Approximate Equivalence of Markov Decision Processes

Eyal Even-Dar and Yishay Mansour

School of Computer Science
Tel Aviv University, Tel-Aviv 69978, Israel
{evend,mansour}@cs.tau.ac.il

Abstract. We consider the problem of finding the minimal ϵ-equivalent MDP for an MDP given in its tabular form. We show that the problem is NP-Hard and then give a bicriteria approximation algorithm to the problem. We suggest that the right measure for finding minimal ϵ-equivalent model is L_1 rather than L_∞ by giving both an example, which demonstrates the drawback of using L_∞, and performance guarantees for using L_1. In addition, we give a polynomial algorithm that decides whether two MDPs are equivalent.

1 Introduction

In Reinforcement Learning, an agent wanders in an unknown environment and tries to maximize its long term return by performing actions and receiving rewards. The challenge is to understand how a current action will affect future rewards. A good way to model this task is the Markov Decision Process (MDP), which has become the dominating approach in Reinforcement Learning [2, 10].

An MDP includes states, which abstract the environment, actions, which are available to the agent, and for each state and action a distribution of next states, the state reached after performing the action in the given state. In addition there is a reward function that assigns a stochastic reward for each state and action; we combine a sequence of rewards into a single value, the return. A policy assigns a distribution over actions for each state.

We focus on the following problem: given an MDP compute an MDP, which is similar to the original MDP and has smaller state space. Our major requirement is that a good policy in the reduced MDP should translate back to a good policy in the original MDP. This will allow us to consider policies in the reduced MDP, and be guaranteed a similar performance in the original MDP. This problem is clearly interesting from the theoretical perspective. Other benefits of a reduced MDP might be, like any other reduced model, better understandability by humans and better generalization ability.

A related line of research is using compact models such as factored MDPs to represent MDPs. Factored MDPs [3] provide a compact representation of MDPs by using state variables instead of enumerating the entire state space. This compact representation has the potential of significantly reducing the number of states. However, this compact representation does not yield neither an efficient method to compute optimal policies [8] nor an efficient way to represent the optimal policy [1]. Givan *et al.* [5] derive an algorithm, which computes an equivalent minimal model for an MDP, when the MDP is given in its tabular form. For a factored MDP they show that the minimization problem is NP-Hard [5] and supply a heuristic that calculates a reduced model, but does not have any performance guarantee. In [4] they supply a method, which computes some ϵ-equivalent model in factored MDPs, but they do not show any performance guarantees or approximation ratio on the resulting MDP.

In this paper we study the problem of finding an ϵ-equivalent model for an MDP in its tabular form. We show that the right measure of computing an ϵ-equivalent model is L_1, rather than L_∞ that was used in [4, 5]. We show that an optimal policy in an ϵ-equivalent model, with respect to L_∞, might be arbitrarily far from the optimal policy in the original MDP, independent of ϵ. On the other hand we show that if a policy is optimal in an ϵ-equivalent model with respect to L_1, then its induced policy in the original model is close to the optimal policy.

We also show that computing the minimal ϵ-equivalent MDP for an MDP, given in its tabular form, is NP-Hard. This holds for both L_1 and L_∞ norm, and is done by a reduction from metric k cluster. Given that the problem is NP-Hard, one would like to derive an approximate solution. Usually in approximation algorithms we are given one objective function, which we try to minimize. In bicriteria approximations we are given two objective functions that we try to minimize. Thus, a bicriteria is an approximate solution in two parameters of the problem. Our criteria are both the MDP size and the ϵ of the ϵ-equivalence. We give an approximation algorithm for finding the ϵ-equivalent MDP and prove that this is an $(2\sqrt{|S|}, \sqrt{|S|})$-approximation. We actually prove that the resulting MDP is either 0-equivalent and with size no more than $2\sqrt{|S|}$ times the size of the minimal ϵ-equivalent MDP, or $\sqrt{|S|}\epsilon$-equivalent and with size no larger than the minimal ϵ-equivalent MDP.

Finally, we also present a simple polynomial algorithm which checks whether two MDPs are equivalent based on the algorithm which computes the minimal MDP [5].

2 Model

We define a Markov Decision process (MDP) as follows

Definition 1. *A Markov Decision process (MDP) M is a 4-tuple (S, A, P, R), where S is a set of the states, A is a set of actions ($A(i)$ is the set of actions available at state i), $P_M(i, j, a)$ is the transition probability from state i to state j when performing action $a \in A(i)$ in state i, and $R_M(s, a)$ is the reward received when performing action a in state s.*

Whenever it is clear from the context we use V^π instead of V^π_M. Next we give a definition of MDP equivalence and of an ϵ-homogenous partition.

Definition 2. *An MDP $M_1 = (S_1, A, R_1, P_1)$ and MDP $M_2 = (S_2, A, R_2, P_2)$ are ϵ-equivalent with respect to L_k norm if there exist mappings $\phi_1 : S_1 \to S$ and $\phi_2 : S_2 \to S$ to MDP $M = (S, A, R, P)$, such that ϕ_1 and ϕ_2 are surjective and for every $s_1 \in S_1, s_2 \in S_2$ such that $\phi_1(s_1) = \phi_2(s_2) = s$ the following holds:*

1. $\forall a \in A \quad \left(\sum_{s' \in S} \left(\sum_{\hat{s};\phi_i(\hat{s})=s'} P_i(s_i, \hat{s}, a) - P(s, s', a)\right)^k\right)^{1/k} \leq \epsilon \quad i = 1, 2$
2. $\forall a \in A \quad |R_i(s_i, a) - R(s, a)| \leq \epsilon \qquad i = 1, 2$

Definition 3. *An MDP $M_1 = (S_1, A, R_1, P_1)$ is an ϵ-homogenous partition of MDP $M = (S, A, R, P)$ with respect to L_k norm if there exists mapping $\phi : S \to S_1$, such that ϕ is surjective and for every s the following holds:*

1. $\forall a \in A \quad \left(\sum_{s' \in S} \left(\sum_{\hat{s};\phi(\hat{s})=s'} P(s, \hat{s}, a) - P_1(\phi(s), s', a)\right)^k\right)^{1/k} \leq \epsilon$
2. $\forall a \in A \quad \max_{\tilde{s}:\phi(\tilde{s})=s} |R(\tilde{s}, a) - R_1(s, a)| \leq \epsilon$

We note that if an MDP M_ϵ is an ϵ-homogenous partition of MDP M then they are ϵ-equivalent. We also note that the minimal equivalent MDP to an MDP M must be an ϵ-homogenous partition of M with $\epsilon = 0$, and that this MDP is unique.

A stochastic stationary policy for an MDP assigns, for each state s a probability for performing action $a \in A(s)$. While following a policy π we perform at time t action a_t at state s_t and observe a reward r_t (distributed according to $R_M(s, a)$), and the next state s_{t+1} (distributed according to $P_M(s_t, s_{t+1}, a_t)$). We combine the sequence of rewards to a single value called the return, and our goal is to maximize the return. The discounted return of policy π is $V_M^\pi = \sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the reward observed at time t. Since all the rewards are bounded by R_{max} the discounted return is bounded by $V_{max} = \frac{R_{max}}{1-\gamma}$.

We define a value function for each state s, under policy π, as $V_M^\pi(s) = E[\sum_{i=0}^{\infty} r_i \gamma^i]$, where the expectation is over a run of policy π starting at state s. We define a state-action value function $Q^\pi(s, a) = R(s, a) + \gamma \sum_{\bar{s}} P(s, \bar{s}, a) V^\pi(\bar{s})$, whose value is the return of initially performing action a at state s and then following policy π.

Let π^* be an optimal policy, which maximizes the return from any start state. (It is well known that there exists an optimal strategy, which is a deterministic policy, see [9].) This implies that for any policy π and any state s we have $V^{\pi^*}(s) \geq V^\pi(s)$, and $\pi^*(s) = argmax_a(R(s, a) + \gamma(\sum_{s'} P(s, s', a) \max_b Q(s', b))$. We say that a policy π is an ϵ-optimal if $\|V^* - V^\pi\|_\infty \leq \epsilon$.

3 Markov Decision Process Minimization

Since we build on some of Givan *et al.* [5] techniques, we describe briefly their minimization process. We first introduce a few notations and definitions. An equivalence relation is defined as a transitive binary relation, E. The equivalence relation groups similar states in the original MDP and in the reduced MDP and each equivalence class translates into one state. We often denote by a block a group of states in a specific equivalence class. Given an MDP, M and an equivalence relation $E \subseteq S \times S$, an operator I is defined as

$$I(E) = \{(i,j) | R(i) = R(j) \wedge E(i,j) \wedge P(i,i'|E,a) = P(j,i'|E,a)\},$$

where $P(i,j'|E,a) = \sum_{j:E(j,j')=1} P(i,j,a)$. The minimal model is the fixed point of the process $E_{n+1} = I(E_n)$, where $I(E) \subseteq E$.

We define the reward partition as the equivalence relation, $E(i,j) = \{i,j| \|R(i,\cdot) - R(j,\cdot)\|_\infty = 0\}$. We say that a block $B \subseteq S$ is stable with respect to block C if every state in B has the same transition probability with respect to C, i.e., for every $i,i' \in B$ we have $P(i,C,a) = P(i',C,a)$, where $P(i',C,a) = \sum_{j\in C} P(i',j,a)$. Let $B = (B_1,\ldots,B_m)$, where if $i \in B_k$ then $j \in B_k \iff E(i,j) = 1$. We define the procedure $SPLIT(B_i,B_j,P)$ that replaces the block B_i by the uniquely determined sub-blocks of B_i, $B_{i1},\ldots,B_{ik}$ such that each sub-block is the maximal stable sub-block with respect to B_j. In other words, all the states $s \in B_{il}$ have the same probability $P(s,B_j,a)$ for every action a and no two sub-blocks B_{il} and B_{ik} have the same set of probabilities.

The algorithm of Givan *et al.* [5] works as follows. It begins by making E_0 the reward partition. It then iterates and checks for every pair of blocks if they are stable; if not, it performs the procedure $SPLIT$. The algorithm terminates when all blocks are stable with respect to every other block. The algorithm clearly terminates, since each SPLIT increases the number of blocks and the number of blocks is bounded by $|S|$.

Theorem 1. *[5] The minimization problem for MDP, given in its tabular form, is in P.*

4 Hardness of epsilon Equivalence

In [5] it was shown that finding the minimal model is in P. On the other hand they have shown that finding the minimal (or minimal ϵ-equivalent) model is NP-Hard for factored MDPs. We show that finding the minimal ϵ-equivalent model is NP-Hard even for MDPs that are represented in a tabular form. We use here a variant of the k-center problem known as **metric k cluster**.

Lemma 1. *[6] Let $G = (V,E)$ be a complete undirected graph with edge costs satisfying the triangle inequality, and let k be a positive integer. Find a partition of V into sets $V_1,...,V_k$ such that the maximum cost of an edge between two vertices in the same set is minimized.*

We will use different notations and a more restricted problem to simplify the reduction.

Corollary 1. *The following ϵ-cover problem is NP-Complete. Given n points, $X = (x_1,...,x_n)$, where x_i is in $\Re^m$, a positive number K and a norm distance, L_q. Partition the points into K sets, $X_1,...X_K$ such that* $\max_{1\le i\le K\ p,p'\in X_i} \|p - p'\|_q \le \epsilon$.

We obtain our theorem by a reduction from the ϵ-cover problem.

Theorem 2. *Given an MDP, M, and a positive number K the problem of determining whether there exists an ϵ-homogenous partition, in L_q, of size no more than K is NP-Complete.*

Proof. The problem is clearly in NP. Next we show a reduction from metric k cluster to this problem. We reduce the n points, $x_1, ..., x_n$ to MDP $M = (s_1, ..., s_n)$ and keep K and ϵ identical in both problems. In M each state has m actions, the reward function is $R(s_i, a_j) = x_i^j$, where x_i^j is the jth coordinate of x_i, and the next state distribution is uniform, i.e. $\forall i, j, k \;\; P(s_i, s_j, a_k) = \frac{1}{n}$. We show that there exists an ϵ - homogenous partition of size no more than K if and only if there exists a partition of X into sets $X_1, ..., X_K$ such that $\max_{1 \le i \le K, p, p' \in X_i} \|p - p'\|_q \le \epsilon$. We first prove that if there exists an ϵ - homogenous partition of size K then there exists a K cluster. Since the next state distribution is uniform for every state-action pair, then an ϵ - homogenous partition of the MDP has to be according to the reward. Let $B_1, \dots, B_K$ be the partition for the MDP, then we have that $\max_{1 \le i \le K, s, s' \in B_i} \|R(s, \cdot) - R(s', \cdot)\|_q \le \epsilon$ and let $X_1, \dots, X_K$ be the matching partition in X. Now we show that the partition that X_i induces on X is an ϵ cover. We observe that

$$\begin{aligned} \max_{1 \le i \le K, u, v \in X_i} \|u - v\|_q &= \max_{1 \le i \le K, u, v \in X_i} \Big(\sum_{j=1}^{m} |u_j - v_j|^q\Big)^{1/q} \\ &= \max_{1 \le i \le K, s, s' \in B_i} \Big(\sum_{j=1}^{m} |R(s, a_j) - R(s', a_j)|^q\Big)^{1/q} \le \epsilon. \end{aligned}$$

Using the fact that every block is stable with respect to every block, the proof of the other direction is identical. □

5 L_1 versus L_∞

In this section we present an example, which demonstrates the drawback of using L_∞ as measure for an ϵ-homogenous partition of the MDP. On the other hand we derive performance guarantees for the L_1 measure.

Lemma 2. *For every $\epsilon > 0$, there exists an MDP M with an ϵ-homogenous partition, M_ϵ, with respect to L_∞, such that an optimal policy in M_ϵ induces a policy in M that is arbitrarily far from from the optimal policy.*

Proof. Consider the MDPs in Figure 1, where M and M_ϵ are ϵ-equivalent for $\epsilon = 1/n$. There are only two policies in M, π_a, which performs action a in state 1 and π_u, which performs action u in state 1. Note that in M_ϵ both policies are optimal. We consider in M_ϵ the policy, π_a, which performs action a in state 1. The induced policy, π_a in M, satisfies $V_M^{\pi_a}(1) \le 6$ for any γ. The optimal policy, π_u, satisfies $V_M^*(1) \ge \frac{R_{max}\gamma^3}{1-\gamma} = \gamma^3 V_{max}$. For constant γ the difference can be arbitrarily large, independent of ϵ. □

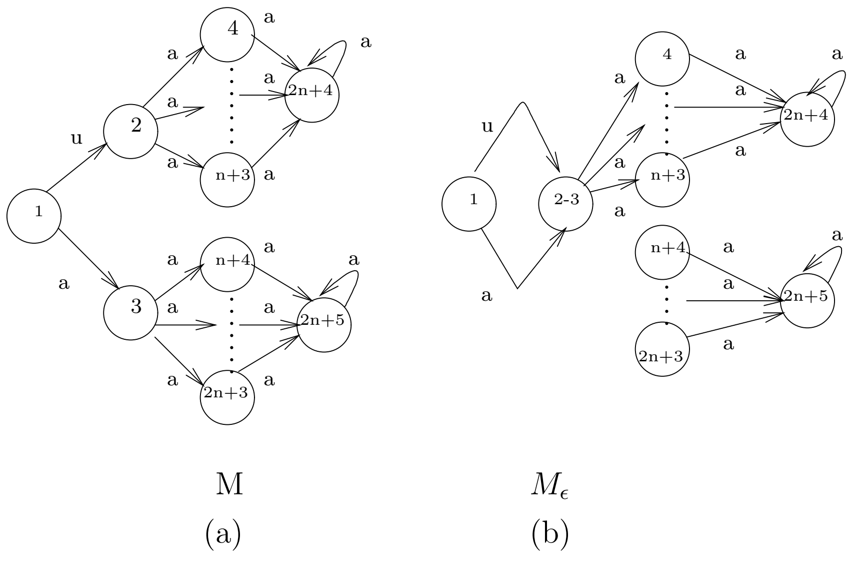

Fig. 1. An example in which two MDPs are ϵ-equivalent in L_∞ norm but an optimal policy in M_ϵ is far from optimal in M. Let $n = \lceil 1/\epsilon \rceil$. The rewards are $R(1, u) = R(1, a) = 0, R(2, a) = R(3, a) = 1, R(2n + 4, a) = R_{max}, R(2n + 5, a) = 0$ $\forall k \in \{4, 2n + 3\}$ $R(k, a) = 2\epsilon k$. In states 2, 3 the next state distribution is uniform. We note that there are only two policies in M, one performs a at state 1 and other performs u at state 1. In M_ϵ the two policies coincide.

In the next lemma we show that for every policy, π^ϵ in an ϵ-homogenous partition of an MDP, with respect to L_1 norm, its induced policy, π is at most $\epsilon\frac{V_{max}}{1-\gamma}$ from the value of π^ϵ in M_ϵ for every state. The proof is done by relying on the following fact:

Fact 3 *For any phase in Value Iteration algorithm we have that if* $\|V_{t+1} - V_t\|_\infty \leq \epsilon$ *then* $\|V_t - V^*\|_\infty \leq \frac{\epsilon}{1-\gamma}$

Lemma 3. *Let M_ϵ be an ϵ-homogenous partition of MDP M, with respect to L_1 norm, and π any policy in M_ϵ, then the induced policy π^M on M, that is $\pi^M(s) = \pi(\phi(s))$, satisfies*

$$\forall s \in S \quad |V_M^{\pi^M}(s) - V_{M_\epsilon}^{\pi}(\phi(s))| \leq \frac{\epsilon V_{max}}{1-\gamma}$$

Proof. The proof is done by estimating the value of π^M using Value Iteration with initial value $V_0^{\pi^M}(s) = V_{M_\epsilon}^{\pi}(\phi(s))$. Since we use L_1 norm, each state can deviate only in ϵ from its representative state in M_ϵ and the difference $\|V_0^{\pi^M} - V_1^{\pi^M}\|_\infty$ is bounded by ϵV_{max}. Using the fact that $\|V_0^{\pi^M} - V_1^{\pi^M}\|_\infty \leq \epsilon V_{max}$ and

Fact 3 we can conclude that

$$\max_{s\in S} |V_M^{\pi^M}(s) - V_{M_\epsilon}^{\pi}(\phi(s))| = \|V_0^{\pi^M} - V_M^{\pi^M}\|_\infty = \|V_0^{\pi^M} - V_\infty^{\pi^M}\|_\infty \leq \frac{\epsilon V_{max}}{1-\gamma},$$

where $V_\infty^{\pi^M} = V_M^{\pi^M}$ is the value of π^M in M. □

The next lemma shows that the optimal policy in M_ϵ induces a nearly optimal policy in M.

Lemma 4. *Let M_ϵ be an ϵ-homogenous partition of MDP M, with respect to L_1 norm, then the optimal policy in M_ϵ induces an $\frac{2\epsilon V_{max}}{1-\gamma}$-optimal policy in M.*

Proof. Let σ be the optimal policy in M_ϵ and σ^M be the induced policy in M. By Lemma 3 we have that $\max_s |V_M^{\sigma^M}(s) - V_{M_\epsilon}^{\sigma}(\phi(s))| \leq \frac{\epsilon V_{max}}{1-\gamma}$. Next we show that if we use a Value Iteration on M with initial value $V_0^M(s) = V_{M_\epsilon}^{\sigma}(\phi(s))$ then we have that

$$\|V_0^M - V_\infty^M\|_\infty = \|V_0^M - V_M^*\|_\infty \leq \frac{\epsilon V_{max}}{1-\gamma}.$$

Since we use L_1 norm, once again we have that $\|V_0^M - V_1^M\|_\infty$ is bounded by ϵV_{max}. Combining the two parts, we obtain that σ^M is an $\frac{2\epsilon V_{max}}{1-\gamma}$-optimal policy in M. □

6 Approximation scheme

Since finding the minimal ϵ-equivalent MDP is NP-Hard, we would like to find an approximate solution to the problem. The natural approximation is to give an MDP, which is ϵ-equivalent and is K times larger than the minimal ϵ-equivalent MDP. Unfortunately, we do not present such approximation. We present a bicriteria approximation algorithm for finding an ϵ-equivalent model for an MDP. Usually in approximation algorithms we are given one objective function, which we try to minimize. In bicriteria approximations two objective functions that we try to minimize are given. Thus, a bicriteria is an approximate solution in two parameters of the problem. Our criteria are both the MDP size and the parameter ϵ in the ϵ-equivalence. We prove that our algorithm is $(2\sqrt{|S|}, \sqrt{|S|})$ approximation with respect to (size, ϵ), that is our MDP might be $\sqrt{|S|}\epsilon$-equivalent and $2\sqrt{|S|}$ times larger than the minimal ϵ-equivalent MDP. In fact, for our algorithm either (1) produces an equivalent MDP, which is $2\sqrt{|S|}$ larger than the minimal size ϵ-equivalent MDP, or (2) produces an MDP which is $\sqrt{|S|}\epsilon$-equivalent, whose size is at most the size of the minimal ϵ-equivalent MDP.

We first give a high level description of our approximation algorithm. We initially build a graph $G = (S, E)$, where the vertices are the MDP states and there is an edge between states if their rewards are ϵ-close. After the initialization, we run in phases. In each phase we check for each edge if it is consistent with some legal partition. A partition is legal with respect to G if every vertex in it

```
Input   : MDP $M$, $\epsilon > 0$ precision
Output : A partition of M to Equivalence Relations
Build a graph $G = (S, E)$, where $(s, t) \in E$ if $\forall a \in A \; |R(s, a) - R(t, a)| \leq \epsilon$;
repeat
    $Deleted = FALSE$;
    foreach $(s, t) \in E$ do
        foreach $a \in A$ do
            if $CheckConsistent(s,t,a,E,S,P) = FALSE$ then
                $Deleted = TRUE$ ;
                $E = E - (s, t)$ ;
            end
        end
    end
until $Deleted = FALSE$;
Compute $C_1, \ldots, C_l$ the connected components of G= (S,E);
if $\max_i Diameter(C_i) \geq \sqrt{S}$ then
    Return $S$
else
    Return $C = \cup_i C_i$
end
```

Algorithm 1: An algorithm which calculates an ϵ-equivalent MDP

is connected to every other vertex. If we observe that the edge is not consistent with any legal partition, then we delete that edge. To check whether an edge (s, t) is consistent, we compute the connected components of G and see if s and t have similar next state transition with respect to them. (The detailed procedure $CheckConsistent$ is formally described in Alg. 2.) We terminate when we cannot delete any edges. Clearly the algorithm terminates after at most $|S|^2$ phases. (The Algorithm is formally described in Alg. 1)

Since our algorithm returns an MDP partition, we show how to construct an MDP from an MDP partition. We note that for an MDP M a partition of S by itself does not define a unique MDP, but a family of MDPs. This family can be viewed also as a bounded MDP [7]; we take a different approach and choose arbitrarily representative MDP for each partition.

Definition 4. *Let $M = (S, A, P, R)$ and ϕ be a mapping from S to S_1. Then the MDP, $M_1 = (S_1, A, P_1, R_1)$ induced by (M, ϕ) is defined as follows. For each $s \in S_1$ we choose arbitrarily s' such that $\phi(s') = s$. The reward in $s \in S_1$ is $R_1(s, a) = R(s', a)$ and the next state distribution is $P_1(s, t, a) = \sum_{\phi(t')=t} P(s', t', a)$.*

In the following lemmas we provide correctness proof of Algorithm 1. We first prove that if s and t are in the same equivalence class in some ϵ-homogenous partition, then the edge (s, t) is never removed from the graph during Algorithm. 1.

Lemma 5. *Let s and t be two states such that there exists an ϵ-homogenous partition MDP in which s and t are in the same equivalence class, then the edge (s, t) is never removed from the graph in Algorithm 1.*

```
Input  : state s , state t, action a, Edges between states E, MDP states S,
         Next state distribution P
Output : Boolean answer whether $s$ and $t$ are consistent
Compute the connected components of $G = (S, E)$, $C_1, \dots, C_l$;
if $\sum_{i=1}^{l} |P(s, C_i, a) - P(t, C_i, a)| \leq \epsilon$ then
    Return TRUE;
else
    Return FALSE;
end
```

Algorithm 2: CheckConsistent(s,t,E,S,P) Procedure

Proof. Let $C_1^k, \dots, C_l^k$ be the connected components of G at the kth stage and $EC_1, \dots, EC_m$ be the equivalence classes of an ϵ-homogenous partition MDP in which s and t are in the same equivalence class. We prove by induction on the algorithm steps (number of calls to *CheckConsistent*) that (1) the edge (s, t) is never removed (2) $C_i^k = \cup_{EC_j \subseteq C_i^k} EC_j$ for every i, i.e., each connected component is a union of equivalence classes from an ϵ-homogenous partition MDP. The basis is due to the fact that the first stage is the reward partition. We assume that the induction hypothesis holds for the first k stages and prove for the $k+1$ stage. WLOG, we assume that the edge (s, t) is checked at this stage. By the induction assumption we have that $C_i^k = \cup_{j:EC_j \subseteq C_i^k} EC_j$ for every i. Therefore, for every action a we obtain the following

$$\begin{aligned}
\sum_{i=1}^{l} |P(s, C_i^k, a) - P(t, C_i^k, a)| &= \sum_{i=1}^{l} \left| \sum_{j:EC_j \subseteq C_i^k} P(s, EC_j, a) - P(t, EC_j, a) \right| \\
&\leq \sum_{i=1}^{l} \sum_{j:EC_j \subseteq C_i^k} |P(s, EC_j, a) - P(t, EC_j, a)| \\
&= \sum_{j=1}^{m} |P(s, EC_j, a) - P(t, EC_j, a)| \leq \epsilon.
\end{aligned}$$

We conclude that (s, t) is never removed from the graph if (s, t) are in the same equivalence class in some ϵ-homogenous partition MDP. Therefore, all states in an equivalence class form a clique at every stage and each equivalence class is contained in a single connected component. □

Next we prove that at the end of the algorithm we have a $(2\sqrt{S}, \sqrt{S})$-approximation. We first show that the size of the resulting MDP is no larger than $2\sqrt{S}$ the minimal ϵ-equivalent MDP.

Lemma 6. *If* $\max_i \mathit{Diameter}(C_i) \geq \sqrt{|S|}$ *then there are at least* $\frac{\sqrt{|S|}}{2}$ *states in the minimal* ϵ*-equivalent MDP.*

Proof. Consider the diameter's path, $s_1, ..., s_{\sqrt{|S|}}$, of a connected component C. By Lemma 5 we have that no three states in the path can be in the same equivalence class. (Otherwise the diameter would have been smaller.) Thus there are at least $\left\lfloor \sqrt{|S|}/2 \right\rfloor$ states in the minimal ϵ-equivalent MDP. $\square$

Next we claim that if we take each connected component as an equivalence class, then they are ϵD-stable, where $D = \max_i Diameter(C_i)$.

Lemma 7. *Consider an MDP, M_1, which is induced by (M, ϕ), such that if $s, s' \in C_i$ in Algorithm 1 then $\phi(s) = \phi(s')$. Then we have that M_1 is ϵD-equivalent to M, where $D = \max_i Diameter(C_i)$.*

Proof. Since ϕ maps all the states in a connected component C_i to a single state in M, we let the states of M_1 be the connected components. This implies that if $s \in C_i$ then $\phi(s) = C_i$. First we show that for every connected component

$$\forall a \forall s, t \in C_i \quad |R(s, a) - R(t, a)| \leq \epsilon D$$

We prove it by induction on the length of the path between the states in C_i. For the basis we have that if the path length between s and t is one, then there is an edge between them. This implies that $max_a|R(s, a) - R(t, a)| \leq \epsilon$. Assume the induction assumption holds for $k - 1$ and prove for k and let $s = s_1, \ldots, s_k = t$ be the path between s and t. From the induction assumption we have that $max_a|R(s_1, a) - R(s_{k-1}, a)| \leq (k-1)\epsilon$ and since there is an edge between s_{k-1} to s_k we have that $max_a|R(s_{k-1}, a) - R(s_k, a)| \leq \epsilon$. Using the triangle inequality we complete the induction. Next we prove that for any s, t in a connected component the following holds:

$$\max_{a \in A} \sum_{C_i} \left| \sum_{\hat{s};\phi(\hat{s})=C_i} P_M(s, \hat{s}, a) - \sum_{\hat{s};\phi(\hat{s})=C_i} P_M(t, \hat{s}, a) \right| \leq \epsilon D$$

Fix an action a. We prove by induction on the path length that $\sum_{i=1}^{m} |P(s_1, C_i, a) - P(s_j, C_i, a)| \leq j\epsilon$, where m is the number of connected component. The basis of the induction is due to the fact that when the algorithm terminates, it cannot delete more edges, and therefore for every edge we have that $\sum_{i=1}^{m} |P(s_j, C_i, a) - P(s_{j+1}, C_i, a)| \leq \epsilon$. We assume that the induction assumption holds for $k-1$ and prove for k. For a path $s = s_1, \ldots, s_k = t$ in G, we show that $\sum_{i=1}^{m} |P(s_1, C_i, a) - P(s_k, C_i, a)| \leq k\epsilon$. Since $\sum_{i=1}^{m} |P(s_{k-1}, C_i, a) - P(s_k, C_i, a)| \leq \epsilon$ and by the induction assumption we have that $\sum_{i=1}^{m} |P(s_1, C_i, a) - P(s_{k-1}, C_i, a)| \leq (k-1)\epsilon$, we can conclude that $\sum_{i=1}^{m} |P(s_1, C_i, a) - P(s_k, C_i, a)| \leq k\epsilon$. $\square$

Combining Lemma 6 and Lemma 7 we obtain our main theorem.

Theorem 4. *Given an MDP M, Algorithm 1 terminates in polynomial time and returns an MDP partition, ϕ with induced MDP, $M_\epsilon = (M, \phi)$, that is at most $\sqrt{|S|}\epsilon$-equivalent and with size no larger than $2\sqrt{|S|}$ time the size of the minimal ϵ-equivalent MDP.*

```
Input  : state s , state t, action a, Edges between states E, MDP states S,
         Next state distribution P
Output : Answer whether $s$ and $t$ are consistent
Build a graph with four layres;
The first layer contains $s$ ;
The second layer contains $S$ ;
The third layer contains $S$ ;
The fourth layer contains $t$;
Connect $s$ to every vertex $s'$ in the second layer with capacity $c(s, s') = P(s, s', a)$;
If $(s', s'') \in E$ then connect $s'$ in the second layer to a vertex $s''$ in the third layer with capacity $c(s', s'') = 1$ ;
Connect every vertex in the third layer to $t$ with capacity $c(s', t) = P(t, s', a)$;
Calculate the max flow from $s$ to $t$;
if $max\ flow < 1 - \epsilon/2$ then
    Return FALSE;
else
    Return TRUE;
end
```

Algorithm 3: CheckConsistent1(s,t,E,S,P) Procedure

6.1 A heuristic Finer Partition

In this section, we suggest a "better" *CheckConsistent* procedure. This procedure always performs at least as good as the original *CheckConsistent* procedure. However, this procedure is more complicated and we could not provide a better performance grantees using it. The procedure *CheckConsistent*1 checks whether an edge (s, t) is consistent by building the graph (in Fig. 2) for each action and check if the max flow is close to 1, more precisely if it is at least $1-\epsilon/2$. (The detailed procedure *CheckConsistent*1 is formally described in Alg. 3.) We also note that the difference between the procedures is that in *CheckConsistent* procedure we connect states between the second and the third layer if they are in the same connected component, while in *CheckConsistent*1 procedure we connect them only if they have an edge in G.

7 Markov Decision Process Equivalence

We present a polynomial method to check whether two MDPs are equivalent. Our algorithm is very simple and uses any procedure to compute the minimal MDP, which we call minimal equivalent MDP. For instance such a procedure is given in [5].

Assuming the algorithm for finding the minimal equivalent model is polynomial in the MDP size as in [5], then our algorithm is polynomial as well.

Lemma 8. *Algorithm 4 runs in polynomial time in the MDP size.*

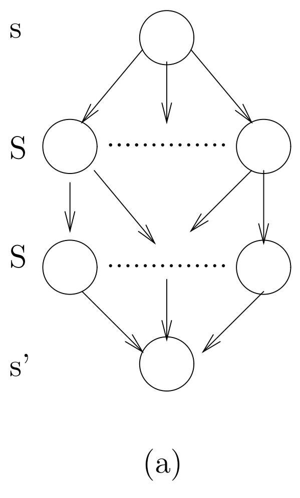

Fig. 2. The graph built by *CheckConsistent* procedure

Next we show the correctness of the algorithm. We first show that any two states from the same minimal MDP D_1 or D_2 have to be in the same equivalence relation of M.

Lemma 9. *No equivalence relation in the minimal partition of M contains two states from either D_1 or D_2.*

Proof. We let $D_1 = (s_1^1, ..., s_1^m)$ and let $D_2 = (s_2^1, ..., s_2^m)$. The proof is done by contradiction. Assume there are two states from D_1, s_1^i and s_1^j, in the same equivalence class in M, then we can construct to M_1 an equivalent model, which consists of at most $size(D_1) - 1$ states. Since s_1^i and s_1^j are stable with respect to any other block in D, then they are stable with respect to the projection of D to D_1 and can be in the same equivalence class. This contradicts the minimality of D_1. □

Lemma 10. *MDPs M_1 and M_2 are equivalent if and only if all the equivalence relations of D consists only pairs (s_i, s_j) for which $s_i \in D_1$ and $s_j \in D_2$, i.e. $size(D) = size(D_1) = size(D_2)$.*

Proof. (a) $\Leftarrow$
Trivial and we have a mapping from M_1 and M_2 to D.
(b)$\Rightarrow$
We assume that M_1 and M_2 are equivalent; thus their minimal models, $D_1 = (s_1^1, ..., s_1^m)$ and $D_2 = (s_2^1, ..., s_2^m)$, satisfy $\forall i \;\; s_1^i \equiv s_2^i$. (we do not assume that we know which state in D_1 corresponds to which state in D_2).

Now we follow the construction of the minimal model as appears in [5]. We prove by induction on the construction steps that for each block B the following

```
Input   : MDPs $M_1, M_2$
Output : Answer whether the MDPs are equivalent
Calculate $D_1$, the minimal Equivalent MDP of $M_1$;
Calculate $D_2$, the minimal Equivalent MDP of $M_2$;
if (size($D_1$) $\neq$ size($D_2$)) then
    Return FALSE;
end
Calculate $D$, the minimal Equivalent MDP of $M = D_1 \cup D_2 = (S_1 \cup S_2, R_1 \cup R_2, P_1 \cup P_2, A)$;
if (size($D$) = size($D_1$)) then
    Return TRUE;
else
    Return FALSE;
end
```

Algorithm 4: An algorithm which finds whether two MDPs are equivalent

holds

$$\forall i \;\; : \;\; s_1^i \in B \iff s_2^i \in B$$

For the induction basis. Since for any i states s_1^i and s_2^i must have the same reward, then they must be in the same block in the initial reward partition. Next we assume that the claim holds for any $j < k$ and prove for k. We assume that in the kth step we check the stability of block B which consists s_1^i and s_2^i with respect to block C. By the induction assumption for every action a we have $P(s_1^i, C, a) = P(s_2^i, C, a)$ and since $SPLIT(B, C, P)$ replaces the block B by the uniquely determined sub-blocks B_i, we have that s_1^i and s_2^i will be in the same sub block after the kth step. Together with Lemma 9 we conclude the lemma. □

By Lemma 10 we have that Algorithm 4 checks whether two MDPs are equivalent and by Lemma 8 we have that it is polynomial. Therefore, we derive the following theorem.

Theorem 5. *The equivalence problem for MDPs, given in their tabular form, is in P.*

8 Acknowledgements

This research was supported in part by a grant from the Israel Science Foundation. Eyal Even-dar was partially supported by the Deutsch Institute.

References

1. E. Allender, S. Arora, M. Kearns, C. Moore, and A. Russell. Note on the representational incompatabilty of function approximation and factored dynamics. In *Advances in Neural Information Processing Systems 15*, 2002.

2. Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.
3. T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.
4. Thomas Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In *UAI*, pages 124–131, 1997.
5. Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 2003. To appear.
6. T.F. Gonzalez. Clustering to minimize the maximum inter-cluster distance. *Theoretical Computer Science*, 38: 293-306, 1985.
7. Robert Givan, Sonia Leach and Thomas Dean. Bounded parameter markov decision processes. *Artificial Intelligence*, 122:71–109, 2000.
8. C. Lusena, J. Goldsmith, and M. Mundhenk. Nonapproximability results for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 14:83–103, 2001.
9. M. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.
10. R. Sutton and A. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.