Acta Cybernetica **18** (2008) 615–635.

Factored Value Iteration Converges

István Szita[*] and András Lőrincz[†]

Abstract

In this paper we propose a novel algorithm, factored value iteration (FVI), for the approximate solution of factored Markov decision processes (fMDPs). The traditional approximate value iteration algorithm is modified in two ways. For one, the least-squares projection operator is modified so that it does not increase max-norm, and thus preserves convergence. The other modification is that we uniformly sample polynomially many samples from the (exponentially large) state space. This way, the complexity of our algorithm becomes polynomial in the size of the fMDP description length. We prove that the algorithm is convergent. We also derive an upper bound on the difference between our approximate solution and the optimal one, and also on the error introduced by sampling. We analyse various projection operators with respect to their computation complexity and their convergence when combined with approximate value iteration.

Keywords: factored Markov decision process, value iteration, reinforcement learning

1 Introduction

Markov decision processes (MDPs) are extremely useful for formalising and solving sequential decision problems, with a wide repertoire of algorithms to choose from [4, 26]. Unfortunately, MDPs are subject to the 'curse of dimensionality' [3]: for a problem with m state variables, the size of the MDP grows exponentially with m, even though many practical problems have polynomial-size descriptions. Factored MDPs (fMDPs) may rescue us from this explosion, because they offer a more compact representation [17, 5, 6]. In the fMDP framework, one assumes that dependencies can be factored to several easy-to-handle components.

For MDPs with known parameters, there are three basic solution methods (and, naturally, countless variants of them): value iteration, policy iteration and linear programming (see the books of Sutton & Barto [26] or Bertsekas & Tsitsiklis [4] for an excellent overview). Out of these methods, linear programming is generally

[*]Eötvös Loránd University, Hungary, Department of Information Systems, E-mail: `szityu@gmail.com`

[†]Eötvös Loránd University, Hungary, Department of Information Systems, E-mail: `andras.lorincz@inf.elte.hu`. Please send correspondence to András Lőrincz.

considered less effective than the others. So, it comes as a surprise that all effective fMDPs algorithms, to our best knowledge, use linear programming in one way or another. Furthermore, the classic value iteration algorithm is known to be divergent when function approximation is used [2, 27], which includes the case of fMDPs, too.

In this paper we propose a variant of the approximate value iteration algorithm for solving fMDPs. The algorithm is a direct extension of the traditional value iteration algorithm. Furthermore, it avoids computationally expensive manipulations like linear programming or the construction of decision trees. We prove that the algorithm always converges to a fixed point, and that it requires polynomial time to reach a fixed accuracy. A bound to the distance from the optimal solution is also given.

In Section 2 we review the basic concepts of Markov decision processes, including the classical value iteration algorithm and its combination with linear function approximation. We also give a sufficient condition for the convergence of approximate value iteration, and list several examples of interest. In Section 3 we extend the results of the previous section to fMDPs and review related works in Section 4. Conclusions are drawn in Section 5.

2 Approximate Value Iteration in Markov Decision Processes

2.1 Markov Decision Processes

An MDP is characterised by a sixtuple $(\mathbf{X}, A, R, P, \mathbf{x}_s, \gamma)$, where $\mathbf{X}$ is a finite set of states;[1] A is a finite set of possible actions; $R : \mathbf{X} \times A \to \mathbb{R}$ is the reward function of the agent, so that $R(\mathbf{x}, a)$ is the reward of the agent after choosing action a in state $\mathbf{x}$; $P : \mathbf{X} \times A \times \mathbf{X} \to [0, 1]$ is the transition function so that $P(\mathbf{y} \mid \mathbf{x}, a)$ is the probability that the agent arrives at state $\mathbf{y}$, given that she started from $\mathbf{x}$ upon executing action a; $\mathbf{x}_s \in \mathbf{X}$ is the starting state of the agent; and finally, $\gamma \in [0, 1)$ is the discount rate on future rewards.

A policy of the agent is a mapping $\pi : \mathbf{X} \times A \to [0, 1]$ so that $\pi(\mathbf{x}, a)$ tells the probability that the agent chooses action a in state $\mathbf{x}$. For any $\mathbf{x}_0 \in \mathbf{X}$, the policy of the agent and the parameters of the MDP determine a stochastic process experienced by the agent through the instantiation

$$\mathbf{x}_0, a_0, r_0, \mathbf{x}_1, a_1, r_1, \ldots, \mathbf{x}_t, a_t, r_t, \ldots$$

The goal is to find a policy that maximises the expected value of the discounted total reward. Let the value function of policy π be

$$V^\pi(\mathbf{x}) := E\Big(\sum_{t=0}^{\infty} \gamma^t r_t \;\Big|\; \mathbf{x} = \mathbf{x}_0\Big)$$

[1]Later on, we shall generalise the concept of the state of the system. A state of the system will be a vector of state variables in our fMDP description. For that reason, we already use the boldface vector notation in this preliminary description.

and let the optimal value function be

$$V^*(\mathbf{x}) := \max_\pi V^\pi(\mathbf{x})$$

for each $\mathbf{x} \in \mathbf{X}$. If V^* is known, it is easy to find an optimal policy π^*, for which $V^{\pi^*} \equiv V^*$. Provided that history does not modify transition probability distribution $P(\mathbf{y}|\mathbf{x}, a)$ at any time instant, value functions satisfy the famous Bellman equations

$$V^\pi(\mathbf{x}) = \sum_a \sum_{\mathbf{y}} \pi(\mathbf{x}, a) P(\mathbf{y} \mid \mathbf{x}, a)\Big(R(\mathbf{x}, a) + \gamma V^\pi(\mathbf{y})\Big) \tag{1}$$

and

$$V^*(\mathbf{x}) = \max_a \sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}, a)\Big(R(\mathbf{x}, a) + \gamma V^*(\mathbf{y})\Big). \tag{2}$$

Most algorithms that solve MDPs build upon some version of the Bellman equations. In the following, we shall concentrate on the value iteration algorithm.

2.2 Exact Value Iteration

Consider an MDP $(\mathbf{X}, A, P, R, \mathbf{x}_s, \gamma)$. The value iteration for MDPs uses the Bellman equations (2) as an iterative assignment: It starts with an arbitrary value function $V_0 : \mathbf{X} \to \mathbb{R}$, and in iteration t it performs the update

$$V_{t+1}(\mathbf{x}) := \max_a \sum_{\mathbf{y} \in \mathbf{X}} P(\mathbf{y} \mid \mathbf{x}, a)\Big(R(\mathbf{x}, a) + \gamma V_t(\mathbf{y})\Big) \tag{3}$$

for all $\mathbf{x} \in \mathbf{X}$. For the sake of better readability, we shall introduce vector notation. Let $N := |\mathbf{X}|$, and suppose that states are integers from 1 to N, i.e. $\mathbf{X} = \{1, 2, \ldots, N\}$. Clearly, value functions are equivalent to N-dimensional vectors of reals, which may be indexed with states. The vector corresponding to V will be denoted as $\mathbf{v}$ and the value of state $\mathbf{x}$ by $\mathbf{v}_\mathbf{x}$. Similarly, for each a let us define the N-dimensional column vector $\mathbf{r}^a$ with entries $\mathbf{r}^a_\mathbf{x} = R(\mathbf{x}, a)$ and $N \times N$ matrix P^a with entries $P^a_{\mathbf{x},\mathbf{y}} = P(\mathbf{y} \mid \mathbf{x}, a)$. With these notations, (3) can be written compactly as

$$\mathbf{v}_{t+1} := \mathbf{max}_{a \in A}\big(\mathbf{r}^a + \gamma P^a \mathbf{v}_t\big). \tag{4}$$

Here, **max** denotes the componentwise maximum operator.

It is also convenient to introduce the *Bellman operator* $\mathcal{T} : \mathbb{R}^N \to \mathbb{R}^N$ that maps value functions to value functions as

$$\mathcal{T}\mathbf{v} := \mathbf{max}_{a \in A}\big(\mathbf{r}^a + \gamma P^a \mathbf{v}\big).$$

As it is well known, $\mathcal{T}$ is a max-norm contraction with contraction factor γ: for any $\mathbf{v}, \mathbf{u} \in \mathbb{R}^N$, $\|\mathcal{T}\mathbf{v} - \mathcal{T}\mathbf{u}\|_\infty \le \gamma \|\mathbf{v} - \mathbf{u}\|_\infty$. Consequently, by Banach's fixed point theorem, exact value iteration (which can be expressed compactly as $\mathbf{v}_{t+1} := \mathcal{T}\mathbf{v}_t$) converges to an unique solution $\mathbf{v}^*$ from any initial vector $\mathbf{v}_0$, and the solution $\mathbf{v}^*$ satisfies the Bellman equations (2). Furthermore, for any required precision $\epsilon > 0$, $\|\mathbf{v}_t - \mathbf{v}^*\|_\infty \le \epsilon$ if $t \ge \frac{\log \epsilon}{\log \gamma} \|\mathbf{v}_0 - \mathbf{v}^*\|_\infty$. One iteration costs $O(N^2 \cdot |A|)$ computation steps.

2.3 Approximate value iteration

In this section we shall review approximate value iteration (AVI) with linear function approximation (LFA) in ordinary MDPs. The results of this section hold for AVI in general, but if we can perform all operations effectively on compact representations (i.e. execution time is polynomially bounded in the number of variables instead of the number of states), then the method can be directly applied to the domain of factorised Markovian decision problems, underlining the importance of our following considerations.

Suppose that we wish to express the value functions as the linear combination of K *basis functions* $h_k : \mathbf{X} \to \mathbb{R}$ $(k \in \{1, \dots, K\})$, where $K << N$. Let H be the $N \times K$ matrix with entries $H_{\mathbf{x},k} = h_k(\mathbf{x})$. Let $\mathbf{w}_t \in \mathbb{R}^K$ denote the weight vector of the basis functions at step t. We can substitute $\mathbf{v}_t = H\mathbf{w}_t$ into the right hand side (r.h.s.) of (4), but we cannot do the same on the left hand side (l.h.s.) of the assignment: in general, the r.h.s. is not contained in the image space of H, so there is no such $\mathbf{w}_{t+1}$ that

$$H\mathbf{w}_{t+1} = \mathbf{max}_{a \in A}\big(\mathbf{r}^a + \gamma P^a H \mathbf{w}_t\big).$$

We can put the iteration into work by projecting the right-hand side into $\mathbf{w}$-space: let $\mathcal{G} : \mathbb{R}^N \to \mathbb{R}^K$ be a (possibly non-linear) mapping, and consider the iteration

$$\mathbf{w}_{t+1} := \mathcal{G}\big[\mathbf{max}_{a \in A}\big(\mathbf{r}^a + \gamma P^a H \mathbf{w}_t\big)\big] \tag{5}$$

with an arbitrary starting vector $\mathbf{w}_0$.

Lemma 1. *If $\mathcal{G}$ is such that $H\mathcal{G}$ is a non-expansion, i.e., for any $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^N$,*

$$\|H\mathcal{G}\mathbf{v} - H\mathcal{G}\mathbf{v}'\|_\infty \le \|\mathbf{v} - \mathbf{v}'\|_\infty,$$

then there exists a $\mathbf{w}^* \in \mathbb{R}^K$ such that

$$\mathbf{w}^* = \mathcal{G}\big[\mathbf{max}_{a \in A}\big(\mathbf{r}^a + \gamma P^a H \mathbf{w}^*\big)\big]$$

and iteration (5) *converges to $\mathbf{w}^*$ from any starting point.*

Proof. We can write (5) compactly as $\mathbf{w}_{t+1} = \mathcal{G}\mathcal{T}H\mathbf{w}_t$. Let $\widehat{\mathbf{v}}_t = H\mathbf{w}_t$. This satisfies

$$\widehat{\mathbf{v}}_{t+1} = H\mathcal{G}\mathcal{T}\widehat{\mathbf{v}}_t. \tag{6}$$

It is easy to see that the operator $H\mathcal{G}\mathcal{T}$ is a contraction: for any $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^N$,

$$\|H\mathcal{G}\mathcal{T}\mathbf{v} - H\mathcal{G}\mathcal{T}\mathbf{v}'\|_\infty \quad \le \quad \|\mathcal{T}\mathbf{v} - \mathcal{T}\mathbf{v}'\|_\infty \le \gamma\|\mathbf{v} - \mathbf{v}'\|_\infty$$

by the assumption of the lemma and the contractivity of $\mathcal{T}$. Therefore, by Banach's fixed point theorem, there exists a vector $\widehat{\mathbf{v}}^* \in \mathbb{R}^N$ such that $\widehat{\mathbf{v}}^* = H\mathcal{G}\mathcal{T}\widehat{\mathbf{v}}^*$ and iteration (6) converges to $\widehat{\mathbf{v}}^*$ from any starting point. It is easy to see that $\mathbf{w}^* = \mathcal{G}\mathcal{T}\widehat{\mathbf{v}}^*$ satisfies the statement of the lemma.

□

Note that if $\mathcal{G}$ is a linear mapping with matrix $G \in \mathbb{R}^{K \times N}$, then the assumption of the lemma is equivalent to $\|HG\|_\infty \le 1$.

2.4 Examples of Projections, Convergent and Divergent

In this section, we examine certain possibilities for choosing projection $\mathcal{G}$. Let $\mathbf{v} \in \mathbb{R}^N$ be an arbitrary vector, and let $\mathbf{w} = \mathcal{G}\mathbf{v}$ be its $\mathcal{G}$-projection. For linear operators, $\mathcal{G}$ can be represented in matrix form and we shall denote it by G.

Least-squares (L_2-)projection. Least-squares fitting is used almost exclusively for projecting value functions, and the term AVI is usually used in the sense "AVI with least-squares projection". In this case, $\mathbf{w}$ is chosen so that it minimises the least-squares error:

$$\mathbf{w} := \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathbf{v}\|_2^2.$$

This corresponds to the linear projection $G_2 = H^+$ (i.e., $\mathbf{w} = H^+\mathbf{v}$), where H^+ is the Moore-Penrose pseudoinverse of H. It is well known, however, that this method can diverge. For an example on such divergence, see, e.g. the book of Bertsekas & Tsitsiklis [4]. The reason is simple: matrix HH^+ is a non-expansion in L_2-norm, but Lemma 1 requires that it should be an L_∞-norm projection, which does not hold in the general case. (See also Appendix A.1 for illustration.)

Constrained least-squares projection. One can enforce the non-expansion property by expressing it as a constraint: Let $\mathbf{w}$ be the solution of the constrained minimisation problem

$$\mathbf{w} := \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathbf{v}\|_2^2, \text{ subject to } \|H\mathbf{w}\|_\infty \leq \|\mathbf{v}\|_\infty,$$

which defines a non-linear mapping $\mathcal{G}_2^c$. This projection is computationally highly demanding: in each step of the iteration, one has to solve a quadratic programming problem.

Max-norm (L_∞-)projection. Similarly to L_2-projection, we can also select $\mathbf{w}$ so that it minimises the max-norm of the residual:

$$\mathbf{w} := \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathbf{v}\|_\infty.$$

The computation of $\mathbf{w}$ can be transcribed into a linear programming task and that defines the non-linear mapping $\mathcal{G}_\infty$. However, in general, $\|H\mathcal{G}_\infty\mathbf{v}\|_\infty \not\leq \|\mathbf{v}\|_\infty$, and consequently AVI using iteration

$$\mathbf{w}_{t+1} := \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathcal{T}H\mathbf{w}_t\|_\infty$$

can be divergent. Similarly to L_2 projection, one can also introduce a constrained version $\mathcal{G}_\infty^c$ defined by

$$\mathcal{G}_\infty^c\mathbf{v} := \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathbf{v}\|_\infty, \text{ subject to } \|H\mathbf{w}\|_\infty \leq \|\mathbf{v}\|_\infty,$$

which can also be turned into a linear program.

It is insightful to contrast this with the approximate linear programming method of Guestrin et al. [14]: they directly minimise the max-norm of the Bellman error, i.e., they solve the problem

$$\mathbf{w}^* := \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathcal{T}H\mathbf{w}\|_\infty,$$

which can be solved without constraints.

L_1-norm projection. Let $\mathcal{G}_{L_1}$ be defined by

$$\mathcal{G}_1\mathbf{v} := \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathbf{v}\|_1.$$

The L_1-norm projection also requires the solution of a linear program, but interestingly, the projection operator $\mathcal{G}_1$ is a non-expansion (the proof can be found in Appendix A.1).

AVI-compatible operators considered so far ($\mathcal{G}_2^c$, $\mathcal{G}_\infty^c$ and $\mathcal{G}_1$) were non-linear, and required the solution of a linear program or a quadratic program in each step of value iteration, which is clearly cumbersome. On the other hand, while $G_2\mathbf{v} = H^+\mathbf{v}$ is linear, it is also known to be incompatible with AVI [2, 27]. Now, we shall focus on operators that are both AVI-compatible and linear.

Normalised linear mapping. Let G be an arbitrary $K \times N$ matrix, and define its normalisation $\mathcal{N}(G)$ as a matrix with the same dimensions and entries

$$[\mathcal{N}(G)]_{ij} = \frac{G_{ij}}{\|[HG]_{i,*}\|_\infty}.$$

that is, $N(G)$ is obtained from G by dividing each element with the norm of the corresponding row of HG. All (absolute) row sums of $H \cdot \mathcal{N}(G)$ are equal to 1. Therefore, (i) $\|H \cdot \mathcal{N}(G)\|_\infty = 1$, and (ii) $H \cdot N(G)$ is maximal in the sense that if the absolute value of any element of $\mathcal{N}(G)$ increased, then for the resulting matrix G', $\|H \cdot G'\|_\infty > 1$.

Probabilistic linear mapping. If all elements of H are non-negative and all the row-sums of H are equal, then $\mathcal{N}(H^T)$ assumes a probabilistic interpretation. This interpretation is detailed in Appendix A.2.

Normalised least-squares projection. Among all linear operators, H^+ is the one that guarantees the best least-squares error, therefore we may expect that its normalisation, $\mathcal{N}(H^+)$ plays a similar role among *AVI-compatible* linear projections. Unless noted otherwise, we will use the projection $\mathcal{N}(H^+)$ subsequently.

2.5 Convergence properties

Lemma 2. *Let $\mathbf{v}^*$ be the optimal value function and $\mathbf{w}^*$ be the fixed point of the approximate value iteration (5). Then*

$$\|H\mathbf{w}^* - \mathbf{v}^*\|_\infty \le \frac{1}{1-\gamma}\|H\mathcal{G}\mathbf{v}^* - \mathbf{v}^*\|_\infty.$$

Proof. For the optimal value function, $\mathbf{v}^* = \mathcal{T}\mathbf{v}^*$ holds. On the other hand, $\mathbf{w}^* = \mathcal{G}\mathcal{T}H\mathbf{w}^*$. Thus,

$$\begin{aligned}
\|H\mathbf{w}^* - \mathbf{v}^*\|_\infty &= \|H\mathcal{G}\mathcal{T}H\mathbf{w}^* - \mathcal{T}\mathbf{v}^*\|_\infty \\
&\le \|H\mathcal{G}\mathcal{T}H\mathbf{w}^* - H\mathcal{G}\mathcal{T}\mathbf{v}^*\|_\infty + \|H\mathcal{G}\mathcal{T}\mathbf{v}^* - \mathcal{T}\mathbf{v}^*\|_\infty \\
&\le \|\mathcal{T}H\mathbf{w}^* - \mathcal{T}\mathbf{v}^*\|_\infty + \|H\mathcal{G}\mathbf{v}^* - \mathbf{v}^*\|_\infty \\
&\le \gamma\|H\mathbf{w}^* - \mathbf{v}^*\|_\infty + \|H\mathcal{G}\mathbf{v}^* - \mathbf{v}^*\|_\infty,
\end{aligned}$$

from which the statement of the lemma follows. For the transformations we have applied the triangle inequality, the non-expansion property of $H\mathcal{G}$ and the contraction property of $\mathcal{T}$. $\square$

According to the lemma, the error bound is proportional to the projection error of $\mathbf{v}^*$. Therefore, if $\mathbf{v}^*$ can be represented in the space of basis functions with small error, then our AVI algorithm gets close to the optimum. Furthermore, the lemma can be used to check *a posteriori* how good our basis functions are. One may improve the set of basis functions iteratively. Similar arguments have been brought up by Guestrin et al. [14], in association with their LP-based solution algorithm.

3 Factored value iteration

MDPs are attractive because solution time is polynomial in the number of states. Consider, however, a sequential decision problem with m variables. In general, we need an exponentially large state space to model it as an MDP. So, the number of states is *exponential* in the size of the description of the task. Factored Markov decision processes may avoid this trap because of their more compact task representation.

3.1 Factored Markov decision processes

We assume that $\mathbf{X}$ is the Cartesian product of m smaller state spaces (corresponding to individual variables):

$$\mathbf{X} = X_1 \times X_2 \times \ldots \times X_m.$$

For the sake of notational convenience we will assume that each X_i has the same size, $|X_1| = |X_2| = \ldots = |X_m| = n$. With this notation, the size of the full state space is $N = |\mathbf{X}| = n^m$. We note that all derivations and proofs carry through to different size variable spaces.

A naive, tabular representation of the transition probabilities would require exponentially large space (that is, exponential in the number of variables m). However, the next-step value of a state variable often depends only on a few other variables, so the full transition probability can be obtained as the product of several simpler factors. For a formal description, we introduce several notations:

For any subset of variable indices $Z \subseteq \{1, 2, \ldots, m\}$, let $\mathbf{X}[Z] := \times_{i \in Z} X_i$, furthermore, for any $\mathbf{x} \in \mathbf{X}$, let $\mathbf{x}[Z]$ denote the value of the variables with indices in Z. We shall also use the notation $\mathbf{x}[Z]$ without specifying a full vector of values $\mathbf{x}$, in such cases $\mathbf{x}[Z]$ denotes an element in $\mathbf{X}[Z]$. For single-element sets $Z = \{i\}$ we shall also use the shorthand $\mathbf{x}[\{i\}] = \mathbf{x}[i]$.

A function f is a *local-scope* function if it is defined over a subspace $\mathbf{X}[Z]$ of the state space, where Z is a (presumably small) index set. The local-scope function f can be extended trivially to the whole state space by $f(\mathbf{x}) := f(\mathbf{x}[Z])$. If $|Z|$

is small, local-scope functions can be represented efficiently, as they can take only $n^{|Z|}$ different values.

Suppose that for each variable i there exist neighbourhood sets Γ_i such that the value of $\mathbf{x}_{t+1}[i]$ depends only on $\mathbf{x}_t[\Gamma_i]$ and the action a_t taken. Then we can write the transition probabilities in a factored form

$$P(\mathbf{y} \mid \mathbf{x}, a) = \prod_{i=1}^{n} P_i(\mathbf{y}[i] \mid \mathbf{x}[\Gamma_i], a) \tag{7}$$

for each $\mathbf{x}, \mathbf{y} \in \mathbf{X}$, $a \in A$, where each factor is a local-scope function

$$P_i : \mathbf{X}[\Gamma_i] \times A \times X_i \to [0,1] \qquad \text{(for all } i \in \{1, \ldots, m\}). \tag{8}$$

We will also suppose that the reward function is the sum of J local-scope functions:

$$R(\mathbf{x}, a) = \sum_{j=1}^{J} R_j(\mathbf{x}[Z_j], a), \tag{9}$$

with arbitrary (but preferably small) index sets Z_j, and local-scope functions

$$R_j : \mathbf{X}[Z_j] \times A \to \mathbb{R} \qquad \text{(for all } j \in \{1, \ldots, J\}). \tag{10}$$

To sum up, a factored Markov decision process is characterised by the parameters $\Big(\{X_i : 1 \le i \le m\}; A; \{R_j : 1 \le j \le J\}; \{\Gamma_i : 1 \le i \le n\}; \{P_i : 1 \le i \le n\}; \mathbf{x}_s; \gamma\Big)$, where $\mathbf{x}_s$ denotes the initial state.

Functions P_i and R_i are usually represented either as tables or dynamic Bayesian networks. If the maximum size of the appearing local scopes is bounded by some constant, then the description length of an fMDP is polynomial in the number of variables n.

3.1.1 Value functions

The optimal value function is an $N = n^m$-dimensional vector. To represent it efficiently, we should rewrite it as the sum of local-scope functions with small domains. Unfortunately, in the general case, no such factored form exists [14].

However, we can still approximate V^* with such an expression: let K be the desired number of basis functions and for each $k \in \{1, \ldots, K\}$, let C_k be the domain set of the local-scope basis function $h_k : \mathbf{X}[C_k] \to \mathbb{R}$. We are looking for a value function of the form

$$\tilde{V}(\mathbf{x}) = \sum_{k=1}^{K} w_k \cdot h_k(\mathbf{x}[C_k]). \tag{11}$$

The quality of the approximation depends on two factors: the choice of the basis functions and the approximation algorithm. Basis functions are usually selected by the experiment designer, and there are no general guidelines how to automate this process. For given basis functions, we can apply a number of algorithms to determine the weights w_k. We give a short overview of these methods in Section 4. Here, we concentrate on value iteration.

3.2 Exploiting factored structure in value iteration

For fMDPs, we can substitute the factored form of the transition probabilities (7), rewards (9) and the factored approximation of the value function (11) into the AVI formula (5), which yields

$$\begin{aligned}\sum_{k=1}^{K} h_k(\mathbf{x}[C_k]) \cdot w_{k,t+1} \quad &\approx \quad \max_a \sum_{\mathbf{y}\in\mathbf{X}} \Big(\prod_{i=1}^{m} P_i(\mathbf{y}[i] \mid \mathbf{x}[\Gamma_i], a)\Big) \cdot \\ &\cdot \Big(\sum_{j=1}^{J} R_j(\mathbf{x}[Z_j], a) + \gamma \sum_{k'=1}^{K} h_{k'}(\mathbf{y}[C_{k'}]) \cdot w_{k',t}\Big).\end{aligned}$$

By rearranging operations and exploiting that all occurring functions have a local scope, we get

$$\begin{aligned}\sum_{k=1}^{K} h_k(\mathbf{x}[C_k]) \cdot w_{k,t+1} = \mathcal{G}_k \max_a \Bigg[\sum_{j=1}^{J} R_j(\mathbf{x}[Z_j], a) \\ + \gamma \sum_{k'=1}^{K} \sum_{\mathbf{y}[C_{k'}]\in\mathbf{X}[C_{k'}]} \Big(\prod_{i\in C_{k'}} P_i(\mathbf{y}[i] \mid \mathbf{x}[\Gamma_i], a)\Big) h_{k'}(\mathbf{y}[C_{k'}]) \cdot w_{k',t} \Bigg]\end{aligned} \qquad (12)$$

for all $\mathbf{x} \in \mathbf{X}$. We can write this update rule more compactly in vector notation. Let

$$\mathbf{w}_t := (w_{1,t}, w_{2,t}, \ldots, w_{K,t}) \in \mathbb{R}^K,$$

and let H be an $|\mathbf{X}| \times K$ matrix containing the values of the basis functions. We index the rows of matrix H by the elements of $\mathbf{X}$:

$$H_{\mathbf{x},k} := h_k(\mathbf{x}[C_k]).$$

Further, for each $a \in A$, let B^a be the $|\mathbf{X}| \times K$ *value backprojection* matrix defined as

$$B^a_{\mathbf{x},k} := \sum_{\mathbf{y}[C_k]\in\mathbf{X}[C_k]} \Big(\prod_{i\in C_k} P_i(\mathbf{y}[i] \mid \mathbf{x}[\Gamma_i], a)\Big) h_k(\mathbf{y}[C_k])$$

and for each a, define the reward vector $\mathbf{r}^a \in \mathbb{R}^{|\mathbf{X}|}$ by

$$\mathbf{r}^a_{\mathbf{x}} := \sum_{j=1}^{n_r} R_j(\mathbf{x}[Z_j], a).$$

Using these notations, (12) can be rewritten as

$$\mathbf{w}_{t+1} := \mathcal{G}\mathbf{max}_{a\in A}\Big(\mathbf{r}^a + \gamma B^a \mathbf{w}_t\Big). \qquad (13)$$

Now, all entries of B^a, H and $\mathbf{r}^a$ are composed of local-scope functions, so any of their individual elements can be computed efficiently. This means that the time required for the computation is exponential in the sizes of function scopes, but only polynomial in the number of variables, making the approach attractive. Unfortunately, the matrices are still exponentially large, as there are exponentially many equations in (12). One can overcome this problem by sampling as we show below.

3.3 Sampling

To circumvent the problem of having exponentially many equations, we select a random subset $\widehat{\mathbf{X}} \subseteq \mathbf{X}$ of the original state space so that $|\widehat{\mathbf{X}}| = \mathrm{poly}(m)$, consequently, solution time will scale polynomially with m. On the other hand, we will select a sufficiently large subset so that the remaining system of equations is still over-determined. The necessary size of the selected subset is to be determined later: it should be as small as possible, but the solution of the reduced equation system should remain close to the original solution with high probability. For the sake of simplicity, we assume that the projection operator $\mathcal{G}$ is linear with matrix G. Let the sub-matrices of G, H, B^a and $\mathbf{r}^a$ corresponding to $\widehat{\mathbf{X}}$ be denoted by $\widehat{G}$, $\widehat{H}$, $\widehat{B}^a$ and $\widehat{\mathbf{r}}^a$, respectively. Then the following value update

$$\mathbf{w}_{t+1} := \widehat{G} \cdot \mathbf{max}_{a \in A}\Big(\widehat{\mathbf{r}}^a + \gamma \widehat{B}^a \mathbf{w}_t\Big) \tag{14}$$

can be performed effectively, because these matrices have polynomial size. Now we show that the solution from sampled data is close to the true solution with high probability.

Theorem 1. *Let* $\mathbf{w}^*$ *be the unique solution of* $\mathbf{w}^* = G\mathcal{T}H\mathbf{w}^*$ *of an FMDP, and let* $\mathbf{w}'$ *be the solution of the corresponding equation with sampled matrices,* $\mathbf{w}' = \widehat{G}\mathcal{T}\widehat{H}\mathbf{w}'$. *Suppose that the projection matrix* G *has a factored structure, too. Then iteration* (14) *converges to* $\mathbf{w}'$, *furthermore, for a suitable constant* Ξ *(depending polynomially on* n^z, *where* z *is the maximum cluster size), and for any* $\epsilon, \delta > 0$, $\|\mathbf{w}^* - \mathbf{w}'\|_\infty \leq \epsilon$ *holds with probability at least* $1 - \delta$, *if the sample size satisfies* $N_1 \geq \Xi \dfrac{m^2}{\epsilon^2} \log \dfrac{m}{\delta}$.

The proof of Theorem 1 can be found in Appendix A.3. The derivation is closely related to the work of Drineas and colleagues [11, 12] with the important exception we use the infinity-norm instead of the L_2-norm. The resulting *factored value iteration* algorithm is summarised in Algorithm 1.

4 Related work

The exact solution of factored MDPs is infeasible. The idea of representing a large MDP using a factored model was first proposed by Koller & Parr [17] but similar ideas appear already in the works of Boutilier, Dearden, & Goldszmidt [5, 6]. More recently, the framework (and some of the algorithms) was extended to fMDPs with

Algorithm 1 Factored value iteration with a linear projection matrix G.

```
% inputs:
% factored MDP, $\mathcal{M} = \big(\{X_i\}_{i=1}^m; A; \{R_j\}_{j=1}^J; \{\Gamma_i\}_{i=1}^m; \{P_i\}_{i=1}^m; \mathbf{x}_s; \gamma\big)$
% basis functions, $\{h_k\}_{k=1}^K$
% required accuracy, $\epsilon$
$N_1$ := number of samples
$\widehat{\mathbf{X}}$ := uniform random $N_1$-element subset of $\mathbf{X}$
create $\widehat{H}$ and $\widehat{G}$
create $\widehat{B}^a = \widehat{P^a H}$ and $\widehat{\mathbf{r}}^a$ for each $a \in A$
$\mathbf{w}_0 = \mathbf{0}$, $t := 0$
repeat
    $\mathbf{w}_{t+1} := \widehat{G} \cdot \max_{a \in A}\Big(\widehat{\mathbf{r}}^a + \gamma \widehat{B}^a \mathbf{w}_t\Big)$
    $\Delta_t := \|\mathbf{w}_{t+1} - \mathbf{w}_t\|_\infty$
    $t := t + 1$
until $\Delta_t \geq \epsilon$
return $\mathbf{w}_t$
```

hybrid continuous-discrete variables [18] and factored partially observable MDPs [23]. Furthermore, the framework has also been applied to structured MDPs with alternative representations, e.g., relational MDPs [13] and first-order MDPs [24].

4.1 Algorithms for solving factored MDPs

There are two major branches of algorithms for solving fMDPs: the first one approximates the value functions as decision trees, the other one makes use of linear programming.

Decision trees (or equivalently, decision lists) provide a way to represent the agent's policy compactly. Koller & Parr [17] and Boutilier et al. [5, 6] present algorithms to evaluate and improve such policies, according to the policy iteration scheme. Unfortunately, the size of the policies may grow exponentially even with a decision tree representation [6, 20].

The exact Bellman equations (2) can be transformed to an equivalent linear program with N variables $\{V(\mathbf{x}) : \mathbf{x} \in \mathbf{X}\}$ and $N \cdot |A|$ constraints:

$$\begin{array}{ll} \text{maximise:} & \displaystyle\sum_{\mathbf{x} \in \mathbf{X}} \alpha(\mathbf{x}) V(\mathbf{x}) \\ \text{subject to} & \displaystyle V(\mathbf{x}) \leq R(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}' \in \mathbf{X}} P(\mathbf{x}' \mid \mathbf{x}, a) V(\mathbf{x}'), \quad (\forall \mathbf{x} \in \mathbf{X}, a \in A). \end{array}$$

Here, weights $\alpha(\mathbf{x})$ are free parameters and can be chosen freely in the following sense: the optimum solution is V^* independent of their choice, provided that each of them is greater than 0. In the approximate linear programming approach, we approximate the value function as a linear combination of basis functions (11), resulting in an approximate LP with K variables $\{w_k : 1 \leq k \leq K\}$ and $N \cdot |A|$ constraints:

$$\begin{aligned}
\text{maximise:} \quad & \sum_{k=1}^{K} \sum_{\mathbf{x} \in \mathbf{X}} w_k \cdot \alpha(\mathbf{x}) h_k(\mathbf{x}[C_k]) \qquad (15) \\
\text{subject to} \quad & \sum_{k=1}^{K} w_k \cdot h_k(\mathbf{x}[C_k]) \leq \\
& \leq R(\mathbf{x}, a) + \gamma \sum_{k'=1}^{K} w_{k'} \sum_{\mathbf{x}' \in \mathbf{X}} P(\mathbf{x}' \mid \mathbf{x}, a) \cdot h_{k'}(\mathbf{x}'[C_{k'}]).
\end{aligned}$$

Both the objective function and the constraints can be written in compact forms, exploiting the local-scope property of the appearing functions.

Markov decision processes were first formulated as LP tasks by Schweitzer and Seidmann [25]. The approximate LP form is due to de Farias and van Roy [7]. Guestrin et al. [14] show that the maximum of local-scope functions can be computed by rephrasing the task as a non-serial dynamic programming task and eliminating variables one by one. Therefore, (15) can be transformed to an equivalent, more compact linear program. The gain may be exponential, but this is not necessarily so in all cases: according to Guestrin et al. [14], "as shown by Dechter [9], [the cost of the transformation] is exponential in the induced width of the cost network, the undirected graph defined over the variables $X_1; \ldots; X_n$, with an edge between X_l and X_m if they appear together in one of the original functions f_j. The complexity of this algorithm is, of course, dependent on the variable elimination order and the problem structure. Computing the optimal elimination order is an NP-hard problem [1] and elimination orders yielding low induced tree width do not exist for some problems." Furthermore, for the approximate LP task (15), the solution is no longer independent of α and the optimal choice of the α values is not known.

The approximate LP-based solution algorithm is also due to Guestrin et al. [14]. Dolgov and Durfee [10] apply a primal-dual approximation technique to the linear program, and report improved results on several problems.

The approximate policy iteration algorithm [17, 14] also uses an approximate LP reformulation, but it is based on the policy-evaluation Bellman equation (1). Policy-evaluation equations are, however, linear and do not contain the maximum operator, so there is no need for the second, costly transformation step. On the other hand, the algorithm needs an explicit decision tree representation of the policy. Liberatore [20] has shown that the size of the decision tree representation can grow exponentially.

4.1.1 Applications

Applications of fMDP algorithms are mostly restricted to artificial test problems like the problem set of Boutilier et al. [6], various versions of the SysAdmin task [14, 10, 21] or the New York driving task [23].

Guestrin, Koller, Gearhart and Kanodia [13] show that their LP-based solution algorithm is also capable of solving more practical tasks: they consider the real-time strategy game *FreeCraft*. Several scenarios are modelled as fMDPs, and solved successfully. Furthermore, they find that the solution generalises to larger tasks with similar structure.

4.1.2 Unknown environment

The algorithms discussed so far (including our FVI algorithm) assume that all parameters of the fMDP are known, and the basis functions are given. In the case when only the factorisation structure of the fMDP is known but the actual transition probabilities and rewards are not, one can apply the factored versions of E^3 [16] or R-max [15].

Few attempts exist that try to obtain basis functions or the structure of the fMDP automatically. Patrascu et al. [21] select basis functions greedily so that the approximated Bellman error of the solution is minimised. Poupart et al. [22] apply greedy selection, too, but their selection criteria are different: a decision tree is constructed to partition the state space into several regions, and basis functions are added for each region. The approximate value function is piecewise linear in each region. The metric they use for splitting is related to the quality of the LP solution.

4.2 Sampling

Sampling techniques are widely used when the state space is immensely large. Lagoudakis and Parr [19] use sampling without a theoretical analysis of performance, but the validity of the approach is verified empirically. De Farias and van Roy [8] give a thorough overview on constraint sampling techniques used for the linear programming formulation. These techniques are, however, specific to linear programming and cannot be applied in our case.

The work most similar to ours is that of Drineas et al. [12, 11]. They investigate the least-squares solution of an overdetermined linear system, and they prove that it is sufficient to keep polynomially many samples to reach low error with high probability. They introduce a non-uniform sampling distribution, so that the variance of the approximation error is minimised. However, the calculation of the probabilities requires a complete sweep through all equations.

5 Conclusions

In this paper we have proposed a new algorithm, factored value iteration, for the approximate solution of factored Markov decision processes. The classical approximate value iteration algorithm is modified in two ways. Firstly, the least-squares projection operator is substituted with an operator that does not increase max-norm, and thus preserves convergence. Secondly, polynomially many samples are sampled uniformly from the (exponentially large) state space. This way, the complexity of our algorithm becomes polynomial in the size of the fMDP description length. We prove that the algorithm is convergent and give a bound on the difference between our solution and the optimal one. We also analysed various projection operators with respect to their computation complexity and their convergence when combined with approximate value iteration. To our knowledge, this is the first algorithm that (1) provably converges in polynomial time and (2) avoids linear programming.

Acknowledgements

The authors are grateful to Zoltán Szabó for calling our attention to the articles of Drineas et al. [11, 12]. This research has been supported by the EC FET 'New and Emergent World models Through Individual, Evolutionary, and Social Learning' Grant (Reference Number 3752). Opinions and errors in this manuscript are the author's responsibility, they do not necessarily reflect those of the EC or other project members.

A Proofs

A.1 Projections in various norms

We wish to know whether $\mathbf{w}_0 = \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathbf{v}\|_p$ implies $\|H\mathbf{w}_0\|_\infty \le \|\mathbf{v}\|_\infty$ for various values of p. Specifically, we are interested in the cases when $p \in \{1, 2, \infty\}$. Fig. 1 indicates that the implication does not hold for $p = 2$ or $p = \infty$, only for the case $p = 1$. Below we give a rigorous proof for these claims.

Consider the example $\mathbf{v} = [1, 1]^T \in \mathbb{R}^2$, $H = [1, 2]^T$, $\mathbf{w} \in \mathbb{R}^1$. For these values easy calculation shows that $\|H[\mathbf{w}_0]_{L_2}\|_\infty = 6/5$ and $\|H[\mathbf{w}_0]_{L_\infty}\|_\infty = 4/3$, i.e., $\|H\mathbf{w}_0\|_\infty \not\le \|\mathbf{v}\|_\infty$ for both cases. For $p = 1$, we shall prove the following lemma:

Lemma 3. *If* $\mathbf{w}_0 = \arg\min_{\mathbf{w}} \|H\mathbf{w} - \mathbf{v}\|_1$*, then* $\|H\mathbf{w}_0\|_\infty \le \|\mathbf{v}\|_\infty$.

Proof. Let $\mathbf{z} := H\mathbf{w}_0 \in \mathbb{R}^N$. If there are multiple solutions to the minimisation task, then consider the (unique) $\mathbf{z}$ vector with minimum L_2-norm. Let $r := \|\mathbf{z} - \mathbf{v}\|_1$ and let $S(\mathbf{v}, r)$ be the L_1-sphere with centre $\mathbf{v}$ and radius r (this is an N-dimensional *cross polytope* or *orthoplex*, a generalisation of the octahedron).

Suppose indirectly that $\|\mathbf{z}\|_\infty > \|\mathbf{v}\|_\infty$. Without loss of generality we may assume that z_1 is the coordinate of $\mathbf{z}$ with the largest absolute value, and that it is positive. Therefore, $z_1 > \|\mathbf{v}\|_\infty$. Let $\mathbf{e}_i$ denote the ith coordinate vector $(1 \le i \le N)$, and let $\mathbf{z}_\delta = \mathbf{z} - \delta\mathbf{e}_1$. For small enough δ, $S(\mathbf{z}_\delta, \delta)$ is a cross polytope such that (a) $S(\mathbf{z}_\delta, \delta) \subset S(\mathbf{v}, r)$, (b) $\forall \mathbf{z}' \in S(\mathbf{z}_\delta, \delta)$, $\|\mathbf{z}'\|_\infty > \|\mathbf{v}\|_\infty$, (c) $\forall \epsilon > 0$ sufficiently small, $(1 - \epsilon)\mathbf{z} \in S(\mathbf{z}_\delta, \delta)$. The first two statements are trivial. For the third statement, note that $\mathbf{z}$ is a vertex of the cross polytope $S(\mathbf{z}_\delta, \delta)$. Consider the cone whose vertex is $\mathbf{z}$ and its edges are the same as the edges of $S(\mathbf{z}_\delta, \delta)$ joining $\mathbf{z}$. It is easy to see that the vector pointing from $\mathbf{z}$ to the origo is contained in this

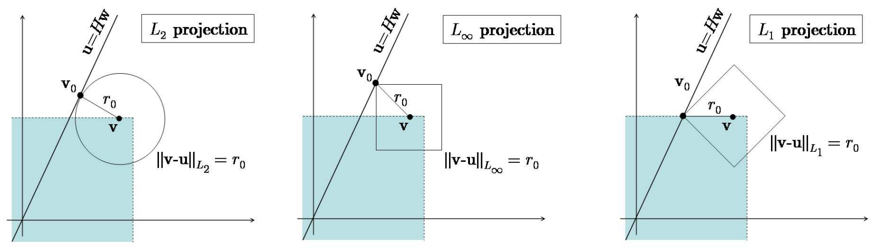

Figure 1: Projections in various norms. The vector $\mathbf{v}$ is projected onto the image space of H, i.e., the subspace defined by $\mathbf{u} = H\mathbf{w}$. Consider the smallest sphere around $\mathbf{v}$ (in the corresponding norm) that touches the subspace $\mathbf{u} = H\mathbf{w}$ (shown in each figure). The radius r_0 of this sphere is the distance of $\mathbf{v}$ from the subspace, and the tangent point $\mathbf{v}_0$ (which is not necessarily unique for L_1 projection) is the projection of $\mathbf{v}$. For this point, $\mathbf{v}_0 = H\mathbf{w}_0$ holds. The shaded region indicates the region $\{\mathbf{u} : \|\mathbf{u}\|_\infty \le \|\mathbf{v}\|_\infty\}$. To ensure the convergence of FVI, the projected vector $\mathbf{v}_0$ must fall into the shaded region.

cone: for each $1 < i \le N$, $|z_i| \le z_1$ (as z_1 is the largest coordinate). Consequently, for small enough ϵ, $\mathbf{z} - \epsilon\mathbf{z} \in S(\mathbf{z}_\delta, \delta)$.

Fix such an ϵ and let $\mathbf{q} = (1-\epsilon)\mathbf{z}$. This vector is (a) contained in the image space of H because $H[(1-\epsilon)\mathbf{w}] = \mathbf{q}$; (b) $\|\mathbf{q} - \mathbf{v}\|_1 \le \|\mathbf{z} - \mathbf{v}\|_1 = r$. The vector $\mathbf{z}$ was chosen so that it has the smallest L_1-norm in the image space of H, so the inequality cannot be sharp, i.e., $\|\mathbf{q} - \mathbf{v}\|_1 = r$. However, $\|\mathbf{q}\|_2 = (1-\epsilon)\|\mathbf{z}\|_2 < \|\mathbf{z}\|_2$ with strict inequality, which contradicts our assumption about $\mathbf{z}$, thus completing the proof. □

A.2 Probabilistic interpretation of $N(H^T)$

Definition 1. *The basis functions* $\{h_k\}_{k=1}^{n_b}$ *have the* uniform covering (UC) *property, if all row sums in the corresponding* H *matrix are identical:*

$$\sum_{k=1}^{n_b} H_{\mathbf{x},k} = B \qquad \textit{for all } \mathbf{x} \in \mathbf{X},$$

and all entries are non-negative. Without loss of generality we may assume that all rows sum up to 1, i.e., H *is a stochastic matrix.*

We shall introduce an auxiliary MDP $\overline{\mathcal{M}}$ such that exact value iteration in $\overline{\mathcal{M}}$ is identical to the approximate value iteration in the original MDP $\mathcal{M}$. Let $\mathbf{S}$ be an K-element state space with states $\mathbf{s}_1, \ldots, \mathbf{s}_K$. A state $\mathbf{s}$ is considered a discrete observation of the true state of the system, $\mathbf{x} \in \mathbf{X}$.

The action space A and the discount factor γ are identical to the corresponding items of $\mathcal{M}$, and an arbitrary element $\mathbf{s}_0 \in \mathbf{S}$ is selected as initial state. In order to obtain the transition probabilities, let us consider how one can get from observing $\mathbf{s}$ to observing $\mathbf{s}'$ in the next time step: from observation $\mathbf{s}$, we can infer the hidden state $\mathbf{x}$ of the system; in state $\mathbf{x}$, the agent makes action a and transfers to state $\mathbf{x}'$ according to the original MDP; after that, we can infer the probability that our observation will be $\mathbf{s}'$, given the hidden state $\mathbf{x}'$. Consequently, the transition probability $\overline{P}(\mathbf{s}' \mid \mathbf{s}, a)$ can be defined as the total probability of all such paths:

$$\overline{P}(\mathbf{s}' \mid \mathbf{s}, a) := \sum_{\mathbf{x},\mathbf{x}' \in \mathbf{X}} \Pr(\mathbf{x} \mid \mathbf{s}) \Pr(\mathbf{x}' \mid \mathbf{x}, a) \Pr(\mathbf{s}' \mid \mathbf{x}).$$

Here the middle term is just the transition probability in the original MDP, the rightmost term is $H_{\mathbf{x},\mathbf{s}}$, and the leftmost term can be rewritten using Bayes' law (assuming a uniform prior on $\mathbf{x}$):

$$\Pr(\mathbf{x} \mid \mathbf{s}) \quad = \quad \frac{\Pr(\mathbf{s} \mid \mathbf{x}) \Pr(\mathbf{x})}{\sum_{\mathbf{x}'' \in \mathbf{X}} \Pr(\mathbf{s} \mid \mathbf{x}'') \Pr(\mathbf{x}'')} = \frac{H_{\mathbf{x},\mathbf{s}} \cdot \frac{1}{|\mathbf{X}|}}{\sum_{\mathbf{x}'' \in \mathbf{X}} H_{\mathbf{x}'',\mathbf{s}} \cdot \frac{1}{|\mathbf{X}|}} = \frac{H_{\mathbf{x},\mathbf{s}}}{\sum_{\mathbf{x}'' \in \mathbf{X}} H_{\mathbf{x}'',\mathbf{s}}}.$$

Consequently,

$$\overline{P}(\mathbf{s}' \mid \mathbf{s}, a) = \sum_{\mathbf{x},\mathbf{x}' \in \mathbf{X}} \frac{H_{\mathbf{x},\mathbf{s}}}{\sum_{\mathbf{x}'' \in \mathbf{X}} H_{\mathbf{x}'',\mathbf{s}}} P(\mathbf{x}' \mid \mathbf{x}, a) H_{\mathbf{x},\mathbf{s}} = \left[\mathcal{N}(H)^T P^a H\right]_{\mathbf{s},\mathbf{s}'}.$$

The rewards can be defined similarly:

$$\overline{R}(\mathbf{s}, a) := \sum_{\mathbf{x} \in \mathbf{X}} \Pr(\mathbf{x} \mid \mathbf{s}) R(\mathbf{x}, a) = \left[\mathcal{N}(H)^T \mathbf{r}^a\right]_{\mathbf{s}}.$$

It is easy to see that approximate value iteration in $\mathcal{M}$ corresponds to exact value iteration in the auxiliary MDP $\overline{\mathcal{M}}$.

A.3 The proof of the sampling theorem (theorem 1)

First we prove a useful lemma about approximating the product of two large matrices. Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times k}$ and let $C = A \cdot B$. Suppose that we sample columns of A uniformly at random (with repetition), and we also select the corresponding rows of B. Denote the resulting matrices with $\widehat{A}$ and $\widehat{B}$. We will show that $A \cdot B \approx c \cdot \widehat{A} \cdot \widehat{B}$, where c is a constant scaling factor compensating for the dimension decrease of the sampled matrices. The following lemma is similar to Lemma 11 of [11], but here we estimate the infinity-norm instead of the L_2-norm.

Lemma 4. *Let $A \in \mathbb{R}^{m \times N}$, $B \in \mathbb{R}^{N \times k}$ and $C = A \cdot B$. Let N' be an integer so that $1 \le N' \le N$, and for each $i \in \{1, \dots, N'\}$, let r_i be an uniformly random integer from the interval $[1, N]$. Let $\widehat{A} \in \mathbb{R}^{m \times N'}$ be the matrix whose i^{th} column is the ${r_i}^{th}$ column of A, and denote by $\widehat{B}$ the $N' \times k$ matrix that is obtained by sampling the rows of B similarly. Furthermore, let*

$$\widehat{C} = \frac{N}{N'} \widehat{A} \cdot \widehat{B} = \frac{N}{N'} \sum_{i=1}^{N'} A_{*,r_i} B_{r_i,*}.$$

Then, for any $\epsilon, \delta > 0$, $\|\widehat{C} - C\|_\infty \le \epsilon N \|A\|_\infty \|B^T\|_\infty$ with probability at least $1-\delta$, if the sample size satisfies $N' \ge \frac{2m^2}{\epsilon^2} \log \frac{2km}{\delta}$.

Proof. We begin by bounding individual elements of the matrix $\widehat{C}$: consider the element

$$\widehat{C}_{pq} = \frac{N}{N'} \sum_{i=1}^{N'} A_{p,r_i} B_{r_i,q}.$$

Let $\mathcal{C}_{pq}$ be the discrete probability distribution determined by mass points $\{A_{p,i} \cdot B_{i,q} \mid 1 \le i \le N\}$. Note that $\widehat{C}_{pq}$ is essentially the sum of N' random variables drawn uniformly from distribution $\mathcal{C}_{pq}$. Clearly,

$$|A_{pi} B_{iq}| \le \max_{ij} |A_{ij}| \max_{ij} |B_{ij}| \le \max_i \sum_j |A_{ij}| \max_i \sum_j |B_{ij}| = \|A\|_\infty \|B\|_\infty,$$

so we can apply Hoeffding's inequality to obtain

$$\Pr\Big(\Big| \frac{\sum_{i=1}^{N'} A_{p,r_i} B_{r_i,q}}{N'} - \frac{\sum_{j=1}^{N} A_{p,j} B_{j,q}}{N} \Big| > \epsilon_1 \Big) < 2 \exp\Big(- \frac{N' \epsilon_1^2}{2\|A\|_\infty^2 \|B\|_\infty^2} \Big),$$

or equivalently,

$$\Pr\Big(\Big| \frac{N}{N'} [\widehat{A}\widehat{B}]_{pq} - [AB]_{pq} \Big| > N\epsilon_1 \Big) < 2 \exp\Big(- \frac{N' \epsilon_1^2}{2\|A\|_\infty^2 \|B\|_\infty^2} \Big),$$

where $\epsilon_1 > 0$ is a constant to be determined later. From this, we can bound the row sums of $\widehat{C} - C$:

$$\Pr\Big(\sum_{p=1}^{m} \Big| \widehat{C}_{pq} - C_{pq} \Big| > m \cdot N\epsilon_1 \Big) < 2m \exp\Big(- \frac{N' \epsilon_1^2}{2\|A\|_\infty^2 \|B\|_\infty^2} \Big),$$

which gives a bound on $\|\widehat{C} - C\|_\infty$. This is the maximum of these row sums:

$$\begin{aligned} \Pr\Big(\|\widehat{C} - C\|_\infty > mN\epsilon_1 \Big) &= \Pr\Big(\max_q \sum_{p=1}^{m} \Big| \widehat{C}_{pq} - C_{pq} \Big| > mN\epsilon_1 \Big) \\ &< 2km \exp\Big(- \frac{N' \epsilon_1^2}{2\|A\|_\infty^2 \|B\|_\infty^2} \Big). \end{aligned}$$

Therefore, by substituting $\epsilon_1 = \epsilon \|A\|_\infty \|B\|_\infty / m$, the statement of the lemma is satisfied if $2km \exp\Big(- \frac{N' \epsilon^2}{2m^2} \Big) \le \delta$, i.e, if $N' \ge \frac{2m^2}{\epsilon^2} \log \frac{2km}{\delta}$. □

If both A and B are structured, we can sharpen the lemma to give a much better (potentially exponentially better) bound. For this, we need the following definition:

For any index set Z, a matrix A is called Z-local-scope matrix, if each column of A represents a local-scope function with scope Z.

Lemma 5. *Let A^T and B be local-scope matrices with scopes Z_1 and Z_2, and let $N_0 = n^{|Z_1|+|Z_2|}$, and apply the random row/column selection procedure of the previous lemma. Then, for any $\epsilon, \delta > 0$, $\|\widehat{C} - C\|_\infty \le \epsilon N_0 \|A\|_\infty \|B\|_\infty$ with probability at least $1-\delta$, if the sample size satisfies $N' \ge \frac{2m^2}{\epsilon^2} \log \frac{2km}{\delta}$.*

Proof. Fix a variable assignment $\mathbf{x}[Z_1 \cup Z_2]$ on the domain $Z_1 \cup Z_2$ and consider the rows of A that correspond to a variable assignment compatible to $\mathbf{x}[Z_1 \cup Z_2]$, i.e., they are identical to it for components $Z_1 \cup Z_2$ and are arbitrary on

$$W := \{1, 2, \ldots, m\} \setminus (Z_1 \cup Z_2).$$

It is easy to see that all of these rows are identical because of the local-scope property. The same holds for the columns of B. All the equivalence classes of rows/columns have cardinality

$$N_1 := n^{|W|} = N/N_0.$$

Now let us define the $m \times N_0$ matrix A' so that only one column is kept from each equivalence class, and define the $N_0 \times k$ matrix B' similarly, by omitting rows. Clearly,

$$A \cdot B = N_1 A' \cdot B',$$

and we can apply the sampling lemma to the smaller matrices A' and B' to get that for any $\epsilon, \delta > 0$ and sample size $N' \ge \frac{2m^2}{\epsilon^2} \log \frac{2km}{\delta}$, with probability at least $1-\delta$,

$$\|\widehat{A' \cdot B'} - A' \cdot B'\|_\infty \le \epsilon N_0 \|A'\|_\infty \|B'\|_\infty.$$

Exploiting the fact that the max-norm of a matrix is the maximum of row norms, $\|A'\|_\infty = \|A\|_\infty / N_1$ and $\|B'\|_\infty = \|B\|_\infty$, we can multiply both sides to get

$$\|N_1 \widehat{A' \cdot B'} - A \cdot B\|_\infty \le \epsilon N_0 N_1 \|A\|_\infty / N_1 \|B\|_\infty = \epsilon N_0 \|A\|_\infty \|B\|_\infty,$$

which is the statement of the lemma. □

Note that if the scopes Z_1 and Z_2 are small, then the gain compared to the previous lemma can be exponential.

Lemma 6. *Let $A = A_1 + \ldots + A_p$ and $B = B_1 + \ldots + B_q$ where all A_i and B_j are local-scope matrices with domain size at most z, and let $N_0 = n^z$. If we apply the random row/column selection procedure, then for any $\epsilon, \delta > 0$, $\|\widehat{C} - C\|_\infty \le \epsilon N_0 pq \max_i \|A_i\|_\infty \max_j \|B_j\|_\infty$ with probability at least $1-\delta$, if the sample size satisfies $N' \ge \frac{2m^2}{\epsilon^2} \log \frac{2km}{\delta}$.*

Proof.

$$\|\widehat{C} - C\|_\infty \le \sum_{i=1}^{p} \sum_{j=1}^{q} \|\widehat{A_i \cdot B_j} - A_i \cdot B_j\|_\infty.$$

For each individual product term we can apply the previous lemma. Note that we can use the same row/column samples for each product, because independence is required only *within* single matrix pairs. Summing the right-hand sides gives the statement of the lemma. □

Now we can complete the proof of Theorem 1:

Proof.

$$\begin{aligned} \|\mathbf{w}^* - \mathbf{w}'\|_\infty &= \|G\mathcal{T}H\mathbf{w}^* - \widehat{G}\mathcal{T}\widehat{H}\mathbf{w}'\|_\infty \\ &\le \|G\mathcal{T}H\mathbf{w}^* - \widehat{G}\mathcal{T}\widehat{H}\mathbf{w}^*\|_\infty + \|\widehat{G}\mathcal{T}\widehat{H}\mathbf{w}^* - \widehat{G}\mathcal{T}\widehat{H}\mathbf{w}'\|_\infty \\ &\le \|G\mathcal{T}H\mathbf{w}^* - \widehat{G}\mathcal{T}\widehat{H}\mathbf{w}^*\|_\infty + \gamma\|\mathbf{w}^* - \mathbf{w}'\|_\infty, \end{aligned}$$

i.e., $\|\mathbf{w}^* - \mathbf{w}'\|_\infty \le \frac{1}{1-\gamma}\|G\mathcal{T}H\mathbf{w}^* - \widehat{G}\mathcal{T}\widehat{H}\mathbf{w}^*\|_\infty$. Let π_0 be the greedy policy with respect to the value function $H\mathbf{w}^*$. With its help, we can rewrite $\mathcal{T}H\mathbf{w}^*$ as a linear expression: $\mathcal{T}H\mathbf{w}^* = \mathbf{r}^{\pi_0} + \gamma P^{\pi_0}H\mathbf{w}^*$. Furthermore, $\mathcal{T}$ is a componentwise operator, so we can express its effect on the downsampled value function as $\mathcal{T}\widehat{H}\mathbf{w}^* = \widehat{\mathbf{r}}^{\pi_0} + \gamma\widehat{P^{\pi_0}H}\mathbf{w}^*$. Consequently,

$$\|G\mathcal{T}H\mathbf{w}^* - \widehat{G}\mathcal{T}\widehat{H}\mathbf{w}^*\|_\infty \le \|G\mathbf{r}^{\pi_0} - \widehat{G}\widehat{\mathbf{r}}^{\pi_0}\|_\infty + \gamma\|GP^{\pi_0}H - \widehat{G}\widehat{P^{\pi_0}H}\|_\infty\|\mathbf{w}^*\|_\infty$$

Applying the previous lemma two times, we get that with probability greater than $1-\delta_1$, $\|G\mathbf{r}^{\pi_0} - \widehat{G}\widehat{\mathbf{r}}^{\pi_0}\|_\infty \le \epsilon_1 C_1$ if $N' \ge \frac{2m^2}{\epsilon_1^2}\log\frac{2m}{\delta_1}$ and with probability greater than $1-\delta_2$, $\|GP^{\pi_0}H - \widehat{G}\widehat{P^{\pi_0}H}\|_\infty \le \epsilon_2 C_2$ if $N' \ge \frac{2m^2}{\epsilon_2^2}\log\frac{2m^2}{\delta_2}$; where C_1 and C_2 are constants depending polynomially on N_0 and the norm of the component local-scope functions, but independent of N.

Using the notation $M = \frac{1}{1-\gamma}\Big(C_1 + \gamma C_2\|\mathbf{w}^*\|_\infty\Big)$, $\epsilon_1 = \epsilon_2 = \epsilon/M$, $\delta_1 = \delta_2 = \delta/2$ and $\Xi = M^2$ proves the theorem. □

Informally, this theorem tells that the required number of samples grows quadratically with the desired accuracy $1/\epsilon$ and logarithmically with the required certainty $1/\delta$, furthermore, the dependence on the number of variables m is slightly worse than quadratic. This means that even if the number of equations is exponentially large, i.e., $N = O(e^m)$, we can select a polynomially large random subset of the equations so that with high probability, the solution does not change very much.

References

[1] Arnborg, Stefan, Corneil, Derek G., and Proskurowski, Andrzej. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[2] Baird, Leemon C. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the International Conference on Machine Learning*, pages 30–37, 1995.

[3] Bellman, Richard E. *Adaptive Control Processes.* Princeton University Press, Princeton, NJ., 1961.

[4] Bertsekas, Dimitri P. and Tsitsiklis, John N. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[5] Boutilier, Craig, Dearden, Richard, and Goldszmidt, Moises. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1104–1111, 1995.

[6] Boutilier, Craig, Dearden, Richard, and Goldszmidt, Moises. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000.

[7] de Farias, Daniela Pucci and van Roy, Benjamin. Approximate dynamic programming via linear programming. In *Advances in Neural Information Processing Systems 14*, pages 689–695, 2001.

[8] de Farias, Daniela Pucci and van Roy, Benjamin. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[9] Dechter, Rina. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[10] Dolgov, Dmitri A. and Durfee, Edmund H. Symmetric primal-dual approximate linear programming for factored MDPs. In *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics (AI&M 2006)*, 2006.

[11] Drineas, Petros, Kannan, Ravi, and Mahoney, Michael W. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal of Computing*, 36:132–157, 2006.

[12] Drineas, Petros, Mahoney, Michael W., and Muthukrishnan, S. Sampling algorithms for l2 regression and applications. In *Proc. 17-th Annual SODA*, pages 1127–1136, 2006.

[13] Guestrin, Carlos, Koller, Daphne, Gearhart, Chris, and Kanodia, Neal. Generalizing plans to new environments in relational MDPs. In *Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[14] Guestrin, Carlos, Koller, Daphne, Parr, Ronald, and Venkataraman, Shobha. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2002.

[15] Guestrin, Carlos, Patrascu, Relu, and Schuurmans, Dale. Algorithm-directed exploration for model-based reinforcement learning in factored mdps. In *Proceedings of the International Conference on Machine Learning*, pages 235–242, 2002.

[16] Kearns, Michael J. and Koller, Daphne. Efficient reinforcement learning in factored MDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 740–747, 1999.

[17] Koller, Daphne and Parr, Ronald. Policy iteration for factored mdps. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 326–334, 2000.

[18] Kveton, Branislav, Hauskrecht, Milos, and Guestrin, Carlos. Solving factored MDPs with hybrid state and action variables. *Journal of Artificial Intelligence Research*, 27:153–201, 2006.

[19] Lagoudakis, Michail G. and Parr, Ronald. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[20] Liberatore, Paolo. The size of MDP factored policies. In *Proceedings of the 18th National Conference on Artificial intelligence*, pages 267–272, 2002.

[21] Patrascu, Relu, Poupart, Pascal, Schuurmans, Dale, Boutilier, Craig, and Guestrin, Carlos. Greedy linear value-approximation for factored markov decision processes. In *Proceedings of the 18th National Conference on Artificial intelligence*, pages 285–291, 2002.

[22] Poupart, Pascal, Boutilier, Craig, Patrascu, Relu, and Schuurmans, Dale. Piecewise linear value function approximation for factored mdps. In *Proceedings of the 18th National Conference on AI*, 2002.

[23] Sallans, Brian. *Reinforcement Learning for Factored Markov Decision Processes.* PhD thesis, University of Toronto, 2002.

[24] Sanner, Scott and Boutilier, Craig. Approximate linear programming for first-order MDPs. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 509–517, 2005.

[25] Schweitzer, Paul J. and Seidmann, Abraham. Generalized polynomial approximations in markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(6):568–582, 1985.

[26] Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, 1998.

[27] Tsitsiklis, John N. and Roy, Benjamin Van. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.