An Evolutionary Random Policy Search Algorithm for Solving Markov Decision Processes

Jiaqiao Hu

Department of Electrical and Computer Engineering &
Institute for Systems Research

jqhu@glue.umd.edu

May , 2005

UNIVERSITY OF MARYLAND 1856
A. JAMES CLARK SCHOOL OF ENGINEERING

The Institute for Systems Research

Problem Setting - MDPs

- MDP is defined as a tuple (X, A, P, R, α)
  - finite state space X ; x_t : state at time t
  - general action space A
  - transition probabilities $P_{x,y}(a)$
  - non-negative bounded one-stage cost function $R(x,a)$
  - discount factor $\alpha \in (0,1)$
- Objective: find a stationary policy π^* to minimize the infinite-horizon expected total discounted cost

$$J^{\pi^*}(x) = \inf_{\pi \in \Pi} J^{\pi}(x), \quad \text{where}$$

$$J^{\pi}(x) = E\left[\sum_{t=0}^{\infty} \alpha^t R(x_t, \pi(x_t)) \,\middle|\, x_0 = x \right], \forall\, x \in X$$

Solution Methods Overview - MDPs

- **Standard techniques**
  - **value iteration** (**VI**): compute a sequence of functions $\{J_k : k = 0,1,\ldots\}$ via the recursion

$$J_{k+1}(x) = \min_{a\in A}[R(x,a) + \alpha \sum_{y\in X} P_{x,y}(a) J_k(y)],\ \forall x \in X.$$

  - **policy iteration** (**PI**)
    - policy evaluation

$$J^{\pi_k}(x) = R(x,\pi(x)) + \alpha \sum_{y\in X} P_{x,y}(\pi(x)) J^{\pi_k}(y),\ \forall x \in X$$

    - policy improvement

$$\pi_{k+1}(x) = \arg\min_{a\in A}[R(x,a) + \alpha \sum_{y\in X} P_{x,y}(a) J^{\pi_k}(y)],\ \forall x \in X$$

  - **modified policy iteration** (**MPI**)

Solution Methods Overview - MDPs

- **State space reduction techniques**
  - **state aggregation** (Bertsekas and Castanon 1989)
  - **value function approximation** (Bellman et al. 1963; Tsitsiklis and Van Roy 1994; Trick and Zin 1997; De Farias and Van Roy 2003 etc.)
  - **randomization** (Rust 1997)
  - **simulation-based approaches**
    - ***temporal difference*** (Sutton 1988)
    - ***Q-learning*** (Watkins 1989)
    - ***other techniques*** (Chang et al. 2003; Chang et al. 2004; Mannor et al. 2003)
- **Action space reduction techniques**
  - **action elimination procedures** (McQueen 1966; Even-Dar et al. 2003)

ERPS for Solving MDPs

- **Motivation:** solving general MDPs is at least as hard as solving non-linear optimization problems

- **Methodology:** use global optimization strategies to improve the performance of the current MDP solution techniques

  - **evolutionary, population-based approaches directly searching the policy space**

    * *complement state space reduction techniques*

    * *avoid optimization over the entire action space*

    * *robustness*

ERPS for Solving MDPs

- **Target problems:** ***X*** small; ***A*** large or uncountable
  - **examples :** queueing control; job-shop scheduling etc.
- **Key steps:** For a set of policies $\Lambda_k = \{\pi_1, \pi_2, \ldots, \pi_N\}$
  - generate elite policy π_*^{k+1} via Policy Improvement with Cost Swapping (PICS)

$$\pi_*^{k+1}(x) = \arg\min_{u \in \Lambda_k(x)} \{R(x,u) + \alpha \sum_{y \in X} P_{x,y}(u)[\min_{\pi_j \in \Lambda} J^{\pi_j}(y)]\},$$

  - construct the next population of policies Λ_{k+1} based on π_*^{k+1} and random sampling of the entire action space.

ERPS for Solving MDPs

- **Initialization:** select initial population Λ_0 , exploitation probability q_0 , action selection distribution P, set $k=0$.
- **Repeat until a specified stopping rule is satisfied:**
  - generate elite policy π_*^{k+1} via **PICS**
  - generate other policies in the next population: based on $U_j \sim U(0,1)$ i.i.d.,
    - * if $\boldsymbol{U_j} \le q_0$ (exploitation)

      choose $\pi_j^{k+1}(x)$ *from small neighborhood of* $\pi_*^{k+1}(x)$,
    - * else (exploration)

      choose $\pi_j^{k+1}(x)$ *according to* P.
  - Construct $\Lambda_{k+1}=\{\pi_*^{k+1}, \pi_2^{k+1}, \ldots, \pi_N^{k+1}\}$, Set $k = k+1$.

ERPS for Solving MDPs

- **Properties**
  - avoid optimization over the entire action space
  - improve a population of policies, i.e.,

$$J^{\pi_*^{k+1}}(x) \le \min_{\pi_j^k \in \Lambda_k} J^{\pi_j^k}(x), \qquad \forall x \in X.$$

  - monotonicity among elite policies

$$J^{\pi_*^{k+1}}(x) \le J^{\pi_*^k}(x), \quad \forall x \in X, \forall k = 0,1,\ldots$$

  - convergence w.p.1. to optimal value function

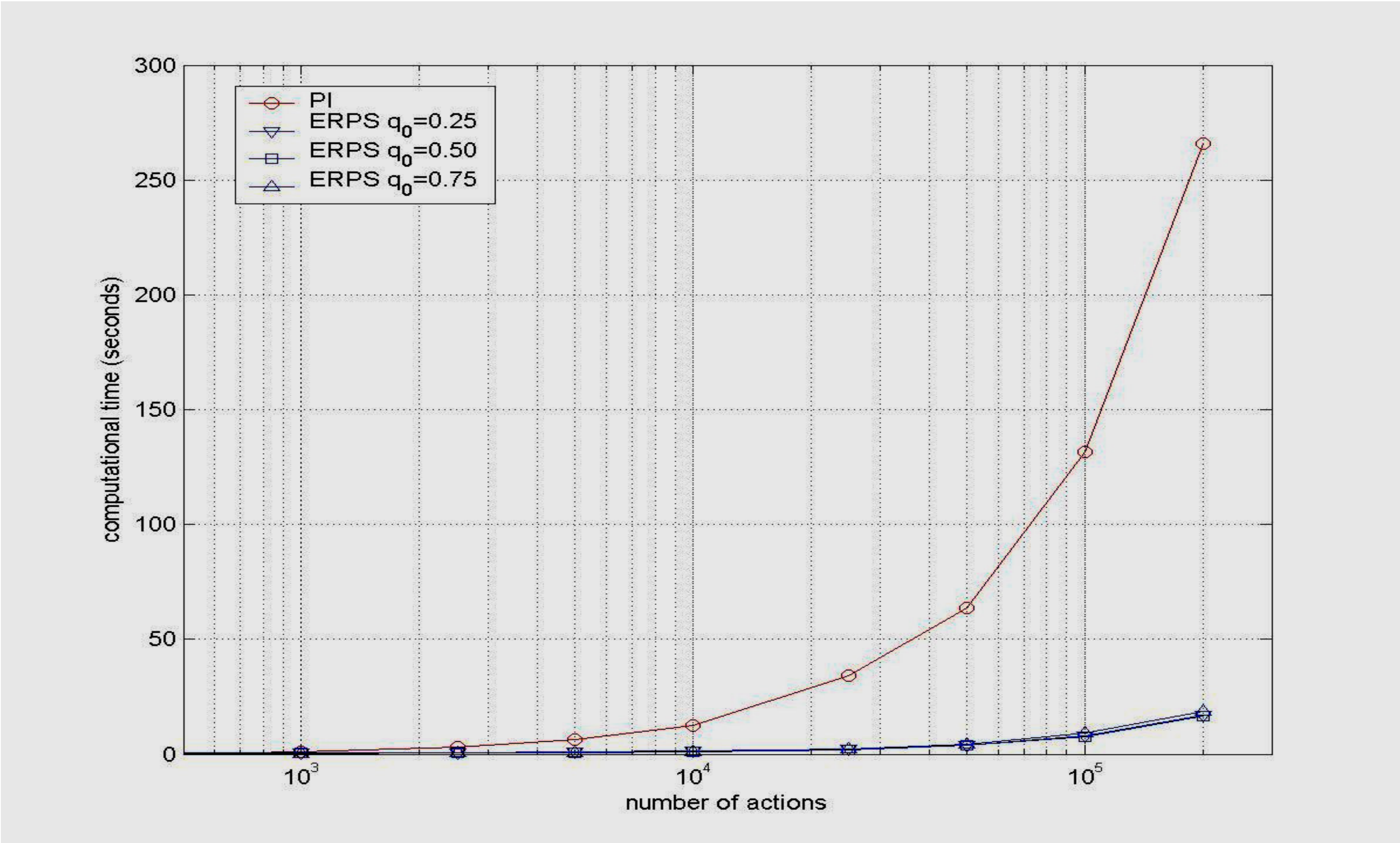
$$\lim_{k \to \infty} J^{\pi_*^k}(x) = J^*(x), \ \forall x \in X \ \text{ w.p.1.}$$

ERPS for Solving MDPs

- **Numerical examples:** single-server queue
  - finite buffer size L=48 (state space size 50)
  - at most one arrival each period of time with probability 0.2, service completion probability $a \in [0,1]$.
  - state x_t= # jobs in system
  - action: service completion probability a
  - discount factor α=0.98
  - one-stage cost function: $R(x,a) = x+50\,a^2$
  - population size N=10

ERPS for Solving MDPs

Running time required for PI and ERPS to find optimal solution as a function of the size of the action space



ERPS for Solving MDPs

Results for continuous action space $A=[0,1]$. relerr= $\|J - J^*\|_\infty / \|J^*\|_\infty$

algorithms	parameters	Avg. time (std err)	Mean relerr (std err)
ERPS r=1/4000	q_0=0.50	2.27 (0.09)	6.41e-13 (7.07e-14)
	q_0=0.75	2.92 (0.08)	1.92e-13 (2.69e-14)
ERPS r=1/8000	q_0=0.50	2.91 (0.10)	1.08e-13 (1.59e-14)
	q_0=0.75	3.50 (0.11)	6.84e-14 (1.03e-14)
ERPS r=1/16000	q_0=0.50	3.25 (0.10)	3.06e-14 (4.56e-15)
	q_0=0.75	3.68 (0.10)	1.89e-14 (2.50e-15)
PI	h=1/16000	23 (N/A)	4.74e-10 (N/A)
	h=1/32000	47 (N/A)	9.52e-11 (N/A)
	h=1/128000	191 (N/A)	6.12e-12 (N/A)
	h=1/512000	781 (N/A)	3.96e-13 (N/A)

ERPS for Solving MDPs

- **Conclusions**
  - evolutionary, population-based approach with guaranteed theoretical convergence
  - much lower computational time than standard PI
  - parallel computing
- **Open problem**
  - performance relies on neighborhood structure, action selection distribution P , and exploitation probability q_0.