On Constraints on the Search Path of Policy Iteration

Omid Madani

March 30, 1999

Abstract

We describe a few structural properties enjoyed by the policy space of problems such as infinite-horizon MDPs. From these properties we derive constraints limiting the number of iterations of algorithms such as the policy iteration algorithm for infinite-horizon MDPs and the Hoffman-Karp algorithm for simple stochastic games. An open problem is to characterize the growth of the worst-case number of iterations of these algorithms subject to the derived constraints.

1 Introduction

This paper reports on an investigation of some open computational complexity questions regarding problems under infinite-horizon MDPs [Mad98, MC94b] and Simple Stochastic Games (SSGs) [Con92]. The best known algorithm for SSGs is a randomized algorithm running in $O(2^{\sqrt{n}})$ expected time [Lud95], where n denotes the number of states of the system. Infinite-horizon MDPs are known to be in P as they can be cast as linear programs (LPs). However, since there is no known strongly polynomial time algorithm for LPs, it is not known whether MDPs have strongly polynomial time algorithms.

Policy iteration for infinite-horizon MDPs is a search algorithm that converges to an optimal policy in few iterations in practice[1]. In this report we consider the variant of policy iteration where all improvable states change actions in each iteration. No lower-bound other than the trivial $\Omega(n)$ iterations is known for this algorithm, and our experiments on random graph inputs have shown no input requiring more than $n+2$ iterations. On the other hand, the Hoffman-Karp algorithm for SSGs is closely related to policy iteration, and we expect that a close study of policy iteration would yield fruitful insights on the structure of both problems, perhaps leading to strongly polynomial time algorithms for both infinite-horizon MDPs and SSGs.

This report is an attempt in that direction. In Section 2 we describe a few structural properties enjoyed by the policy space of a class of problems including infinite-horizon MDPs.

[1] We assume two actions per state in this report for simplicity.

We use these properties in Section 3 to derive constraints limiting the number of iterations of algorithms such as the policy iteration algorithm and the Hoffman-Karp algorithm for SSGs. An open problem is to characterize the growth of the worst-case number of iterations of algorithms that must satisfy the constraints. The treatment in Section 2 is in an abstract though simple and self-contained setting. We refer the reader to [Mad98, Con92, Con93, Put94] for background on MDPs and SSGs.

2 Monotone Spaces

Consider policy spaces which come under consideration in problems such as MDPs under various finite or infinite-horizon criteria, or their generalizations such as the SSGs. Assume the system under consideration has n vertices (states), and for our purposes it suffices to assume that each vertex has two control choices, 0 or 1. Each policy is an assignment of a control choice to each vertex. Thus policies can be thought of as bit vectors and there are 2^n many of them. We call such set of 2^n policies a *policy space*. For a policy p, let v_p denote its value vector assigned by a particular optimality criterion (an n-dimensional real valued vector for n vertices), and $v_p[i]$ denote the value of vertex i using policy p. The optimality criterion can be an infinite horizon (or finite-horizon) average or total expected reward criterion, or in case of 2-player games, the state values obtained for the max player when it plays the policy p, and the min player choses its best policy against policy p.

Consider the partial ordering on value vectors of policies. This partial ordering is the natural one, derived from the component-wise comparison of values. We write $v_1 < v_2$ for two vectors v_1 and v_2, if $v_1[i] \leq v_2[i], 1 \leq i \leq n$, and for some $j, 1 \leq j \leq n$, $v_1[j] < v_2[j]$. Likewise, we write $v_1 \leq v_2$ if $v_1[i] \leq v_2[i], 1 \leq i \leq n$. We extend this notation to policies: $p < q$ if $v_p < v_q$ and $p \leq q$ if $v_p \leq v_q$. Finally $p <> q$ denotes the situation where neither $p < q$ nor $q < p$ (incomparable policies). So the partial ordering on the value vectors imposes a semi-partial ordering (the ordering is not anti-symmetric) on the set of policies. The partial ordering on the policy space of MDPs and models alike has additional structural properties which depend on the optimality criteria in question. As consequences of these properties, we obtain characteristics such as existence of optimal policies, notions of local optimality being equivalent to global optimality, correctness of policy improvement search algorithms such as parallel switch (described below), and constraints on the search sequences of algorithms. We will next explore these properties and their consequences.

Definition 1 *Switching a vertex in a policy means toggling its control choice from* 0 *to* 1 *or* 1 *to* 0 *(complementing the control bit), while leaving the rest of the policy unchanged. Similarly, switching a set of vertices in a policy means switching every vertex in the set, and leaving the rest of vertices unchanged. Given a policy p:*

- *A set S of vertices is said to be switchable if switching the choices of the vertices results*

in an increase in the value of at least one vertex in S, and the value of no vertex in S decreases.

- *A vertex v is switchable if the set $\{v\}$ is switchable.*
- *Define an unswitchable set (and vertex) in a symmetric way, i.e. the values don't increase, and the value of at least one vertex in the set decreases.*
- *If the value of no vertex in the set changes with switching the set, the set is called a don't-care set. Similarly define a don't-care vertex.*
- *Finally, switching a mixed set results in an increase in the value of at least one vertex and a decrease in the value of another.*

Switchability properties by definition are always with respect to a policy under consideration, and in what follows mention of an explicit policy is omitted. MDPs enjoy a nice monotonicity property in that switching a switchable set is not only locally beneficial (with respect to vertices in the set) but also globally beneficial (at a minimum harmless to other vertices). This motivates the next definition:

Definition 2 *A policy space is called monotone if the following are satisfied:*

- *Given any policy p, switching any switchable subset of p results in a policy q, with $p < q$*
- *Given any policy p, switching any unswitchable subset of p results in a policy q, with $q < p$*
- *Given any policy p, switching any don't-care subset of p results in q, with $v_q = v_p$*

The monotonicity of a policy set is desirable but still *not* sufficient for important properties such as the existence of upper or lower elements (e.g. optimal value vectors). However we describe two characteristics that policy spaces may have and from each of which existence of extreme elements and other desirable properties follow.

Definition 3 *A policy space has the decomposability property if,*

1. *The union of a switchable set S_1 and a switchable or don't-care set S_2, is a switchable set.*
2. *(Similar for unswitchable sets)*
3. *The union of a don't-care sets S_1 and S_2, is a don't-care set.*

Definition 4 *For any policy and any subset S of vertices, let $\delta(i)$ denote the change in value of vertex i when S is switched. Let $\delta_S = \max_{i \in S} |\delta(i)|$, and $\delta_V = \max_{i \in S} |\delta(i)|$. Note that $\delta_V \geq \delta_S$ by definition. In damped policy spaces, for any policy and any subset S of vertices $\delta_V \geq \delta_S$.*

Policy spaces of MDPs, under various criteria such as average expected-reward, total reward, or model generalizations such as 2-player games, enjoy the damping property, but systems such as pre-Leontief system of inequalities (see [MC94a]) need not. On the other hand, MDPs under total reward (discounted or undiscounted) criteria enjoy the decomposability properties, while MDPs under average-reward criteria don't (they fail property 3 of decomposability).

We can first show that the combination of monotonicity and decomposability or monotonicity and dampness establish the existence of extreme lattice points (optimal policies and/or value vectors), and that policy-improvement algorithms are correct in that they find these extreme points. The basic proof argument works by showing that if a policy has a mixed set then it has a switchable set, so the policy can be improved. As there are only a finite number of policies, there must exist an upper (optimal) element. The following lemma is what we need then:

Lemma 2.1 *Consider a monotone policy space satisfying either decomposability (type one) or dampness (type two). Then any mixed set in any policy p has a switchable (and an unswitchable) subset.*

Proof. Consider a type one space and a mixed set in a policy. The mixed set must have a switchable vertex in it, otherwise it will be don't-care or unswitchable by decomposability.

Consider a type two space, and a smallest mixed set S in it. Let S_2 be those vertices of S that increase value when switching S. S_2 cannot be empty, otherwise S would be don't-care or unswitchable and couldn't be mixed by monotonicity. If S_1 is switchable, we are done. S_1 can't be don't-care, otherwise switching it causes no vertex value to change due to dampness, and then switching S_1 can only decrease any value due to monotonicity and S would not be mixed. A similar argument shows that S_1 can't be unswitchable, otherwise, let δ denote the maximum decrease in values of vertices in S_1 which would be maximum change in values of all vertices. Now switching the rest of vertices S_2 of S can't improve any vertex value in S_2 by more than δ, otherwise some vertex in S_2 will have increased value, contradicting our assumption on S_1 and S_2, therefore some vertices of S_1 will have no increase in value again contradicting our assumption on S_1. □

Definition 5 *A policy r is an optimal policy if for all policies p, $p \leq r$.*

Hence optimal policies exist in monotone/decomposable (type 1) or monotone/damped (type 2) policy spaces, and the following generic policy-improvement algorithm stops and finds the optimal.

```
Start with an arbitrary policy p
While p contains a switchable subset do
  change p by switching a switchable subset of p
Output p
```

So far we have described several sufficient conditions for existence of optimal policies and correctness of the generic policy improvement algorithm in monotone policy spaces. The decomposability property of type 2 spaces translates to several elegant constraints on the search sequence of parallel switch. We will describe such in the next section.

3 Constraints on Parallel Switch in Type One Spaces

In this section we assume that the underlying policy space is a type one space. Let us focus on the parallel switch algorithm which switches the set of all switchable vertices in each iteration:

```
1 Start with an arbitrary policy p
2 While p contains a switchable vertex do
3   change p by switching all switchable vertices of p
4 Output p
```

Note that the most common variant of policy iteration for infinite-horizon MDPs is a parallel-switch algorithm. The argument for correctness of parallel switch is straight forward: from decomposability, if policy p has a switchable set (or a mixed set), it must have a switchable vertex (hence line 2 checks for optimality), and line 3 must improve p due to the decomposability and monotonicity properties.

Consider the sequence of policies that parallel switch goes through (policy i is the one at the start of iteration i). An example policy sequence is shown next, where policy 000 is the policy at iteration 1 (the start of the algorithm), the first iteration of the algorithm switches all vertices (bits) to get 111 for iteration 2:

1	000
2	111
3	001
4	101
5	100

We will see that, as a consequence of the Constraint corollary given below, the pattern of bit (vertex) switches in any iteration constrains the pattern of bit switches in later iterations. These constraints accumulate and eventually no additional policy can be added to the policy sequence. In this section we will explore the growth of the length of the longest possible sequence of bit vectors (policies) under these constraints as a function of n.

Lemma 3.1 *Consider any two policies p and q in a type 1 space. Let $S = \{i | v_p(i) < v_q(i)\}$, and assume $S \neq \emptyset$. Then the set $S' = S \cap \{i | p(i) \neq q(i)\}$ is not empty, and furthermore some vertex of S' is switchable in p.*

Proof. An extension of the proof of lemma 2.1 for type one spaces. □

Corollary 3.2 *(Constraint) Consider two policies p and q and assume $p < q$. Let S denote the vertices where the choice of edges is different in p and q. There is some vertex $v \in S$ that is switchable in p.*

We say a policy is *implied* at (iteration) $i < m$, if it can be constructed by switching a subset of switchable vertices of policy i (so policy $i + 1$ is by definition a policy implied at i). In our

example, at iteration 2, policies 011 (switch bit 1 only), 101 (switch bit 2 only), and 001 (switch the set of all switchable vertices) are all implied. From the decomposability and monotonicity properties, all policies implied at i are superior to policy i and hence all policies $j, j \leq i$. In the above example, we have $111 < 011$, $111 < 101$, and $111 < 001$.

Consider any implied policy q at some iteration $i \geq 2$, and a policy p at iteration $j \leq i$. We have $p < q$, so it follows from the Constraint corollary that some vertex from the set of vertices where q and p differ must be switchable in p (in iteration j) and must have the value it has in q in iteration $j+1$. Put another way, the switched bit value of at least one switchable vertex in p, must be the same as the bit value in q. We shall formulate this constraint as a Boolean disjunctive clause. Each transition from iteration i to iteration $i+1$ (transition i) introduces a clause that all implied policies at iterations $i+1$ and later should satisfy. Let x_v denote the boolean variable for bit (vertex) v. If vertex v in transition i changes from 1 to 0, then the clause for transition i will contain $\bar{x}_v$, and if it changes from 0 to 1, it will contain x_v. The clausal constraints for the example above are shown next. The (clausal) constraint for transition i is shown next to policy $i+1$:

1	000	
2	111	$(x_1 \vee x_2 \vee x_3)$
3	001	$(\bar{x}_1 \vee \bar{x}_2)$
4	101	x_1
5	100	$\bar{x}_3$

Hence we define a *legal* sequence of bit vectors to be one where:

1. No vector repeats immediately (no empty clausal constraint).

2. The disjunctive clausal constraint produced from transition i is satisfied by all the implied bit vectors at transition $i+1$ and later.

Of course, due to the clausal constraints, an immediate consequence is that no vector can appear more than once in the entire sequence. We are interested in the growth of the length of a longest legal sequence as a function of n, the number of bits (vertices). We denote this measure by $L(n)$. It is easy to show that $L(n) = \Omega(n^2)$. An interesting open problem is to obtain tight upper and lower bounds on $L(n)$.

We close with a bipartite graph representation of legal sequences. This view can aid in visualization and analysis of the process. Consider a bipartite graph $G = (V_1, V_2)$, where V_1 is a set of n vertices, one for each bit of the n bits of the policy vectors. Each vertex in V_2 corresponds to a switch group, *i.e.* the set of bits that switch together in an iteration of the algorithm. A vertex in V_1 has an edge to a vertex in V_2 if and only if the corresponding bit participates in the corresponding group. To reflect clausal constraints, some edges may be directed as follows. For any $u \in V_1 \cup V_2$, let $d(u)$ be the total number of edges incident on u,

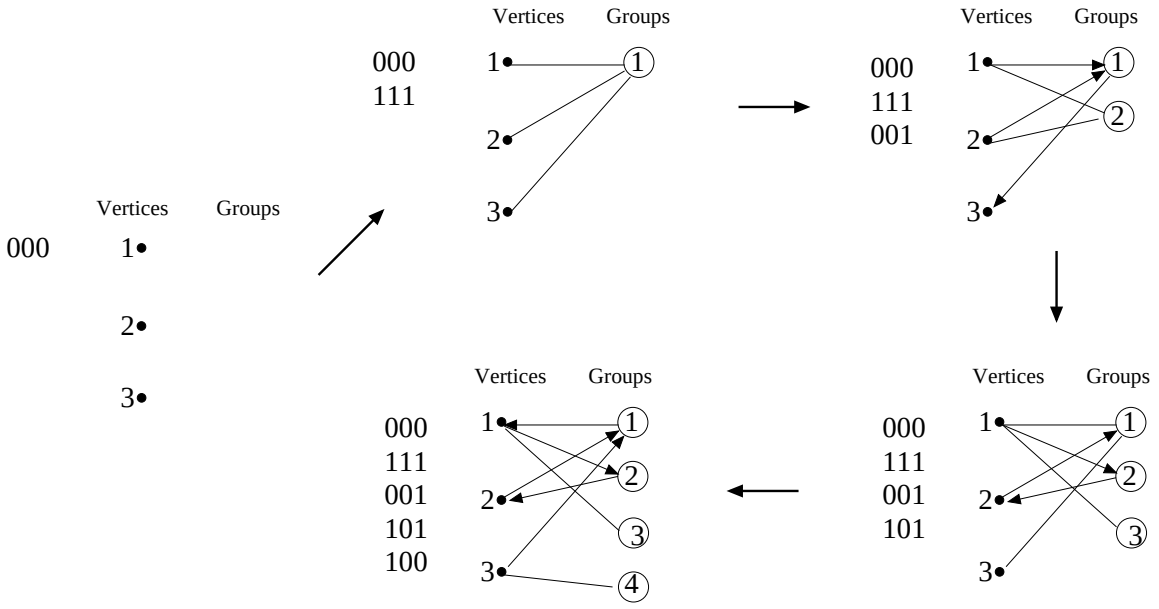

Figure 1: The process of growing a legal graph which includes toggling vertices and groups.

$out(u)$ be the number of edges directed away from u, and $in(i)$ be the number of edges directed toward u. For any vertex v and a neighboring group $g \in V_2$ the edge from vertex v is directed toward group g if and only if the current value of v is different from its value in g (v does not satisfy the clausal constraint corresponding to g). For a group $g \in V_2, in(g) \leq d(g) - 1$ and $out(g) = 0$ if $in(g) < d(g) - 1$, and $out(g) = 1$, if $in(g) = d(g) - 1$. Note that due to constraints of the process, a vertex $v \in V_1$ cannot switch if $in(v) > 0$, and a set of vertices in V_1 that contain all the vertices of some group with undirected incident edges cannot switch together. Figure 1 shows the the graphical representation for each iteration of the our example sequence. We refer to bipartite graphs that represent a legal sequence as legal graphs. Such *legal* graphs enjoy properties such as acyclicity with respect to directed edges, and it is not hard to see that for $v \in V_1, out(v) = \lfloor \frac{d(v)}{2} \rfloor$.

How fast does the maximum size for V_2 grow as a function of the size of V_1 in legal graphs? This is equivalent to the question about the growth of $L(n)$. An essentially equivalent question is how large can the maximum number of edges of legal graphs be as a function of $|V_1|$. We have conducted computer experiments to investigate the growth of $L(n)$. These experiments provide some evidence that $L(n)$ may grow super-polynomially, but perhaps sub-exponentially. We are currently investigating analytic methods for bounding $L(n)$, using the graph representation as an aid.

4 Summary and Future Work

In this paper we described several properties of policy spaces of MDPs in a uniform setting. From these properties, constraining rules on the search path of policy iteration were developed. It is an open question whether these constraining rules are strong enough to limit the run-time of policy iteration to a significantly lower than an exponential bound. On the other hand, we believe that there is more structure to MDPs, especially in the connectivity of the underlying graph, that should further limit the length of the search path of policy iteration. Discovering and characterizing such properties if they exist, and/or looking for bad example graphs for policy iteration is a promising research direction.

References

[Con92] Anne Condon. The complexity of simple stochastic games. *Information and Computation*, 96(2):203–224, February 1992.

[Con93] Anne Condon. On algorithms for simple stochastic games. In *Advances in computational complexity theory*, volume 13 of *DIMACS series in discrete mathematics and theoretical computer science*. 1993.

[Lud95] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and computation*, 117:151–155, 1995.

[Mad98] Omid Madani. Models for decision making in dynamic and uncertain domains. Technical report, University of Washington, 1998.

[MC94a] Nimrod Megiddo and Edith Cohen. Improved algorithms for linear inequalities with two variables per inequality. *SIAM Journal on Computing*, 23(6):1313–47, Dec 1994.

[MC94b] Mary Melekopoglou and Anne Condon. On the complexity of the policy improvement algorithm for markov decision processes. *ORSA Journal on Computing*, 6(2), 1994.

[Put94] Martin L. Puterman. *Markov Decision Processes*. Wiley Inter-science, 1994.