Efficiently Exploiting Symmetries in Real Time Dynamic Programming

Shravan Matthur Narayanamurthy and **Balaraman Ravindran**
Indian Institute of Technology, Madras
Computer Science Department
shravmn@cse.iitm.ernet.in and ravi@cs.iitm.ernet.in

Abstract

Current approaches to solving Markov Decision Processes (MDPs) are sensitive to the size of the MDP. When applied to real world problems though, MDPs exhibit considerable implicit redundancy especially in the form of symmetries. Existing model minimization methods do not exploit the redundancy due to symmetries well. In this work, given such symmetries, we present a time-efficient algorithm to construct a functionally equivalent reduced model of the MDP. Further we present a Real Time Dynamic Programming (RTDP) algorithm which obviates an explicit construction of the reduced model by integrating the given symmetries into it. The RTDP algorithm solves the reduced model, while working with parameters of the original model and the given symmetries. As RTDP uses its experience to determine which states to backup, it focuses on parts of the reduced state set that are most relevant. This results in significantly faster learning and a reduced overall execution time. The algorithms proposed are particularly effective in the case of structured automorphisms even when the reduced model does not have fewer features. We demonstrate the results empirically on several domains.

1 Introduction

Markov Decision Processes (MDPs) are a popular way to model stochastic sequential decision problems. But most modeling and solution approaches to MDPs scale poorly with the size of the problem. Real world problems often tend to be very large and hence do not yield readily to the current solution techniques. However models of real world problems exhibit redundancy that can be eliminated, reducing the size of the problem.
One way of handling redundancy is to form abstractions, as we humans do, by ignoring details not needed for performing the immediate task at hand. Researchers in artificial intelligence and machine learning have long recognized the importance of abstracting away redundancy for operating in complex and real-world domains [Amarel, 1968].
Identifying symmetrically equivalent situations frequently results in useful abstraction. Informally, a symmetric system is one which is invariant under certain transformations onto itself. An obvious class of symmetries is based on geometric transformations such as, rotations, reflections and translations. Existing work on model minimization of MDPs does not handle symmetries well. The MDP minimization algorithms analyzed by [Givan *et al.*, 2003] do not consider symmetries of MDPs. While it is possible to extend these algorithms to accommodate symmetric equivalence of states, without considering state-action equivalence they cannot model many interesting kinds of symmetry. [Zinkevich and Balch, 2001] consider state-action equivalence, but do not provide any minimization algorithms.

In this article we consider a notion of symmetries, in the form of symmetry groups, as formalized in [Ravindran and Barto, 2002]. Our objective here is to present algorithms that provide a way of using the symmetry information to solve MDPs thereby achieving enormous gains over normal solution approaches. First, we present a time-efficient algorithm (G-reduced Image Algorithm) to construct a reduced model given the symmetry group. The reduced model obtained is functionally equivalent to the original model in that, it preserves the dynamics of the original model and hence a solution of the reduced model will lead to a solution in the original model. However the reduced model can be significantly smaller when compared to the original model depending on the amount symmetry information supplied. Thus solving the reduced model can be a lot easier and faster. Further, we identify that an explicit construction of a reduced model is not essential to use the symmetry information in solving the MDP. We use the G-reduced Image Algorithm as a basis to present the Reduced Real Time Dynamic Programming(RTDP) algorithm which integrates the symmetry information into the RTDP algorithm [Barto *et al.*, 1995] used for solving MDPs. Though the algorithm works directly with the original model it considers only a portion of the original model that does not exhibit redundancy and is rel-

evant in achieving its goals. This focus on the relevance of states results in significantly faster learning leading to huge savings in overall execution time. To make the algorithms more effective, especially in terms of space, we advocate the use of certain structural assumptions about MDPs. We use several domains to demonstrate the improvement obtained by using the reduced RTDP algorithm.

After introducing some notation and background information in Sec. 2, we present the G-reduced Image Algorithm in Sec. 3. We then present the reduced RTDP algorithm in Sec. 4. The experiments done and results achieved are presented in Sec. 5. Finally we conclude the article by giving some directions for future work in Sec. 6.

2 Notation and Background

2.1 Markov Decision Processes

A *Markov Decision Process* is a tuple $\langle S, A, \Psi, P, R\rangle$, where $S = \{1, 2, \ldots, n\}$ is a set of states, A is a finite set of actions, $\Psi \subseteq S \times A$ is the set of admissible state-action pairs, $P : \Psi \times S \to [0, 1]$ is the transition probability function with $P(s, a, s')$ being the probability of transition from state s to state s' under action a, and $R : \Psi \to \mathbb{R}$ is the expected reward function, with $R(s, a)$ being the expected reward for performing action a in state s. Let $A_s = \{a|(s, a) \in \Psi\} \subseteq A$ denote the set actions admissible in state s. We assume that $\forall s \in S, A_s$ is non-empty.

A *stochastic policy* π is a mapping $\Psi \to [0, 1]$, s.t., $\sum_{a \in A_s} \pi(s, a) = 1 \forall s \in S$. The *value* of a state s under policy π is the expected value of the discounted sum of future rewards starting from state s and following policy π thereafter. The *value function* V^π corresponding to a policy π is the mapping from states to their values under π. It can be shown that V^π satisfies the bellman equation:

$$V^\pi(s) = \sum_{a \in A_s} \pi(s, a)\Big[R(s, a) + \gamma \sum_{s' \in S} P(s, a, s')V^\pi(s')\Big] \quad (1)$$

where $0 \leq \gamma < 1$ is a discount factor.

The solution of an MDP is an *optimal policy* π^* that uniformly dominates all other policies for that MDP. In other words $V^{\pi^*}(s) \geq V^\pi(s)$ for all $s \in S$ and for all π.

2.2 Factored Markov Decision Processes

Factored MDPs are a popular way to model structure in MDPs. A factored MDP is defined as a tuple $\langle S, A, \Psi, P, R\rangle$. The state set, given by M features or variable, $S \subseteq \prod_{i=1}^{M} S_i$, where S_i is the set of permissible values for the feature i. A is a finite set of actions, $\Psi \subseteq S \times A$ is the set of admissible state-action pairs. The transition probabilities P are often described by a two-slice *Temporal Bayesian Network* (2-TBN). The state transition probabilities can be factored as:

$$P(s, a, s') = \prod_{i=1}^{M} Prob(s'_i|Pre(s'_i, a)) \quad (2)$$

where $Pre(s'_i, a)$ denotes the parents of node s'_i in the 2-TBN corresponding to action a and each of the probabilities $Prob(s'_i|Pre(s'_i, a))$ is given by a conditional probabilitiy table associated with node s'_i. The reward function may be similarly represented.

2.3 Homomorphisms and Symmetry Groups

This section has been adapted from [Ravindran and Barto, 2002].

Let B be a partition of a set X. For any $x \in X, [x]_B$ denotes the block of B to which x belongs. Any function f from a set X to a set Y induces a partition (or equivalence relation) on X, with $[x]_f = [x']_f$ if and only if $f(x) = f(x')$ and x, x' are f-equivalent written $x \equiv_f x'$. Let B be a partition of $Z \subseteq X \times Y$, where X and Y are arbitrary sets. The projection of B onto X is the partition $B|X$ of X such that for any $x, x' \in X, [x]_{B|X} = [x']_{B|X}$ if and only if every block containing a pair in which x is a component also contains a pair in which x' is a component or every block containing a pair in which x' is a component also contains a pair in which x is a component.

Definition 1. An *MDP homomorphism* h from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to an MDP $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $\langle f, \{g_s|s \in S\}\rangle$, with $h((s, a)) = (f(s), g_s(a))$, where $f : S \to S'$ and $g_s : A_s \to A'_{f(s)}$ for $s \in S$, such that: $\forall s, s' \in S, a \in A_s$

$$P'(f(s), g_s(a), f(s')) = \sum_{s'' \in [s']_f} P(s, a, s'') \quad (3)$$

$$R'(f(s), g_s(a)) = R(s, a) \quad (4)$$

We use the shorthand $h(s, a)$ for $h((s, a))$.

Definition 2. An MDP homomorphism $h = \langle f, \{g_s|s \in S\}\rangle$ from MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to MDP $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$ is an MDP *isomorphism* from $\mathcal{M}$ to $\mathcal{M}'$ if and only if f and g_s, are bijective. $\mathcal{M}$ is said to be *isomorphic* to $\mathcal{M}'$ and vice versa. An MDP isomorphism from MDP $\mathcal{M}$ to itself is call an *automorphism* of $\mathcal{M}$.

Definition 3. The set of all automorphisms of an MDP $\mathcal{M}$, denoted by $\mathrm{Aut}\mathcal{M}$, forms a group under composition of homomorphisms. This group is the *symmetry group* of $\mathcal{M}$.

Let $\mathcal{G}$ be a subgroup of $\mathrm{Aut}\mathcal{M}$. The subgroup $\mathcal{G}$ induces a partition $B_\mathcal{G}$ of Ψ: $[(s_1, a_1)]_{B_\mathcal{G}} = [(s_2, a_2)]_{B_\mathcal{G}}$ if and only if there exists $h \in \mathcal{G}$ such that $h(s_1, a_1) = (s_2, a_2)$ and $(s_1, a_1), (s_2, a_2)$ are said to be $\mathcal{G}$ equivalent written $(s_1, a_1) \equiv_\mathcal{G} (s_2, a_2)$. Further if $s_1 \equiv_{B_\mathcal{G}|S} s_2$ then we write as shorthand $s_1 \equiv_{\mathcal{G}|S} s_2$. It can be proved that there exists a homomorphism $h^\mathcal{G}$ from $\mathcal{M}$ to some $\mathcal{M}'$, such that the partition induced by $h^\mathcal{G}$, $B_{h^\mathcal{G}}$, is the same $B_\mathcal{G}$. The image of $\mathcal{M}$ under $h^\mathcal{G}$ is called the *$\mathcal{G}$-reduced image* of $\mathcal{M}$.

Adding structure to the state space representation allows us to consider morphisms that are structured, e.g., Projection homomorphisms (see sec. 5 of [Ravindran and Barto, 2003]). It can be shown that symmetry groups do not result in projection homomorphisms, except in a

few degenerate cases.
Another simple class of structured morphisms that do lead to useful symmetry groups are those generated by *permutations* of feature values. Let $\sum_M$ be the set of all possible permutations of $\{1, \ldots, M\}$. Given a structured set $X \subseteq \prod_{i=1}^{M} X_i$ and a permutation $\sigma \in \sum_M$, we can define a permutation on X by $\sigma(\langle x_1, \ldots, x_M \rangle) = \langle x_{\sigma(1)}, \ldots, x_{\sigma(M)} \rangle$, and it is a *valid* permutation on X if $x_{\sigma(i)} \in X_i$ for all i and for all $\langle x_1, \ldots, x_M \rangle \in X$.

Definition 4. A *permutation automorphism* h on a structured MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ is a bijection on Ψ , defined by a tuple of bijections $\langle f(s), g_s(a) \rangle$, with $h((s,a)) = (f(s), g_s(a))$, where $f \in \sum_M : S \to S$ is a valid permutation on S and $g_s : A_s \to A'_{f(s)}$ for $s \in S$, such that: $\forall s, s' \in S, a \in A_s$

$$P'(f(s), g_s(a), f(s')) = P(s, a, s')$$
$$= \prod_{i=1}^{M} Prob(s'_{f(i)} | f(Pre_{f(s)}(s'_{f(i)}, a))) \quad (5)$$
$$R'(f(s), g_s(a)) = R(s, a) \quad (6)$$

Here $f(Pre_{f(s)}(s'_{f(i)}, a)) = \{s_{f(j)} | s_j \in Pre(s'_{f(i)}, a)\}$ with $s_{f(j)}$ assigned according to $f(s)$.

3 G-reduced Image Algorithm

3.1 Motivation

In a large family of tasks, the *symmetry groups* are known beforehand or can be specified by the designer through a superficial examination of the problem. A straight forward approach to minimization using symmetry groups would require us to enumerate all the state-action pairs of the MDP. Even when the *symmetry group*, $\mathcal{G}$, is given, constructing the reduced MDP by explicit enumeration takes time proportional to $|\Psi|.|\mathcal{G}|$.

We present in Fig. 1, an efficient incremental algorithm for building the reduced MDP given a symmetry group or subgroup. This is an adaptation of an algorithm proposed by [Emerson and Sistla, 1996] for constructing reduced models for concurrent systems.

3.2 Comments

The algorithm does a breadth-first enumeration of states skipping states and state-action pairs that are equivalent to those already visited. On encountering a state-action pair not equivalent to one already visited, it examines the states reachable from it to compute the image MDP parameters. The algorithm terminates when at least one representative from each equivalence class of $\mathcal{G}$ has been examined. For a proof that the transition probabilities actually represent those for the reduced image, see App. A. The algorithm as presented assumes that all states are reachable from the initial state. It is easy, however, to modify the algorithm suitably. Assuming an explicit representation for the symmetry group and that table look-up takes constant time, the algorithm will run in time proportional to $|\Psi|'.|\mathcal{G}|$. However an explicit representation of $\mathcal{G}$ demands exorbitant memory of the order of $|\mathcal{G}|.|\Psi|$.

```
01 Given M = ⟨S, A, Ψ, P, R⟩ and G ≤ AutM,
02 Construct M/B_G = ⟨S', A', Ψ', P', R'⟩.
03 Set Q to some initial state {s_0}, S' ← {s_0}
04 While Q is non-empty
05     s = dequeue{Q}
06     For all a ∈ A_s
07         If (s, a) ≢_G (s', a') for some (s', a') ∈ Ψ', then
08             Ψ' ← Ψ' ∪ (s, a)
09             A' ← A' ∪ a
10             R'(s, a) = R(s, a)
11             For all t ∈ S such that P(s, a, t) > 0
12                 If t ≡_{G|S} s', for some s' ∈ S',
13                     P'(s, a, s') ← P'(s, a, s') + P(s, a, t)
14                 else
15                     S' ← S' ∪ t
16                     P'(s, a, t) = P(s, a, t)
17                     add t to Q.
```

Figure 1: Incremental algorithm for constructing the G-reduced image given MDP $\mathcal{M}$ and some $\mathcal{G} \leq$'Aut$\mathcal{M}$. Q is the queue of states to be examined. This algorithm terminates when at least one representative from each equivalence class of $\mathcal{G}$ has been examined.

Nevertheless many classes of problems modeled as MDPs exhibit some inherent structure. To accommodate these structures we can consider special forms of homomorphism on factored representations(see Sec. 2.2) such as *projection homomorphism*. Similarly special forms of automorphisms on factored representations like permutation automorphisms (see Definition 4) can also be used advantageously. The advantage here is that the morphisms forming the symmetry group need not be stored explicitly as they are defined on the features instead of states.

For example, let us consider the case of permutation automorphisms. To check for $(s,a) \equiv_{\mathcal{G}} (s', a')$, we need to generate $|\mathcal{G}|$ states that are equivalent to (s', a') by applying each $h \in \mathcal{G}$. Each application of h incurs a time linear in the number of features. Thus in this case the time complexity of the algorithm presented is of the order of $|\Psi|'.|\mathcal{G}|.M$, where M is the number of features whereas no space is needed for storing the $\mathcal{G}$ explicitly.

Thus by restricting the class of automorphisms to functions that are defined on features instead of states, we only incur additional time which is a function of the number of features (significantly less than the number of states) along with a drastic decrease in the space complexity. The use of factored representations leads to further reduction in space needed for storing the transition probabilities and the reward function. Hence the use of these representations and some special forms of automorphisms makes the algorithm presented more effective than its use in the generic case. Also as $\mathcal{G}$ is just a subgroup, the algorithm can work with whatever little symmetry information the designer might have.

4 Reduced RTDP Algorithm

4.1 Motivation

Given a real world problem modeled as an MDP, invariably the state space consists of vast regions which are not relevant in achieving the goals. The minimization approach leads to a certain degree of abstraction which reduces the extent of such regions. Nonetheless the reduced image still contains regions which are not relevant in achieving the goals even in the reduced model. If one has to construct a reduced image then there is no way to avoid these irrelevant regions. But our goal here is to find a policy for acting in the original model. To achieve this, using the algorithm presented in Fig. 1, the following are the steps:

1. Construct the reduced image.
2. Solve the reduced image.
3. *Lift* the policy found in the previous step to the original model.

Since it is this final policy that we want, we can forgo the explicit construction of the reduced model by integrating the information in the symmetry group into the algorithm which solves the original model. Though there are a variety of ways to solve an MDP, we choose to take up RTDP as it uses the experience of the agent to focus on the relevant sections of the state space. This saves the time spent on explicit construction of the reduced model.

Also the G-reduced Image algorithm as presented doesn't preserve any structure in the transition probabilities or the reward function that might have existed because of the use of factored representations. Consequently the reduced image might take considerably more space than the original model. The algorithm we present in Fig. 2 tries to achieve the best of both worlds as it not only works with the original model but also includes the state space reduction by integrating the symmetry group information into the RTDP algorithm.

4.2 Convergence of Reduced RTDP

The algorithm is a modification of the RTDP algorithm with steps from the previous algorithm integrated into lines 7 to 17. If we assume that we have the reduced MDP M', then leaving out lines 7 to 9 and lines 12 to 17 leaves us with the normal RTDP algorithm being run on the reduced image since as explained below, $(s,a) \in \Psi', R'(s,a) = R(s,a)$. Due to the equivalence tests done at lines 7 and 13, the algorithm maintains a policy for and considers only the reduced state space. From App. A, lines 12 to 17 compute the transition probabilities for the reduced image. From Eqn. 6, $R(s,a)$ is the expected reward under the reduced image. So for all $(s,a) \in \Psi', R'(s,a) = R(s,a)$. Thus the update equation in line 21 can be rewritten as,

$$Q(s,a) = R'(s,a) + \sum_{s'' \in S'} \gamma.P'(s,a,s''). \max_{a'' \in A_{s''}} Q(s'',a'') \quad (7)$$

01 Given $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ and $\mathcal{G} \leq \text{Aut}\mathcal{M}$,
02 Hashtable $Q \leftarrow Nil$ is the action value function.
03 Repeat (for each episode)
04 Initialize s and $S' \leftarrow \{s\}$
05 Choose a from s using policy derived from Q (e.g. ϵ-greedy policy)
06 Repeat (for each step in the episode)
07 if $(s,a) \equiv_{\mathcal{G}} (s'',a'')$ for some $(s'',a'') \in Q$ where $(s'',a'') \neq (s,a)$
08 $s \leftarrow s''; a \leftarrow a''$
09 continue.
10 Take action a and observe reward r and next state s'
11 Choose a' from s' using policy derived from Q (e.g.ϵ-greedy policy)
12 For all t such that $P(s,a,t) > 0$
13 If $t \equiv_{\mathcal{G}|S} s''$, for some $s'' \in S'$,
14 $P'(s,a,s'') \leftarrow P'(s,a,s'') + P(s,a,t)$
15 else
16 $S' \leftarrow S' \cup t$
17 $P'(s,a,t) = P(s,a,t)$
18 if $(s,a) \notin Q$
19 add (s,a) to Q.
20 $Q(s,a) \leftarrow 0$
21 $Q(s,a) \leftarrow R(s,a) + \sum_{s'' \in S'} \gamma.P'(s,a,s''). \max_{a'' \in A_{s''}} Q(s'',a'')$
22 $s \leftarrow s'; a \leftarrow a'$

Figure 2: RTDP algorithm with integrated symmetries which computes the Action Value function for the reduced MDP without explicitly constructing it.

which is nothing but the update equation for the reduced image. Thus it is exactly similar to running normal RTDP on the reduced image. As normal RTDP converges to an optimal action value function, the reduced RTDP also converges, as long as it continues to back up all states in the reduced image.

The complete construction of the reduced image can take up considerable amount of time mapping all the irrelevant states into the reduced model whereas with the use of this algorithm one can get near optimal policies even before the construction of a reduced image is complete. It is also faster than the normal RTDP algorithm as its state space is reduced by the use of the symmetry group information.

5 Experiments and Results

Experiments were done on three domains that are explained below. To show the effect of the degree of symmetry considered in the domain we consider a 2-fold symmetry for which $\mathcal{G} < \text{Aut}\mathcal{M}$ and full symmetry $\mathcal{G} = \text{Aut}\mathcal{M}$. We compare the reduced RTDP algorithm using 2 degrees of symmetry with the normal RTDP algorithm. We present learning curves representing the decrease in the number of steps taken to finish each

episode. To show the time efficiency of the reduced RTDP algorithm we present a bar chart of times taken by the reduced RTDP algorithm using 2 degrees of symmetry and the normal RTDP algorithm for completing 200 episodes of each domain. All the algorithms used a discount factor, $\gamma = 0.9$. An epsilon greedy policy with $\epsilon = 0.1$ was used to choose the actions at each step. Due to lack of space we present one graph per domain though experiments were done with different sizes of each domain. The results are similar in other cases. We note exceptions if any as is relevant.

5.1 Deterministic Grid-World

Two Grid-Worlds of sizes 10x10 and 25x25 with four deterministic actions of going UP, DOWN, RIGHT and LEFT were implemented. The initial state was (0,0) and the goal states were {(0,9),(9,0)} and {(0,24),(24,0)} respectively. For the 2-fold symmetry, states about NE-SW diagonal, i.e., (x,y) and (y,x) were equivalent. If the grid is of size $M \times N$ then let maxX = $M - 1$ and masY = $N - 1$. For the full symmetry case, states (x,y), (y,x), (maxX-x,maxY-y) and (maxY-y,maxX-x) were equivalent. State action equivalence was defined accordingly.

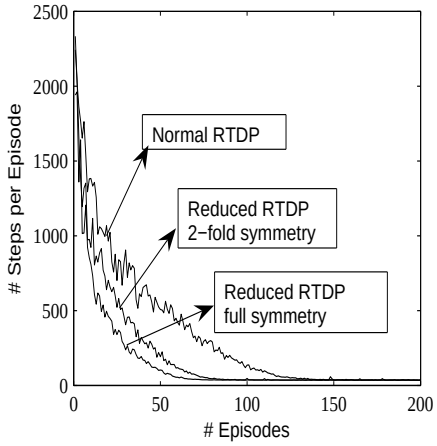

Figure 3: Learning curves for the Deterministic Grid-World domain with a 25x25 grid.

5.2 Probabilistic Grid-World

Two Grid-Worlds of sizes 10x10 and 25x25 with four actions of going UP, DOWN, RIGHT and LEFT were implemented. Unlike the deterministic domain, here actions led to the relevant grid only with a probability of 0.9 and left the state unchanged with a probability of 0.1. The initial state was (0,0) and the goal states were {(0,9),(9,0)} and {(0,24),(24,0)} respectively. For the 2-fold symmetry, states about NE-SW diagonal, i.e., (x,y) and (y,x) were equivalent. For the full symmetry case, states (x,y), (y,x), (maxX-x,maxY-y) and (maxY-y,maxX-x) were equivalent. State action equivalence was defined accordingly.

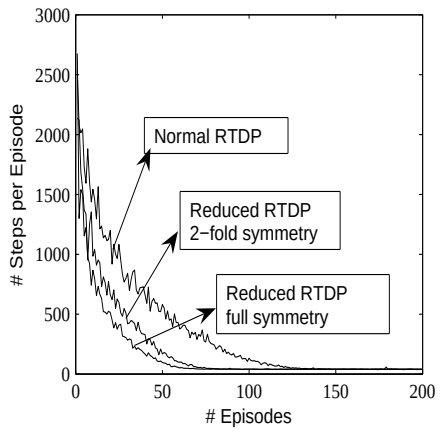

Figure 4: Learning curves for the Probabilistic Grid-World domain with a 25x25 grid.

5.3 Probabilistic Towers of Hanoi

The towers of Hanoi domain as implemented had 3 pegs. Two domains, one with 3 and the other with 5 disks were implemented. Actions that allowed the transfer of a smaller disk onto a larger disk or to an empty peg were permitted. The actions did the transfer of disk with a probability of 0.9 and left the state unchanged with a probability of 0.1. The initial state in the case of 3 disks was {(1,3), (2), ()} and {(4), (1,2), (3,5)} in the 5 disk case. The goal states were designed to allow various degrees of symmetry. For a 2-fold symmetry, the goal states considered were states where, all disks were either on peg 1 or 2. Equivalent states were those that have disk positions of pegs 1 and 2 interchanged. For the full symmetry case, the goal states considered were states where, all disks were on any one peg. Equivalent states were those that have disk positions interchanged by any possible permutation of the pegs. State action equivalence was defined accordingly.

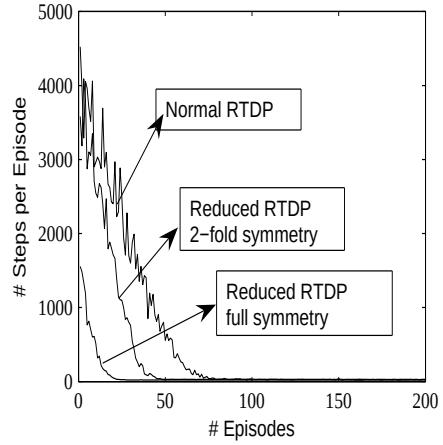

Figure 5: Learning curves for the Probabilistic Towers of Hanoi domain with 5 disks.

5.4 Time efficiency

The bar-graph in Fig. 6 shows the running times (scaled to even the graph) of the normal RTDP, reduced RTDP with a 2-fold symmetry and reduced RTDP with full symmetry. The first cluster is on the Deterministic Grid-World domain with a 25x25 grid, the second cluster is on Probabilistic Grid-World with a 25x25 grid and the third on Probabilistic Towers of Hanoi with 5 disks.[1]

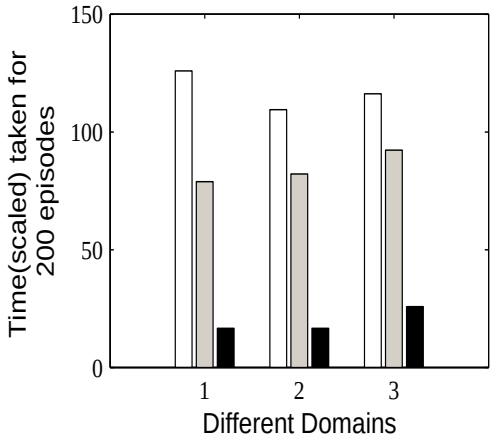

Figure 6: Comparison of running times(scaled).

5.5 Discussion

The results are as expected. The comparisons show that the reduced RTDP algorithm learns faster than the normal RTDP both in the full symmetry case as well as in the 2-fold symmetry case. Further among the reduced RTDP algorithms, the one using more symmetry is better than the one with lesser symmetry. The same is reflected in the running times of algorithms. The full symmetry case is at least 5 times faster than the normal RTDP. The 2-fold symmetry is also faster than the normal RTDP.

One observation contrary to graph shown in the bar graph of Fig. 6 is that when reduced RTDP algorithms are used for very small domains like 3-disk Towers of Hanoi, the overhead involved in checking equivalence of states outweighs the benefit from the reduction due to symmetry. Though we have not been able to quantify the exact extent of the trade-offs involved, we feel that when the expected length of a trajectory to the goal state from the initial state is small in comparison with the state space, the benefits obtained by using the symmetry group information are masked by the overhead involved in doing the equivalence comparisons. However this is true only in case of very small domains. In any domain of reasonable size the agent implementing normal RTDP has to explore vast spaces before arriving at the correct trajectory. But for an agent implementing reduced RTDP the symmetry reduces this space that has to be explored. Also greater the symmetry used lesser the space that has to be explored. This explains the better performance of the reduced RTDP algorithm.

[1]Running times for domains of lesser size does not follow the pattern indicated by the graphs. See Sec. 5.5

6 Conclusions and Future Work

The algorithms presented in this article provide an efficient way of exploiting varying amounts of symmetry present in a domain resulting in faster learning and reduced execution times. With the use of structured morphisms on factored representations the algorithms are even more effective, especially in terms of space.

The notion of equivalence used here is very strict. One direction for future work, that we perceive, is to include notions of approximate equivalence. Another possibility will be to quantify the exact trade-offs involved due to overheads of checking equivalence and the performance gained by the use of symmetries.

A Transition probabilities computed for the reduced model

Let $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ be an MDP and $\mathcal{G}$ the given symmetry group. Let $B_{\mathcal{G}}$ be the partition induced by $\mathcal{G}$. Let $\mathcal{M}/B_{\mathcal{G}} = \langle S', A', \Psi', P', R'\rangle$ be the reduced image. Let $\rho(s,a)$ denote the set of states reachable from state s by doing action a. Let $\overline{B_{\mathcal{G}}|S} = \{[s]_{B_{\mathcal{G}}|S} \mid [s]_{B_{\mathcal{G}}|S} \cap \rho(s,a) = \emptyset\}$. When (s,a) is used with P', they represent blocks whereas when used with P, $(s,a) \in S$ and are representatives for $[s,a]_{B_{\mathcal{G}}}$ From Def. 1 the transition probabilities, P', satisfy,

$$\forall (s,a) \in \Psi',\ \ \forall [s']_{B_{\mathcal{G}}|S} \in B_{\mathcal{G}}|S \qquad P'(s,a,[s']_{B_{\mathcal{G}}|S}) = \sum_{s'' \in [s']_{B_{\mathcal{G}}|S}} P(s,a,s'') \quad (8)$$

By the definition of $\overline{B_{\mathcal{G}}|S}$,

$$\forall (s,a) \in \Psi',\ \ \forall [s']_{B_{\mathcal{G}}|S} \in \overline{B_{\mathcal{G}}|S} \qquad P'(s,a,[s']_{B_{\mathcal{G}}|S}) = 0 \quad (9)$$

As $\sum_{s'' \in ([s']_{B_{\mathcal{G}}|S} - \rho(s,a))} P(s,a,s'') = 0$

$$\forall (s,a) \in \Psi',\ \ \forall [s']_{B_{\mathcal{G}}|S} \in B_{\mathcal{G}}|S - \overline{B_{\mathcal{G}}|S}, \qquad P'(s,a,[s']_{B_{\mathcal{G}}|S}) = \sum_{s'' \in ([s']_{B_{\mathcal{G}}|S} \cap \rho(s,a))} P(s,a,s'') \quad (10)$$

As $B_{\mathcal{G}}|S$ is a partition of S, $\forall t \in \rho(s,a)$, there exists exactly one $[s']_{B_{\mathcal{G}}|S} \in \left(B_{\mathcal{G}}|S - \overline{B_{\mathcal{G}}|S}\right)$ such that $t \in [s']_{B_{\mathcal{G}}|S}$.
Hence Eqn. 10 can be rewritten as

$$\forall (s,a) \in \Psi',\ \ \forall t \in \rho(s,a) \qquad P'(s,a,[t]_{B_{\mathcal{G}}|S}) = \sum_{s'' \in (\rho(s,a) \cap [s']_{B_{\mathcal{G}}|S})} P(s,a,s'') \quad (11)$$

It is evident that lines 12 to 17 of Fig. 1 implement Eqn. 9 and Eqn. 11.

References

[Amarel, 1968] Saul Amarel. On representations of problems of reasoning about actions. In Donald Michie, editor, *Machine Intelligence 3*, volume 3, pages 131–171. Elsevier/North-Holland, Amsterdam, London, New York, 1968. Amarel, S.

[Barto *et al.*, 1995] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[Emerson and Sistla, 1996] F. Allen Emerson and A. Prasad Sistla. Symmetry and model checking. *Formal Methods in System Design: An International Journal*, 9(1/2):105–131, August 1996.

[Givan *et al.*, 2003] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[Ravindran and Barto, 2002] Balaraman Ravindran and Andrew G. Barto. Model minimization in hierarchical reinforcement learning. *Lecture Notes on Computer Science*, 2371:196–211, 2002.

[Ravindran and Barto, 2003] Balaraman Ravindran and Andrew G. Barto. Smdp homomorphisms: An algebraic approach to abstraction in semi markov decision processes. In *IJCAI 03*, pages 1011–1016. AAAI, August 2003.

[Zinkevich and Balch, 2001] M. Zinkevich and T. Balch. Symmetry in markov decision processes and its implications for single agent and multiagent learning. In *ICML 2001*, pages 632–640. Morgan Kaufmann, 2001.