Algebraic MDP Minimization Using the Graph Laplacian

Rajasekaran Masatran MASATRAN AT FREESHELL.ORG

Indian Institute of Technology Madras, Chennai, TN, India

Balaraman Ravindran RAVI AT CSE.IITM.AC.IN

Indian Institute of Technology Madras, Chennai, TN, India

Abstract

The *Markov decision process* is the standard model for sequential decision problems. Identification of MDP isomorphisms allows us to minimize it, speeding up the solution process. To minimize the MDP, we reduce it to a Markov chain using the uniform random walk policy. Markov chain isomorphism is essentially *directed weighted graph matching*. Ours is the first attempt at using the laplacian to compute MDP isomorphisms. We introduce the problem, formulate it, and propose a solution using the graph laplacian.

Eigendecomposition of the adjacency matrix and other laplacian matrices provides descriptive graph invariants[1]. We use the eigendecompositions of the symmetrized adjacency matrices to find all matching close to the optimal one. The proposed method is *analytic*, as opposed to combinatorial or iterative. We use the solution to weighted graph matching to construct the minimized MDP. Experiments show that this method is indeed resistant to noise.

1. Introduction

1.1. Problem

The *Markov decision process* is the standard model for sequential decision problems. Though the MDP is a simple model, computing its optimal policy is computationally intensive, as the time complexity is cubic in the number of states. Identification of isomorphisms allows us to do MDP minimization. This reduces the number of states in the MDP and substantially speeds up the solution process. Our problem is to develop an efficient method for noise-resistant identification of MDP isomorphisms.

Consider the example MDP, in Figure 1 (a), from (Ravindran & Barto, 2004). This is a gridworld with deterministic actions. The four actions are N, E, W and S, denoting North, East, West and South respectively. In each state, the four adjacent states are accessible, via the respective action. The goal state is G.

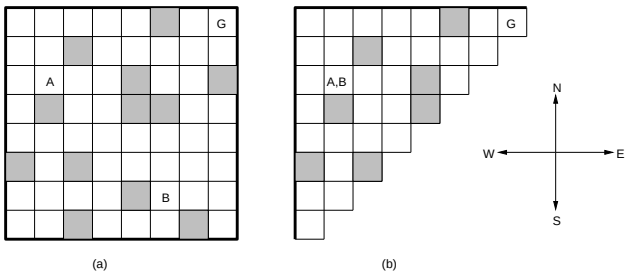

Figure 1. Example Gridworld

There are some equivalences in this gridworld. The state-action pairs (A, E) and (B, N) are equivalent, as taking action E in state A, and taking action N in state B, go to equivalent states. The gridworld in Figure 1 (b) is a reduced model of the gridworld in Figure 1 (a). The state-action pairs (A, E) and (B, N) in the original problem are merged to form the state ($\{$A, B$\}$, E) in the reduced problem. A solution to this reduced gridworld corresponds to a solution to the original problem.

[1]Graph invariant: A property of the graph that is unchanged under relabeling of the vertices.

1.2. Analytic Algorithms

The parameters of the MDP are generally not known precisely as they are estimated empirically. It is necessary to ensure that this inaccuracy does not hinder identification of MDP isomorphisms. Because of this requirement, we keep our algorithm *analytic*, as opposed to combinatorial or iterative. Using analytic methods has these advantages:

1. Our algorithm is simple and elegant.
2. Since the parameters of the MDP are estimated empirically, they have noise which will affect the algorithm. Our algorithm handles numerical inaccuracy to some extent.
3. Since there are various transition probabilities that are estimated, it is helpful to have a single tolerance that is shared among all of them. Since our algorithm is analytic, it can trivially share numerical tolerance among different variables.
4. There is scope for providing theoretical error bounds, like in (Taylor et al., 2009).
5. Since efficient algorithms for eigendecomposition are available, our algorithm is relatively efficient.

1.3. The Decoupling Principle

The use of spectral techniques is an instance of the decoupling principle described in (Sadun, 2008):

Many phenomena in mathematics, physics, and engineering are described by linear evolution equations. Light and sound are linear waves. Population growth and radioactive decay are described well by first order linear differential equations.

When faced with coupled equations involving variables $x_1, x_2, \cdots, x_m$, we define new variables $y_1, y_2, \cdots, y_m$. These variables can always be chosen so that the evolution of y_1 depends only on y_1 (and not on $y_2, \cdots, y_m$), the evolution of y_2 depends only on y_2, and so on. To find $x_1(t), x_2(t), \cdots, x_m(t)$ in terms of the initial conditions $x_1(0), x_2(0), \cdots, x_m(0)$, we convert $\mathbf{x}(0)$ to $\mathbf{y}(0)$, then solve for $\mathbf{y}(t)$, then convert to $\mathbf{x}(t)$.

We use the eigenvectors of the coefficient matrix as to determine $\mathbf{y}$ in terms of $\mathbf{x}$. When this transformation is applied, the coefficient matrix becomes a diagonal matrix, and decouples the equations. Many powerful techniques such as fourier analysis are just special applications of the decoupling principle.

The graph laplacian is a decoupling technique as well. It represents the geodesics of the graph in terms of independent eigenfunctions.

1.4. Directed Weighted Graphs

A directed weighted graph G is an ordered pair (V, A), where V is the set of nodes of the graph, and A is a weighting function which gives a real non-negative value $A(v_i, v_j)$ to each ordered pair of nodes (v_i, v_j), $v_i \in V$, $v_j \in V$.

$|V| = n$. The nodes of the graph are v_1, v_2, ..., v_n. The graph is *undirected* if its weight function is *symmetric*, i. e., $A(v_i, v_j) = A(v_j, v_i)\ \forall\ (v_i, v_j)$. We allow the weight function to take any real value.

1.4.1. Directed Weighted Graph Matching

Directed weighted graph matching is the problem of finding the optimum matching between two directed weighted graphs. Self-loops are permitted. This type of graph occurs frequently, for instance, as the transition matrix of a Markov chain.

We formulate it based on the weighted graph matching problem formulation in (Umeyama, 1988).

Let $G = (V_G, E_G)$, and $H = (V_H, E_H)$ be directed weighted graphs with adjacency matrices A_G and A_H. DWGM is the problem of finding a one-to-one correspondence ϕ between $V_G = v_1, v_2, \ldots, v_n$ and $V_H = v'_1, v'_2, \ldots, v'_n$, which minimizes a *difference* between G and H. We use a sum-of-squares criterion for difference:

$$J(\phi) = \sum_i^n \sum_j^n (A_G(v_i, v_j) - A_H(\phi(v_i), \phi(v_j)))$$

$J(\phi)$ can be reformulated using the permutation matrix P

correspoding to ϕ.

$$J(P) = \|PA_GP' - A_H\|^2$$

where $\|.\|$ is the euclidean norm $\|A\| = (\sum_i^n \sum_j^n |a_{i,j}|^2)^{\frac{1}{2}}$. Thus, DWGM is reduced to the problem of finding the permutation matrix P which minimizes $J(P)$.

Two ordinary graphs are isomorphic if there exists a one-to-one correspondence between their nodesets that preserves adjacency. We call two directed weighted graphs isomorphic if there exists a one-to-one correspondence ϕ that makes $J(\phi)$ zero. In this case, we have

$$PA_GP' = A_H$$

Because ordinary graphs are directed weighted graphs with a symmetric boolean adjacency matrix, DWGM includes the graph isomorphism problem.

In general, it is not easy to find the exact solution for DWGM since it is a combinatorial problem. Thus, we need to develop an efficient method which gives a nearly optimum solution, i.e., a permutation matrix $\widehat{P}$ whose criterion value $J(\widehat{P})$ is very close to the optimum value. Since we do not know the optimum criterion value in advance, what we can do is to give an algorithm which determines a permutation matrix of small criterion value. In this paper, we propose a fast algorithm giving such permutation matrices.

1.4.2. Prior Work

(Shapiro & Brady, 1992) is an analytic algorithm for the weighted graph matching problem. We remove the direction information in our graph, and use this algorithm. (Umeyama, 1988) is another algorithm for the weighted graph matching problem, but it cannot use partial eigendecomposition. (Chung, 2005) is a directed laplacian for graphs defined using the perron eigenvector, that might be usable for directed weighted graph matching.

2. Our Solution

In **Section 2: MDP Isomorphism to WGM**, we reduce the MDP isomorphism problem into the directed weighted graph matching problem. To do this, we convert the MDP into a Markov chain using the uniform random walk policy. Markov chain isomorphism is essentially directed weighted graph matching.

In **Section 3: WGM to MSAP**, we reduce the directed weighted graph matching problem into the multi-solution assignment problem. To do this, we symmetrize the adjacency matrix, and use a variant of (Shapiro & Brady, 1992).

In **Section 4: MSAP**, we formulate the multi-solution assignment problem, and solve it by backtracking in the hungarian algorithm.

2.1. MDP Isomorphism to WGM

In this section, we reduce the MDP isomorphism problem to weighted graph matching.

2.1.1. Challenges

We need to find state-action equivalences, not just state equivalences.

2.1.2. Contributions

Prior approaches to MDP isomorphism have relied on analytical techniques. Ours is probably the first algorithm that is algebraic. We solve the MDP isomorphism problem by reducing it to weighted graph matching, and reducing weighted graph matching to the multi-solution assignment problem. This structure of our solution is a novel contribution.

2.1.3. Algorithm

To minimize the MDP, we reduce it to a graph, identify the vertex equivalence partitioning on the graph, and use those to determine the state-action equivalence partitioning on the original MDP.

It is important to not restrict the algorithm to state equivalence. State-action equivalence, i.e., permutation of actions at one state to correspond to actions at another state, should also be considered.

The uniform random walk policy is used to reduce the MDP to a graph, since this is the only one of the candidate reduc-

tions to graph isomorphism that is compatible with both state-action isomorphisms and purely analytic techniques. The resultant Markov chain is a directed weighted graph.

Algorithm 1:]

Input: $P^a_{s,s'}$
Output: $O^a_{s,s'}$
repeat
  $P(s'|s) \leftarrow \frac{1}{|A|} \sum_{a \in A} P^a_{s,s'}$
  $stateeq \leftarrow DWGM(P(s'|s), P(s'|s))$
  if $stateeq = I_n$ **then**
    $exit$
  end
  $saeq \leftarrow saeq\text{-}from\text{-}stateeq(stateeq, P^a_{s,s'})$
  $O^a_{s,s'} \leftarrow reduce(P^a_{s,s'}, saeq)$
until *false*
MDP Isomorphism to WGM [1]

$P^a_{s,s'}$ is reduced in two stages using the state-action equivalences.

First, the destination states are merged to form destination superstates. The probabilities of landing into the destination states within each superstate are summed up.

Next, the source states are merged to form source superstates, during which the action mapping within the superstate is computed.

2.1.4. Uniform Random Walk

A policy for an MDP gives the probability of taking each action, given the current state.

$$\pi(s, a) = P(a_{t+1} = a | s_{t+1} = s)$$

For a fixed policy, the MDP acts like a Markov chain.

$$P(s'|s) = P(s_{t+1} = s' | s_t = s) = \sum_{a \in A} \pi(s, a)\ P^a_{s,s'}$$

2.1.5. State-Action Equivalences

State equivalence is an equivalence relation, i.e., graph isomorphism is reflexive, symmetric and transitive. It maps each state into a partition, which we call a superstate.

In each state, each action is a point in the m-dimensional standard simplex, where each dimension is the probability that taking the action in the current state will lead to a particular superstate. So, given a state and an action, we can identify the equivalent action for all states in its superstate. This gives us the state-action equivalences.

If A is the set of all possible actions and z is the size of the superstate, then a simple algorithm can identify the state-action equivalences in the superstate in time $z \times |A|^2$. Thus all state-action equivalences can be found in time $|S| \times |A|^2$, where S is the set of states.

2.2. WGM to MSAP

In this section, we use an algorithm based on (Shapiro & Brady, 1992) for reducing weighted graph matching to the multi-solution assignment problem.

2.2.1. Challenges

The transition matrix is large and its eigendecomposition is the primary bottleneck in our algorithm.

2.2.2. Contributions

Since the transition matrix is large, we use an algorithm for partial eigendecomposition, and filter out any spurious isomorphisms that are generated.

2.2.3. Algorithm

Our algorithm for weighted graph matching takes two directed weighted graphs as input, and returns a vertex equivalence matrix as output.

Let $G = (V_G, E_G)$, and $H = (V_H, E_H)$ be the input graphs, with adjacency matrices A_G and A_H respectively.

The output is a matrix $M : V_G \times V_H \rightarrow \{false, true\}$ that represents all possible matchings.

2.2.4. Hermetization

Eigendecomposition of a general real-valued matrix $M \in \mathbb{R}^{n \times n}$ gives complex eigenvalues and complex eigenfunctions. Hermetization makes the eigenvalues real, but the eigenfunctions are still complex. Symmetrization makes both eigenvalues and eigenfunctions real.

Algorithm 2:]

$sym_G \leftarrow \frac{A_G + A'_G}{2}$;
$sym_H \leftarrow \frac{A_H + A'_H}{2}$;
$m \leftarrow \sqrt{|V_G|}$;
$(\Lambda_G, U_G) \leftarrow parteig(sym_G, m)$;
$(\Lambda_H, U_H) \leftarrow parteig(sym_H, m)$;
$(\overline{\Lambda_G}, \overline{U_G}) \leftarrow unique(\Lambda_G, U_G)$;
$(\overline{\Lambda_H}, \overline{U_H}) \leftarrow unique(\Lambda_H, U_H)$;
$vertexeq \leftarrow \langle \overline{U_G}\, \overline{U_H} \rangle$;
$matchings \leftarrow MSAP(vertexeq)$;
$matchings \leftarrow WGM(G, H)$ [2]

$$herm = \left(\frac{A + A'}{2} + i \frac{A - A'}{2} \right)$$

The real part of the hermetized matrix is the same as the symmetric matrix. If the MDP is symmetric, then the hermetized matrix is purely real.

2.2.5. Truncation

Eigendecomposition is an expensive operation and is the bottleneck in our algorithm. Since there is significant redundant information for our use of the eigendecomposition, it is sufficient to compute a limited number of eigenfunctions, provided that we also have eigenvalue multiplicity information.

We compute the eigenfunctions corresponding to the m largest eigenvalues. We use the square root of the number of states as a rule-of-thumb value for m.

2.2.6. Eigendecomposition

$U \in \mathbb{C}^{n \times m}$ is a partial eigendecomposition of $herm$. Each column is an eigenfunction of an eigenvalue, while each row is a label for a vertex.

$U \in \mathbb{R}^{n \times m}$ is a partial eigendecomposition of sym. Each column is an eigenfunction of an eigenvalue, while each row is a label for a vertex.

$U = parteig(herm)$

2.2.7. Eigenvalue Multiplicity

Each eigenvalue has an associated eigenspace for which the corresponding eigenfunctions form the basis.

Theorem 2.18 of (Bıyıkoğlu et al., 2007) analyses eigenvalue multiplicity. If the eigenvalue is not shared, the eigenfunction is uniquely determined except for sign. Sign correction must be done before comparing vertices.

If the eigenvalue is shared, then the eigenfunctions that share it are together a basis for the eigenspace. In this case, we remove these eigenfunctions from consideration, since finding the optimal rotation of the basis for matching is computationally expensive.

2.2.8. Vertex Equivalence

Let $U_i \in \mathbb{C}^m$ be the eigendecomposition label on vertex v_i. We use the standard hermetian form $\langle XY \rangle = XY^* = \sum_i X_i Y_i^*$, as a similarity measure between labels, and use it to determine vertex equivalence.

Y^* denotes the transpose of the element-wise complex conjugate of Y. $vertexeq : V_G \times V_H \to [0, 1]$ is defined as:

$vertexeq_{ij} \leftarrow \langle U_{Gi}\, U_{Hj} \rangle$.

This similarity measure has the elegant property that if all eigenfunctions are used, it takes value 1 only if the two vertices are equivalent. It can be expressed in matrix form as:

$$vertexeq \leftarrow U_G U_H^*$$

2.3. MSAP

In this section, we modify the hungarian algorithm (Kuhn, 1955) to find multiple solutions and approximate solutions.

2.3.1. Challenges

The hungarian algorithm returns only one solution, but we need all solutions that are almost optimal.

2.3.2. Contributions

We use backtracking in the hungarian algorithm to return all solutions that are almost optimal.

2.3.3. Assignment Problem

Given sets X and Y of equal size, and a cost function $C : X \times Y \rightarrow \mathbb{R}$, find the assignment $W \in \mathbb{R}^{n \times n}$ that minimizes $\sum_{x \in X} \sum_{y \in Y} W(x, y)\ C(x, y)$, with constraints $\sum_{x \in X} C(x, y) = 1$, $\sum_{y \in Y} C(x, y) = 1$, and $C(x, y) \geq 0 \ \forall\ x \in X, y \in Y$.

2.3.4. Hungarian Method

Jacobi[2] invented the Hungarian method to solve the assignment problem. It was posthumously published in (Jacobi, 1865).

Kőnig[3]' theorem, published in (König, 1931), gives an equivalence between the maximum matching problem and the minimum vertex cover problem in bipartite graphs. Egerváry[4] published his rank reduction theorem in (Egerváry, 1956). Kuhn combined Kőnig's theorem and Egerváry's rank reduction into the hungarian algorithm, and published it in (Kuhn, 1955). At that time, Jacobi's paper was not known, and Kuhn named the algorithm the hungarian method.

2.3.5. MSAP Concepts

We enhanced the Hungarian method as described below. Our implementation is based on Markus Bühren's implementation ((Bühren, 2009)) of the Hungarian method.

All Solutions The processed distance matrix is intercepted, and backtracking is used to determine all solutions, instead of just the single solution found by the original algorithm.

Tolerance to Inaccuracy Checks for zero are replaced by checks for being close to zero. This allows us to handle noise in the cost matrix.

Algorithm 3: $assignments \leftarrow backtrack(distMatrix, row, epsilon\text{-}assignment)$

Input: $distMatrix$, row, $epsilon\text{-}assignment$
Output: $assignments$
// $distMatrix$ contains the costs, and is non-negative.
// Backtracking on row, the row number in $distMatrix$.
$assignments \leftarrow \{\}$
if *row = (size(distMatrix) + 1)* **then** // Found assignment.
  $assignments \leftarrow mybuildassignment(distMatrix)$
  // One assignment pair for each row of distMatrix.
  // Index of -1, of each row, together give full assignment.
  return
end
for $column \in$ {*columns of distMatrix whose* row^{th} *element is almost zero*}
do
  if *there are no assignment pairs in* $column^{th}$ *column* **then**
    $distMatrix(row, column) \leftarrow -1$
    // Marking assignment pair by -1.
    $assignments \leftarrow \{assignments \bigcup backtrack(distMatrix, row+1, epsilon\text{-}assignment)\}$
    // Backtracking, next row onwards.
    $distMatrix(row, column) \leftarrow 0$
    // Remove -1 mark for current column, move onto next column.
  end
end

2.4. Results

As expected, noise resistance is found to decrease with increase in number of states.

[2]Jacobi: Carl Gustav Jacob Jacobi, 1804–1851, German mathematician.

[3]Kőnig: Dénes Kőnig, 1884–1944, Hungarian mathematician, wrote the first book on graph theory.

[4]Egerváry: Jenő Egerváry, 1891–1958, Hungarian mathematician.

Table 1. Results

World Size	Noise Tolerance
2^3	.25
3^3	.23
4^3	.15
5^3	.12
6^3	.04
7^3	.01

The eigenvalues of the normalized Laplacian are bounded between 0 and 2. The number of states in the MDP equals the number of vertices in the graph equals the number of eigenvalues. As many eigenvalues as the number of states in the MDP are packed into the $[0, 2]$ interval. As the size of the MDP increases, the number of eigenvalues in the interval increases as well, and a small amount of noise will suffice to change the order between the eigenvalues.

Our algorithm can handle noise only upto the level where the order among eigenvalues is corrupted by the noise.

3. Conclusion

Spectral techniques were used to design a purely analytic algorithm for directed weighted graph matching. Experiments show that this algorithm has some resistance to noise when used for MDP minimization. As expected, noise resistance is found to decrease with increase in number of states.

3.1. Challenges

We need to find state-action equivalences, not just state equivalences.

The transition matrix is large and its eigendecomposition is the primary bottleneck in our algorithm.

The hungarian algorithm returns only one solution, but we need all solutions that are almost optimal.

3.2. Contributions

Prior approaches to MDP isomorphism have relied on analytical techniques. Ours is probably the first algorithm that is algebraic. We solve the MDP isomorphism problem by reducing it to weighted graph matching, and reducing weighted graph matching to the multi-solution assignment problem. This structure of our solution is a novel contribution.

Since the transition matrix is large, we use an algorithm for partial eigendecomposition, and filter out any spurious isomorphisms that are generated.

We use backtracking in the hungarian algorithm to return all solutions that are almost optimal.

3.3. Future Work

The multi-solution assignment problem is the bottleneck in our algorithm. For each vertex, considering only those vertices witin ϵ from its similarity to itself, and restricting the multi-solution assignment algorithm to these vertices, will make it faster.

A theoretical error bound for approximate isomorphisms, like (Taylor et al., 2009), could be provided.

References

Bühren, Markus. Functions for the rectangular assignment problem [computer software], version 2009-12-01, 2009.

Bıyıkoğlu, Türker, Leydold, Josef, and Stadler, Peter F. *Laplacian Eigenvectors of Graphs*, volume 1915 of *Lecture Notes in Mathematics*. Springer, 2007. ISBN 978-3-540-73509-0.

Chung, Fan R. K. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005.

Egerváry, Jenő. Régi és új módszerek lineáris egyenletrendszerek megoldására. *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei*, 1:109–123, 1956.

Jacobi, Carl Gustav Jacob. De investigando ordine systematis aequationum differentialum vulgarium cujuscunque. *Journal für die reine und angewandte Mathematik*, 64: 193–216, 1865.

Kuhn, Harold W. The Hungarian method for the assign-

ment problem. *Naval Research Logistics Quarterly*, 2 (1–2):83–97, 1955.

König, Dénes. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

Ravindran, Balaraman and Barto, Andrew G. Approximate homomorphisms: A framework for non-exact minimization in markov decision processes. In *KBCS 2004: Proceedings of the Fifth International Conference on Knowledge Based Computer Systems*, 2004.

Sadun, Lorenzo. *Applied Linear Algebra: The Decoupling Principle*. American Mathematical Society, second edition, 2008. ISBN 0-8218-4441-5.

Shapiro, Larry S. and Brady, J. Michael. Feature-based correspondence: an eigenvector approach. *Image and Vision Computing (British Machine Vision Conference 1991)*, 10(5):283–288, 1992. ISSN 0262-8856.

Taylor, Jonathan, Panangaden, Prakash, and Precup, Doina. Bounding performance loss in approximate MDP homomorphisms. In *NIPS 2008: Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, 2009.

Umeyama, Shinji. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5): 695–703, 1988.