  $(\mathbf{r}_t^{(f)})_{t \in \mathbb{N}}$  where  $P_t^{(f)}$  resp.  $\mathbf{r}_t^{(f)}$  is the transition matrix resp. the reward in the  $t$ -th time step. Using this sequence, we can derive a probability distribution on sequences of states and corresponding rewards, from which the *expected discounted reward* can be defined as

$$\text{Ex} \left[ \sum_{t \in \mathbb{N}} \left( \gamma^{t-1} \mathbf{r}_t^{(f)} \right) \right] = \sum_{t \in \mathbb{N}} \gamma^{t-1} \text{Ex} \left[ \mathbf{r}_t^{(f)} \right].$$

By computing the expected discounted reward for all states, one obtains a *value vector*  $\mathbf{v}$  where each entry equals the expected discounted reward one obtains starting from the respective state. The mathematical properties of expected discounted rewards on which we shall rely in the sequel are widely discussed in [Put94].

We assume that the goal is the maximization of the discounted reward. For a BMDP policies assuring best and worst case behavior can be determined by solving the following Bellman equations.

$$\begin{aligned} v_{s\downarrow} &= \max_{a \in A} \left( r_{s\downarrow}^a + \min_{\mathbf{p}_s^a \in \mathbf{p}_{s\downarrow}^a} (\mathbf{p}_s^a \mathbf{v}_{\downarrow}) \right) \\ v_{s\uparrow} &= \max_{a \in A} \left( r_{s\uparrow}^a + \max_{\mathbf{p}_s^a \in \mathbf{p}_{s\uparrow}^a} (\mathbf{p}_s^a \mathbf{v}_{\uparrow}) \right) \end{aligned} \quad (1)$$

As shown in [GLD00] the minimum and maximum inside the equations can be solved easily for BMDPs such that the computation of the vectors  $\mathbf{v}_{\downarrow}$  and  $\mathbf{v}_{\uparrow}$  is a standard MDP problem which can be solved with value iteration, policy iteration or linear programming [Put94]. The optimal policies are pure and denoted as  $\pi_{\downarrow}$  and  $\pi_{\uparrow}$ , respectively.

In the extended model (cf. Definition 4) we are additionally interested in the “average-case” properties of a SBMDP  $(S, A, T_{\downarrow}, R, Pr)$  and a policy that maximizes the performance under the “average-case” assumption. The reasoning here is fairly straightforward: Since the probability distribution for the transition matrices is known, we can define for each action  $a \in A$ ,  $\bar{P}^a = \text{Ex}[P^a] = \int_{P^a} P \cdot p_a(P) dP$  and  $\bar{\mathbf{r}}^a = \text{Ex}[\mathbf{r}^a] = \int_{\mathbf{r}^a} \mathbf{r} \cdot p_a(\mathbf{r}) d\mathbf{r}$ . The Bellman equations become

$$\bar{v}_s = \max_{a \in A} (\bar{r}_s^a + \bar{\mathbf{p}}_s^a \bar{\mathbf{v}}) \quad (2)$$

which is a standard MDP problem. The optimal policy is pure and is denoted as  $\bar{\pi}$ . For some policy  $f$ , by  $\mathbf{v}_{\downarrow}^f$ ,  $\mathbf{v}_{\uparrow}^f$  and  $\bar{\mathbf{v}}^f$  we designate the lower, upper and average value vectors. Obviously  $\mathbf{v}_{\downarrow}^f \leq \bar{\mathbf{v}}^f \leq \mathbf{v}_{\uparrow}^f$  and also  $\mathbf{v}_{\downarrow} \leq \bar{\mathbf{v}} \leq \mathbf{v}_{\uparrow}$  hold. Usually the policies  $\pi_{\downarrow}$ ,  $\pi_{\uparrow}$  and  $\bar{\pi}$  differ and a compromise between two or all three goals have to be found. We consider in the sequel mainly policies that simultaneously consider  $\mathbf{v}_{\downarrow}$  and  $\bar{\mathbf{v}}$  but the corresponding approaches can be easily extended to two other vectors or to all three or even more vectors. Before we introduce different approaches to handle the problem, a few remarks should be made.

**Extended models** The BMDP framework is not the only way to describe parameter uncertainty in Markov decision processes. An alternative is to parameterize the uncertainty sets of transition matrices with variables  $\lambda_1, \dots, \lambda_k$  such that for each state  $s$  and each action  $a$ , there is a function  $h_{s,a}: \mathbb{R}^k \rightarrow \mathbb{R}^n$  that maps the parameter space to a set of stochastic vectors. By restricting the functions  $h_{s,a}$  to simple classes, such as linear or affine functions, one can handle this perspective in a similar fashion to the one presented in this paper although the computation of the minimum or maximum in (1) might be more complex and requires in extreme cases the solution of an LP problem itself.

**Continuous time models** We consider here MDPs in discrete time. MDPs in continuous time can be transformed into discrete time MDPs using uniformization [Ser79]. The resulting discrete time MDP shows an equivalent behavior with respect to the discounted reward. In a similar way, it is possible to transform BMDPs in continuous time into discrete time BMDPs using uniformization. We will not formalize the approach here but keep in mind that continuous time models can be handled as well with the following approaches.

**Mixed policies** The most evident approach here is to compute separate policies, an optimal policy  $\bar{\pi}$  for the expected Markov decision process, a worst-case optimal one  $\pi_{\downarrow}$  and a best case optimal policy  $\pi_{\uparrow}$  for the stochastic BMDP. Then, it is possible to define a mixed policy by defining a probability vector  $\mathbf{q} = (q_{\downarrow}, \bar{q}, q_{\uparrow})$  ( $q_{\downarrow} + \bar{q} + q_{\uparrow} = 1$ ). The mixed policy  $\pi_{\mathbf{q}}$  is the policy which, in every state, chooses with probability  $q_{\downarrow}$  the action defined by  $\pi_{\downarrow}$ , with probability  $\bar{q}$  the action defined by  $\bar{\pi}$  and with probability  $q_{\uparrow}$  the action defined by  $\pi_{\uparrow}$ . This seems to be a reasonable decision which seems to define a compromise between worst, average and best case. This is indeed the case for certain processes, e.g. if several expected values are considered [CMH06]. However, if the worst and best case behavior is analyzed, the resulting mixed policy is usually provably suboptimal since it results in the minimum of the worst and best case rewards of all three policies.

We show the mentioned effect by means of a small example where we consider only the average and worst case. Consider, for an arbitrary small  $\epsilon > 0$ , a BMDP with four states, a reward vector  $\mathbf{r} = (0, 2 + 2\epsilon, 1, 0)^T$  that is shared by all actions, and two actions given by the transition matrices

$$P^1_{\uparrow} = \left\{ \left( \begin{array}{cccc} 0 & \lambda & 0 & 1 - \lambda \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right) \mid \lambda \in [0, 1] \right\}, P^2_{\downarrow} = \left\{ \left( \begin{array}{cccc} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right) \right\}$$

where  $\lambda$  is uniformly distributed on  $[0, 1]$ . It is easy to see that choosing the second action in state 1 is pessimistically optimal; the optimal average-case policy will, however, choose the first action in state 1. The resulting rewards are  $(\bar{r}_1, r_{1\downarrow}) = (1 + \epsilon, 0)$  and  $(1, 1)$  for choosing the first and second action, respectively. A mixed policy will choose the first action in state 1 with probability  $q$  and thus, risk to gain a reward of zero with combined probability  $\frac{q}{2}$  over the values of  $\lambda$  and the result of the coin toss. One can see that with the combined policy, one can only (deterministically) guarantee the lower bounds that result from the average-case policy while the pessimistically optimal policy will only be used

with a certain probability. Thus, mixed policies that rely on probabilistic choices cannot deliver strong deterministic performance guarantees. This is different from models with several expected rewards as in [CMH06] where mixed policies can be applied to reach convex linear combinations of the value vectors resulting from pure policies.

### 3 Optimization relative to a weight vector

To overcome the limitations of the naive mixing approach discussed in the previous section, we propose a derivation from value iteration by introducing a weight or preference vector  $\mathbf{w}$ . We assume that we have  $G$  different goals represented by value vectors. Then  $\mathbf{w}$  has  $G$  components and  $w_g \geq 0$ ,  $\sum_{g=1}^G w_g = 1$ . Thus, we build a convex linear combination of the value vectors. Specifically consider a stochastic BMDP  $(S, A, T_{\downarrow}, R_{\downarrow}, Pr)$  and a discount factor  $\gamma \in [0, 1)$ . We optimize two value vectors,  $\mathbf{v}_{\downarrow}$  and  $\bar{\mathbf{v}}$  simultaneously, by computing, for each state  $s \in S$  and the previously fixed weight  $w \in [0, 1]$ ,

$$a_s^* = \arg \max_{a \in A} \left[ w (\bar{r}_s^a + \gamma \bar{\mathbf{p}}_s^a \cdot \bar{\mathbf{v}}) + (1 - w) \left( r_{s\downarrow}^a + \gamma \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} \mathbf{p}_s^a \cdot \mathbf{v}_{\downarrow} \right) \right] \quad (3)$$

and derive from the decisions new value vectors

$$\bar{\mathbf{v}} = \bar{\mathbf{r}}^{\pi^*} + \gamma \bar{P}^{\pi^*} \cdot \bar{\mathbf{v}}, \quad \mathbf{v}_{\downarrow} = \mathbf{r}_{\downarrow}^{\pi^*} + \gamma \min_{\mathbf{P}^{\pi^*} \in \mathbf{P}_{\downarrow}^{\pi^*}} P^{\pi^*} \cdot \mathbf{v}_{\downarrow}. \quad (4)$$

where  $\pi^* = (a_1^*, \dots, a_n^*)$ . These steps define an iterative procedure and we show that the mapping

$$K(\mathbf{v}_1, \mathbf{v}_2)(s) := \left( \bar{r}_s^{a^*} + \gamma \bar{\mathbf{p}}_s^{a^*} \cdot \mathbf{v}_1, r_{s\downarrow}^{a^*} + \gamma \min_{\mathbf{p}_s^{a^*} \in \mathbf{P}_{s\downarrow}^{a^*}} \mathbf{p}_s^{a^*} \cdot \mathbf{v}_2 \right)$$

converges and that the (pure) policy  $\pi^*$  that corresponds to its fixed point is optimal.

**Lemma 1.** *The mapping  $K$  as defined above is a contraction mapping for non-negative value vectors.*

*Proof.* First, for any norm  $\|\cdot\|$  on  $\mathbb{R}^n$  and a positive real weight  $w \in (0, 1)$ , we define a mapping  $\|\mathbf{x}, \mathbf{y}\|_{+,w}$  on  $\mathbb{R}^{2n}$  as  $w\|\mathbf{x}\| + (1-w)\|\mathbf{y}\|$ . We show that this mapping is a norm on  $\mathbb{R}^{2n}$ , as it is, using the norm properties of  $\|\cdot\|$ , for all  $a \in \mathbb{R}$  and all  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ ,

- $\|a\mathbf{x}, a\mathbf{y}\|_{+,w} = w\|a\mathbf{x}\| + (1-w)\|a\mathbf{y}\| = aw\|\mathbf{x}\| + a(1-w)\|\mathbf{y}\| = a\|\mathbf{x}, \mathbf{y}\|_{+,w}$ .
- $\|\mathbf{x}, \mathbf{y}\|_{+,w} = 0 \Leftrightarrow w\|\mathbf{x}\| + (1-w)\|\mathbf{y}\| = 0 \Leftrightarrow \mathbf{x} = \mathbf{y} = 0$ , as norms are always greater than zero if and only if the value is different from zero.
- $\|\mathbf{x}_1 + \mathbf{x}_2, \mathbf{y}_1 + \mathbf{y}_2\|_{+,w} = w\|\mathbf{x}_1 + \mathbf{x}_2\| + (1-w)\|\mathbf{y}_1 + \mathbf{y}_2\| \leq w\|\mathbf{x}_1\| + w\|\mathbf{x}_2\| + (1-w)\|\mathbf{y}_1\| + (1-w)\|\mathbf{y}_2\| = \|\mathbf{x}_1, \mathbf{y}_1\|_{+,w} + \|\mathbf{x}_2, \mathbf{y}_2\|_{+,w}$ .

As an auxiliary function, we define

$$L(\mathbf{v}_1, \mathbf{v}_2)(s, a) := w \left( \bar{r}_s^a + \gamma \bar{\mathbf{p}}_s^a \mathbf{v}_1 \right) + (1-w) \left( r_{s\downarrow}^a + \gamma \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} \mathbf{p}_s^a \cdot \mathbf{v}_2 \right)$$

Let  $s \in S$  be an arbitrary state. Then we consider the expression

$$\max_{a \in A} L(\mathbf{u}_1, \mathbf{u}_2)(s, a) - \max_{a \in A} L(\mathbf{v}_1, \mathbf{v}_2)(s, a)$$

which is the basis for computing  $\|K(\mathbf{u}_1, \mathbf{u}_2) - K(\mathbf{v}_1, \mathbf{v}_2)\|_{+,w}$ . This term is a special case of the expression  $\max_x f(x) - \max_y g(y)$  for two functions  $f, g: X \rightarrow \mathbb{R}$  on some set  $X$ . By definition, it is  $\max_x f(x) - \max_y g(y) \leq \max_x f(x) - g(y) = \max_x (f(x) - g(y))$ , for all  $y \in X$ . By setting  $y = x$ , we obtain the general inequality  $\max_x f(x) - \max_y g(y) \leq \max_x (f(x) - g(x))$ , which justifies the following operations.

$$\begin{aligned} & \max_{a \in A} L(\mathbf{u}_1, \mathbf{u}_2)(s, a) - \max_{a \in A} L(\mathbf{v}_1, \mathbf{v}_2)(s, a) \\ & \leq \max_{a \in A} (L(\mathbf{u}_1, \mathbf{u}_2)(s, a) - L(\mathbf{v}_1, \mathbf{v}_2)(s, a)) \\ & \leq w \max_{a \in A} \gamma [\bar{\mathbf{p}}_s^a \mathbf{u}_1 - \bar{\mathbf{p}}_s^a \mathbf{v}_1] + (1-w) \max_{a \in A} \gamma \left[ \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} \mathbf{p}_s^a \mathbf{u}_2 - \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} \mathbf{p}_s^a \mathbf{v}_2 \right] \\ & \leq \gamma (w \|\mathbf{u}_1 - \mathbf{v}_1\| + (1-w) \|\mathbf{u}_2 - \mathbf{v}_2\|) \\ & \leq \gamma \|(\mathbf{u}_1, \mathbf{u}_2) - (\mathbf{v}_1, \mathbf{v}_2)\|_{+,w} \end{aligned}$$

From these computations it also follows that for  $(\mathbf{w}_1, \mathbf{w}_2) = K(\mathbf{u}_1, \mathbf{u}_2) - K(\mathbf{v}_1, \mathbf{v}_2)$ , it is  $w \|\mathbf{w}_1\| \leq \gamma w \|\mathbf{u}_1 - \mathbf{v}_1\|$  and  $(1-w) \|\mathbf{w}_2\| \leq \gamma (1-w) \|\mathbf{u}_2 - \mathbf{v}_2\|$ . Hence, we can derive the contraction property

$$\|K(\mathbf{u}_1, \mathbf{u}_2) - K(\mathbf{v}_1, \mathbf{v}_2)\|_{+,w} \leq \gamma \|(\mathbf{u}_1, \mathbf{u}_2) - (\mathbf{v}_1, \mathbf{v}_2)\|_{+,w}$$

For the cases  $w = 0$  or  $w = 1$  the desired properties also hold: For  $w = 0$ , the algorithm reduces itself to the standard value iteration and for  $w = 1$ , the algorithm reduces itself to interval value iteration. In both cases, the contraction mapping properties have been shown [Put94, GLD00].  $\square$

To prove correctness, we show now that the optimal policy (relative to the chosen measure) is the fixed point of the value iteration scheme.

**Theorem 1.** *The fixed point of  $K$  is an optimal value vector relative to the weighted sum of the average-case and the worst-case transition probabilities  $\mathbf{v}^* = (\mathbf{v}_1^*, \mathbf{v}_2^*)$  and the corresponding policy  $\pi^*$ .*

*Proof.* The existence of the fixed point follows from Banach's fixed point theorem and Lemma 1 that shows that  $K$  is a contraction mapping.

Suppose for the sake of contradiction that for  $\mathbf{u} = K(\mathbf{v}^*)$ ,  $\mathbf{u} \neq \mathbf{v}^*$ . Then there exists a state  $s \in S$  such that  $u_s \neq v_s^*$ . Now we can consider several cases.

**Case 1** If the policy  $\pi$  that results from application of  $K$  is the same as the by assumption optimal policy  $\pi^*$ , then we interpret the BMDP under  $\pi$  as a BMRP; thus,  $\mathbf{v}^*$  is the value vector of a BMRP if and only if  $\mathbf{v}^*$  satisfies the equation  $\mathbf{v}^* = \left( \bar{\mathbf{r}}^{(\pi)} + \gamma \bar{P}^{(\pi)} \mathbf{v}_1^*, \mathbf{r}_{\downarrow}^{(\pi)} + \gamma \min_{P^{(\pi)} \in \mathbf{P}^{(\pi)}_{\downarrow}} P^{(\pi)} \mathbf{v}_2^* \right)$ . Thus, in this case  $\mathbf{u} \neq \mathbf{v}^*$  leads to a contradiction.



**Case 2.1** If the fixed-point policy  $\pi$  and vector  $\mathbf{u}^* = (\mathbf{u}_1^*, \mathbf{u}_2^*)$  that result from the application of  $K$  are different from  $\pi^*$  and  $\mathbf{v}^*$  and it is  $wv_{s,1}^* + (1-w)v_{s,2}^* < wu_{s,1}^* + (1-w)u_{s,2}^*$ , then  $\pi'$ , the policy that results from one application of  $K$  to  $\mathbf{v}^*$  is a policy that delivers a higher sum of worst-case and average-case rewards and thus,  $\pi^*$  cannot be the optimal policy.

**Case 2.2** If the fixed-point policy  $\pi$  and vector  $\mathbf{u}^* = (\mathbf{u}_1^*, \mathbf{u}_2^*)$  that result from the application of  $K$  are different from  $\pi^*$  and  $\mathbf{v}^*$  and it is  $wv_{s,1}^* + (1-w)v_{s,2}^* > wu_{s,1}^* + (1-w)u_{s,2}^*$ , then there is an action  $a \in A$  such that  $w(\bar{r}_s^a + \gamma \bar{\mathbf{p}}_s^a \cdot \mathbf{v}_1^*) + (1-w)(r_{s\downarrow}^a + \gamma \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} \mathbf{p}_s^a \cdot \mathbf{v}_2^*) > wv_{s,1}^* + (1-w)v_{s,2}^*$ . However, then it follows that the updated policy  $\pi'$  with action  $a$  in state  $s$  is better than  $\pi^*$ , as  $\pi^*$  was chosen to be optimal for all choices of  $P(\pi)$ . This is a contradiction to the assumption that  $\pi^*$  is the policy for which, in every state, and thus also in state  $s$ , the sum  $wv_{s,1}^* + (1-w)v_{s,2}^*$  is maximal. □

As we can see, the proposed policy iteration scheme converges to a unique value vector with the maximal sum of the worst-case and expected rewards. With such a policy, one can guarantee a lower bound for the sum of the worst-case and the average-case value vectors. Furthermore, the convergence properties are, as with all contraction mappings, such that the fixed point will be reached after a polynomial number of iterations, for a polynomial number of bits in the representation of a number. It is easy to see that one application of  $K$  can be evaluated in polynomial time, so, the total time complexity of this policy iteration-based procedure is polynomial. We note that this does not contradict the NP-hardness result of [CMH06] for multi-objective MDPs since the hardness result is shown for the problem of deciding if there is a policy whose value vector  $\mathbf{v}$  dominates a given vector  $\mathbf{w}$ .

**Corollary 1.** *Optimizing the weighted sum of the minimal and the expected discounted total rewards can be done in polynomial time in the size of the BMDP<sup>1</sup>.*

As a side result, we observe that the resulting optimal policies are pure. This means that the convex hull of Pareto optimal policies, i.e., of the policies whose value vectors are not dominated by value vectors of other policies is spanned by pure policies.

With this result, it is possible to compute policies that are, in a sense, controllably robust, i.e., have a performance measure that incorporates the worst-case as well as the best-case performance. To extend our discussion further, it is possible to include the upper bound of the value vector as a third component of the performance measure. It is easy to see that the reasoning used above also applies for the upper bound component, and thus, we can compute policies that also consider the best-case expected discounted reward.

<sup>1</sup>The size of the BMDP is measured in  $n \cdot m$  and the complexity of computing  $\max_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a}$  which can be done with an effort in  $\mathcal{O}(n)$  for BMDPs

## 4 Multi-Objective Approaches

An intuitive expansion of the preceding discussion is to generalize the considered problem as a variant of a multi-objective Markov decision process problem, with the components being the worst-case, best-case, or average-case expected discounted rewards. It is tempting to consider known techniques for multi-objective optimization in this context.

For a given weight (3) and (4) describe a variant of value iteration which converges to a optimal pure policy. We denote the corresponding policy as  $\pi^{\mathbf{w}}$ . However, if  $\mathbf{w}$  is not known a priori, one is usually interested in the set of pure policies which are optimal for some weight vector. More formally, let  $\mathcal{W} = \{\mathbf{w} = (w_{\downarrow}, \bar{w}, w_{\uparrow}) \mid \mathbf{w} \geq \mathbf{0}, \mathbf{w}\mathbf{1} = 1\}$  the set of weight vectors and let  $\mathcal{P}_{pure}$  be the set of pure policies. Then define

$$\mathcal{P}_{weight} = \{\pi \mid \pi \in \mathcal{P}_{pure} \exists \mathbf{w} \in \mathcal{W} : \pi = \pi^{\mathbf{w}}\}$$

as the set of pure policies that are optimal for some weight vector. Furthermore, define

$$\begin{aligned} \mathcal{P}_{Pareto} = & \\ & \left\{ \pi \mid \pi \in \mathcal{P}_{pure}, \neg \exists \pi' \in \mathcal{P}_{pure} : \mathbf{v}_{\downarrow}^{\pi} \leq \mathbf{v}_{\downarrow}^{\pi'} \wedge \bar{\mathbf{v}}^{\pi} \leq \bar{\mathbf{v}}^{\pi'} \wedge \mathbf{v}_{\uparrow}^{\pi} \leq \mathbf{v}_{\uparrow}^{\pi'} \wedge \right. \\ & \left. \left( \mathbf{v}_{\downarrow}^{\pi} \neq \mathbf{v}_{\downarrow}^{\pi'} \vee \bar{\mathbf{v}}^{\pi} \neq \bar{\mathbf{v}}^{\pi'} \vee \mathbf{v}_{\uparrow}^{\pi} \neq \mathbf{v}_{\uparrow}^{\pi'} \right) \right\} \end{aligned}$$

the set of pure policies that result in Pareto optimal value vectors. It is easy to see that  $\mathcal{P}_{weight} \subseteq \mathcal{P}_{Pareto} \subseteq \mathcal{P}_{pure}$  holds and usually the subsets are proper subsets. We denote by  $\mathcal{V}_{weight}$ ,  $\mathcal{V}_{Pareto}$  and  $\mathcal{V}_{pure}$  be the corresponding sets of value vectors.

Often the convex hull of Pareto optimal policies is considered in multi-objective MDPs (e.g., in [CMH06, BN08]). The assumption in those approaches is that a randomized policy which probabilistically chooses an action from one of several Pareto-optimal pure policies results in a value vector that is given by the convex linear combination of the value vectors of the pure policies. However, this only holds if several expected values are considered in a multi-objective setting. In our case we have for two pure policies  $\pi, \pi' \in \mathcal{P}_{Pareto}$  with value vectors  $(\mathbf{v}_{\downarrow}^{\pi}, \bar{\mathbf{v}}^{\pi}, \mathbf{v}_{\uparrow}^{\pi})$  and  $(\mathbf{v}_{\downarrow}^{\pi'}, \bar{\mathbf{v}}^{\pi'}, \mathbf{v}_{\uparrow}^{\pi'})$  that are combined to a randomized policy  $\tilde{\pi}$  which selects with probability  $q \in (0, 1)$  an action from policy  $\pi$  and with probability  $(1 - q)$  an action from policy  $\pi'$  that  $\bar{\mathbf{v}}^{\tilde{\pi}} = q\bar{\mathbf{v}}^{\pi} + (1 - q)\bar{\mathbf{v}}^{\pi'}$ ,  $\mathbf{v}_{\downarrow}^{\tilde{\pi}} = \min(\mathbf{v}_{\downarrow}^{\pi}, \mathbf{v}_{\downarrow}^{\pi'})$  and  $\mathbf{v}_{\uparrow}^{\tilde{\pi}} = \max(\mathbf{v}_{\uparrow}^{\pi}, \mathbf{v}_{\uparrow}^{\pi'})$  where the minimum and maximum are computed per state. Thus, the worst case and also best case behavior does not profit from randomized policies which makes them less attractive in our case.

It is of interest to consider a related, but different problem: Given minimal average and worst-case value vectors  $(\bar{\mathbf{u}}, \mathbf{u}_{\downarrow})$ , does there exist a pure policy  $\pi$  such that  $\bar{\mathbf{v}}^{\pi} \geq \bar{\mathbf{u}}$  and  $\mathbf{v}_{\downarrow}^{\pi} \geq \mathbf{u}_{\downarrow}$ ? It has been shown in [CMH06] that the corresponding decision problem is NP-hard for general multi-objective MDPs where different expected values are computed. The following theorem shows that the hardness result also holds for SBMDPs even if rewards do not depend on the chosen action.

**Theorem 2.** *The problem of deciding existence of a pure policy  $\pi$  for a given bounded-parameter Markov decision process that delivers a worst-case expected*

discounted reward  $\mathbf{v}_{\downarrow}^{(\pi)}$  and an average-case expected discounted reward  $\bar{\mathbf{v}}^{(\pi)}$  such that  $\mathbf{v}_{\downarrow}^{(\pi)} \geq \mathbf{v}_1, \bar{\mathbf{v}}^{(\pi)} \geq \mathbf{v}_2$  for given vectors  $\mathbf{v}_1, \mathbf{v}_2$  is NP-complete.

*Proof.* The stated problem is obviously in NP because the evaluation of a policy for a BMDP can be done in polynomial time. As for NP-hardness, we show a reduction from the subset sum problem. Given a subset sum instance  $M = \{m_1, \dots, m_n\}$  we construct a BMDP and two vectors  $\mathbf{v}_1, \mathbf{v}_2$  such that there is a policy  $\pi$  with  $\mathbf{v}_{\downarrow}^{(\pi)} \geq \mathbf{v}_1, \bar{\mathbf{v}}^{(\pi)} \geq \mathbf{v}_2$  if and only if there is a subset  $I \subseteq [n]$  such that  $\sum_{i \in I} m_i = \frac{1}{2} \sum_{i=1}^n m_i$ .

The BMDP which we shall construct will have a pre-defined discount factor  $\gamma$ ,  $4n + 1$  states which have the identifiers  $t$  and  $q_i, \bar{s}_i, s_{\downarrow i}, s_{\uparrow i}$  for  $i \in [n]$ . The general idea of the reduction is the following: The MDP shall proceed through all states  $q_i$  and end up in the absorbing state  $t$ . In the state  $q_i$ , it shall be possible to choose between the (adjusted for the discount factor) reward pairs  $(2m_i, 0)$  and  $(m_i, m_i)$  with the first component being the average case and the second component being the worst case part. This way, a pure policy  $\pi$  will induce a subset  $I \subseteq [n]$  such that the total reward, if one starts in state  $q_1$ , will be  $(\sum_{i=1}^n m_i + \sum_{i \in I} m_i, \sum_{i \notin I} m_i)$ . Technically, we model this by enabling two actions in states  $q_i$ ,  $a$  and  $b$ . These actions do not generate rewards, but lead to different outcomes: The action  $a$  leads to the state  $\bar{s}_i$  unconditionally with reward 0, and all actions from  $\bar{s}_i$  lead to the state  $q_{i+1}$  with reward  $\frac{m_i}{\gamma^{2^i-1}}$ . The action  $b$  leads to either  $s_{\downarrow i}$  or  $s_{\uparrow i}$ , with probability in the interval  $[0, 1]$ ; all actions from these two states lead, in turn, to  $q_{i+1}$ , and the difference between  $s_{\downarrow i}$  and  $s_{\uparrow i}$  lies in the reward: the reward in  $s_{\downarrow i}$  is 0, the reward in  $s_{\uparrow i}$  is  $\frac{4m_i}{\gamma^{2^i-1}}$ .

Finally, we define the expected discounted rewards we would like to get with a pure policy  $\pi$ , it should be  $\frac{1}{2} \sum_{i \in [n]} m_i$  in the worst case and  $\frac{3}{2} \sum_{i \in [n]} m_i$  in the average case.

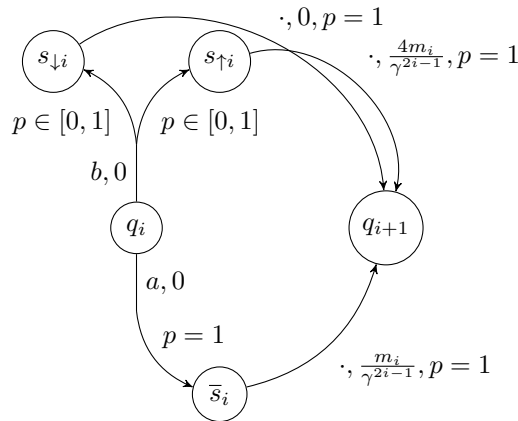


Figure 1: Construction from Theorem 2

It is easy to see that the construction can be done in polynomial time. We want now to show correctness. For every subset  $I \subseteq [n]$  we define a policy  $\pi_I$  with  $\pi_I(q_i) = a$  if  $i \in I$  and  $\pi_I(q_i) = b$  if  $i \notin I$ . The expected discounted total reward for policy  $\pi$  in state  $q_1$  will then be  $\sum_{i \in I} \gamma^{2^i-1} \cdot \frac{m_i}{\gamma^{2^i-1}} + \sum_{i \in [n] \setminus I} \gamma^{2^i-1} \cdot \frac{2m_i}{\gamma^{2^i-1}} =$

$\sum_{i \in [n]} m_i + \sum_{i \in [n] \setminus I} m_i$  in the average case and  $\sum_{i \in I} \gamma^{2i-1} \cdot \frac{m_i}{\gamma^{2i-1}} = \sum_{i \in I} m_i$  in the worst case. If there is a subset  $I \subseteq [n]$  such that  $\sum_{i \in I} m_i = \sum_{i \in [n] \setminus I} m_i$ , then there is a policy  $\pi_I$  that delivers the requested expected discounted rewards. Furthermore, as any pure policy  $\pi$  induces a subset  $I \subseteq [n]$  by defining  $i \in I \Leftrightarrow \pi(q_i) = a$ , the existence of a pure policy  $\pi$  with the requested expected discounted rewards implies the existence of a subset with sum  $\frac{1}{2} \sum_{i \in [n]} m_i$ .  $\square$

Up to now we have only considered optimal stationary policies and in our setting pure policies are of special importance. However, if we allow general policies which consider the history, then additional Pareto optimal solutions may be found. This will be shown by a simple example (shown in Fig. 2). The SBMDP has two states, in state 1 one can choose between action  $a$  and  $b$ . If  $a$  is selected the process goes with probability  $p$ , which is uniformly  $[0, 1]$  distributed, into state 2 and stays with probability  $1 - p$  in state 1. If  $b$  is selected, then the process makes with probability  $q$ , which is uniformly  $[0.5, 0.7]$  distributed, a transition to state 2 and stays with probability  $1 - q$  in state 1. From state 2 the process always returns to state 1 with probability 1 no matter which action has been chosen. The reward vector is  $\mathbf{r}^a = \mathbf{r}^b = (1, 0)^T$ .

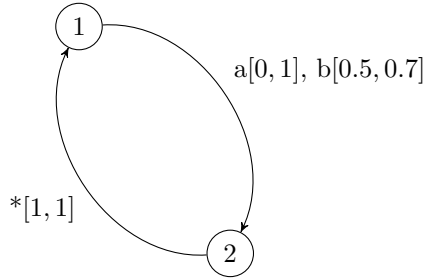


Figure 2: Simple example BMDP

Since decisions in state 2 are irrelevant, we have two pure policies namely ( $a$ ) which chooses always action  $a$  and ( $b$ ) that chooses always  $b$ . Obviously, ( $a$ ) is better for maximizing the average reward and  $b$  is better for the worst case reward. For  $\gamma = 0.9$  the corresponding values are  $\mathbf{v}^a = (\bar{v}_1^a, v_{1\downarrow}^a) = (6.8966, 5.2632)$  and  $\mathbf{v}^b = (\bar{v}_1^b, v_{1\downarrow}^b) = (6.4935, 6.1350)$ . Both vectors are Pareto optimal. Now consider the non-stationary policies  $ab$  and  $ba$  which collect alternating sequences of the actions  $a$  and  $b$ . We have  $\mathbf{v}^{ab} = (\bar{v}_1^{ab}, v_{1\downarrow}^{ab}) = (6.7024, 5.2632)$  and  $\mathbf{v}^{ba} = (\bar{v}_1^{ba}, v_{1\downarrow}^{ba}) = (6.6398, 5.5913)$ .  $\mathbf{v}^{ab}$  is dominated by  $\mathbf{v}^a$  whereas  $\mathbf{v}^{ba}$  is another Pareto optimal vector which cannot be reached with any stationary policy. By extending the history, we obtain additional Pareto optimal policies.

We define  $\mathcal{P}_{opt}$  as the set of Pareto optimal policies. In general  $\mathcal{P}_{opt}$  can be an infinite set but since decisions  $k$  steps apart receive a discounting factor  $\gamma^k$ , the influence past decisions vanishes exponentially. Again we denote by  $\mathcal{V}_{opt}$  the set of value vectors belonging to policies from  $\mathcal{P}_{opt}$ .

## 5 Computation of Optimal Policies

The computation of Pareto optimal policies in multi-objective MDPs is a challenge since the number of optimal policies can be large or even infinite. Several papers consider methods to compute or approximate the Pareto set (e.g., [PW10, BN08, CMH06, Whi82, WdJ07, RWO14]) but most of the methods can only be applied to small processes. Furthermore, the methods consider the computation of several expected values. In SBMDPs we have the worst and best case vectors which are harder to compute.

As in the single objective case, in principle value iteration, policy iteration or linear programming [Put94] can be applied in the multi-objective case. However, for SBMDPs linear programming is not recommended since the computation of minimum inside the maximum computation is not linear such that we end up with a problem with a linear goal function but non-linear constraints. In the cases we consider here, namely PBMDPs, the computation in the minimum can be transformed into a set of linear constraints but the number of additional variables is huge such that the approach becomes inefficient. In the following we consider three algorithms to compute, respectively approximate the sets  $\mathcal{V}_{opt}$ ,  $\mathcal{V}_{Pareto}$  and  $\mathcal{V}_{weight}$ . The first approach is based on value iteration, the other two on policy iteration.

**Pareto Optimal Policies from  $\mathcal{P}_{opt}$ :** The set  $\mathcal{V}_{opt}$  can be computed with value iteration based on the results of [BN08, Whi82]. Let  $V$  be a set of value vectors  $(\mathbf{v}_\downarrow, \bar{\mathbf{v}}, \mathbf{v}_\uparrow)$ .  $V_s = \{(v_{s\downarrow}, \bar{v}_s, v_{s\uparrow}) | (\mathbf{v}_\downarrow, \bar{\mathbf{v}}, \mathbf{v}_\uparrow) \in V\}$  is the subset that considers only the elements with index  $s$  from the vectors. Furthermore, we use  $V_{s\downarrow}$  for the subset containing only the first component of  $(v_\downarrow, \bar{v}, v_\uparrow) \in V_s$ ,  $\bar{V}_s$  for the set which contains only the second component and  $V_{s\uparrow}$  as the subset containing only the third component. For a set  $V_s$  we define

$$popt(V_s) = \{(u_{s\downarrow}, \bar{u}_s, u_{s\uparrow}) \in V_s | \neg \exists (v_{s\downarrow}, \bar{v}_s, v_{s\uparrow}) \in V_s : u_{s\downarrow} \leq v_{s\downarrow} \wedge \bar{u}_s \leq \bar{v}_s \wedge u_{s\uparrow} \leq v_{s\uparrow}\}$$

as the set of Pareto optimal pairs. Since  $V_s$  is a set,  $(u_\downarrow, \bar{u}, u_\uparrow), (v_\uparrow, \bar{v}, v_\downarrow) \in V_s$  implies  $u_{s\downarrow} \neq v_{s\downarrow}$  or  $\bar{u}_s \neq \bar{v}_s$  or  $u_{s\uparrow} \neq v_{s\uparrow}$ .

Algorithm 1 computes an approximation of the set  $\mathcal{V}_{opt}$  with a predefined error bound. Before we prove the error bound of the algorithm, a short explanation of the steps of the algorithm is given. The computation starts with three value vectors which are zero. Then for each state and action pair and each pair of value vectors that has been computed in the previous step new average and worst case values are computed. The newly computed pairs are added to the current set and all non-Pareto optimal pairs are removed from the resulting set (step 10). Iteration  $k$  mimics the computation of policies that consider a history of  $k$  steps and assume that the average and worst case reward  $k$  steps before was 0 in any state. In step 11/12 the complete set of Pareto optimal vectors is computed. It is obviously possible to combine the value pairs per state arbitrarily because decisions are made independently in any state. However, the set  $V^k$  needs not to be computed explicitly, it is sufficient to store the sets  $V_s^k$ .

The algorithm computes only the value vectors and not the optimal policies. It can be easily extended to generate also the policies by storing action  $a$  and vectors  $(\mathbf{v}_\downarrow, \bar{\mathbf{v}}, \mathbf{v}_\uparrow)$  with the value pair  $u_\downarrow, \bar{u}, u_\uparrow$  in the steps 7 – 10. In this way

---

**Algorithm 1** Value iteration to approximate  $\mathcal{V}_{opt}$ 


---

```

1: function PARETO-OPT( $(S, A, T_{\downarrow}, R_{\downarrow}, Pr), \gamma$ )
2:   initialize  $k = 0, V^0 = \{\mathbf{0}, \mathbf{0}, \mathbf{0}\}$ ;
3:   repeat
4:      $V_s^k = \emptyset$ ;
5:      $k = k + 1$ ;
6:     for  $s \in S, a \in A, (\mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow}) \in V^{k-1}$  do
7:        $u_{\downarrow} = r_{s\downarrow}^a + \gamma \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} (\mathbf{p}_s^a \mathbf{v}_{\downarrow})$ ;
8:        $\bar{u} = \bar{r}_s^a + \gamma \bar{\mathbf{P}}_s^a \bar{\mathbf{v}}$ ;
9:        $u_{\uparrow} = r_{s\uparrow}^a + \gamma \max_{\mathbf{p}_s^a \in \mathbf{P}_{s\uparrow}^a} (\mathbf{p}_s^a \mathbf{v}_{\uparrow})$ ;
10:       $V_s^k = \text{popt}(V_s^k \cup \{u_{\downarrow}, \bar{u}, u_{\uparrow}\})$ ;
11:       $V^k = \times_{(v_{1\downarrow}, \bar{v}_1, v_{1\uparrow}) \in V_1^k \dots (v_{n\downarrow}, \bar{v}_n, v_{n\uparrow}) \in V_n^k}$ 
12:         $((v_{1\downarrow}, \dots, v_{n\downarrow})^T, (\bar{v}_1, \dots, \bar{v}_n)^T, (v_{1\uparrow}, \dots, v_{n\uparrow})^T)$ ;
13:    until  $\text{maxerr}(V^k) < (\epsilon_{\downarrow}, \bar{\epsilon}, \epsilon_{\uparrow})$ ;
14:    return  $V^k$ 

```

---

the optimal policies, which are deterministic but consider the last  $k - 1$  decisions for the decision in the  $k$ th step, can be generated

The following theorem proves an error bound which is used to implement the function *maxerr*.

**Theorem 3.** Let  $V^k$  be a set of value vectors resulting from Algorithm 1. For each  $(\mathbf{u}_{\downarrow}, \bar{\mathbf{u}}, \mathbf{u}_{\uparrow}) \in \mathcal{V}_{opt}$  exists some  $(\mathbf{v}_{\uparrow}, \bar{\mathbf{v}}, \mathbf{v}_{\downarrow}) \in V^k$  such that

$$\|\mathbf{u}_{\downarrow} - \mathbf{v}_{\downarrow}\|_{\infty} \leq v_{\downarrow}^* \frac{\gamma^k}{1 - \gamma^k}, \quad \|\bar{\mathbf{u}} - \bar{\mathbf{v}}\|_{\infty} \leq \bar{v}^* \frac{\gamma^k}{1 - \gamma^k}, \quad \|\mathbf{u}_{\uparrow} - \mathbf{v}_{\uparrow}\|_{\infty} \leq v_{\uparrow}^* \frac{\gamma^k}{1 - \gamma^k},$$

where  $v_{\downarrow}^* = \max_{s \in S} \max_{v_{\downarrow} \in V_{s\downarrow}^k} (v_{\downarrow})$ ,  $\bar{v}^* = \max_{s \in S} \max_{\bar{v} \in \bar{V}_s^k} (\bar{v})$  and  $v_{\uparrow}^* = \max_{s \in S} \max_{v_{\uparrow} \in V_{s\uparrow}^k} (v_{\uparrow})$ .

*Proof.* Since the policies computed in Algorithm 1 are deterministic, they can be represented by vectors  $\pi_i$  for the decisions in the  $i$ th ( $1 \leq i \leq k$ ) step. Each  $(\mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow}) \in V^k$  results from a specific sequence  $\pi_1, \dots, \pi_k$ .  $(\mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow})$  has been computed starting with value vector  $(\mathbf{0}, \mathbf{0}, \mathbf{0})$   $k$  steps apart. Now assume that we start with  $(\mathbf{w}_{\downarrow}, \bar{\mathbf{w}}, \mathbf{w}_{\uparrow})$  instead of  $(\mathbf{0}, \mathbf{0}, \mathbf{0})$  and use the same policy  $\pi_1, \dots, \pi_k$ , then the resulting value vectors are

$$\begin{aligned} \mathbf{v}_{\downarrow} + \gamma^k \prod_{i=1}^k P(\pi_i) \mathbf{w}_{\downarrow} &\leq \mathbf{v}_{\downarrow} + \gamma^k \max_{s \in S} (\mathbf{w}_{s\downarrow}) \mathbf{1}, \\ \bar{\mathbf{v}} + \gamma^k \prod_{i=1}^k P(\pi_i) \bar{\mathbf{w}} &\leq \bar{\mathbf{v}} + \gamma^k \max_{s \in S} (\bar{\mathbf{w}}_s) \mathbf{1} \quad \text{and} \\ \mathbf{v}_{\uparrow} + \gamma^k \prod_{i=1}^k P(\pi_i) \mathbf{w}_{\uparrow} &\leq \mathbf{v}_{\uparrow} + \gamma^k \max_{s \in S} (\mathbf{w}_{s\uparrow}) \mathbf{1}. \end{aligned}$$

Consequently,  $(\mathbf{v}_{\uparrow}, \bar{\mathbf{v}}, \mathbf{v}_{\downarrow})$  contains the discounted reward accumulated in the last  $k$  steps and is a lower bound for the discounted reward accumulated over an infinite number of steps. Since  $V^k$  contains all Pareto optimal vectors,  $v_{\downarrow}^*$  the maximal lower bound of the accumulated reward in  $k$  steps,  $\bar{v}^*$  is the maximal

average reward that can be accumulated in  $k$  steps and  $v_{\uparrow}^*$  the maximal upper bound of the accumulated reward in  $k$  steps. This implies

$$\begin{aligned} \max_{s \in S} (\mathbf{w}_{s\downarrow}) &\leq \sum_{i=0}^{\infty} (\gamma^k)^i v_{\downarrow}^* = \frac{1}{1-\gamma^k} v_{\downarrow}^*, \\ \max_{s \in S} (\bar{\mathbf{w}}_s) &\leq \sum_{i=0}^{\infty} (\gamma^k)^i \bar{v}^* = \frac{1}{1-\gamma^k} \bar{v}^* \quad \text{and} \\ \max_{s \in S} (\mathbf{w}_{s\uparrow}) &\leq \sum_{i=0}^{\infty} (\gamma^k)^i v_{\uparrow}^* = \frac{1}{1-\gamma^k} v_{\uparrow}^* \end{aligned}$$

for every value vector  $(\mathbf{w}_{\downarrow}, \bar{\mathbf{w}}, \mathbf{w}_{\uparrow})$  reachable by an arbitrary policy and also for every value vector  $(\mathbf{w}_{\downarrow}, \bar{\mathbf{w}}, \mathbf{w}_{\uparrow}) \in \mathcal{V}_{opt}$ .

Now let  $(\mathbf{u}_{\downarrow}, \bar{\mathbf{u}}, \mathbf{u}_{\uparrow}) \in \mathcal{V}_{opt}$  be a Pareto optimal pair of vectors resulting from policy  $f = \pi_1, \dots, \pi_k, \dots \in \mathcal{P}_{opt}$ . First, assume that  $\pi_1, \dots, \pi_k$  is a policy that has been used in Algorithm 1 to compute  $(\mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow}) \in V^k$ . Then  $\bar{\mathbf{v}}$  contains the average reward accumulated during the last  $k$  steps by policy  $f$ . Additionally,  $\bar{\mathbf{u}}$  contains the discounted reward accumulated in the steps more than  $k$  steps apart. A bound for this reward is given above and this bound is weighted by  $\gamma^k$  which results in the bound given in the theorem. With the same arguments the bound for the worst and best case behavior can be derived.

Now assume that  $f = \pi_1, \dots, \pi_k; \dots \in \mathcal{P}_{opt}$  but  $\pi_1, \dots, \pi_k$  has not been used to compute a vector from  $V^k$ . Then  $(\mathbf{u}_{\downarrow}, \bar{\mathbf{u}}, \mathbf{u}_{\uparrow})$  consists of a part  $(\mathbf{w}_{\downarrow}, \bar{\mathbf{w}}, \mathbf{w}_{\uparrow})$  that includes the reward accumulated during the last  $k$  steps and the rest which can be bounded as in the previous case. However, since  $V^k$  contains all Pareto optimal vector with respect to the reward accumulated in  $k$  steps, it contains a vector pair  $(\mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow})$  with  $\mathbf{v}_{\downarrow} \geq \mathbf{w}_{\downarrow}$ ,  $\bar{\mathbf{v}} \geq \bar{\mathbf{w}}$  and  $\mathbf{v}_{\uparrow} \geq \mathbf{w}_{\uparrow}$ . These vectors plus the bound for the reward accumulated in the steps  $k+1, \dots$  define a bound for  $(\mathbf{u}_{\downarrow}, \bar{\mathbf{u}}, \mathbf{u}_{\uparrow})$ .  $\square$

The definition of function *maxerr* follows immediately from the previous theorem. The algorithm approximates the, possibly infinite, Pareto front by a finite set of points with a predefined error bound. In the worst case, the number of value vectors and therefore the computational effort grows exponentially with  $k$ . From a practical point of view it is, of course, impossible to implement an exponentially growing number of policies. Therefore we consider in the following paragraphs the computation of pure policies.

**Pareto Optimal Policies from  $\mathcal{P}_{Pareto}$ :** In contrast to the previous case, now only pure policies are considered. However, even the number of Pareto optimal pure policies can be exponential in the size of the model since the number of pure policies equals  $m^n$ . Value iteration and policy iteration can be applied to compute policies from  $\mathcal{P}_{Pareto}$  and the corresponding value vectors from  $\mathcal{V}_{Pareto}$ . We present an algorithm based on policy iteration which outperforms value iteration if the set  $\mathcal{P}_{Pareto}$  is not too large and if it is possible to identify *good* policies with a low effort. Both conditions hold for most practical problems. However, in the worst case, where (almost) all pure policies are Pareto optimal, all policies have to be evaluated to compute  $\mathcal{V}_{Pareto}$ .

In the following algorithm we use sets  $PV$  and  $PV'$  that contain quadruples  $(\pi, \mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow})$  where  $\pi$  is the pure policy and  $(\mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow})$  are the value vectors. With a slight extension, we use the function *popt* also for these sets. Function *eval* evaluates a pure policy for a SBMDP. Additionally, we use a set  $P$  where all

evaluated policies are stored to avoid a reevaluation of a policy. Let  $(\pi^{lb}, \bar{\mathbf{v}}^{lb}, \mathbf{v}_\downarrow^{lb})$ ,  $(\pi^{av}, \bar{\mathbf{v}}^{av}, \mathbf{v}_\downarrow^{av})$  and  $(\pi^{ub}, \bar{\mathbf{v}}^{ub}, \mathbf{v}_\downarrow^{ub})$  the policies and value vectors resulting from the optimization of the lower bound, average case and upper bound of the discounted reward, respectively. It is known that these policies belong to  $\mathcal{P}_{Pareto}$ .

Algorithm 2 is an optimized version of a policy iteration approach to compute  $\mathcal{P}_{Pareto}$  and the corresponding value vectors. The algorithm is based on the observation that if for  $(\pi, \mathbf{v}_\downarrow, \bar{\mathbf{v}}, \mathbf{v}_\uparrow)$  a state, action pair  $(s, a)$  can be found such that for the policy  $\pi'$  which results from  $\pi$  by setting  $\pi'_s = a$  and  $\pi'_{s'} = \pi_{s'}$  for  $s' \neq s$  one of the three relations

$$r_{s\downarrow}^a + \gamma \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} \mathbf{p}_s^a \mathbf{v}_\downarrow > v_{s\downarrow} \text{ or } \bar{r}_s^a + \gamma \bar{\mathbf{P}}_s^a \bar{\mathbf{v}} > \bar{v}_s \text{ or } r_{s\uparrow}^a + \gamma \max_{\mathbf{p}_s^a \in \mathbf{P}_{s\uparrow}^a} \mathbf{p}_s^a \mathbf{v}_\uparrow > v_{s\uparrow}$$

holds, then for the value vectors  $(\mathbf{u}_\downarrow, \bar{\mathbf{u}}, \mathbf{u}_\uparrow)$  belonging to  $\pi'$ ,  $\mathbf{u}_\downarrow > \mathbf{v}_\downarrow$  or  $\bar{\mathbf{u}} > \bar{\mathbf{v}}$  or  $\mathbf{u}_\uparrow > \mathbf{v}_\uparrow$  holds. On the other hand, if no pair  $(s, a)$  can be found that observes the above conditions, then  $\pi$  is for sure Pareto optimal in the set of pure policies. If the new policy  $\pi'$  assures all of the above relations, then  $\mathbf{u}_\downarrow > \mathbf{v}_\downarrow$  and  $\bar{\mathbf{u}} > \bar{\mathbf{v}}$  and  $\mathbf{u}_\uparrow > \mathbf{v}_\uparrow$  such that  $\pi'$  dominates  $\pi$ . As usual for policy iteration, the algorithm runs until the no more policies can be found with value vectors that are Pareto optimal with respect to the current set of value vectors. Since the number of pure policies is finite and all evaluated policies are stored, the algorithm will terminate.

Algorithm 2 starts with up to three policies which are known to be Pareto optimal. In an outer loop, new policies are generated from a current set of Pareto optimal policies. In the first inner loop, the algorithm tries to find policies that dominate a policy collected from the current set of Pareto optimal policies. In this way one tries to find good policies in the first iterations of the algorithms to avoid the evaluation of too many suboptimal policies. In the second inner loop, policies from the set of Pareto optimal policies are modified in one component, it is checked whether the resulting policy has not been evaluated and whether it improves the worst, average or best case reward. If this is the case, the policy is evaluated and it is put into the set of Pareto optimal policies. Non-optimal policies are always removed immediately from this set.

In function *eval* the following three sets of equations have to be solved to compute the value vectors for some pure policy  $\pi$ .

$$\mathbf{u}_\downarrow = \mathbf{r}_\downarrow^\pi + \gamma \min_{\mathbf{P} \in \mathbf{P}_\downarrow^\pi} (\mathbf{P}^\pi \mathbf{u}_\downarrow), \bar{\mathbf{u}} = \bar{\mathbf{r}}^\pi + \gamma \bar{\mathbf{P}}^\pi \bar{\mathbf{u}} \text{ and } \mathbf{u}_\uparrow = \mathbf{r}_\uparrow^\pi + \gamma \max_{\mathbf{P} \in \mathbf{P}_\uparrow^\pi} (\mathbf{P}^\pi \mathbf{u}_\uparrow)$$

The equations for the average values define a set of linear equations which can be solved with standard means. For the vector of the worst and best case an iterative approach is applied. Vector  $\mathbf{u}_\downarrow$  (or  $\mathbf{u}_\uparrow$ ) is initialized, the minimum (or maximum) is computed and with this minimum (or maximum) a new vector is computed which is then used to find a new minimum (or maximum). This sequence defines a converging sequence of value vectors.

**Optimal Policies from  $\mathcal{P}_{weight}$ :** For the computation of the set of policies that are optimal under some weight vector  $(w_\downarrow, \bar{w}, w_\uparrow)$  ( $w_\downarrow + \bar{w} + w_\uparrow = 1$ ), the approaches from [RWO13, RWO14, RSS<sup>+</sup>14] can in principle be applied with some slight modifications. The resulting optimization problem is bilinear with linear constraints. The optimization problem is not convex but can usually be



---

**Algorithm 2** Policy iteration to compute  $\mathcal{P}_{Pareto}$  and  $\mathcal{V}_{Pareto}$ 


---

```

1: function PURE-OPT( $SBMDP = (S, A, T_{\downarrow}, R_{\downarrow}, Pr), \gamma$ )
2:   initialize  $PV = \{(\pi^{lb}, \bar{\mathbf{v}}^{lb}, \mathbf{v}_{\downarrow}^{lb}), (\pi^{av}, \bar{\mathbf{v}}^{av}, \mathbf{v}_{\downarrow}^{av}), (\pi^{ub}, \bar{\mathbf{v}}^{ub}, \mathbf{v}_{\downarrow}^{ub})\}$ 
3:   initialize  $P = \{\pi^{lb}, \pi^{av}, \pi^{ub}\}$ ;
4:   while true do
5:      $PV' = \emptyset$ ;
6:      $PV'' = PV$ ;
7:     repeat
8:       remove  $(\pi, \mathbf{u}_{\downarrow}, \bar{\mathbf{u}}, \mathbf{u}_{\uparrow})$  from  $PV''$ ;
9:        $\pi' = \pi$ ;
10:      for  $s \in S$  and  $a \in A$  do
11:         $v_{s\downarrow} = r_{s\downarrow}^a + \gamma \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} \mathbf{p}_s^a \mathbf{u}_{\downarrow}$ ;
12:         $\bar{v}_s = \bar{r}_s^a + \gamma \bar{\mathbf{p}}_s^a \bar{\mathbf{u}} > \bar{u}_s$ ;
13:         $v_{s\uparrow} = r_{s\uparrow}^a + \gamma \max_{\mathbf{p}_s^a \in \mathbf{P}_{s\uparrow}^a} \mathbf{p}_s^a \mathbf{u}_{\uparrow}$ ;
14:        if  $v_{s\downarrow} > u_{s\downarrow}$  and  $\bar{v}_s > \bar{u}_s$  and  $v_{s\uparrow} > u_{s\uparrow}$  then
15:           $\pi'_s = a$ ;
16:        if  $\pi' \neq \pi$  then
17:           $(\mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow}) = eval(SBMDP, \pi')$ ;
18:           $PV'' = popl(PV'' \cup \{(\pi', \mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow})\})$ ;
19:           $PV' = popl(PV' \cup \{(\pi', \mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow})\})$ ;
20:           $P = P \cup \{\pi'\}$ ;
21:        else
22:           $PV' = popl(PV' \cup \{(\pi, \mathbf{u}_{\downarrow}, \bar{\mathbf{u}}, \mathbf{u}_{\uparrow})\})$ ;
23:      until  $PV'' = \emptyset$ ;
24:       $PV'' = PV'$ ;
25:      for all  $(\pi, \mathbf{u}_{\downarrow}, \bar{\mathbf{u}}, \mathbf{u}_{\uparrow}) \in PV'$  do
26:        for all  $s \in S$  and  $a \in A$  where  $\pi_s \neq a$  do
27:           $\pi' = \pi$  and  $\pi'_s = a$ ;
28:           $v_{s\downarrow} = r_{s\downarrow}^a + \gamma \min_{\mathbf{p}_s^a \in \mathbf{P}_{s\downarrow}^a} \mathbf{p}_s^a \mathbf{u}_{\downarrow}$ ;
29:           $\bar{v}_s = \bar{r}_s^a + \gamma \bar{\mathbf{p}}_s^a \bar{\mathbf{u}} > \bar{u}_s$ ;
30:           $v_{s\uparrow} = r_{s\uparrow}^a + \gamma \max_{\mathbf{p}_s^a \in \mathbf{P}_{s\uparrow}^a} \mathbf{p}_s^a \mathbf{u}_{\uparrow}$ ;
31:          if  $\pi' \notin P$  and  $(v_{s\downarrow} > u_{s\downarrow}$  or  $\bar{v}_s > \bar{u}_s$  or  $v_{s\uparrow} > u_{s\uparrow})$  then
32:             $(\mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow}) = eval(SBMDP, \pi')$ ;
33:             $P = P \cup \{\pi'\}$ ;
34:             $PV'' = popl(PV'' \cup \{(\pi', \mathbf{v}_{\downarrow}, \bar{\mathbf{v}}, \mathbf{v}_{\uparrow})\})$ ;
35:      if  $PV = PV''$  then
36:        return  $PV$ 
37:      else
38:         $PV = PV''$ ;

```

---

solved if the dimension of the goal function is not too high which is the case for SBMDPs with two or three dimensional goal functions.

For some weight vector  $\mathbf{w}$  let  $\mathbf{v}^{\mathbf{w}} = (\mathbf{v}_{\downarrow}^{\mathbf{w}}, \bar{\mathbf{v}}^{\mathbf{w}}, \mathbf{v}_{\uparrow}^{\mathbf{w}})$  be the optimal value vector. Vector  $\mathbf{v}^{\mathbf{w}}$  corresponds to a pure policy and can be computed with a standard approach for solving single objective MDPs.

For the algorithm assume that for a set of weight vectors  $\mathcal{W}_{cur}$  the optimal value vectors  $\mathcal{V}_{cur}$  have been computed. Let  $c$  be the cardinality of  $\mathcal{V}_{cur}$  and assume that vectors are number consecutively from 1 through  $c$ . Then  $\mathbf{w}^i = (w_{\downarrow}^i, \bar{w}^i, w_{\uparrow}^i)$  and  $\mathbf{v}^i = (\mathbf{v}_{\downarrow}^i, \bar{\mathbf{v}}^i, \mathbf{v}_{\uparrow}^i)$  are the  $i$ -th weight and value vector from the sets  $\mathcal{W}_{cur}$  and  $\mathcal{V}_{cur}$ , respectively. Again we use the notation  $\mathcal{V}_{s,cur}$  to consider only the elements belonging to  $s \in S$  in the value vectors. We obtain for some weight vector  $(w_{\downarrow}, \bar{w}, w_{\uparrow})$  and  $s \in S$  the following bounds

$$\begin{aligned} lb_s(\mathbf{w}, \mathcal{V}_{cur}) &= \max_{v_{s,\downarrow}, (\bar{v}_s, v_{s,\uparrow}) \in \mathcal{V}_{cur}} (w_{\downarrow} v_{s,\downarrow}^{\mathbf{w}} + \bar{w} \bar{v}_s + w_{\uparrow} v_{s,\uparrow}) \\ &\leq w_{\downarrow} v_{s,\downarrow}^{\mathbf{w}} + \bar{w} \bar{v}_s + w_{\uparrow} v_{s,\uparrow}^{\mathbf{w}} \\ &\leq \max_{(z_{\downarrow}, \bar{z}, z_{\uparrow}) \in Z_s} (w_{\downarrow} z_{s,\downarrow} + \bar{w} \bar{z}_s + w_{\uparrow} z_{s,\uparrow}) = ub_s(\mathbf{w}, \mathcal{V}_{cur}) \end{aligned} \quad (5)$$

where

$$Z_s = \left\{ (z_{\downarrow}, \bar{z}, z_{\uparrow}) \mid \forall \mathbf{w} \in \mathcal{W}_{cur} : \right. \\ \left. w_{\downarrow} z_{\downarrow} + \bar{w} \bar{z} + w_{\uparrow} z_{\uparrow} \leq w_{\downarrow} v_{s,\downarrow}^{\mathbf{w}} + \bar{w} \bar{v}_s + w_{\uparrow} v_{s,\uparrow}^{\mathbf{w}} \wedge z_{\uparrow} \geq \bar{z} \geq z_{\downarrow} \right\}.$$

The lower bound holds since  $\mathcal{V}_{cur} \subseteq \mathcal{V}_{pure}$  such that only valid value vectors are considered. The upper bound holds since any valid value vector has to be consistent with the optimal value vectors that have been computed, i.e., it cannot result in a larger reward for one of the optimal value vectors that are known. For some weight vector from  $\mathcal{W}_{cur}$  the upper and lower bound are equal to the exact maximum.

The functions  $lb_s(\mathbf{w}, \mathcal{V})$  and  $ub_s(\mathbf{w}, \mathcal{V})$  are piecewise linear and convex in  $\mathbf{w}$  for a fixed set  $\mathcal{V}$ . Furthermore, for  $\mathcal{V}' \subset \mathcal{V}$ , the relations  $lb_s(\mathbf{w}, \mathcal{V}) \leq lb_s(\mathbf{w}, \mathcal{V}')$  and  $ub_s(\mathbf{w}, \mathcal{V}) \geq ub_s(\mathbf{w}, \mathcal{V}')$  hold.  $Z_s$  is a convex set.

The value vector  $\mathbf{v}_s^i = (v_{s,\downarrow}^i, \bar{v}_s^i, v_{s,\uparrow}^i) \in \mathcal{V}_{s,cur}$  is optimal for weight  $\mathbf{w} = (w_{\downarrow}, \bar{w}, w_{\uparrow})$  and set  $\mathcal{V}_{s,cur}$ , iff  $w_{\downarrow}(v_{s,\downarrow}^i - v_{s,\downarrow}^j) + \bar{w}(\bar{v}_s^i - \bar{v}_s^j) + w_{\uparrow}(v_{s,\uparrow}^i - v_{s,\uparrow}^j) \geq 0$  for all  $j = 1, \dots, c, j \neq i$ . The set of weight vectors for which  $\mathbf{v}_s^i$  is optimal is also convex and the lower bound results from the product of the weight vector  $\mathbf{w}$  and value vector  $\mathbf{v}_s^i$ . The maximal difference between lower and upper bound in this region can be computed from the following bilinear problem.

$$\begin{aligned} &\max_{\mathbf{w}, \mathbf{u}} \left( w_{\downarrow} u_{\downarrow} + \bar{w} \bar{u} + w_{\uparrow} u_{\uparrow} - \left( w_{\downarrow} v_{s,\downarrow}^i + \bar{w} \bar{v}_s^i + w_{\uparrow} v_{s,\uparrow}^i \right) \right) \\ &\text{with the constraints} \\ i) \quad &w_{\downarrow} (\mathbf{v}_{\downarrow}^i - \mathbf{v}_{\downarrow}^j) + \bar{w} (\bar{\mathbf{v}}^i - \bar{\mathbf{v}}^j) + w_{\uparrow} (\mathbf{v}_{\uparrow}^i - \mathbf{v}_{\uparrow}^j) \geq 0 \\ ii) \quad &w_{\downarrow} + \bar{w} + w_{\uparrow} = 1 \\ iii) \quad &w_{\downarrow} (v_{s,\downarrow}^j - u_{\downarrow}) + \bar{w}^j (\bar{v}_s^j - \bar{u}) + w_{\uparrow} (v_{s,\uparrow}^j - u_{\uparrow}) \geq 0 \\ iv) \quad &u_{\uparrow} \geq \bar{u} \geq u_{\downarrow} \\ &\text{for all } j \in \{1, \dots, c\} \setminus \{i\} \end{aligned} \quad (6)$$

The set of constraints  $i)$  defines a polyhedron for the weight vectors  $\mathbf{w}$ . If we fix  $\mathbf{w}$  or  $\mathbf{u}$ , then the problem becomes a linear program. This means that for a given

$\mathbf{u}$  the maximum is achieved by solving the linear program with the constraints  $i$  and  $ii$ . This implies that the maximum can be computed with the following steps:

1. Determine the vertices of the polyhedron defined by  $i$ ) and  $ii$ ).
2. Solve the linear program with constraints  $iii$ ) and  $iv$ ) for all  $\mathbf{w}$  given as a vertex of the polyhedron defined by  $i$ ) and  $ii$ ). The maximum is the solutions of the problem.

The set of extremal points of the polyhedron in principle can grow exponentially in the number of constraints. However, since the dimension of the problem is only 3, the problem is usually manageable.

---

**Algorithm 3** Algorithm to compute  $\mathcal{P}_{weight}$  and  $\mathcal{V}_{weight}$

---

```

1: function WEIGHT-OPT( $SBMDP = (S, A, T_{\downarrow}, R_{\downarrow}, Pr), \gamma, \epsilon$ )
2:   initialize  $PV = \{(\pi^{lb}, \mathbf{v}_{\downarrow}^{lb}, \bar{\mathbf{v}}^{lb}, \mathbf{v}_{\uparrow}^{lb}, (1, 0, 0)), (\pi^{av}, \mathbf{v}_{\downarrow}^{av}, \bar{\mathbf{v}}^{av}, \mathbf{v}_{\uparrow}^{av}, (0, 1, 0)),$ 
3:      $(\pi^{ub}, \mathbf{v}_{\downarrow}^{ub}, \bar{\mathbf{v}}^{ub}, \mathbf{v}_{\uparrow}^{ub}, (0, 0, 1))\}$  ;
4:   for  $s \in S$  do
5:      $PV' = PV$  ;
6:     while  $PV' \neq \emptyset$  do
7:       remove  $(\pi, \mathbf{v}, \mathbf{w})$  from  $PV'$  ;
8:       repeat
9:         solve the optimization problem (6) and obtain  $(\mathbf{w}, \mathbf{u}_s)$  ;
10:        if  $\mathbf{w}(\mathbf{u}_s - \mathbf{v}_s) > \epsilon$  then
11:           $(\pi, (\mathbf{x}_{\downarrow}, \bar{\mathbf{x}}, \mathbf{x}_{\uparrow})) =$  solution of MDP with weight vector  $\mathbf{w}$  ;
12:           $PV = PV \cup \{(\pi, (\mathbf{x}_{\downarrow}, \bar{\mathbf{x}}, \mathbf{x}_{\uparrow}), \mathbf{w})\}$  ;
13:           $PV' = PV' \cup \{(\pi, (\mathbf{x}_{\downarrow}, \bar{\mathbf{x}}, \mathbf{x}_{\uparrow}), \mathbf{w})\}$  ;
14:        else
15:           $\mathbf{u}_s = \mathbf{v}_s$  ;
16:        until  $\mathbf{u}_s = \mathbf{v}_s$  ;
17:   return  $PV$ 

```

---

The complete solution approach is given in Algorithm 3. The algorithm first solves the MDP for the extreme weight vectors, where only the average or only the lower bound are considered. The corresponding value vectors assure that the optimization problem (6) is bounded since  $\mathbf{u}_{\downarrow} \leq \mathbf{v}_{\downarrow}^{lb}$ ,  $\bar{\mathbf{u}} \leq \bar{\mathbf{v}}^{av}$  and  $\mathbf{u}_{\uparrow} \leq \mathbf{v}_{\uparrow}^{lb}$  holds then due to the set of constraints  $iii$ ). Then for each state the set of pure policies that is optimal for some weight vector is computed. In step 6 one policy is selected from the set of policies that have already been found as optimal policies for some weight vector. The solution of the optimization problem (6) determines an upper bound for the difference between the policy that is known and any other policy in the range where the selected policy is currently assumed to be optimal. If the difference is larger than some predefined error bound  $\epsilon$ , a new policy is computed for the weight vector that has been determined in the optimization step. The new policy, value vectors and weight vectors are added to the set of available policies and adds constraints to the optimization problem. After a region has been explored for a state (lines 7-16), the computed upper bound remains valid since constraints are added but never

removed. After termination of the algorithm, the result is an  $\epsilon$ -approximation of the set of policies from  $\mathcal{P}_{weight}$ .

The effort of the algorithm depends on the number of policies that have to be analyzed. Of course, this number can be exponential in the size of the MDP. The solution of the optimization problem is usually not critical since the number of variables is small. If the optimal values are only needed for some states, then the outer *for*-loop can be modified to consider only the interesting states.

## 6 Example

The current implementation of the algorithms is a prototype that can only be applied for very small examples. Therefore we present a first small example. Implementation of a more efficient version of the algorithms that allows one to analyze much larger examples is underway.

As an example we consider a simple system of a component with failures and possible maintenance. The component can be in one of 5 state  $\{new, good, adequate, obsolete, unusable\}$ . In each state three possible actions exist, namely  $\{ignore, maintenance, buy\}$ . The state transition diagram of the example is shown in Figure 3. The following table includes the bounds for the transition probabilities in the form *action*, [*minimum*, *average*, *maximum*].

from/to	new	good	adequate	obsolete	unusable
new	i[0.45, 0.5, 0.6] m[0.4, 0.9, 1] b[1, 1, 1]	i[0.35, 0.45, 0.5] m[0, 0.05, 0.4]			i[0, 0.05, 0.3] m[0, 0.05, 0.4]
good	b[1, 1, 1] m[0.4, 0.85, 1]	i[0.45, 0.5, 0.6] m[0, 0.05, 0.4]	i[0.35, 0.45, 0.5] m[0, 0.05, 0.4]		i[0, 0.05, 0.3] m[0, 0.05, 0.4]
adequate	b[1, 1, 1]	m[0.4, 0.85, 1]	i[0.45, 0.5, 0.6] m[0, 0.05, 0.4]	i[0.35, 0.45, 0.5] m[0, 0.05, 0.4]	i[0, 0.05, 0.3] m[0, 0.05, 0.4]
obsolete	b[1, 1, 1]		m[0.4, 0.85, 1]	i[0.35, 0.7, 1.0] m[0, 0.05, 0.4]	i[0.15, 0.3, 0.45] m[0, 0.1, 0.8]
unusable	b[1.0, 1.0, 1.0]				i,m[1, 1, 1]

Rewards depend on the state and action but are all exactly known. The following table includes the expected reward vectors. We assume that the lower bound results from the expected values multiplied by 0.8 and the upper bound results from the expected values multiplied by 1.2.

action/state	new	good	adequate	obsolete	unusable
ignore	30	28	26	20	20
maintenance	20	20	20	20	20
buy	0	0	0	0	0

For  $\gamma = 0.9$  the policy (i, m, m, m, b) is optimal in the average case. However, since the maintenance operation is unreliable and may even result in a system failure with a relatively high probability in the worst case, also the policies (i, m, m, i, b), (i, i, m, m, b), (i, m, i, m, b), (i, i, i, m, b) and (i, i, m, i, b) are Pareto optimal if one considers the average and worst case behavior of the system. The Figures 4 and 5 show the location of optimal points according to the average and minimal reward starting in the state *new* and according to the sum of rewards over all states. In the latter case only 4 policies are Pareto optimal. The dashed lines indicate solutions that are reachable by mixed policies.

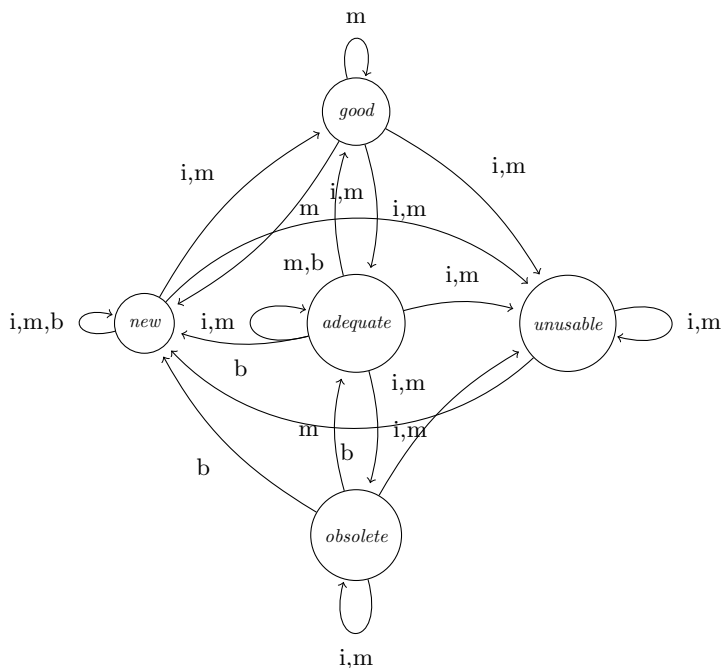


Figure 3: State transition diagram of the example.

The set  $\mathcal{P}_{weight}$  contains only the policies  $(i, m, m, m, b)$  and  $(i, i, i, m, b)$ . The reachable rewards  $w\bar{v}_s + (1 - w)v_{s,\downarrow}$  are shown in Figure 6 for state *new*.

The set  $\mathcal{P}_{opt}$  contains an exponentially growing number of policies which cannot be computed for sufficiently small error bounds. Therefore it is only possible to approximate the set which is done by restricting the number of value vectors in the sets  $V_s^k$  in Algorithm 1. For the following results up to 50 vectors are kept for state *new* and up to 5 vectors are kept for the remaining states which still means that more than 50000 value vectors are computed. 100 iterations of the value iterations are performed resulting in error bounds  $\bar{\epsilon} = 0.0128$  and  $\epsilon_{\downarrow} = 0.0094$ . Results are shown in the Figures 7 and 8. It can be seen that with the use of non-stationary policies much better compromise solutions can be found than with pure policies.

## 7 Conclusions

Summing up, we have presented different algorithms to analyze bounded-parameter Markov decision processes. In contrast to known approaches [GLD00] that usually analyze the worst case behavior and result in a variant of robust optimization, the problem is handled here as a multi-objective problem. Of course, the price for this extension can be an exponential complexity due to an exponential number of *optimal* policies in this case. However, in practice often the number of optimal policies is fairly small and the combination of different goal functions often allows one to find compromise solutions with an acceptable worst case behavior combined with good results in the average case. The problem differs from many other multi-objective optimization approaches for MDPs where several expected values are analyzed, whereas here expected and worst case behavior are considered together. This has for example the consequence that randomized policies resulting from the combination of two or

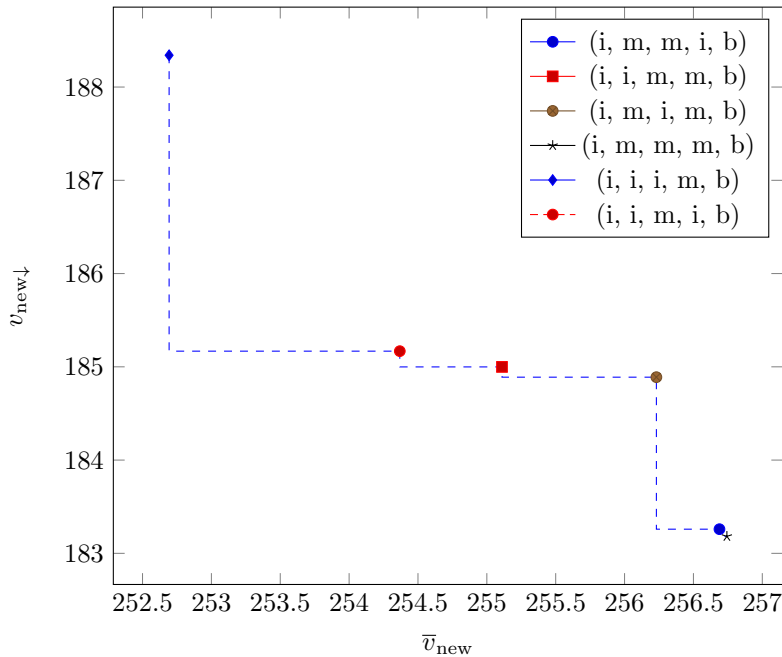


Figure 4: Values in state *new*

more deterministic policies are not applicable since the worst case corresponds to the worst case of the set of policies that are used in the approach.

For three specific problems of multi-objective optimization of MDPs computational algorithms are presented. The algorithms are tuned for the problems but can, of course, not avoid the exponential complexity, if the problem is exponential. However, in two of the three cases it is possible to compute error bounds, if the algorithm is stopped before the optimum has been reached. This is important from a practical point of view, since usually only a relatively small number of policies can be implemented for the practical operation of a system.

The approach presented here can be extended in various directions. As mentioned several times it is fairly easy to consider additional goals beyond worst, average and best cases. The algorithms can be easily extended but the complexity of multi-objective optimization, of course, usually grows rapidly with the dimension of the problem. Our results not only apply to bounded-parameter Markov decision processes, but can also be utilized for Markov decision processes with convex uncertainties, as long as the convex program  $\min \mathbf{c}^T \mathbf{x}$  subject to  $\mathbf{x} \in C$  with a convex set  $C$  can be solved efficiently [PLSVS13, NN94]. Then, the same basic algorithms can be used in order to compute Pareto optimal policies and extreme points in the value vector space, if one adjusts them to solve the convex program in the iteration step.

For future research, it would be interesting to extend the methods to unrestricted stochastic games as defined in [Sha53] and other formalisms like partially observable MDPs [Lov91]. It would be furthermore interesting to consider other optimality criteria such as expected gain.

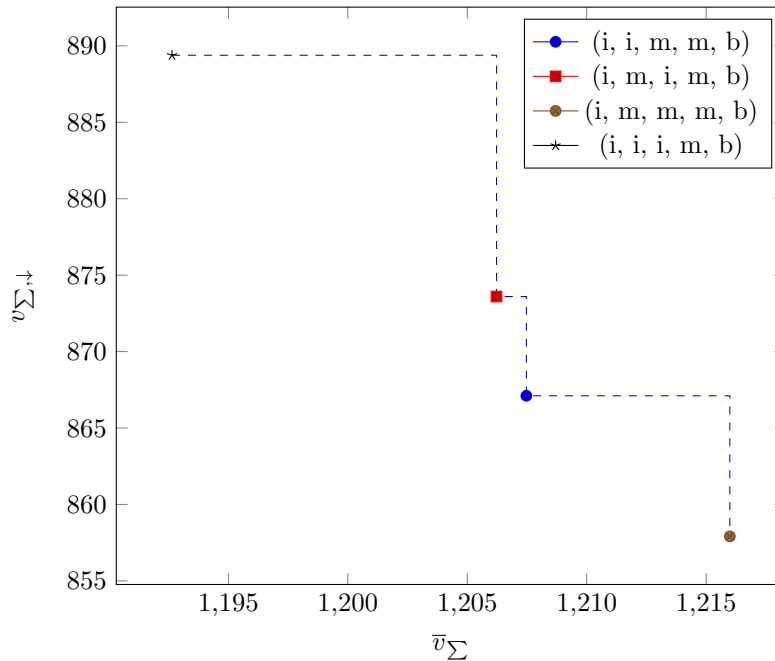


Figure 5: Values in state *new*

## References

- [BN08] Leon Barrett and Sridhar Narayanan. Learning all optimal policies with multiple criteria. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 41–47. ACM, 2008.
- [CFK<sup>+</sup>13] Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. On stochastic games with multiple objectives. In Krishnendu Chatterjee and Jiri Sgall, editors, *MFCS*, volume 8087 of *Lecture Notes in Computer Science*, pages 266–277. Springer, 2013.
- [CMH06] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov decision processes with multiple objectives. In Bruno Durand and Wolfgang Thomas, editors, *STACS*, volume 3884 of *Lecture Notes in Computer Science*, pages 325–336. Springer, 2006.
- [DSdB11] Karina Valdivia Delgado, Scott Sanner, and Leliane Nunes de Barros. Efficient solutions to factored mdps with imprecise transition probabilities. *Artif. Intell.*, 175(9-10):1498–1527, 2011.
- [FV96] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.
- [GLD00] Robert Givan, Sonia M. Leach, and Thomas L. Dean. Bounded-parameter Markov decision processes. *Artif. Intell.*, 122(1-2):71–109, 2000.
- [Lov91] William S. Lovejoy. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research*, 39(1):162–175, 1991.
- [NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

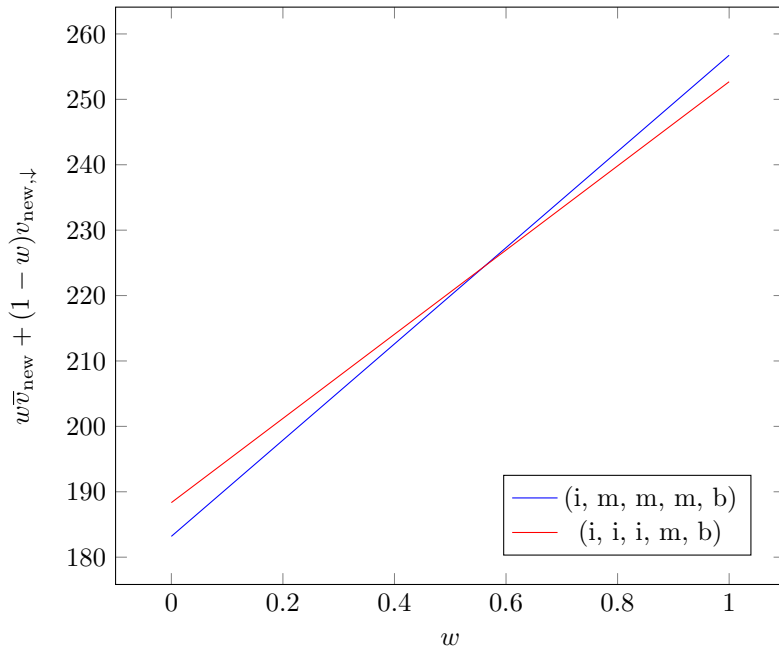


Figure 6: Values in state *new*

- [PLSVS13] Alberto Puggelli, Wenchao Li, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In Natasha Sharygina and Helmut Veith, editors, *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 527–542. Springer, 2013.
- [Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.
- [PW10] Patrice Perny and Paul Weng. On finding compromise solutions in multi-objective Markov decision processes. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 969–970. IOS Press, 2010.
- [RSS<sup>+</sup>14] Diederik Marijn Roijers, Joris Scharpff, Matthijs T. J. Spaan, Frans A. Oliehoek, Mathijs de Weerd, and Shimon Whiteson. Bounded approximations for linear multi-objective planning under uncertainty. In Steve Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *ICAPS*. AAAI, 2014.
- [RWO13] Diederik M. Roijers, Shimon Whiteson, and Frans A. Oliehoek. Computing convex coverage sets for multi-objective coordination graphs. In Patrice Perny, Marc Pirlot, and Alexis Tsoukiàs, editors, *ADT*, volume 8176 of *Lecture Notes in Computer Science*, pages 309–323. Springer, 2013.
- [RWO14] Diederik M. Roijers, Shimon Whiteson, and Frans A. Oliehoek. Linear support for multi-objective coordination graphs. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS ’14, pages 1297–1304, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.



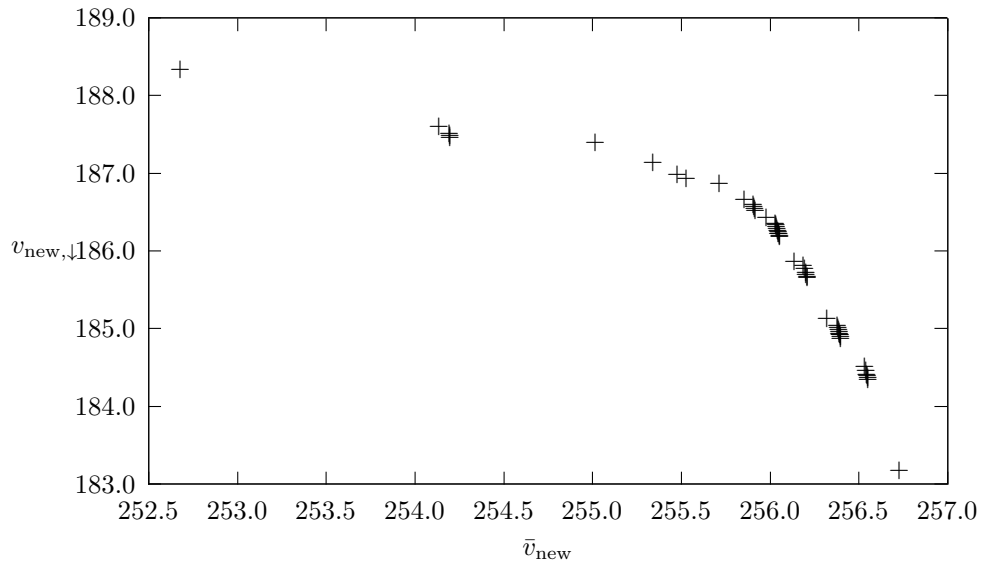


Figure 7: Approximation of the Pareto front for the reward in state *new*.

- [Ser79] R. F. Serfozo. An equivalence between continuous and discrete time Markov decision processes. *Operations Research*, 27(3):616–620, 1979.
- [Sha53] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39:1095–1100, 1953.
- [SL73] Jay K. Satia and Roy E. Lave. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973.
- [WdJ07] M.A. Wiering and E.D. de Jong. Computing optimal stationary policies for multi-objective Markov decision processes. In *ADPRL 2007, IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 158–165, April 2007.
- [WE94] Chelsea C. White and Hany K. Eldeib. Markov decision processes with imprecise transition probabilities. *Operations Research*, 42(4):739–749, 1994.
- [Whi82] D.J White. Multi-objective infinite-horizon discounted Markov decision processes. *Journal of Mathematical Analysis and Applications*, 89(2):639 – 647, 1982.
- [WKR13] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Math. Oper. Res.*, 38(1):153–183, 2013.

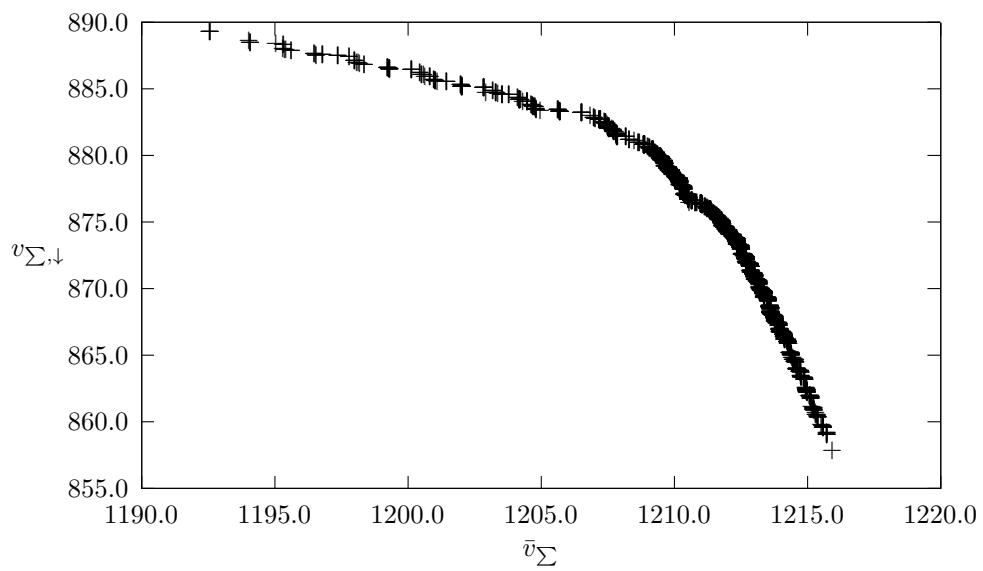


Figure 8: Approximation of the Pareto front for the sum of all state rewards.