On Policy Iteration as a Newton's Method and Polynomial Policy Iteration Algorithms

Omid Madani
Department of Computing Science
University of Alberta
Edmonton, AL
Canada T6G 2E8
madani@cs.ualberta.ca

Abstract

Policy iteration is a popular technique for solving Markov decision processes (MDPs). It is easy to describe and implement, and has excellent performance in practice. But not much is known about its complexity. The best upper bound remains exponential, and the best lower bound is a trivial $\Omega(n)$ on the number of iterations, where n is the number of states.

This paper improves the upper bounds to a polynomial for policy iteration on MDP problems with special graph structure. Our analysis is based on the connection between policy iteration and Newton's method for finding the zero of a convex function. The analysis offers an explanation as to why policy iteration is fast. It also leads to polynomial bounds on several variants of policy iteration for MDPs for which the linear programming formulation requires at most two variables per inequality (MDP(2)). The MDP(2) class includes deterministic MDPs under discounted and average reward criteria. The bounds on the run times include $O(mn^2 \log m \log W)$ on MDP(2) and $O(mn^2 \log m)$ for deterministic MDPs, where m denotes the number of actions and W denotes the magnitude of the largest number in the problem description.

1 Introduction

Markov decision processes offer a clean and rich framework for problems of control and decision making under uncertainty [BDH99; RN95]. A set of central problems in this family is the fully observable Markov decision problems under infinite-horizon criteria [Ber95]. We refer to these as MDP problems in this paper. Not only are the MDP problems significant on their own, but solutions to these problems are used repeatedly in solving problem variants such as stochastic games and partially observable MDPs [Con93; HZ01]. In an MDP model, the system is in one of a finite set of states at any time point. In each state an agent has a number of actions to choose from. Execution of an action gives the agent a reward and causes a stochastic change in the system state. The problem is, given a full description of the system and actions, to find a policy, that is a mapping from states to actions, so that the expected (discounted) total reward over an indefinite (or infinite) number of action executions is maximized.

Policy improvement is a key technique in solving MDPs. It is simple to describe and easy to implement, and quickly converges to optimal solutions in practice. The improvement method begins with an arbitrary policy, and improves the policy iteratively until an optimal policy is found. In each improvement step, the algorithm changes choice of action for a subset of the states, which leads to an improved policy. These algorithms differ on how the states are picked. In addition to policy improvement algorithms, other methods for solving MDPs include algorithms for linear programming, but variants of policy improvement are preferred due to speed and ease of implementation [Lit96; Han98; GKP01]. We remark that policy improvement can be viewed as a special linear programming solution method [Lit96; Ber95].

Unfortunately, the worst-case bounds on several implementations of policy improvement are exponential [MC94]. The exponential lower bounds have been shown for those policy improvement algorithms that in each iteration attempt to pick a single most promising state to improve, and are established on plausible heuristics for such a choice, such as looking ahead one or a constant number of steps. Let us call these 'selective' policy improvement. In a sense, the bounds imply that attempting to be smart about choosing which state to improve can lead to an exponentially long path to the optimal. On the other hand, the policy improvement technique is naturally implemented in a manner in which all states are examined, and any improvable state changes action. We will refer to this variation as policy iteration (PI) (see Section 2.1). It is known that PI is no worse than pseudo-polynomial[1] time [Ber95], while the exponential lower bounds on selective algorithms apply irrespective of the number representation [Lit96]. This suggests that the constraints on how PI advances may be inherently different than those for the selective policy improvement algorithms, and leaves hope for PI. But quantifying the advancement of PI has been difficult. The best upper bound on PI, besides the pseudo-polynomial bound, is also exponential $O(2^n/n)$ [MS99], where n is the number of states and each state has two actions. This upper bound is derived using certain par-

[1] That is, polynomial if the numbers are written in unary.

tial order and monotonicity properties of the policy space. No lower bound other than the trivial $\Omega(n)$ on the number of iterations is known.

This paper describes a measure of advancement for PI that offers an explanation of why PI is fast. While we don't derive polynomial bounds for PI on general MDPs, we give the first polynomial bounds for PI on many significant subclasses of MDPs and related problems. Fig. 1 shows the MDP problems in context. The problems get roughly harder from left to right. Our results apply to the enclosed problems. Let m be the number of actions, and W be the largest number in magnitude in the problem description[2]. We give an $O(n^2 \log nW)$ bound on the number of iterations of PI on MDP graphs that have a special structure: there exists a certain state b such that any sequence of visited states under any policy repeats state b before repeating another state. Each iteration takes $O(m)$ time or $O(m+n^2)$ time depending on the edges (Section 4.1). For the rest of the enclosed problems, we show that a variant of PI is polynomial. We discuss these problems next.

The general MDP problem can be formulated as a linear program (LP) in which the feasibility constraints have at most *three* variables per inequality [Mad02]. The subclasses of the MDPs for which we give polynomial policy improvement can be summarized as those for which the corresponding LPs require at most *two* variables per inequality (Section 4.4). We use the names MDP(2) and MDP(3) to differentiate. In the LP formulation of the MDP, finding the feasibility region under the system of inequalities is sufficient to solve the problem. We have shown that, conversely, an extension of policy iteration solves the two variable per inequality linear feasibility (TVPI) problem and have given a polynomial bound for one such type of algorithm. The way policy improvement solves the TVPI problem is different from previous algorithms, as will be described in the paper. The best bound that we give for policy improvement on MDP(2) is $O(mn^2 \log m \log W)$ and for deterministic MDPs (discounted or average reward) is $O(mn^2 \log m)$, but we expect that the bounds can be improved. In the paper, at times, we will use $\widetilde{O}()$ notation to hide extra $\log()$ factors for simplicity.

A main tool we use is the analysis of Newton's method for finding the zero of a function [Rad92; Mad00]. It was known that policy iteration was a form of Newton's method, but the well-known local quadratic convergence of Newton's method is not useful for deriving polynomial bounds. The analysis given here relates the way the process converges to attributes of the problem size. We point out that our results on deterministic MDPs do not require this analysis.

We begin with the definition of the MDP problems and policy iteration. Then we show how PI is a Newton's method on a single state problem, and describe its analysis that is used in showing polynomial bounds. We then show how this tool can be leveraged to establish polynomial bounds for various policy improvement algorithms on different problems. Complete proofs appear in [Mad02; Mad00].

[2]We make the standard assumption that a fraction such as a probability is represented by the ratio of two integers.

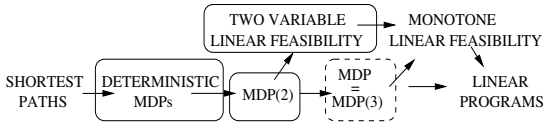

Figure 1: MDP problems in context. Problems are generalized in the direction of the arrows. Our results apply to the enclosed problems.

2 Preliminaries

A Markov decision process (MDP) consists of a set V of n vertices or system states, and for each vertex $v \in V$, a finite set E_v of edge choices or actions. Let m denote the total number of edges. Time is discretized and at each time point t, $t = 0, 1, \cdots$ the system occupies one vertex $v^{(t)}$, which is called the state (or vertex) of the system at time t, and $v^{(0)}$ is the initial state of the system. The means of change of vertex (system state) is edge choice. Each edge can branch out and have one or more end-vertices (Fig. 3a). By choosing edge $e \in E_u$ when the system is in state u, a *reward* of $r(e) \in R$ is obtained and the system transitions to an end-vertex v, with probability $Pr(e, v)$, where $\sum_{v \in V} Pr(e, v) = 1$. The effects of edges, *i.e.* the rewards and the transition probabilities, do not change with time, and they are completely specified as part of the problem instance. We will also use shorthand r_e to denote reward of edge e. A *policy* $\mathcal{P}$ is a mapping assigning to each vertex v a single edge choice $\mathcal{P}(v) \in E_v$. The value (vector) of a policy, denoted $\mathcal{V}_\mathcal{P}$ is a vector of n values, where $\mathcal{V}_\mathcal{P}[i]$ is the *value* of vertex v_i under policy $\mathcal{P}$, defined as the expectation of total reward $\sum_{t=0}^{\infty} r(\mathcal{P}(v^{(t)}))$, when $v^{(0)} = v_i$. We assume here that $\mathcal{V}_\mathcal{P}[i]$ is bounded and well behaved for any policy $\mathcal{P}$ and initial vertex[3].

Define the *optimal value* of a vertex v, denoted x^*_v, to be the maximum value of vertex v over all policies. A desirable and simplifying property of MDPs is that there exists an *optimal policy* which simultaneously maximizes the value of all vertices [Ber95]. Therefore the MDP problem is to find an optimal policy or compute the optimal value of all vertices.

2.1 Policy Iteration

The generic policy improvement procedure is given in Fig. 2. Evaluating a policy means finding its corresponding value vector. Policy improvement algorithms differ on how they improve the policy. In a common technique which we will refer to as (classic) policy iteration (PI), the improvement is done in 'parallel' as follows: each vertex u picks its

[3]Policy improvement algorithms can be extended to discover ill-defined problems as discussed later. Whenever a discount is used (i.e. maximize expected $\sum_{t=0}^{\infty} \beta^t r_{\mathcal{P}(v^{(t)})}$, with $\beta < 1$), the problem is well-defined, and a discount is modeled by each edge having a transition probability to an absorbing zero-reward state.