On the Hardness of Finding Symmetries in Markov Decision Processes

Shravan Matthur Narayanamurthy SHRAVANMN@GMAIL.COM
Yahoo! Labs, Bangalore

Balaraman Ravindran RAVI@CSE.IITM.AC.IN
Dept. of Comp. Sci. and Engg., Indian Institute of Technology Madras, Chennai

Abstract

In this work we address the question of finding symmetries of a given MDP. We show that the problem is *Isomorphism Complete*, that is, the problem is polynomially equivalent to verifying whether two graphs are isomorphic. Apart from the theoretical importance of this result it has an important practical application. The reduction presented can be used together with any off-the-shelf Graph Isomorphism solver, which performs well in the average case, to find symmetries of an MDP. In fact, we present results of using NAutY (the best Graph Isomorphism solver currently available), to find symmetries of MDPs.

1. Introduction

Markov Decision Processes (MDPs) are widely employed to model sequential decision problems. But current solution techniques for MDPs do not scale well with the size of the MDPs, and hence are proving inadequate in solving large real-world problems. While building abstract models of real-world problems, it can be seen that a high degree of redundancy is present which can be exploited to reduce size of the model. This reduction in size could possibly lead to more efficient solution methods.

One such notion of redundancy is a degree of symmetry that is present in any real-world problem. (Amarel, 1968) first looked at exploiting such symmetries in solving a missionaries and cannibals problem. In this work we use the notion of symmetries in MDPs introduced in (Ravindran, 2004). While it is widely believed that finding symmetries in MDPs is a hard problem, no one has investigated before exactly how hard this problem is.

Intuitively this seems harder than finding symmetries in graphs, due to the additional structure introduced by MDPs. In this work we show that finding symmetries in MDPs is no harder than the problem of graph isomorphism. We also show that existing graph isomorphism solvers can be used to find symmetries in MDPs.

We present some notation in the next section, and some related work in Section 3. In Section 4 we formally define the problem, and present a constructive algorithm in Section 5 for showing the equivalence to graph isomorphism. We discuss some results Section 6 and conclude in Section 7.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Homomorphisms and Symmetry Groups

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 $< f, \{g_s | s \in S\} >$, with $h((s,a)) = (f(s), g_s(a))$, where $f : S \rightarrow S'$ and $g_s : A_s \rightarrow A'_{f(s)}$ for $s \in S$, such that: $\forall s, s' \in S, a \in A_s$

$$P'(f(s), g_s(a), f(s')) = \sum_{s'' \in [s']_f} P(s, a, s'') \quad (1)$$

$$R'(f(s), g_s(a)) = R(s, a) \quad (2)$$

We use the shorthand $h(s,a)$ for $h((s,a))$. Often for convenience, we use $< f, \{g_s\} >$ to denote $< f, \{g_s | s \in S\} >$.

Definition 2. Let $\mathcal{M}$' be an image of the MDP $\mathcal{M}$ under homomorphism $h =< f, \{g_s\} >$. For any $s \in S, g_s^{-1}(a')$ denotes the set of actions that have the same image $a' \in A'_{f(s)}$ under g_s. Let π' be a stochastic policy in $\mathcal{M}'$. Then π' lifted to $\mathcal{M}$ is the policy $\pi_{\mathcal{M}'}$ such that for any $a \in g_s^{-1}(a'), \pi'_{\mathcal{M}}(s,a) = \pi'(f(s),a')/|g_s^{-1}(a')|$

Definition 3. An MDP homomorphism $h =< f, \{g_s\} >$ 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 called an *automorphism* of $\mathcal{M}$.

Definition 4. The set of all automorphisms of an MDP $\mathcal{M}$, denoted by $Aut\mathcal{M}$, forms a group under composition of homomorphisms. This group is the *symmetry group* of $\mathcal{M}$.

Let $\mathcal{G}$ be a subgroup of $Aut\mathcal{M}$. The image of $\mathcal{M}$ under $\mathcal{G}$ is called the *$\mathcal{G}$-reduced image* of $\mathcal{M}$.

Definition 5. An MDP $\mathcal{M}'$ is said to be a *reduced model* of an MDP $\mathcal{M}$, iff there exists an MDP homomorphism $h : \mathcal{M} \to \mathcal{M}'$.

3. Related Work

MDP Minimization is a well studied problem. As stated earlier, in the model minimization approach, a reduced MDP that that preserves some key properties as the original MDP is found by combining "equivalent" states. The reduced MDP found depends on the notion of equivalence between states used in the aggregation. The notion of equivalence chosen will be fundamental in designing and analyzing algorithms for reducing MDPs. In (Dean & Givan, 1997) a minimization algorithm is proposed based on the notion of *stochastic bi-simulation homogeneity*. Informally, a partition of the state space for an MDP is said to be homogeneous if for each action, states in the same block have the same probability of transitioning to each other block. They also provide an algorithm for finding the coarsest homogeneous refinement of any partition of the state space of an MDP. The algorithm starts with an initial partition P_0 and iteratively refines it by splitting the blocks until the coarsest homogeneous refinement of P_0 is obtained. A notion of stability of a block with respect to another is defined and unstable blocks are split till all blocks of the partition are stable. The complexity of the algorithm is expressed in terms of the partition manipulation operations. Hence, the actual complexity depends on the underlying partition representation and manipulation algorithms. (Givan et al., 2003) discuss the application of the algorithm to solving factored MDP problems. Enumerating the state space is avoided by describing large blocks of equivalent states in factored form with the block descriptions being inferred directly from the original factored representation.

(Ravindran, 2004) proposes a more generic framework based on the notion of MDP homomorphisms with *state-dependent action recoding* as introduced in Section 2. This allows a greater reduction in problem size and aids in modeling many other notions of equivalence like symmetries. A polynomial time algorithm to find the reduced model under the notion of MDP homomorphisms is also proposed by extending the algorithm proposed by (Givan et al., 2003) and (Lee & Yannakakis, 1992). Again, the algorithm is polynomial in the number of block operations, the stability criterion is modified to suit the equivalence notion and the same process of iterative splitting is used. The notion of stability used is called the *stochastic substitution property*, which is an extension of the *substitution property* for finite state machines (Hartmanis, 1966).

However, literature on MDP minimization using symmetries is sparse. (Zinkevich & Balch, 2001) define symmetries based on state-action equivalence but do not make any connections to group-theoretic concepts or minimization algorithms.

Another dimension to analyze the literature is the approach to symmetry finding. Two main approaches exist:

1. To derive a set of necessary conditions for elements to be symmetric
2. Prove Isomorphism Completeness and use a graph isomorphism finding system

Intuitively symmetries seem easier to identify than homomorphisms and we tried the first approach to find a polynomial time algorithm for symmetry finding, along the lines of the MDP homomorphism finding, with the motivation of finding better algorithms for MDP minimization. The MDP homomorphism definition allows for deriving this easily because, two state action pairs $(s_1, a_1), (s_2, a_2)$ are homomorphically equivalent if

$$\begin{aligned} h(s_1,a_1) &= h(s_2,a_2) \\ P'(f(s_1), g_{s_1}(a_1), f(s')) &= P'(f(s_2), g_{s_2}(a_2), f(s')) \\ T(s_1,a_1,[s']_{B_h|S}) &= T(s_2,a_2,[s']_{B_h|S}) \end{aligned}$$

for all $s' \in S$. This is the stochastic substitution property and it allows us to deal just with blocks without worrying about the actual functions. However, a similar attempt for symmetries still needs the symmetry f in the necessary condition as below:

$$\begin{aligned} h(s_1,a_1) &= (s_2,a_2) \\ P(f(s_1), g_{s_1}(a_1), f(s')) &= P(s_2,a_2,f(s')) \\ P(s_1,a_1,s') &= P(s_2,a_2,f(s')) \end{aligned}$$

(Flener et al., 2002) and (Crawford, 1992) point that symmetry finding for CSPs in general is Isomorphism Com-

plete. However, there also exist results showing that symmetry finding is NP-complete (in case of geometric automorphism of graphs (Manning, 1990)). So we were still unclear whether symmetry finding for MDPs is Isomorphism Complete or NP-complete due to the presence of factorially many action recoding functions. A better understanding of the use of symmetries for abstraction in MDPs is the motivation for this work.

4. Problem Definition

To exploit the power of abstraction using symmetries, we identify them and construct a reduced model by abstracting away the symmetric portions. As the reduced model can be significantly smaller, it can be easier to solve. We use the notion of automorphisms to model symmetries. So formally, given an MDP $\mathcal{M}$,

1. Find the automorphism group, $Aut\mathcal{M}$ and
2. Given the automorphism group, $Aut\mathcal{M}$ find the corresponding reduced model, the $Aut\mathcal{M}$-Reduced Image

5. Finding Symmetries

5.1. Problem Simplification

Let us consider the first part of our problem, i.e., given an MDP $\mathcal{M}$, find the automorphism group of $\mathcal{M}$, $Aut\mathcal{M}$. We know that a group can be specified using its generators. So we simplify the problem to finding the generators of $Aut\mathcal{M}$. Let $AMGEN(\mathcal{M})$ denote the problem of finding the generators of $Aut\mathcal{M}$. We write $A \propto B$ if a problem A is polynomially reducible to B. We say that problems A and B are polynomially equivalent iff $A \propto B$ and $B \propto A$. We denote polynomial equivalence by $\equiv_\propto$.

Definition 6. A problem A is *Isomorphism Complete* iff A is polynomially equivalent to finding whether two graphs are isomorphic.

Let G_1, G_2 be two simple graphs unless otherwise mentioned. The following is a list of relevant *Isomorphism Complete* problems (Booth & Colbourn, 1977) on graphs:

- $ISO(G_1, G_2)$: Isomorphism recognition for G_1 and G_2
- $IMAP(G_1, G_2)$: Isomorphism Map from G_1 to G_2(if it exists),
- $AGEN(G_1)$: Generators of the automorphism group, $AutG_1$
- $DGEN(G)$: Generators of the automorphism group, $AutG$, where G is a weighted digraph

From (Mathon, 1979), (Read & Corneil, 1977), (Miller, 1977) we have,
$DGEN(G) \equiv_\propto AGEN(G) \equiv_\propto IMAP(G_1, G_2) \equiv_\propto ISO(G_1, G_2)$.

We intend to prove that $AMGEN(\mathcal{M})$ is *Isomorphism Complete*. We are done if we prove that $AMGEN(\mathcal{M}) \equiv_\propto DGEN(G_\mathcal{M})$, where $G_\mathcal{M}$ is a weighted graph constructed in polynomial time from $\mathcal{M}$, that is, $AMGEN(\mathcal{M}) \propto DGEN(G_\mathcal{M})$ and $DGEN(G_\mathcal{M}) \propto AMGEN(\mathcal{M})$. It is easy to see that $DGEN(G_\mathcal{M}) \propto AMGEN(\mathcal{M})$ is true because we can always construct a degenerate MDP from a digraph. So we need to prove that $AMGEN(\mathcal{M}) \propto DGEN(G_\mathcal{M})$.

5.2. *Isomorphism Completeness* of the problem

An MDP $\mathcal{M}$ can be considered as a pseudograph with states acting as vertices and actions acting as edges. Since there can be more than one action affecting the transition between 2 states, we need to represent this as a pseudograph. The transition probabilities and rewards can be thought of as weight functions. Next, we formally pose $AMGEN(\mathcal{M})$ as a problem on a weighted pseudograph.

Let $G_\mathcal{M} =< \Sigma_a, V, E, W_P, W_R >$ be the pseudograph corresponding to $\mathcal{M}$, where

$$
\begin{aligned}
\Sigma_a &: \text{Alphabet for labelling corresponding to actions} \\
V &: \text{Set of vertices corresponding to states} \\
E &: \text{Set of edges, where each edge is a triple } (u,a,v) \text{ where, } u,v \in V \text{ and } a \in \Sigma_a \text{ corresponding to state transitions} \\
W_P &: E \to \mathbb{R} \text{ corresponding to transition probabilities} \\
W_R &: E \to \mathbb{R} \text{ corresponding to rewards with } W_R(u,a,v) = W_R(u,a,v') \\
&\quad \forall\, (u,a,v),(u,a,v') \in E \\
\text{Note, } E &= \bigcup_{u,v \in V} E_{uv} \text{ where, } E_{uv} = \{\, (u',a,v') \in E \mid u' = u \text{ and } v' = v \,\}
\end{aligned}
$$

$AMGEN(\mathcal{M})$ can be formulated as finding the generators of the group of bijections $h : V \times \Sigma_a \to V \times \Sigma_a$. h is defined by $h(u,a) = (f(u), g_u(a))$, where

$$
\begin{aligned}
f &: V \to V \text{ and} \\
g_u &: \Sigma_a \to \Sigma_a \text{ defined for each } u \in V \text{ are bijections s. t.} \\
W_P(f(u), g_u(a), f(v)) &= W_P(u,a,v) \text{ and} \\
W_R(f(u), g_u(a), f(v)) &= W_R(u,a,v) \;\forall\; (u,a,v) \in E
\end{aligned}
$$

These two components of each generator can be interpreted as follows:

1. f is a function that permutes the states/vertices
2. The set of functions $\{g_u\}$ defined for each state/vertex permutes the actions/edge labels. These are called the State-Dependent Action Recoding (SDAR) functions.

5.2.1. Set Bijections

Let us assume, for a moment, that we have a procedure that constructs a weighted digraph $WD_{\mathcal{M}}$ from $G_{\mathcal{M}}$. Now, solving $DGEN(WD_{\mathcal{M}})$ gives the generators of $WD_{\mathcal{M}}$. Even if these were somehow same as the fs we are looking for, we still need a way to find the SDAR functions. To achieve this, we define the notion of a *set bijection* which represents a set of bijections very compactly. In the worst case, for each f, there can be factorially many SDAR functions. So a normal explicit representation cannot be used. We also define the operations of intersection between two *set bijections* to find the bijections that are common to both *set bijections*, composition between two *set bijections* and an inverse of a *set bijection*. All these operations can be done in time polynomial of the number of elements in the domain of a bijection belonging to the *set bijection*.

Definition 7. Consider two finite sets A and B. Let $U_A = \{U_{A_1}, U_{A_2}, \ldots, U_{A_k}\}$ and $U_B = \{U_{B_1}, U_{B_2}, \ldots, U_{B_k}\}$ be partitions of A and B respectively. U_A and U_B are said to be *similar* iff $|U_A| = |U_B|$ and for each $U_{A_i} \in U_A$ there exists a unique $U_{B_j} \in U_B$ such that $|U_{A_i}| = |U_{B_j}|$. We denote it by $U_A \sim U_B$.

Note that, by definition the sets A and B will be of the same size.

Definition 8. Let A and B be two finite sets and $U_A = \{U_{A_1}, U_{A_2}, \ldots, U_{A_k}\}$ and $U_B = \{U_{B_1}, U_{B_2}, \ldots, U_{B_k}\}$ be partitions of A and B respectively such that $U_A \sim U_B$. A bijective map $X : U_A \to U_B$ where $X(U_{A_i}) = U_{B_j}$ implies $|U_{A_i}| = |U_{B_j}|$ for all $U_{A_i} \in U_A$ is called a *set bijection*.

Informally, a *set bijection* can be interpreted as representing a set of bijections from A to B. $X(U_{A_i}) = U_{B_j}$ represents all possible bijective mappings from elements in U_{A_i} to elements in U_{B_j}. A bijection from A to B in the set of bijections that represent the *set bijection*, can be formed by collating mappings from each $X(U_{A_i}) = U_{B_j}$. The *set bijection* represents all mappings that can be formed by such collations. To formalize this notion, we define the *interpretation function* next.

Let $X_{AB} \triangleq \{$ all bijections $X : U_A \to U_B$ such that U_A and U_B are similar partitions of A and B respectively $\}$ be the set of all *set bijections*. Let $2^{S_{|V|}}$ be the powerset set of all permutations from $A \to B$. Define, $\hat{I} : X_{AB} \to 2^{S_{|V|}}$ such that $\hat{I}(X : U_A \to U_B) = \{$ all bijections $l : A \to B \mid l(x \in U_{A_i}) \in X(U_{A_i})\ \forall U_{A_i} \in U_A\}$. Evidently, $\hat{I}$ is only injective and not surjective as there exist sets of $2^{S_{|V|}}$ that cannot be represented by a *set bijection*. For example, consider the set of bijections, between $\{a, b, c\}$ and $\{1, 2, 3\}$, $L = \{(a \to 1, b \to 2, c \to 3), (a \to 2, b \to 1, c \to 3), (a \to 2, b \to 3, c \to 1)\}$. Clearly there does not exist an $X : U_A \to U_B$ such that $\hat{I}(X) = L$. All we can say is that there exists an X such that $L \subset \hat{I}(X)$. To get a bijective interpretation function, we define, $I : X_{AB} \to image(\hat{I})$ such that $I(X : U_A \to U_B) = \hat{I}(X : U_A \to U_B)$. Clearly I is a bijection and we call this the interpretation function.

Definition 9. Let A be a finite set and let $U_A^1 = \{U_{A_1}^1, U_{A_2}^1, \ldots, U_{A_k}^1\}$, $U_A^2 = \{U_{A_1}^2, U_{A_2}^2, \ldots, U_{A_k}^2\}$ be two partitions of A such that, $U_A^1 \sim U_A^2$. We define the intersection of two similar partitions of a finite set as $U_A^1 \cap U_A^2 = \{U_{A_i}^1 \cap U_{A_j}^2 \mid U_{A_i}^1 \in U_A^1, U_{A_j}^2 \in U_A^2$ *and* $U_{A_i}^1 \cap U_{A_j}^2 \neq \emptyset\}$.

Definition 10. Let A and B be two finite sets and $U_A^1 = \{U_{A_1}^1, U_{A_2}^1, \ldots, U_{A_k}^1\}$, $U_A^2 = \{U_{A_1}^2, U_{A_2}^2, \ldots, U_{A_k}^2\}$ be two partitions of A and $U_B^1 = \{U_{B_1}^1, U_{B_2}^1, \ldots, U_{B_k}^1\}$, $U_B^2 = \{U_{B_1}^2, U_{B_2}^2, \ldots, U_{B_k}^2\}$ be two partitions of B. Also let $U_A^1 \sim U_B^1$ and $U_A^2 \sim U_B^2$. Let two *set bijections* X_1 and X_2 be defined from U_A^1 to U_B^1 and from U_A^2 to U_B^2 respectively. If $(U_A^1 \cap U_A^2) \sim (U_B^1 \cap U_B^2)$, we define the intersection between the two set bijections $X = X_1 \cap X_2$ as follows: $\forall U_{A_i}^1 \in U_A^1, U_{A_j}^2 \in U_A^2$ such that $U_{A_i}^1 \cap U_{A_j}^2 \neq \emptyset$, $X(U_{A_i}^1 \cap U_{A_j}^2) = X_1(U_{A_i}^1) \cap X_2(U_{A_j}^2)$. Note that, $X : U_A^1 \cap U_A^2 \to U_B^1 \cap U_B^2$ and it can be shown that $I(X) = I(X_1) \cap I(X_2)$.

Definition 11. Let A be a finite set. Let $U_A^1 = \{U_{A_1}^1, U_{A_2}^1, \ldots, U_{A_k}^1\}$, $U_A^2 = \{U_{A_1}^2, U_{A_2}^2, \ldots, U_{A_k}^2\}$ be two *similar* partitions of A. Let X be a *set bijection* defined from U_A^1 to U_A^2. We define the *inverse* of X as $X^{-1} : U_A^2 \to U_A^1$ such that $X^{-1}(U_{A_i}^2) = U_{A_j}^1$ iff $X(U_{A_j}^1) = U_{A_i}^2$.

Definition 12. Let A, B and C be three finite sets and $U_A = \{U_{A_1}, U_{A_2}, \ldots, U_{A_k}\}$, $U_B = \{U_{B_1}, U_{B_2}, \ldots, U_{B_k}\}$ and $U_C = \{U_{C_1}, U_{C_2}, \ldots, U_{C_k}\}$ be partitions of A, B and C respectively . Also let U_A, U_B and U_C be pairwise similar to each other. Let two *set bijections* X_1 and X_2 be defined from U_B to U_C and U_A to U_B respectively. We define the composition of X_1 and X_2, $X = X_1 \odot X_2$ as the *set bijection* from U_A to U_C defined by $X(U_{A_i}) = X_1(X_2(U_{A_i}))$, for each $U_{A_i} \in U_A$. It can be shown that for each $l \in I(X)$ there exist, $l_1 \in I(X_1)$ and $l_2 \in I(X_2)$ such that $l = l_1 \circ l_2$ where $\circ$ denotes normal function composition.

5.2.2. Vector-Weighted Digraph

We assume that Σ_a can be ordered and let O be such an ordering.

Without loss of generality, we can assume that $|E_{uv}| =$

$k, \forall u, v \in V$ because, we can always take $max_{u,v \in V}|E_{uv}| = k$ and if $\exists u, v \in V$ such that $(u, a, v) \in E$ for some $a \in \Sigma_a$ and $|E_{uv}| < k$, then add dummy labels (chosen from the remaining labels in Σ_a) and zero weights to make $|E_{uv}| = k$. This corresponds to the general assumption in MDPs that $|A_s| = k, \forall s \in S$.

Let $< a_1, a_2, \ldots, a_k >$ ordered as per O be the k-tuple representing the label of each edge in E_{uv}. This being the same for all edges, we leave out labeling from the graph definition.

Now we define the vector-weighted digraph corresponding to $\mathcal{M}$, $VWG_{\mathcal{M}} =< V, E', \mathbf{W_P}, \mathbf{W_R} >$, as follows:

$$
\begin{aligned}
E' &= \{(u,v) \mid \exists a \in \Sigma_a \text{ and } (u,a,v) \in E\} \\
\mathbf{W_P} &: E' \to \mathbb{R}^k \text{ defined by} \\
&\quad \mathbf{W_P}(\mathbf{u},\mathbf{v}) =< W_P(u,a_1,v), \ldots, W_P(u,a_k,v) > \\
\mathbf{W_R} &: E' \to \mathbb{R}^k \text{ defined by} \\
&\quad \mathbf{W_R}(\mathbf{u},\mathbf{v}) =< W_R(u,a_1,v), \ldots, W_R(u,a_k,v) >
\end{aligned}
$$

where $a_1, a_2, \ldots, a_k$ are ordered as per O.

5.2.3. Sorted Vector-Weighted Digraph

We define the sorted vector-weighted digraph, $SVWG_{\mathcal{M}} =< V, E', \mathbf{W_{P_s}}, \mathbf{W_{R_s}} >$, as follows:

$$
\begin{gathered}
\mathbf{W_{P_s}} : E' \to \mathbb{R}^k \text{ defined by} \\
\mathbf{W_{P_s}}(\mathbf{u},\mathbf{v}) =< W_P(u, p_{uv}(1), v), \ldots, W_P(u, p_{uv}(k), v) > \\
\text{where, } p_{uv} : \mathbb{N}_k \to \Sigma_a \text{ such that} \\
W_P(u, p_{uv}(1), v) \leq \ldots \leq W_P(u, p_{uv}(k), v) \\
\mathbf{W_{R_s}} : E' \to \mathbb{R}^k \text{ defined by} \\
\mathbf{W_{R_s}}(\mathbf{u},\mathbf{v}) =< W_R(u, r_{uv}(1), v), \ldots, W_R(u, r_{uv}(k), v) > \\
\text{where, } r_{uv} : \mathbb{N}_k \to \Sigma_a \text{ such that} \\
W_R(u, r_{uv}(1), v) \leq \ldots \leq W_R(u, r_{uv}(k), v)
\end{gathered}
$$

Note that, p_{uv} and r_{uv} are not unique. So we choose them such that the order O is preserved.

5.2.4. *Set Bijections* that sort the vector-weights

Here we show that there exists a *set bijection* whose interpretation is the set of permutations that sort the vector-weights. Let $\mathbb{N}_k$ be the set of first k natural numbers. Let $D^P_{uv} \triangleq \{$ all permutations $l : \mathbb{N}_k \to \Sigma_a \mid l$ sorts $\mathbf{W_P}(\mathbf{u},\mathbf{v})\}$ be defined for each $(u,v) \in E'$. So, $\mathbf{W_{P_s}}(\mathbf{u},\mathbf{v}) =< W_P(u,l(1),v), \ldots, W_P(u,l(k),v) >$ and $W_P(u,l(1),v) \leq W_P(u,l(2),v) \leq \ldots \leq W_P(u,l(k),v)$. Clearly, $\mathbb{N}_k$ can be partitioned into $U^{uv}_{\mathbb{N}_k} = \{\mathbb{N}^1_k, \mathbb{N}^2_k, \ldots, \mathbb{N}^j_k\}$ such that, $\forall t \in \mathbb{N}^y_k$, $W_P(u,l(t),v)$ has the same value for each $y = 1, 2, \ldots, j$ and if $t \in \mathbb{N}^y_k$ and $t' \in \mathbb{N}^{y+1}_k$ then $W_P(u,l(t),v) < W_P(u,l(t'),v)$. This partition induces a corresponding partition $U^{uv}_{\Sigma_a} = \{\Sigma^1_a, \Sigma^2_a, \ldots, \Sigma^j_a\}$ where $\Sigma^i_a = \{l(t) \mid t \in \mathbb{N}^i_k\}$. Since, each l sorts $\mathbf{W_P}(\mathbf{u},\mathbf{v})$, they satisfy the property that $l(x \in \mathbb{N}^i_k) \in \Sigma^i_a$. Therefore, there exists a set bijection $Q^P_{uv} : U^{uv}_{\mathbb{N}_k} \to U^{uv}_{\Sigma_a}$ such that, $I(Q^P_{uv}) = D^P_{uv}$.

Using a similar procedure, we can show that there exists set bijection $Q^R_{uv} : U^{uv}_{\mathbb{N}_k} \to U^{uv}_{\Sigma_a}$ whose interpretation is the set of permutations that sort $\mathbf{W_R}(\mathbf{u},\mathbf{v})$.

Let $Q_{uv} = Q^P_{uv} \cap Q^R_{uv}$. If $Q_{uv} = \emptyset$, then there doesn't exist an automorphism for the MDP $\mathcal{M}$.

5.2.5. Weighted Digraph

Now we define the weighted digraph $WG_{\mathcal{M}} =< V, E', W' >$ as follows:

$$
\begin{aligned}
W' &: E' \to \mathbb{R} \text{ such that } W'(u,v) = m(\mathbf{W_{P_s}}(u,v). \\
&\quad \mathbf{W_{R_s}}(u,v)) \text{ where m is a bijection from } \mathbb{R}^{2k} \to \mathbb{R} \\
&\quad \text{and . denotes concatenation}
\end{aligned}
$$

Algorithm 1 Construction

Given $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$
Let *SOLN* be an empty set
Construct the pseudograph $G_{\mathcal{M}} =< \Sigma_a, V, E, W_P, W_R >$ as defined in Section 5.2
Construct the vector-weighted digraph $VWG_{\mathcal{M}} =< V, E', \mathbf{W_P}, \mathbf{W_R} >$ as defined in Section 5.2.2
Construct the sorted vector-weighted digraph $SVWG_{\mathcal{M}} =< V, E', \mathbf{W_{P_s}}, \mathbf{W_{R_s}} >$ as defined in Section 5.2.3
for each $(u,v) \in E'$ **do**
  Compute Q^P_{uv} and Q^R_{uv} by finding the partition of $\mathbb{N}_k$ as described in Section 5.2.4
  $Q_{uv} \leftarrow Q^P_{uv} \cap Q^R_{uv}$
  if $Q^P_{uv} \cap Q^R_{uv}$ does not exist **then**
    exit
  end if
end for
Construct the weighted digraph $WG_{\mathcal{M}} =< V, E', W' >$ using m as described in Section 5.2.5
$F \leftarrow DGEN(WG_{\mathcal{M}})$ where F is the set of generators of $AutWG_{\mathcal{M}}$
for each $f \in F$ **do**
  for each $(u,v) \in E'$ **do**
    $G_{uv} \leftarrow Q_{f(u)f(v)} \odot Q^{-1}_{uv}$
  end for
  Let $\hat{H^f}$ be an empty set
  for each $u \in V$ **do**
    $G_u \leftarrow G_{uv}$ from some $v \in V$
    for each $v \in V$ **do**
      $G_u \leftarrow G_u \cap G_{uv}$
    end for
    Add G_u to $\hat{H^f}$
  end for
  Add $< f, \hat{H^f} >$ to *SOLN*
end for

5.2.6. CONSTRUCTION

The procedure for finding symmetries of an MDP $\mathcal{M}$ is given in Algorithm 1.

The complexity of the algorithm is as follows. The construction steps in lines 3 to 5, are at most polynomial in $|E|$. Using a constant access time data structure like a hashtable, Q^P_{uv} and Q^R_{uv} can be constructed in $O(|E_{uv}|)$ time.The intersection takes $O(|E_{uv}|^2)$ time. Since this runs for $|E'|$ iterations, computation of Q_{uv} is at most polynomial in $|E|$. Since m is known, the construction of weighted digraph in line 13, is polynomial in $|E|$. With the use of procedures that return at most $|V|$ automorphisms of $AutWG_{\mathcal{M}}$ (Mathon, 1979), the construction of G_u for each f, from lines 15 to 26, runs for at most $|V|$ iterations.

The most expensive part of the loop from lines 20 to 26 is the computation of $|V|^2$ intersections. But this is still polynomial in $|V||E|$ time. Hence the algorithm takes polynomially more time than the solution time of *DGEN*. Also to extract a solution from *SOLN*, we need to extract $|V|$ SDAR functions from $\hat{H^f}$ for each f, which takes $|E_{uv}|$ time if we use a constant access time data structure. So extraction of a solution takes $O(|V|^2|E|)$ which is still polynomial in $|V||E|$. While one can intuitively see that the reduction is indeed polynomial time, the proof is presented in an associated technical report (Narayanamurthy & Ravindran, 2008), due to lack of space.

5.3. Significance

The above result is significant both theoretically and practically. Practically speaking, the reduction to Graph Isomorphism allows us to use any of the numerous off-the-shelf Graph Isomorphism solvers to find symmetries on MDPs. In fact, we use NAutY - No Automorphisms, Yes?, the best Graph Isomorphism solver currently available (Skiena, 1997) to find out symmetries in MDPs. NAutY solves *AGEN*(*G*). It uses backtracking and a refinement procedure to find the canonical labeling. If two different labelings lead to the same graph, then an automorphism can be found using these labelings (McKay, 1981). In the worst case it can take exponential time. So it allows the use of a variety of vertex invariants, which act like heuristics, to solve harder problems. However, for random graphs with n vertices and edge probability 0.5, average execution times for large n are about n^2 nanosecs.We use NAutY in the fourteenth line in the construction, where we need to solve *DGEN*(*G*). We first convert the weighted digraph into an unweighted digraph using standard procedure. We then use NAutY to find the generators of the automorphism group of the so found digraph. From these we extract generators of *AutWG* as per the above procedure. We present some results in Section 6.

6. Results

The experiments were run on the following two domains. We describe results per domain.

6.1. Probabilistic GridWorld

The domain is an $N \times N$ GridWorld with four probabilistic actions of going UP, DOWN, RIGHT and LEFT having a 90% success probability. The initial state was (0,0) and the goal states were $\{(0, N-1), (N-1, 0)\}$. We used Algorithm 1 to find the symmetries with NAutY being used as the *DGEN* solver. We then used the symmetries to find the partition of Ψ. We were able to find the partition corresponding to the symmetry group, that is, for a grid of size $M \times N$, states (x,y), (y,x), (M-1-x,N-1-y) and (N-1,M-1-x) are equivalent. We present the time taken by the algorithm for GridWorlds of different sizes.

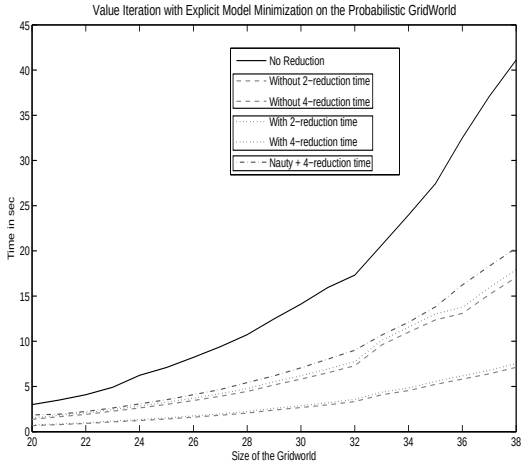

Figure 1. Average running times of the value iteration algorithm with explicit model minimization on Probabilistic GridWorld vs size of the GridWorld. Each of the 3 sets should be compared with the graph for no reduction. Curves in a set represent different degrees of symmetry. Each set shows the time reduction with reduced model usage. First one discounts the time taken to find symmetries and for reduction. The next set includes the time for reduction but discounts time taken to find symmetries. The last one includes both the time taken to find symmetries using NAutY and time for reduction.

To complete the end-to-end approach, we ran the $\mathcal{G}$-reduced image algorithm, presented in (Ravindran, 2004), to find the reduced image and ran the Value Iteration algorithm on the reduced image. To show the efficiency of reduction, we show the time taken for reduction and solution separately. We also present the case of a handcrafted 2-folded symmetry which is used with the $\mathcal{G}$-reduced image algorithm and reduced model is used with Value Iteration.

From Figure 1 it is evident that the reduced model con-

struction is efficient and adds little overhead. However, the results of the end-to-end approach show a significant overhead due to symmetry finding. It cuts the saving by almost half. Still the results are significant because they double the size of the largest GridWorld that can be solved in some given time.

6.2. GridWorld Soccer

The domain is a soccer-inspired grid domain. It is a slightly modified version of that described in (Bowling, 2003). We first describe the original version of the domain and then state the modification.

It is an $M \times N$ grid with two agents. One is denoted the attacker (**A**) who holds the ball and the other as the defender (**B**) who tries to snatch the ball from the attacker. The center lines/grids(depending on whether M and N are even or odd) for both x-axis and y-axis are chosen naturally. The state is defined by the non-identical positions of the attacker and the defender. This defines the state space with $(MN)^2 - (MN)$ states. The actions are movements in the four compass directions: **N**, **E**, **W**, **S** and the hold action **H**. It is a single player game, in that, only the attacker chooses actions deliberately while the defender executes random actions. The action chosen by the attacker and the random action of the defender are executed in random order, which determines the next state. However if the defender tries to move into the attacker's location then the state is unchanged and if the attacker tries to move into the defender's location, the game is reset to the initial state which is shown in Figure 2. The right hand section of the grid is the attacker's half and the left hand section that of the defender. The goal is chosen to be situated beyond the first column of grids occupying one grid on each side of the y-axis central line/grid. A **W** action from the squares in front of the goal state leads to a goal with a reward of 1 and to the end of an episode. Everywhere else the reward is 0. A 5×4 domain is shown in Figure 2.

Intuitively, the domain seems symmetric around the y-axis center line. However, the results of using Algorithm 1 on this domain showed us that the domain is not symmetric due to the existence of the reset action when the attacker tries to move into the defender's position. So we modified the domain to have symmetric reset, that is, reset happens to the initial state and its symmetric state around the y-axis center line with equal probability. This makes the domain symmetric as per intuition, which the algorithm confirms.

Interestingly, the algorithm also finds that the existence of the hold action adds further symmetry. The grids along the border of the domain act as walls. For example, the northern wall stops the **N** action leaving the state unchanged which is the same result if the agent were to execute a **H** action. These additional symmetries which we did not think

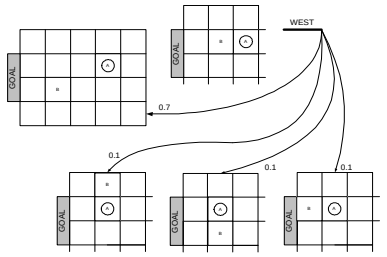

Figure 2. Single Player grid soccer where agent **B** selects it actions randomly. The initial state is shown on the left and an example of transitions and associated probabilities are given for a particular state and action on the right. Notice that fifty percent of the time **A**'s actions are executed first causing it to lose the ball and the game reset to the initial state. In addition, if **B** selects **H** or **E** it does not move and so **A** still loses the ball and returns to the initial state. The other outcomes are equiprobable.

of before running algorithm were found by the algorithm. This suggests that there might exist complicated symmetries that will be discovered by the algorithm, which are hard to find, even upon a close examination. Also in many cases, symmetries are size invariant. So we can use the algorithm on a relatively smaller version of the domain and find symmetries which might still hold on the larger version.

We present the time taken by the algorithm for different sizes. An increment of one here means an increase of one along both axes. The presence of two agents, blows up the state space very rapidly and we hit the limit on the order of the graph imposed by NAutY very soon (for a 11×10 grid).To present similar graphs as in the probabilistic GridWorld case, we use the explicit model minimization approach with Value Iteration. The results are presented in Figure 3.

In this case, we find that the overheads due to the construction and the $\mathcal{G}$-reduced image algorithm is negligible. Though efficiency of the $\mathcal{G}$-reduced image algorithm is expected, the efficiency of the construction can be possibly because of the structure of the domain yielding an easy graph to find automorphisms on.

7. Conclusion

In this work, we have provided a constructive proof for the *Isomorphism Completeness* of the problem of finding symmetries. We have also proposed the use of this constructive proof along with an efficient minimization algorithm to solve an MDP using symmetries and demonstrated it empirically. As part of future work, we are looking at adapting approximation algorithms for finding graph isomorphisms to finding approximate symmetries in MDPs.

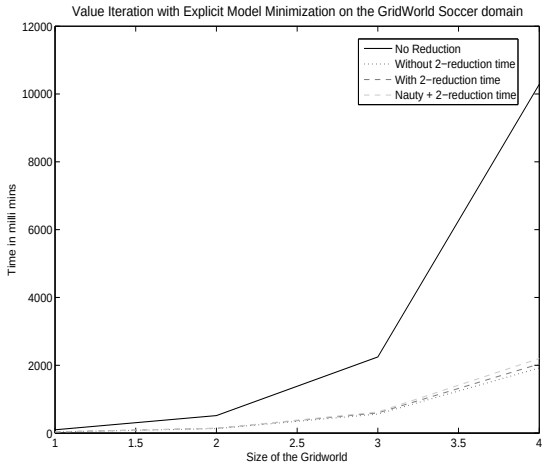

Figure 3. Average running times of the value iteration algorithm with explicit model minimization on GridWorld Soccer domain vs size of the domain. Size of one represents the 5×4 grid. Thereafter an increment of one means an increment of one along both axes. Each graph should be compared with the graph for no reduction. The other graphs show the time reduction with reduced model usage. First one discounts the time taken to find symmetries and for reduction. The next one includes the time for reduction but discounts time taken to find symmetries. The last one includes both the time taken to find symmetries using NAutY and time for reduction.

Acknowledgements

We would like to thank the reviewers for their valuable comments and inputs. We would also like to thank Google Research for supporting Dr. Ravindran's participation in the conference.

References

Amarel, S. (1968). On representations of problems of reasoning about actions. In D. Michie (Ed.), *Machine intelligence 3*, vol. 3, 131–171. Amsterdam, London, New York: Elsevier/North-Holland. Amarel, S.

Booth, K. S., & Colbourn, C. J. (1977). *Problems polynomially equivalent to graph isomorphism* (Technical Report). University of Waterloo.

Bowling, M. (2003). *Multiagent learning in the presence of agents with limitations*. Doctoral dissertation, Carnegie Mellon University.

Crawford, J. (1992). A theoretical analysis of reasoning by symmetry in first-order logic.

Dean, T., & Givan, R. (1997). Model minimization in markov decision processes. *AAAI/IAAI* (pp. 106–111).

Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., & Walsh, T. (2002). Breaking row and column symmetries in matrix models.

Givan, R., Dean, T., & Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, *147*, 163–223.

Hartmanis, J. (1966). *Algebraic structure theory of sequential machines (prentice-hall international series in applied mathematics)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Lee, D., & Yannakakis, M. (1992). Online minimization of transition systems (extended abstract). *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing* (pp. 264–274). New York, NY, USA: ACM Press.

Manning, J. B. (1990). *Geometric symmetry in graphs*. Doctoral dissertation, Purdue University.

Mathon, R. (1979). A note on the graph isomorphism counting problem. *Information Processing Letters*, *8*, 131–132.

McKay, B. D. (1981). Practical graph isomorphism. *Congressus Numerantium*, *30*, 45–87.

Miller, G. L. (1977). Graph isomorphism, general remarks. *STOC '77: Proceedings of the ninth annual ACM symposium on Theory of computing* (pp. 143–150). New York, NY, USA: ACM Press.

Narayanamurthy, S. M., & Ravindran, B. (2008). *On the hardness of finding symmetries in markov decision processes* (Technical Report IITMCSETR-01-08). Indian Institute of Technology, Madras.

Ravindran, B. (2004). *An algebraic approach to abstraction in reinforcement learning*. Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst.

Read, R. C., & Corneil, D. G. (1977). The graph isomorphism disease. *Journal of Graph Theory I*, 339–363.

Skiena, S. (1997). The stony brook algorithm repository.

Zinkevich, M., & Balch, T. (2001). Symmetry in markov decision processes and its implications for single agent and multiagent learning. *Proceedings of the ICML-01* (pp. 632–640). Morgan Kaufmann.