A Least Squares Temporal Difference Actor-Critic Algorithm with Applications to Warehouse Management *

Reza Moazzez Estanjini† Keyong Li‡ Ioannis Ch. Paschalidis§

July 2010
Revised: May 2011, January 2012

(submitted, *Naval Research Logistics*)

Abstract

This paper develops a new approximate dynamic programming algorithm for Markov decision problems and applies it to a vehicle dispatching problem arising in warehouse management. The algorithm is of the actor-critic type and uses a least squares temporal difference learning method. It operates on a sample-path of the system and optimizes the policy within a pre-specified class parameterized by a parsimonious set of parameters. The method is applicable to a partially observable Markov decision process setting where the measurements of state variables are potentially corrupted and the cost is only observed through the imperfect state observations. We show that under reasonable assumptions, the algorithm converges to a locally optimal parameter set. We also show that the imperfect cost observations do not affect the policy and the algorithm minimizes the true expected cost.

In the warehouse application, the problem is to dispatch sensor-equipped forklifts in order to minimize operating costs involving product movement delays and forklift maintenance. We consider instances where standard dynamic programming is computationally intractable. Simulation results confirm the theoretical claims of the paper and show that our algorithm converges more smoothly than earlier actor-critic algorithms while substantially outperforming heuristics used in practice.

Keywords: Markov decision processes; partial observability, approximate dynamic programming, actor-critic algorithms, warehouse management, vehicle routing.

1 Introduction

Markov Decision Processes (MDPs) are widely used for modeling sequential decision-making problems that arise in engineering, economics, computer science, and social sciences. However, it is

*Research partially supported by the NSF under grant EFRI-0735974, by the DOE under grant DE-FG52-06NA27490, by the ARO under grant W911NF-09-1-0492, and by the ODDR&E MURI10 program under grant N00014-10-1-0952.

†Division of Systems Eng., Boston University, reza2007@bu.edu.

‡Center for Information & Systems Eng., Boston University, likeyong@ieee.org.

§Corresponding author. Department of Electrical and Computer Engineering and Division of Systems Engineering, Boston University, Boston, MA 02215, yannisp@bu.edu, http://ionia.bu.edu/.

well-known that many such problems suffer from the curse of dimensionality, which makes it difficult to obtain practical solutions for realistically-sized instances. In some cases, the system of interest is so complex that it is not even feasible to determine all the MDP model parameters explicitly. To overcome the difficulty, earlier work has focused on simulation-based *approximate dynamic programming* (see, [33, 17]), treatments of which have appeared under the terms *reinforcement learning* [39] and *neuro-dynamic programming* [10]. *Actor-critic algorithms* ([4]), which we adopt in this work, fall within this broad category.

Actor-critic algorithms are typically used to optimize some *Randomized Stationary Policy (RSP)* using policy gradient estimation. RSPs are parameterized by a parsimonious set of parameters and the objective is to optimize the policy (hence, the use of gradient methods) with respect to these parameters. To that end, one needs to estimate appropriate policy gradients, which can be done using learning methods and is much more efficient than estimating/computing a cost-to-go function in the entire state-action space. Many different versions of actor-critic algorithms have been proposed, e.g., [11, 12, 18, 21, 27, 29, 35], and they have been shown to be quite efficient for various applications (e.g., in robotics [31] and robot navigation [37], power management of wireless transmitters[7], biology [23], and optimal bidding for electricity generation companies [20]).

A particularly attractive design of the actor-critic architecture was proposed in [25], where the critic estimates the policy gradient using sequential observations from a sample path while the actor updates the policy at the same time, although at a slower time-scale. It was proved that the estimate of the critic tracks the slowly varying policy asymptotically under suitable conditions. A center piece of these conditions is a relationship between the actor step-size and the critic step-size, which will be discussed later.

The critic of [25] uses first-order variants of the *Temporal Difference (TD)* algorithm (TD(1) and TD(λ)). However, it has been shown that the least squares TD methods (LSTD and LSPE) are superior in terms of convergence rate; see [9, 13, 15, 24, 28]. LSTD and LSPE were first proposed for discounted cost problems in [15] and [8], respectively. Later, [24] showed that the convergence rate of LSTD is optimal. More recently, [42] proved the convergence of both LSTD and LSPE for the average cost problem and showed that both LSTD and LSPE enjoy the optimal convergence rate. Their results clearly demonstrated that LSTD and LSPE converge much faster and more reliably than TD(1) and TD(λ).

In this paper, we focus on average cost problems with finite state and control space (see [34] for an introduction on how to solve average cost/reward problems). Motivated by these findings, we propose an actor-critic algorithm that adopts the least squares learning methods while, at the same time, maintains the concurrent update architecture of the actor and the critic. Note that [32] also used LSTD in an actor-critic method mostly applicable to episodic problems, but the actor there had to wait for the critic to converge before making each policy update. Further, [32] provided an approach to take advantage of the freedom of augmenting the set of basis functions for Q-value approximation; something which is not considered in the present paper. An interesting future direction could be the integration of the work in this paper with elements of [32].

In addition to the use of LSTD, we demonstrate how the proposed algorithm can be modified to apply to a *Partially Observable Markov Decision Process (POMDP)* setting. Such a setting is rather natural in the particular warehouse management application we study. Partial observability in reinforcement learning algorithms has been studied before (see, for example, [26, 22, 5, 6, 1, 38, 40]). [40] first proposed the use of actor-critic algorithms for POMDPs with perfectly observable cost

and showed that the gradient estimation can be done by making the critic compute a value function (conditional mean of the true value function) which does not depend on the states. We built on this result and consider the case where the cost is also not perfectly observable. We show that under some separability assumptions, the corresponding actor-critic algorithm converges to the locally optimal solution despite the imperfect cost information.

The methodological work in this paper is motivated by and applied to the problem of intelligently dispatching forklifts in a warehouse. We are interested in sensor-equipped forklifts that can determine their status (condition monitoring), their physical location in the warehouse, and continuously transmit this information wirelessly to some central gateway. Leveraging the information made available, we design a forklift dispatching algorithm to minimize operational costs associated with product delivery delays and forklift maintenance. In this application, the POMDP setting is appropriate as the forklift localization and condition monitoring system is subject to estimation errors.

In what follows, formulation of the MDP problem is in Section 2. The LSTD actor-critic algorithm with concurrent updates is presented in Section 3. In Section 4, we propose the LSTD *Partially Observable* (LSTD-PO) actor-critic algorithm. The warehouse management application is then discussed in Section 5, where the effectiveness of the proposed algorithm is demonstrated. Conclusions are drawn in Section 6.

Notation: We use bold letters to denote vectors and matrices; typically vectors are lower case and matrices upper case. Vectors are assumed to be column vectors unless explicitly stated otherwise. For economy of space we write $\mathbf{x} = (x_1, \ldots, x_n)$ for the n-dimensional column vector $\mathbf{x} \in \mathbb{R}^n$. Transpose is denoted by prime. For any $m \times n$ matrix $\mathbf{A}$, with rows $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{R}^n$, $\mathbf{v}(\mathbf{A})$ will denote the column vector $(\mathbf{a}_1, \ldots, \mathbf{a}_m)$. $\|\cdot\|$ will denote the Euclidean norm and $\|\cdot\|_{\boldsymbol{\theta}}$ a special norm in the MDP state-action space we will introduce. $\mathbf{0}$ denotes a vector or matrix with all components set to zero and $\mathbf{I}$ is the identity matrix.

2 Problem formulation

The problem considered here is rather standard and we will state it briefly.

2.1 Average-cost MDP

Consider an average-cost MDP with finite state and action spaces. Let k denote time, $\mathbb{X}$ denote the state space, and $\mathbb{U}$ denote the action space. Consider $\mathbf{x} \in \mathbb{X}$ and $u \in \mathbb{U}$. Let $g(\mathbf{x}, u)$ be the one-step cost. The policy candidates are assumed to belong to a parameterized family of *Randomized Stationary Policies* (RSPs) $\{\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \mid \boldsymbol{\theta} \in \mathbb{R}^n\}$. That is, given a state $\mathbf{x}$ and a parameter $\boldsymbol{\theta}$, the policy applies action u with probability $\mu_{\boldsymbol{\theta}}(u|\mathbf{x})$. The goal is to optimize the average cost over the n-dimensional vector $\boldsymbol{\theta}$.

Let $p(\boldsymbol{\xi}|\mathbf{x}, u)$ denote the state transition probabilities (which are typically not explicitly known); that is, $p(\boldsymbol{\xi}|\mathbf{x}, u)$ is the probability of transitioning to state $\boldsymbol{\xi}$ given that action u is taken while the system is at state $\mathbf{x}$. Under suitable ergodicity conditions, $\{\mathbf{x}_k\}$ and $\{\mathbf{x}_k, u_k\}$ are Markov chains with stationary distributions under a fixed policy. These stationary distributions are denoted by $\pi_{\boldsymbol{\theta}}(\mathbf{x})$ and $\eta_{\boldsymbol{\theta}}(\mathbf{x}, u)$, respectively. The average cost of the MDP is thus well defined and given by

$\bar{\alpha}(\boldsymbol{\theta}) = \sum_{\mathbf{x},u} \eta_{\boldsymbol{\theta}}(\mathbf{x},u) g(\mathbf{x},u)$. We will not elaborate on the ergodicity conditions, except to note that it suffices that the process $\{\mathbf{x}_k\}$ is irreducible and aperiodic given any $\boldsymbol{\theta}$, and for any $\mathbf{x} \in \mathbb{X}$ and $u \in \mathbb{U}$,

$$\text{either} \quad \inf_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(u|\mathbf{x}) > 0 \quad \text{or} \quad \mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \equiv 0 \tag{1}$$

(see Theorem 2.6 of [24]). One possible form of RSP that satisfies this assumption uses the Boltzmann-like distribution:

$$\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) = \frac{\chi(\mathbf{x},u) e^{h(u,\mathbf{x},\boldsymbol{\theta})}}{\sum_{a \in \mathbb{U}} \chi(\mathbf{x},u) e^{h(a,\mathbf{x},\boldsymbol{\theta})}}, \tag{2}$$

where $\chi(\mathbf{x},u)$ is nonnegative. Note that it is possible to assign zero to $\chi(\mathbf{x},u)$ for some pairs of $(\mathbf{x},u)$, i.e., exclude certain actions for some state values. The Boltzmann policy offers a nice balance of flexibility and simplicity. Indeed, it turns out to be sufficient for the warehouse vehicle dispatching problem discussed later in the paper. The RSP can be expensive to compute when the action space is large, but in typical applications including the warehouse case, the action space is relatively small.

2.2 Actor-Critic algorithm

With no explicit model of the state transitions but only a sample path denoted by $\{\mathbf{x}_k, u_k\}$, the actor-critic algorithms typically optimize $\boldsymbol{\theta}$ locally in the following way: first, the critic estimates the policy gradient

$$\nabla \bar{\alpha}(\boldsymbol{\theta}) = \left(\frac{\partial \bar{\alpha}(\boldsymbol{\theta})}{\partial \theta_1}, \ldots, \frac{\partial \bar{\alpha}(\boldsymbol{\theta})}{\partial \theta_n}\right) \tag{3}$$

using some type of a *Temporal Difference (TD)* algorithm; then the actor modifies the policy parameter along the gradient direction.

Define the so-called $Q_{\boldsymbol{\theta}}$-value function as any function satisfying the Poisson equation

$$Q_{\boldsymbol{\theta}}(\mathbf{x},u) = g(\mathbf{x},u) - \bar{\alpha}(\boldsymbol{\theta}) + (P_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}})(\mathbf{x},u),$$

where the operator $P_{\boldsymbol{\theta}}$ denotes taking expectation after one transition, namely, $(P_{\boldsymbol{\theta}} Q)(\mathbf{x},u) = \sum_{\boldsymbol{\xi} \in \mathbb{X}, \nu \in \mathbb{U}} \mu_{\boldsymbol{\theta}}(\nu|\boldsymbol{\xi}) p(\boldsymbol{\xi}|\mathbf{x},u) Q(\boldsymbol{\xi},\nu)$. $Q_{\boldsymbol{\theta}}(\mathbf{x},u)$ can be interpreted as the expected future excess cost (on top of the average cost) we incur if we start at state $\mathbf{x}$, apply control u, and then follow the RSP $\mu_{\boldsymbol{\theta}}$. Let now

$$\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x},u) = (\psi^1_{\boldsymbol{\theta}}(\mathbf{x},u), \ldots, \psi^n_{\boldsymbol{\theta}}(\mathbf{x},u)),$$

where

$$\psi^i_{\boldsymbol{\theta}}(\mathbf{x},u) = \frac{\partial}{\partial \theta_i} \ln \mu_{\boldsymbol{\theta}}(u|\mathbf{x}), \ i = 1, \ldots, n.$$

It is assumed that $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x},u)$ is bounded and continuously differentiable, which is true for policies of the form (2). We make the convention $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x},u) = \mathbf{0}$ when $\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) = 0$ for all values of $\boldsymbol{\theta}$ (cf. Assumption (1)).

A key fact underlying the actor-critic algorithm is that the policy gradient can be expressed as (see [25] for more details)

$$\frac{\partial \bar{\alpha}(\boldsymbol{\theta})}{\partial \theta_i} = \langle Q_{\boldsymbol{\theta}}, \psi^i_{\boldsymbol{\theta}} \rangle_{\boldsymbol{\theta}}, \ i = 1, \ldots, n,$$

where the operator $\langle \cdot, \cdot \rangle_{\boldsymbol{\theta}}$ is defined as: for any two functions f_1 and f_2 of $\mathbf{x}$ and u,

$$\langle f_1, f_2 \rangle_{\boldsymbol{\theta}} \stackrel{\triangle}{=} \sum_{\mathbf{x},u} \eta_{\boldsymbol{\theta}}(\mathbf{x},u) f_1(\mathbf{x},u) f_2(\mathbf{x},u). \tag{4}$$

This is very interesting because it suggests an equivalent class of the $Q_{\boldsymbol{\theta}}$-value function in terms of estimating the policy gradient. For any function $f(\mathbf{x},u)$, let $\mathbf{f}$ be its vector representation in $\mathbb{R}^{|\mathbb{X}||\mathbb{U}|}$ — the notations $\mathbf{Q}$ and $\boldsymbol{\psi}^i_{\boldsymbol{\theta}}$ (used next) are instances of this vector representation. Then, the operator (4) defines a inner product in $\mathbb{R}^{|\mathbb{X}||\mathbb{U}|}$. Let $\|\cdot\|_{\boldsymbol{\theta}}$ denote the norm induced by this inner product, i.e., $\|\mathbf{f}\|^2_{\boldsymbol{\theta}} = \langle \mathbf{f}, \mathbf{f} \rangle_{\boldsymbol{\theta}} = \langle f, f \rangle_{\boldsymbol{\theta}}$. Let also $\mathscr{S}_{\boldsymbol{\theta}}$ be the subspace of $\mathbb{R}^{|\mathbb{X}||\mathbb{U}|}$ spanned by the vectors $\boldsymbol{\psi}^i_{\boldsymbol{\theta}}$, $i = 1, \ldots, n$. Next, denote by $\Pi_{\boldsymbol{\theta}}$ the projection with respect to the norm $\|\cdot\|_{\boldsymbol{\theta}}$ onto $\mathscr{S}_{\boldsymbol{\theta}}$, i.e., for any $\mathbf{f} \in \mathbb{R}^{|\mathbb{X}||\mathbb{U}|}$, $\Pi_{\boldsymbol{\theta}}\mathbf{f}$ is the unique vector in $\mathscr{S}_{\boldsymbol{\theta}}$ that minimizes $\|\mathbf{f} - \hat{\mathbf{f}}\|_{\boldsymbol{\theta}}$ over all $\hat{\mathbf{f}} \in \mathscr{S}_{\boldsymbol{\theta}}$. Since for all i

$$\langle Q_{\boldsymbol{\theta}}, \boldsymbol{\psi}^i_{\boldsymbol{\theta}} \rangle_{\boldsymbol{\theta}} = \langle \Pi_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}, \boldsymbol{\psi}^i_{\boldsymbol{\theta}} \rangle_{\boldsymbol{\theta}},$$

it is sufficient to know the projection of $\mathbf{Q}_{\boldsymbol{\theta}}$ onto $\mathscr{S}_{\boldsymbol{\theta}}$ in order to compute $\nabla \bar{\alpha}(\boldsymbol{\theta})$. Consequently, for policy gradient estimation, one may substitute $Q_{\boldsymbol{\theta}}$ with a parametric linear architecture of the following form (see [25]):

$$Q^{\mathbf{r}}_{\boldsymbol{\theta}}(\mathbf{x},u) = \boldsymbol{\psi}'_{\boldsymbol{\theta}}(\mathbf{x},u)\mathbf{r}^*, \qquad \mathbf{r}^* \in \mathbb{R}^n. \tag{5}$$

This dramatically reduces the complexity of learning from the space $\mathbb{R}^{|\mathbb{X}||\mathbb{U}|}$ to the space $\mathbb{R}^n$. Furthermore, the temporal difference algorithms can be used to learn such an $\mathbf{r}^*$ effectively. The elements of $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x},u)$ are understood as basis functions used to develop an approximation of the policy gradient. To sum up the key point of this section: the policy gradient of the average cost MDP can be expressed as

$$\frac{\partial \bar{\alpha}(\boldsymbol{\theta})}{\partial \theta_i} = \langle \boldsymbol{\psi}'_{\boldsymbol{\theta}}(\mathbf{x},u)\mathbf{r}^*, \psi^i_{\boldsymbol{\theta}} \rangle_{\boldsymbol{\theta}}. \tag{6}$$

The actor-critic algorithm aims at finding good approximations of r^*, and consequently of the policy gradient.

3 Actor-Critic algorithm using LSTD

The critic in [25] used either TD(λ) or TD(1). The algorithm we propose uses a least squares TD method (LSTD) instead, as it has been shown to provide far superior performance ([42]).

The actor and the critic updates take place in the course of a simulation of a single sample path of the MDP. Let $\mathbf{x}_k$ denote the state of the system at time k. Let $\mathbf{r}_k$, the iterate for $\mathbf{r}^*$ in (5), be the parameter vector of the critic at time k, $\boldsymbol{\theta}_k$ be the parameter vector of the actor at time k, and $\mathbf{x}_{k+1}$ be the new state, obtained after action u_k is applied when the state is $\mathbf{x}_k$. A new action u_{k+1} is generated according to the RSP corresponding to the actor parameter $\boldsymbol{\theta}_k$ (see [25]). The critic and the actor carry out the following updates where α_k is an estimate of the average cost, $\mathbf{z}_k \in \mathbb{R}^n$ is often called Sutton's eligibility trace [39] which is a weighted sum of the present and past feature vectors obtained from the simulations, $\mathbf{b}_k \in \mathbb{R}^n$ represents the eligibility trace scaled by the drifts of single period reward from the average cost, and $\mathbf{A}_k \in \mathbb{R}^{n \times n}$ is a sample estimate of the matrix formed by $\mathbf{z}_k(\boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_{k+1}, u_{k+1}) - \boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_k, u_k))$.

LSTD Actor-Critic

Initialization:

Set all entries in $\mathbf{z}_0, \alpha_0, \mathbf{A}_0, \mathbf{b}_0$ and $\mathbf{r}_0$ to zeros. Let $\boldsymbol{\theta}_0$ take some initial value, potentially corresponding to a heuristic policy.

Critic:

$$\begin{aligned} \alpha_{k+1} &= \alpha_k + \gamma_k \left[g(\mathbf{x}_k, u_k) - \alpha_k \right], \\ \mathbf{z}_{k+1} &= \lambda \mathbf{z}_k + \boldsymbol{\psi}_{\boldsymbol{\theta}_k}(\mathbf{x}_k, u_k), \\ \mathbf{b}_{k+1} &= \mathbf{b}_k + \gamma_k \left[(g(\mathbf{x}_k, u_k) - \alpha_k)\mathbf{z}_k - \mathbf{b}_k \right], \\ \mathbf{A}_{k+1} &= \mathbf{A}_k + \gamma_k \left[\mathbf{z}_k (\boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_{k+1}, u_{k+1}) - \boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_k, u_k)) - \mathbf{A}_k \right], \end{aligned} \tag{7}$$

where $\lambda \in [0, 1)$, $\gamma_k \stackrel{\triangle}{=} \dfrac{1}{k}$ corresponds to the original LSTD, and finally

$$\mathbf{r}_{k+1} = -\mathbf{A}_k^{-1} \mathbf{b}_k. \tag{8}$$

Actor:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta_k \Gamma(\mathbf{r}_k) \boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_{k+1}, u_{k+1}) \mathbf{r}_k \boldsymbol{\psi}_{\theta_k}(\mathbf{x}_{k+1}, u_{k+1}). \tag{9}$$

In the above, $\{\gamma_k\}$ controls the critic step-size, while $\{\beta_k\}$ and $\Gamma(\mathbf{r})$ control the actor step-size together. An implementation of this algorithm needs to make these choices. The role of $\Gamma(\mathbf{r})$ is mainly to keep the actor updates bounded, and we can for instance use

$$\Gamma(\mathbf{r}) = \begin{cases} \dfrac{D}{||\mathbf{r}||}, & \text{if } ||\mathbf{r}|| > D, \\ 1, & \text{otherwise,} \end{cases}$$

for some $D > 0$. $\{\beta_k\}$ is a deterministic and non-increasing sequence for which we need to have

$$\sum_k \beta_k = \infty, \quad \sum_k \beta_k^2 < \infty, \quad \lim_{k \to \infty} \frac{\beta_k}{\gamma_k} = 0, \tag{10}$$

An example of $\{\beta_k\}$ satisfying Eq. (10) is

$$\beta_k = \frac{c}{k \, \ln k}, \quad k > 1,$$

where $c > 0$ is a constant parameter.

Note that the critic update above is essentially the standard LSTD [42], and the actor update is the same as that of [25]. However, the nontrivial task here is to show that the combination works. Specifically, we need to show that the LSTD critic also converges despite policy updates of the actor, and consequently the actor converges as well. The conclusion is stated in the following theorem, while the proof is given in Appendix A.

Theorem 3.1 *[Convergence] For the LSTD actor-critic with some step-size sequence $\{\beta_k\}$ satisfying (10) and for any $\epsilon > 0$, there exists some λ sufficiently close to 1 such that $\liminf_k ||\nabla \bar{\alpha}(\boldsymbol{\theta}_k)|| < \epsilon$ w.p.1. That is, $\boldsymbol{\theta}_k$ visits an arbitrary neighborhood of a stationary point infinitely often.*

4 Actor-Critic algorithms for POMDPs with indirectly observed cost

A *Partially Observable Markov Decision Process* (POMDP) is a generalization of an MDP in which the system dynamics follow an MDP but the agent cannot directly observe the underlying state. Instead, it may maintain a probability distribution over the set of possible states, based on a set of observations. The POMDP framework is general enough to model a variety of real-world sequential decision processes (for a review, see [16]).

It was shown in [40] that if the cost of a POMDP is perfectly observable, then the analytical results on TD and actor-critic algorithms can be transferred without difficulty. In this section, the application of the LSTD actor-critic algorithm is extended to the case of a POMDP where, in addition to the state being partially observable, the critic observes the cost only through its (noisy) observations of the state. To the best of our knowledge, there is no previous work in the literature that considers such a POMDP model in an actor-critic framework. There can be a number of reasons why the cost is not directly observable. For example, the wear of an equipment can be precisely known only at the time of maintenance, not at the time of operation. However, one may have a less precise but more timely estimate of this cost based on the usage of the equipment. In this section, and under reasonable assumptions, the actor-critic algorithms are modified to overcome the imperfect knowledge of costs. An example scenario is the warehouse vehicle dispatching problem we tackle later.

Consider the formulation introduced in Section 2 and suppose the state $\mathbf{x}$ has a bijective representation $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M)})$ where $\mathbf{x}^{(1)} \in \mathbb{X}^{(1)}, \mathbf{x}^{(2)} \in \mathbb{X}^{(2)}, \ldots, \mathbf{x}^{(M)} \in \mathbb{X}^{(M)}$. Of course, the non-trivial case $M > 1$ is of interest, especially where the size of the problem grows mostly due to M, while the cardinality of each individual $\mathbb{X}^{(l)}$ remains moderate. For the observation model, to which we will also be referring to as the *noise model*, assume that an observation $\mathbf{y}$ is generated from the true state $\mathbf{x}$ according to some known probability distribution $p(\mathbf{y}|\mathbf{x})$ represented as a matrix $\mathbf{P}_{\mathbf{y}|\mathbf{x}}$ (with rows and columns indexed by the possible values of $\mathbf{x}$ and $\mathbf{y}$, respectively). It is further assumed that the observations belong to the state space, i.e., $\mathbf{y}_k$ is represented by $(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(M)})$ where $\mathbf{y}^{(l)} \in \mathbb{X}^{(l)},\ l = 1, \ldots, M$. Note that the partial observability here is in the sense that observations are subject to noise and errors. Furthermore, it is assumed that the cost is *separable* as indicated in the following assumption.

Assumption A
(i) **(Separability)** *The cost function can be written as* $g(\mathbf{x}, u) = g_1(\mathbf{x}^{(1)}) + g_2(\mathbf{x}^{(2)}) + \cdots + g_M(\mathbf{x}^{(M)}) + g_U(u)$.
(ii) The noise model can also be expressed as M *independent distributions represented by full-rank matrices* $\mathbf{P}_{\mathbf{y}^{(l)}|\mathbf{x}^{(l)}},\ l = 1, \ldots, M$.

Note that it is realistic to assume the invertibility of $\mathbf{P}_{\mathbf{y}^{(l)}|\mathbf{x}^{(l)}}$, because in most applications, the quality of state observations is reasonably good, hence, $\mathbf{P}_{\mathbf{y}^{(l)}|\mathbf{x}^{(l)}}$ is close to the identity matrix. The RSP in the POMDP case is chosen to be a function of the observation; that is, $\mu_{\boldsymbol{\theta}}(u_k|\mathbf{y}_k)$. This form of RSP can be accommodated by augmenting the state to $\{\mathbf{x}_k, \mathbf{y}_k\}$. As shown in [40], the actor-critic algorithm still converges to the optimal policy if the cost is perfectly observed. For the case where the cost is not perfectly observable, we introduce the following LSTD *Partially Observable* (LSTD-PO) actor-critic algorithm.

LSTD-PO Actor-Critic for POMDP with Indirectly Observed Cost

Initialization:

Set $\mathbf{z}_0, \alpha_0, \mathbf{A}_0, \mathbf{b}_0$ and $\mathbf{r}_0$ to zeros. Let $\boldsymbol{\theta}_0$ take some initial value, potentially corresponding to a heuristic policy.

Critic:

$$\begin{aligned} \alpha_{k+1} &= \alpha_k + \gamma_k \left[\tilde{g}(\mathbf{y}_k, u_k) - \alpha_k\right], \\ \mathbf{z}_{k+1} &= \lambda \mathbf{z}_k + \boldsymbol{\psi}_{\boldsymbol{\theta}_k}(\mathbf{y}_k, u_k), \\ \mathbf{b}_{k+1} &= \mathbf{b}_k + \gamma_k \left[(\tilde{g}(\mathbf{y}_k, u_k) - \alpha_k)\mathbf{z}_k - \mathbf{b}_k\right], \\ \mathbf{A}_{k+1} &= \mathbf{A}_k + \gamma_k \left[\mathbf{z}_k(\boldsymbol{\psi}'_{\boldsymbol{\theta}_k}(\mathbf{y}_{k+1}, u_{k+1}) - \boldsymbol{\psi}'_{\boldsymbol{\theta}_k}(\mathbf{y}_k, u_k)) - \mathbf{A}_k\right], \\ \mathbf{r}_{k+1} &= -\mathbf{A}_k^{-1}\mathbf{b}_k. \end{aligned} \tag{11}$$

In the above, $\lambda \in [0, 1)$, $\gamma_k \stackrel{\triangle}{=} \dfrac{1}{k}$, and

$$\tilde{g}(\mathbf{y}, u) = \sum_{l=1}^{M} \sum_{\mathbf{x}^{(l)}} \nu^{(l)}_{I(\mathbf{x}^{(l)}), I(\mathbf{y}^{(l)})} g_l(\mathbf{x}^{(l)}) + g_U(u), \tag{12}$$

where $I(\cdot)$ is a function that returns the index of a given element $\mathbf{x}^{(l)}$ in the component space $\mathbb{X}^{(l)}$ (consistently with the ordering of the rows and columns of $\mathbf{P}_{\mathbf{y}^{(l)}|\mathbf{x}^{(l)}}$), and $\nu^{(l)}_{i,j}$ is the element located on the ith row and jth column of $\mathbf{P}^{-1}_{\mathbf{y}^{(l)}|\mathbf{x}^{(l)}}$. For convenience, from now on the notation $\nu^{(l)}_{\mathbf{x}^{(l)}, \mathbf{y}^{(l)}}$ will be used for $\nu^{(l)}_{I(\mathbf{x}^{(l)}), I(\mathbf{y}^{(l)})}$, and $\tilde{g}$ will be called the *cost proxy*. As will be shown, this cost proxy averages to the same value as the true cost.

Actor:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta_k \Gamma(\mathbf{r}_k) \boldsymbol{\psi}'_{\boldsymbol{\theta}_k}(\mathbf{y}_{k+1}, u_{k+1}) \mathbf{r}'_k \boldsymbol{\psi}_{\boldsymbol{\theta}_k}(\mathbf{y}_{k+1}, u_{k+1}). \tag{13}$$

The step size sequences follow the same rules as in the previous section. Note that the only difference here is the introduction of the cost proxy.

Theorem 4.1 *For any policy parameter $\boldsymbol{\theta}$, the average true cost $\bar{\alpha}_{\boldsymbol{\theta}}$ equals the average cost proxy*

$$\tilde{\alpha}_{\boldsymbol{\theta}} = \sum_{\mathbf{y}, u} \tilde{\eta}_{\boldsymbol{\theta}}(\mathbf{y}, u) \tilde{g}(\mathbf{y}, u),$$

where $\tilde{\eta}_{\boldsymbol{\theta}}(\mathbf{y}, u)$ is the stationary distribution of the process $\{\mathbf{y}_k, u_k\}$.

See Appendix B for the proof.

Remarks :

1. We should clarify that the average true cost $\bar{\alpha}_{\boldsymbol{\theta}}$ is the average cost resulting from the RSP $\mu_{\boldsymbol{\theta}}(u|\mathbf{y})$ applied to the "noisy" version of the state and is not equal to the average cost of an RSP $\mu_{\boldsymbol{\theta}}(u|\mathbf{x})$ that has access to the true state. Clearly the optimal (over $\boldsymbol{\theta}$) former RSP is suboptimal to the optimal latter RSP.

2. We also note that the proposed LSTD-PO actor-critic algorithm optimizes the average cost proxy $\tilde{\alpha}_{\boldsymbol{\theta}}$ in the same way the original LSTD actor-critic algorithm optimizes the average true cost. Then, since according to the above theorem these costs coincide for every $\boldsymbol{\theta}$, it follows that the LSTD-PO actor-critic optimizes the average true cost.

3. One possible technical concern here is whether the core iteration of the LSTD-PO critic (defined by (11)) converges like that of the normal LSTD critic (defined by (7)) does. To address this concern, note that (11) differs from (7) in exactly two ways: first, (11) uses the sample path of $\mathbf{y}$ instead of $\mathbf{x}$; and second, (11) substituted the true cost ($\bar{\alpha}$) with the cost proxy ($\tilde{\alpha}$). The first difference could affect the convergence of the algorithm if the $\mathbf{y}$-based process has a different ergodicity property. But that is not possible because $\mathbf{y}$ is generated by $\mathbf{x}$ with a full-rank probability transition matrix — consequently the σ-algebra of $\mathbf{y}$ is topologically equivalent to that of $\mathbf{x}$, and the distribution of $\mathbf{y}$ stabilizes whenever that of $\mathbf{x}$ stabilizes. On the other hand, the second difference turns out not really a difference, because Theorem 4.1 shows $\tilde{\alpha} = \bar{\alpha}$ for any value of $\boldsymbol{\theta}$. In conclusion, we do not need to give a separate proof for the convergence of the LSTD-PO actor-critic.

It should be emphasized that the primary role of *separability* (Assumption A(i)) is to allow the LSTD-PO algorithm be implemented with mild computation cost. If the state space is not separable, then a huge matrix $\mathbf{P}_{\mathbf{y}|\mathbf{x}}$ has to be inverted in lieu of M smaller matrices $\mathbf{P}_{\mathbf{y}^{(l)}|\mathbf{x}^{(l)}}$. Even in the case that one of the latter matrices is relatively large, it will typically be sparse, which makes it appropriate to store and compute its inverse using LU factorization. Very briefly, LU factorization is a known method to solve a general system of linear equations of the form $\mathbf{Ax} = \mathbf{b}$. It factors a non-singular matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ as $\mathbf{A} = \mathbf{PLU}$ where $\mathbf{P} \in \mathbb{R}^{n\times n}$ is a permutation matrix, $\mathbf{L} \in \mathbb{R}^{n\times n}$ is unit lower triangular, and $\mathbf{U} \in \mathbb{R}^{n\times n}$ is upper triangular. The standard algorithm for computing an LU factorization is *Gaussian elimination with partial pivoting* or *Gaussian elimination with row pivoting* ([14]). The cost is $\frac{2n^3}{3}$ *flops* if no structure in $\mathbf{A}$ is exploited where a flop is defined as one addition, or subtraction, or multiplication, or division of two floating-point numbers. To compute $\mathbf{A}^{-1}$, we can solve the equations $\mathbf{Ax}_i = \mathbf{e}_i$, where $\mathbf{x}_i$ is the ith column of $\mathbf{A}^{-1}$, and $\mathbf{e}_i$ is the ith unit vector, resulting in a total cost of $\frac{8n^3}{3}$ flops. However, if $\mathbf{A}$ is sparse, computing the LU factorization is less costly and in many cases the cost grows approximately linearly with n when n is large.

Also note that LSTD-PO requires a single column from each inverse matrix of the noise model at each iteration (the columns corresponding to the observed state), and these could be computed using the technique above. Thus, not only can the computational cost be dissipated over the lifetime of the learning algorithm, but some columns of the inverse might not be computed at all if the corresponding states are never visited. As a final note on computational matters, efficient techniques for storing the sparse matrix can be found in [19, 36] and [3].

5 Forklift dispatching in warehouse management

In this section, we demonstrate how the proposed LSTD actor-critic can be used effectively in dealing with a real-world problem (forklift dispatching in warehouses). Our model of the warehouse described in this section is based on an on-site visit and discussion with the forklift manufacturer and the warehouse managers. Simulation is used to evaluate the efficiency of the proposed algorithm.

5.1 Setup

Consider a warehouse with multiple items being supplied and demanded dynamically. The items come in and out of the warehouse through a central depot. After coming in, they are first stored

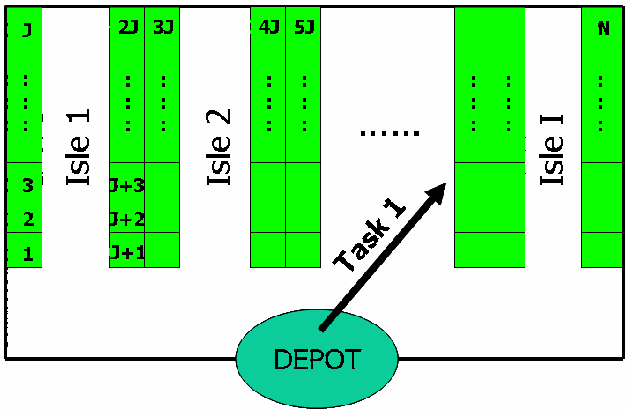

Figure 1: Schematic top-view of the warehouse and an illustration of Task 1.

at some vertically stacked reserve locations, and then moved to pick-up locations that are on the ground level and are specific for each item. Fig. 1 and Fig. 2 show a schematic top-view of the warehouse and a side-view of an isle in the warehouse, respectively.

Three types of tasks are performed in the warehouse:

- **Task 1**: Moving unloaded items from the depot to the reserve locations.
- **Task 2**: Moving items from the reserve locations to fill the corresponding pick-up locations.
- **Task 3**: Moving demanded items from the pick-up locations to the depot.

The forklifts perform Tasks 1 and 2, and Task 3 is done by a different type of vehicle (called pallet-truck). Any item in the warehouse must go through a complete cycle of tasks in the above order (i.e., Task 1 $\rightarrow$ Task 2 $\rightarrow$ Task 3). Figs. 1, 2 and 3 illustrate Tasks 1 and 2 and an overview of all tasks in the warehouse, respectively.

The operation of pallet trucks is simple, rigid, and with no substantial opportunity for optimization. Only the dispatching of the forklifts is considered here, with the goal of moving items through the cycle as efficiently as possible. In addition to Tasks 1 and 2 described above, the forklift may also be commanded to go to the battery/maintenance shop or simply idle. Intuitively, the factors that need to be considered include the urgency of the demand for specific items, the urgency of unloading the supplied items and freeing the suppliers' trucks, traffic congestion experienced by the forklifts, and the "health" of the forklifts (e.g., battery status, or other indications from a machine condition monitoring system). However, with a large number of items that have different values, demand patterns and other contextual variables, forklift dispatching is a high-dimensional dynamic decision problem.

The problem is naturally formulated as an average-cost infinite-horizon MDP. Let N be the number of items, I be the number of isles, and F be the number of forklifts in the warehouse. The state of the system includes:

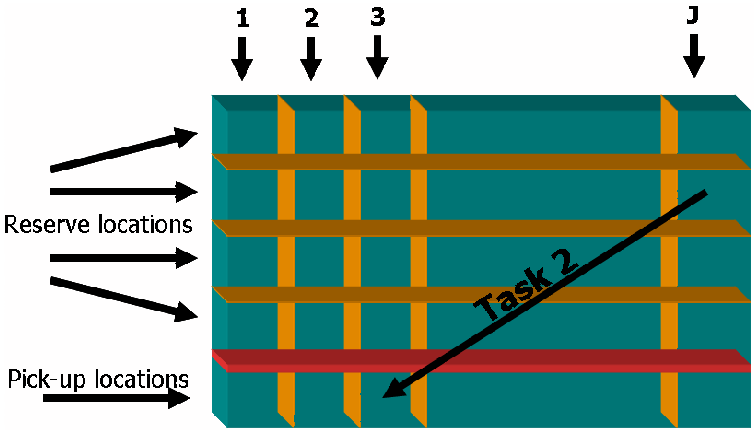

Figure 2: Schematic side-view of an isle in the warehouse and an illustration of Task 2.

- p_i: the level of item i in its pick-up location $(i = 1, \ldots, N)$,
- d_j: the level of items associated with isle j waiting in the depot $(j = 1, \ldots, I)$,
- l_k: the location of forklift k, $\quad (k = 1, \ldots, F)$,
- s_k: the status of forklift k, $\quad (k = 1, \ldots, F)$,
- h_k: health of forklift k, $\quad (k = 1, \ldots, F)$.

The capacities of pick up locations and of the depot are assumed finite, each p_i takes values from a discrete set $\{-P, -P+1, \ldots, -1, 0, 1, \ldots, P-1, P\}$, and each d_j takes values from a discrete set $\{0, 1, \ldots, Q-1, Q\}$. Specifically, $p_i < 0$ indicates that $|p_i|$ items are "backordered" at pick-up location i. The warehouse area is partitioned such that at each time, each forklift is assumed to be located either next to one of the pick-up locations (identified by $1, \ldots, N$), or at the depot (identified by $N+1$), or at the battery/maintenance shop (identified by $N+2$), hence l_k takes values from a discrete set $\{1, \ldots, N, N+1, N+2\}$. Each s_k takes values from a discrete set {*"Idle", "Battery/Maintenance", "Task 1", "Task 2"*}, where "Idle" denotes that the forklift is idle, "Battery/Maintenance" denotes that the forklift is in maintenance or its battery is being replaced, and "Task 1" or "Task 2" denote that the forklift is performing either Task 1 or Task 2, respectively. Each h_k also takes values from a discrete set $\{0, 1, \ldots, B\}$, where 0 indicates that the forklift's battery/health is in "bad" condition, and the higher the value of h_k is, the better the health of the forklift k is. The state of the system, therefore, can be represented by the following vector of size $N + I + 3F$:

$$\{p_1, \ldots, p_N, d_1, \ldots, d_I, l_1, \ldots, l_F, s_1, \ldots, s_F, h_1, \ldots, h_F\},$$

Considering that a good number of variables are involved in describing the problem, let's summarize: F is the number of forklifts; the status of each forklift takes 4 possible values; I is the number of

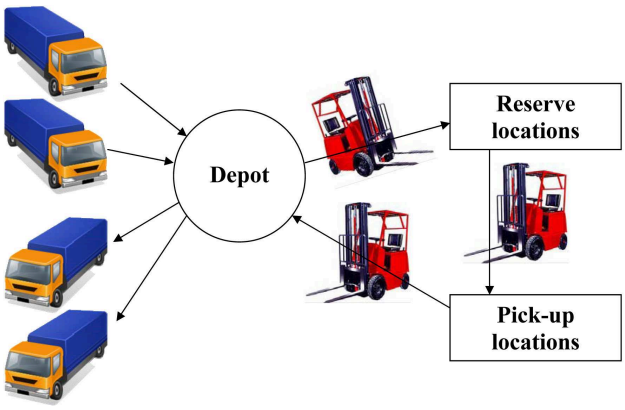

Figure 3: Overview of the activities in the warehouse.

isles; N is the number of pickup locations; $B+1$ is the number of forklift health levels; $2P+1$ is the number of stock levels in each pickup location; and $Q+1$ is the number of stock levels of the depot. Then, the state space of the system has

$$(2P+1)^N(Q+1)^I(N+2)^F 4^F(B+1)^F$$

discrete elements.

The arrival processes of deliveries and demands are assumed to follow the Poisson distribution, whereas the quantities of the deliveries and demands are assumed to follow a discrete uniform distribution. No deliveries are allowed when the depot is full, and no further demand of an item is accepted when the current level of the item in its pick-up location is at the lowest possible value (i.e., the maximum backordered level). *Traffic congestion* in the ith isle, indicated by K_i is defined as

$$K_i = W_1 X_i + W_2 F_i,$$

where F_i is the number of forklifts operating in that isle (known since the position of the forklifts is part of the state), X_i is an estimate of the pallet truck traffic, defined as

$$X_i = \sum_{\{j|\mathrm{IND}(j)=i,\, p_j<0\}} \log(-p_j),$$

and W_1 and W_2 are constant weights capturing the relative degree according to which pallet trucks and forklifts contribute to the traffic, respectively. In the above, $\mathrm{IND}(j)$ is a function that returns the index of the isle in which the pick-up location of the jth item is located.

Tasks 1 and 2 are assumed to have exponentially distributed durations with rates that can depend on the type of the task. Given the current state and the particular forklift (with a known location) to which either Task 1 or 2 gets assigned, the expected task duration is a certain (increasing) function of the traffic congestion present at the destination isle. We assume that this function takes values in a finite interval, hence, the rate at which a task gets completed is finite for all states.

Assume also that when a forklift is ordered to remain "Idle" or go to the "Battery/Maintenance" it remains in the corresponding status for an exponentially distributed amount of time with a rate that may depend on the status. It follows, that a forklift k maintains each status s_k for an exponentially distributed amount of time. Let $\rho_{\max}$ the maximum rate (over all states) corresponding to this exponential distribution.

With the assumptions put in place the state of the system evolves as a continuous time Markov chain. We uniformize this Markov chain by creating a Poisson clock with rate $F\rho_{\max}$ at which transitions occur. Given the rates of various status durations, it is straightforward to compute transition probabilities to the next state each time our uniform Poisson clock ticks; clearly, some of these transitions are self-transitions. We will use our MDP machinery to make (forklift assignment) decisions only at the transition epochs of this uniformized Markov chain.

The one-step cost $g(\cdot)$ includes the opportunity cost of having shortages in the pickup locations ($g_1(\cdot)$), the cost associated with the delay of clearing the depot ($g_2(\cdot)$), and the cost of operating the forklifts ($g_3(\cdot)$), where

$$g_1(\mathbf{x}, u) = \sum_{i=1}^{N} c_i \max(p_i, 0), \quad g_2(\mathbf{x}, u) = c_s \sum_{i=1}^{I} d_i, \quad g_3(\mathbf{x}, u) = c_f n_f.$$

In the above, c_i is the price/value of item i times the chance of having a sale per unit of time if the pickup location is stocked at or above demand; c_s is the average cost of delay in clearing the depot per pallet per unit of time; c_f denotes the average cost of operating a forklift per unit of time; and n_f denotes the number of forklifts that are being operated (i.e., with a status equal to "Task 1" or "Task 2"). The one-step cost rate $g(\cdot)$ is then defined as

$$g(\mathbf{x}, u) = g_1(\mathbf{x}, u) + g_2(\mathbf{x}, u) + g_3(\mathbf{x}, u).$$

As we mentioned in the Introduction, the forklifts are fitted with wireless sensors and communicate with the dispatching system via a warehouse wireless sensor network (see [30] for an overview of the sensor network setup). The sensors on each forklift provide the forklift's location (needed in order to determine congestion at the isles) as well as various forklift health indicators (e.g., battery status). These observations are "noisy" due to estimation errors but they fit our POMDP framework. To see this, we show that the separability assumption of the cost function (cf. Assumption A) holds. We can represent the bijective representation of the state as

$$\begin{aligned}
x^{(1)}, \ldots, x^{(N)} &= p_1, \ldots, p_N, \\
x^{(N+1)}, \ldots, x^{(N+I)} &= d_1, \ldots, d_I, \\
x^{(N+I+1)}, \ldots, x^{(N+I+F)} &= l_1, \ldots, l_F, \\
x^{(N+I+F+1)}, \ldots, x^{(N+I+2F)} &= s_1, \ldots, s_F, \\
x^{(N+I+2F+1)}, \ldots, x^{(N+I+3F)} &= h_1, \ldots, h_F
\end{aligned}$$

and we can write

$$g(\mathbf{x}, u) = g_1(x^{(1)}) + g_2(x^{(2)}) + \cdots + g_{N+I+3F}(x^{(N+I+3F)}) + g_U(u),$$

where

$$g_i(x^{(i)}) = \begin{cases} c_i \max(p_i, 0), & \text{if } 1 \le i \le N, \\ c_s d_{i-N}, & \text{if } N+1 \le i \le N+I, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$g_U(u) = c_f I_1(u).$$

It follows that Assumption A(i) holds. Furthermore, because the processes of location detection and of the battery/health readings from each forklift are all independent of each other, Assumption A(ii) also holds and the noise models can be constructed (e.g., using sampling methods). Therefore, the result of Theorem 4.1 is applicable.

5.2 The RSP

The action space $\mathbb{U}$ contains $N + I + 2$ elements:

1. the first N elements correspond to Task 2 assignments, where each element maps to one of the N pick-up locations;
2. the next I elements concern Task 1 assignments, where element $N+i$, $i = 1, \ldots, I$, corresponds to moving items from the depot to a reserve location in isle i;
3. the $(N + I + 1)$st element corresponds to "going to the battery/maintenance shop"; and, finally,
4. the $(N + I + 2)$nd element corresponds to "idling".

Our RSP is constructed by exponential functions with a 4-dimensional parameter vector

$$\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4).$$

The scalar elements of $\boldsymbol{\theta}$ are used to weigh, respectively, the urgency of the demand (θ_1), the delay at the depot (θ_2), traffic congestion (θ_3), and forklift health (θ_4). The RSP is given by

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(u|\mathbf{x}) = \frac{1}{(\sum_{i=1}^{N+I+2} a_i)} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{N+I+2} \end{pmatrix},$$

where

$$a_i = \begin{cases} P_i \, e^{\theta_1 c_i (U - p_i) + \theta_3 K^{-1}_{I(i)}}, & 1 \le i \le N, \\ D_{i-N} \, e^{\theta_2 d_{i-N} + \theta_3 K^{-1}_{i-N}}, & N+1 \le i \le N+I, \\ B \, e^{\theta_4}, & i = N+I+1, \\ 1, & i = N+I+2, \end{cases}$$

in which U is the maximum capacity of pick-up locations, and

$$\begin{aligned} P_i &= \begin{cases} 0, & \text{if the pick-up location of the } i\text{th item is full,} \\ 1, & \text{otherwise,} \end{cases} \\ D_i &= \begin{cases} 0, & \text{if there is no item associated with the } i\text{th isle waiting in the depot,} \\ 1, & \text{otherwise,} \end{cases} \\ B &= \begin{cases} 1, & \text{if the health condition of the forklift is "bad",} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{14}$$

Note that, we are optimizing each forklift as if it were a single, independent agent. In our uniformized Markov chain at most one forklift completes its current status at any given epoch; when this happens we use the RSP described above to assign a new status to it.

Some observations on the structure of the RSP are in order. The probability of filling pick-up location i increases with the value c_i of item i and the backlog $U - p_i$ but it decreases with the congestion level at the corresponding isle $I(j)$. Note that this probability is controlled by parameters θ_1 and θ_3. Similarly, the probability of assigning a forklift to move material from the depot to a reserve location in isle j is increasing with the amount of such material accumulated at the depot (d_j) and decreasing with the congestion in isle j. The latter probability is controlled by parameters θ_2 and θ_3. Finally, the probability of sending the forklift to the maintenance/battery shop is controlled by the parameter θ_4. The task of the actor-critic algorithm is to optimize the policy performance over $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_4)$.

5.3 Numerical results

We first obtained the exact optimal policy for instances of a very small-scale version of this problem which involves 2 forklifts ($F = 2$), 2 isles ($I = 2$), and 4 items ($N = 4$) using standard *Dynamic Programming* (DP) – value iteration in particular. We set $P = 2$, $Q = 1$, and $B = 1$. We generated 12 instances of this small-scale version (termed S-1 up to S-12) by varying the values of the parameters in the cost function as well as the rates of arrivals of deliveries and demands. We then used our LSTD actor-critic algorithm to optimize the RSP outlined earlier. The results are shown in Table 1 (AC is used as abbreviation for actor-critic algorithm). In addition, Fig. 5 illustrates the progress of the algorithm for the instance S-1. For the small-scale problem, our actor-critic algorithm took a much shorter time than DP and consistently found near-optimal solutions along different sample paths (within 10% of the optimal DP cost). We also tracked the changes in the average congestion (Fig. 6), average availability of items to fulfill demand (Fig. 7), average number of items waiting at the depot (Fig. 8), and average delay at the depot (Fig. 9). The purpose is to qualitatively understand how the policy we obtained trades-off these factors, which is useful in designing a good heuristic policy. For instance, we found from the DP solution that Task 2 has higher priority than Task 1.

We then used this insight to design a heuristic policy that we used as a benchmark for larger-scale problems where DP becomes computational intractable. The description of the heuristic is given in Fig. 4. It is interesting that it is consistent with state-of-the-art policies used by practitioners in actual warehouses. When applied to the small-scale problem above it performs within 20% of the optimal cost.

In order to evaluate the efficiency of our approach, we used it to learn a locally optimal RSP for a larger-scale problem involving 10 forklifts, 4 isles and 32 items. Note that with only 2 forklifts, 2 isles and 4 items, the state space contains $5,760,000$ discrete elements. The size of the state space for the larger-scale problem is much greater ($8.26e + 47$ discrete elements), hence, it cannot be solved with exact DP in a reasonable amount of time.

We generated 10 instances of the larger-scale version (termed L-1 up to L-10) by varying the values of the parameters in the cost function as well as the parameters of the distribution functions characterizing arrivals of deliveries and demands. We then tested our LSTD actor-critic algorithm to see if it is able to learn a better policy than the heuristic policy of Fig. 4. The results are

As soon as a forklift finishes its task, assign a new task to it according to the following order of priorities:

1. If the forklift's health is "bad", then the forklift is commanded to go to the battery/maintenance shop.
2. If there are shortages in the pick-up locations, then Task 2 is assigned. When more than one items are in shortage, the item with the highest unit price is served first.
3. If there are items waiting at the depot, then Task 1 is assigned.
4. Otherwise, the forklift is instructed to idle.

Figure 4: The description of the heuristic policy for the forklift dispatching problem.

Table 1: Results obtained by the algorithms for small-scale problems.

Instance	S-1	S-2	S-3	S-4	S-5	S-6	S-7	S-8	S-9	S-10	S-11	S-12
Optimal cost (DP)	5.44	12.97	7.43	6.32	14.65	9.26	5.82	14.02	9.07	8.18	15.14	9.91
AC cost	5.95	13.23	7.95	6.51	15.97	10.02	5.98	14.86	9.43	8.75	16.53	10.48
$\frac{\text{AC cost}}{\text{Optimal cost}}$	1.09	1.02	1.07	1.03	1.09	1.08	1.03	1.06	1.04	1.07	1.09	1.06
Heuristic cost	6.52	15.05	9.36	7.27	17.29	10.93	6.93	16.82	11.16	9.73	17.71	11.69
$\frac{\text{Heuristic cost}}{\text{Optimal cost}}$	1.20	1.16	1.26	1.15	1.18	1.18	1.19	1.20	1.23	1.19	1.17	1.18

encouraging (see Table 2). In addition, Fig. 10 illustrates the progress of the algorithm for the instance L-1. The LSTD actor-critic converged to policies that represent more than a 15% cost reduction compared to the heuristic policy. In some instances, this improvement exceeds 25%. In particular, we noticed that the actor-critic algorithm did a good job avoiding traffic congestion.

In addition, Fig. 5 and Fig. 10 also compare the performance of our proposed LSTD actor-critic with the traditional TD-based actor-critic developed in [25] (with λ in the TD(λ) critic set to 0.75) and the algorithm in [32] (natural actor-critic). Note that our LSTD actor-critic shows smoother and faster convergence behavior than both alternatives.

We also used the LSTD-PO actor-critic to learn the optimal policy under partial observability of states and cost. As shown in Fig. 5 and Fig. 10, the simulation results confirm Theorem 4.1. We mention that the LSTD-PO algorithm converges to 10.63 for the S-1 instance which, as shown in Fig. 5, is much higher than the cost achieved under perfect state observations. To compare the former with a more precise baseline, a lower bound of the optimal average cost for the imperfect observation case was computed using the methods given in [41]. To obtain this lower-bound, we discretized the belief space of the POMDP into a finite number of belief points and computed the

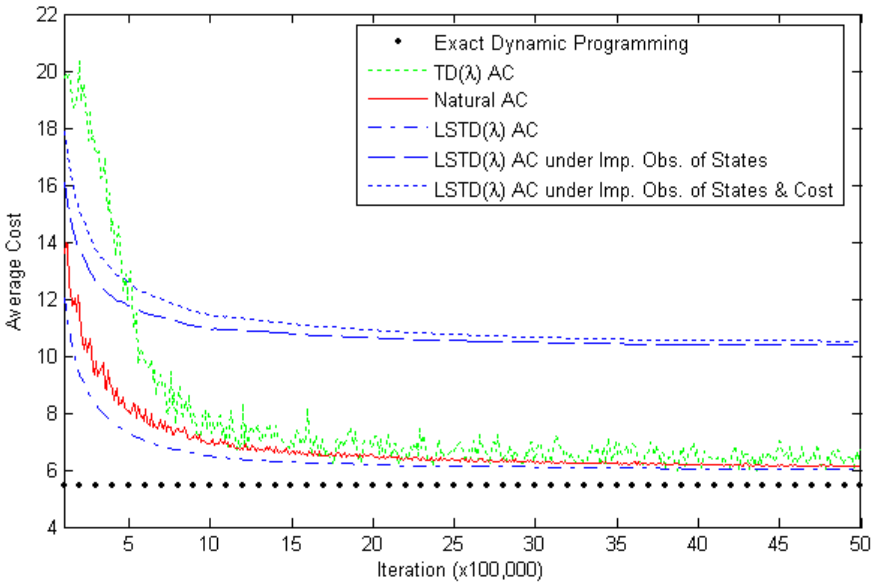

Figure 5: Results obtained by the algorithms for the S-1 instance.

Table 2: Results obtained by the algorithms for larger-scale problems.

Instance	L-1	L-2	L-3	L-4	L-5	L-6	L-7	L-8	L-9	L-10
Heuristic cost	18.07	26.11	22.44	31.72	27.32	38.87	31.06	40.43	33.38	41.82
AC cost	14.11	21.41	16.61	26.64	21.58	31.48	24.85	33.15	25.37	34.29
$\frac{\text{Heuristic cost - AC cost}}{\text{Heuristic cost}}$	0.22	0.18	0.26	0.16	0.21	0.19	0.20	0.18	0.24	0.18

approximating functions at these points using multi-chain algorithms. We discretized the belief space (which is a simplex in an $|\mathbb{X}|$-dimensional space) into a regular grid according to pattern k-E introduced in Section 4 of [41]. Briefly, the pattern k-E consists of k grid points on each edge, in addition to the vertices of the belief simplex. We then used the pattern $10 - E$ and obtained the value 9.7 for the lower-bound on the optimal average cost which implies that LSTD-PO finds near-optimal solutions. Finally, we mention that the dimensionality of the belief space for the larger-scale problem is too large, hence, it is impractical to obtain such a lower-bound in a reasonable amount of time.

The numerical experiments above were implemented in C on a PC with 1.4 GHz CPU speed. On average, the exact DP for the small-scale instances took about 10 hours to converge whereas the actor-critic converged within 20 minutes. For the larger-scale instances, the actor-critic algorithm took about 2 hours to converge.

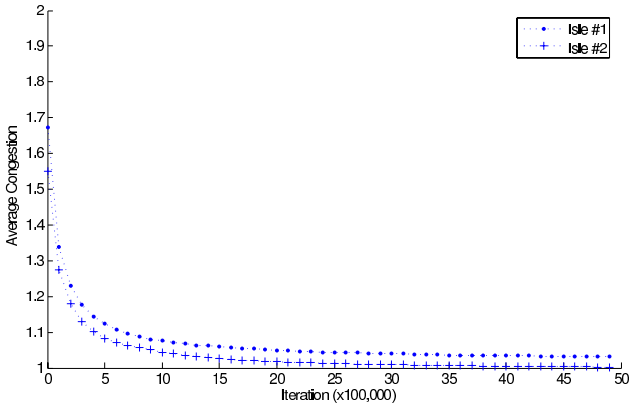

Figure 6: Congestion data obtained by the proposed actor-critic algorithm for the S-1 instance.

6 Conclusions

This paper integrated a least-squares temporal difference learning method into an efficient actor-critic architecture where the actor and the critic updates are carried out concurrently. The convergence of the algorithm was established, and the performance of the algorithm was demonstrated by solving a forklift dispatching problem arising in warehouse management. For that application, and for instances of realistic size, we demonstrated a 20% performance improvement over a heuristic similar to the ones used in practice. We also extended the applicability of actor-critic algorithms to general cases of POMDPs where even the cost is not perfectly observable. The algorithms presented in this paper use policies that are functions of immediate observations. Although in a POMDP setting, more complex policies such as controllers with internal states can be used ([6, 2]), the actor-critic methods can also be used as alternatives to these methods in learning finite-state controllers (see [40]). In the POMDPs with indirectly observed cost, the use of such alternatives could be an interesting direction for further research.

Acknowledgments

We would like to thank Raymond Corp. for its support and Paul McCabe and Steve Medwin for useful discussions. We also thank the anonymous reviewers who have helped us improve the paper.

Appendix

A Proof of Theorem 3.1

We first cite the theory of *linear stochastic approximation driven by a slowly varying Markov chain* [24] (with simplifications).

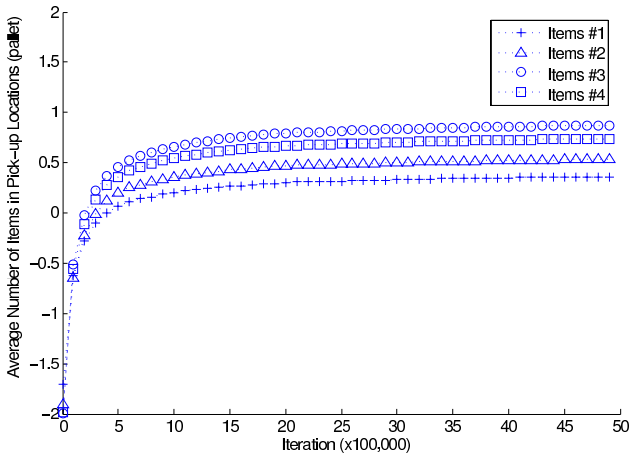

Figure 7: Pick-up location data obtained by the proposed actor-critic algorithm for the S-1 instance.

Let $\{\mathbf{y}_k\}$ be a finite Markov chain whose transition probabilities depend on a parameter $\boldsymbol{\theta} \in \mathbb{R}^n$. Consider a generic iteration of the form

$$\mathbf{s}_{k+1} = \mathbf{s}_k + \gamma_k(\mathbf{h}_{\boldsymbol{\theta}_k}(\mathbf{y}_{k+1}) - \mathbf{G}_{\boldsymbol{\theta}_k}(\mathbf{y}_{k+1})\mathbf{s}_k) + \gamma_k \boldsymbol{\Xi}_k \mathbf{s}_k, \tag{15}$$

where $\mathbf{s}_k \in \mathbb{R}^m$, $\boldsymbol{\Xi}_k$ is some $m \times m$-matrix, and $\mathbf{h}_{\boldsymbol{\theta}}(\cdot) \in \mathbb{R}^m, \mathbf{G}_{\boldsymbol{\theta}}(\cdot) \in \mathbb{R}^{m\times m}$ are $\boldsymbol{\theta}$-parameterized vector and matrix functions, respectively. This equation sets up a structure of a regression process whose weight is also determined by a stochastic process. The simultaneous-update actor-critic we are considering is a special case. It has been shown in [24] that the critic in (15) converges if the following set of conditions are met.

Condition 1

1. *The sequence $\{\gamma_k\}$ is deterministic, non-increasing, and*

$$\sum_k \gamma_k = \infty, \quad \sum_k \gamma_k^2 < \infty.$$

2. *The random sequence $\{\boldsymbol{\theta}_k\}$ satisfies $||\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k|| \le \beta_k H_k$ for some process $\{H_k\}$ with bounded moments, where $\{\beta_k\}$ is a deterministic sequence such that*

$$\sum_k \left(\frac{\beta_k}{\gamma_k}\right)^d < \infty \quad \textit{for some } d > 0.$$

3. *$\boldsymbol{\Xi}_k$ is an $m \times m$-matrix valued martingale difference with bounded moments.*

4. *For each $\boldsymbol{\theta}$, there exist $\bar{\mathbf{h}}(\boldsymbol{\theta}) \in \mathbb{R}^m$, $\bar{\mathbf{G}}(\boldsymbol{\theta}) \in \mathbb{R}^{m\times m}$, and corresponding m-vector and $m \times m$-matrix functions $\hat{\mathbf{h}}_{\boldsymbol{\theta}}(\cdot)$, $\hat{\mathbf{G}}_{\boldsymbol{\theta}}(\cdot)$ that satisfy the Poisson equation. That is, for each $\mathbf{y}$,*

$$\hat{\mathbf{h}}_{\boldsymbol{\theta}}(\mathbf{y}) = \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{y}) - \bar{\mathbf{h}}(\boldsymbol{\theta}) + (P_{\boldsymbol{\theta}}\hat{\mathbf{h}}_{\boldsymbol{\theta}})(\mathbf{y}),$$

$$\hat{\mathbf{G}}_{\boldsymbol{\theta}}(\mathbf{y}) = \mathbf{G}_{\boldsymbol{\theta}}(\mathbf{y}) - \bar{\mathbf{G}}(\boldsymbol{\theta}) + (P_{\boldsymbol{\theta}}\hat{\mathbf{G}}_{\boldsymbol{\theta}})(\mathbf{y}).$$

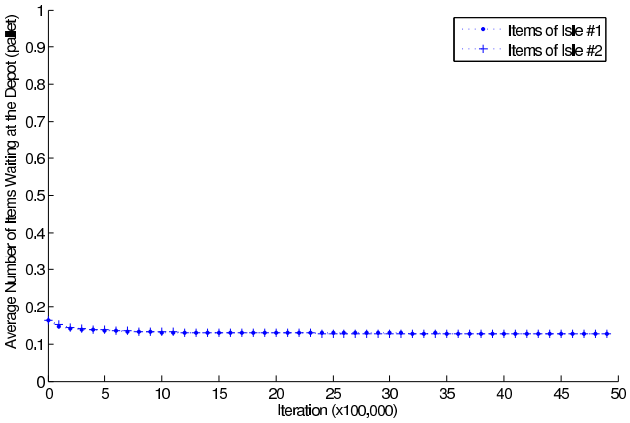

Figure 8: Depot data obtained by the proposed actor-critic algorithm for the S-1 instance.

5. *For some constant* C *and for all* $\boldsymbol{\theta}$, *we have* $\max(||\bar{\mathbf{h}}(\boldsymbol{\theta})||, ||\bar{\mathbf{G}}(\boldsymbol{\theta})||) \leq C$.
6. *For any* $d > 0$, *there exists* $C_d > 0$ *such that* $\sup_k \mathbf{E}[||\mathbf{f}_{\boldsymbol{\theta}_k}(\mathbf{y}_k)||^d] \leq C_d$, *where* $\mathbf{f}_{\boldsymbol{\theta}}(\cdot)$ *represents any of the functions* $\hat{\mathbf{h}}_{\boldsymbol{\theta}}(\cdot)$, $\mathbf{h}_{\boldsymbol{\theta}}(\cdot)$, $\hat{\mathbf{G}}_{\boldsymbol{\theta}}(\cdot)$ *and* $\mathbf{G}_{\boldsymbol{\theta}}(\cdot)$.
7. *For some constant* $C > 0$ *and for all* $\boldsymbol{\theta}, \bar{\boldsymbol{\theta}} \in \mathbb{R}^n$, $\max(||\bar{\mathbf{h}}(\boldsymbol{\theta}) - \bar{\mathbf{h}}(\bar{\boldsymbol{\theta}})||, ||\bar{\mathbf{G}}(\boldsymbol{\theta}) - \bar{\mathbf{G}}(\bar{\boldsymbol{\theta}})||) \leq C||\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}||$.
8. *There exists a positive measurable function* $C(\cdot)$ *such that for every* $d > 0$, $\sup_k \mathbf{E}[C(\mathbf{y}_k)^d] < \infty$, *and* $||\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{y}) - \mathbf{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{y})|| \leq C(\mathbf{y})||\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}||$.
9. *There exists* $a > 0$ *such that for all* $\mathbf{s} \in \mathbb{R}^m$ *and* $\boldsymbol{\theta} \in \mathbb{R}^n$

$$\mathbf{s}'\bar{\mathbf{G}}(\boldsymbol{\theta})\mathbf{s} \geq a||\mathbf{s}||^2.$$

If these conditions are all met, then

$$\lim_{k\to\infty} \mathbf{s}_k = \bar{\mathbf{G}}(\boldsymbol{\theta}_k)^{-1}\bar{\mathbf{h}}(\boldsymbol{\theta}_k).$$

For now, let's focus on the first two items of Condition 1. For any matrix $\mathbf{A}$ let $\mathbf{v}(\mathbf{A})$ be a column vector that stacks all row vectors of $\mathbf{A}$ (also written as column vectors). Simple algebra suggests that the core iteration of the LSTD critic can be written as (15) with

$$\mathbf{s}_k = \begin{bmatrix} M\alpha_k \\ \mathbf{b}_k \\ \mathbf{v}(\mathbf{A}_k) \\ 1 \end{bmatrix}, \qquad \mathbf{y}_k = (\mathbf{x}_k, u_k, \mathbf{z}_k),$$

$$\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{y}) = \begin{bmatrix} Mg(\mathbf{x},u) \\ g(\mathbf{x},u)\mathbf{z} \\ \mathbf{v}(\mathbf{z}((P_{\boldsymbol{\theta}}\boldsymbol{\psi}'_{\boldsymbol{\theta}})(\mathbf{x},u) - \boldsymbol{\psi}'_{\boldsymbol{\theta}}(\mathbf{x},u))) \\ 1 \end{bmatrix}, \quad \mathbf{G}_{\boldsymbol{\theta}}(\mathbf{y}) = \begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{z}/M & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \end{bmatrix}, \qquad (16)$$

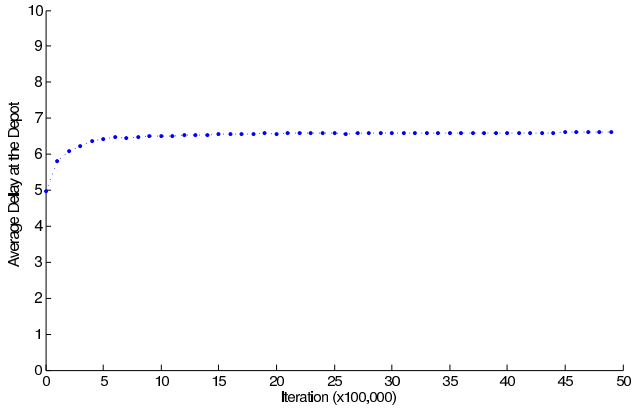

Figure 9: Delay data obtained by the proposed actor-critic algorithm for the S-1 instance.

$$\mathbf{\Xi}_k = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{v}(\mathbf{z}_k(\boldsymbol{\psi}'_{\boldsymbol{\theta}_k}(\mathbf{x}_{k+1}, u_{k+1}) - (P_{\boldsymbol{\theta}}\boldsymbol{\psi}_{\boldsymbol{\theta}})'(\mathbf{x}_k, u_k))) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where M is an arbitrary (large) positive constant whose role is to facilitate the convergence proof.

The step-sizes γ_k and β_k in (7) and (9) correspond exactly to the γ_k and β_k in Condition 1.(1) and 1.(2), respectively. If the MDP has finite state and action space, then the conditions on $\{\beta_k\}$ reduce to ([24])

$$\sum_k \beta_k = \infty, \quad \sum_k \beta_k^2 < \infty, \quad \lim_{k\to\infty} \frac{\beta_k}{\gamma_k} = 0, \tag{17}$$

where $\{\beta_k\}$ is a deterministic and non-increasing sequence. Note that we can use $\gamma_k = 1/k$ (cf. Condition 1). The following theorem establishes the convergence of the critic.

Lemma A.1 *[Critic Convergence] For the LSTD actor-critic (7) and (8) with some step-size sequence $\{\beta_k\}$ satisfying (17), the sequence $\mathbf{s}_k$ is bounded, and*

$$\lim_{k\to\infty} |\bar{\mathbf{G}}(\boldsymbol{\theta}_k)\mathbf{s}_k - \bar{\mathbf{h}}(\boldsymbol{\theta}_k)| = 0. \tag{18}$$

Proof : To show that (15) converges with $\mathbf{s}, \mathbf{y}, \mathbf{h}_{\boldsymbol{\theta}}(\cdot), \mathbf{G}_{\boldsymbol{\theta}}(\cdot)$ and $\mathbf{\Xi}$ substituted by (16), the conditions 1.(1)-(9) should be checked. However, a comparison with the convergence proof for the TD(λ) critic in [25] gives a simpler proof. Let

$$\mathbf{F}_{\boldsymbol{\theta}}(\mathbf{y}) = \mathbf{z}(\boldsymbol{\psi}'_{\boldsymbol{\theta}}(\mathbf{x}, u) - (P_{\boldsymbol{\theta}}\boldsymbol{\psi}_{\boldsymbol{\theta}})'(\mathbf{x}, u)).$$

While proving the convergence of the TD(λ) critic operating concurrently with the actor, [25] showed that

$$\tilde{\mathbf{h}}_{\boldsymbol{\theta}}(\mathbf{y}) = \begin{bmatrix} Mg(\mathbf{x}, u) \\ g(\mathbf{x}, u)\mathbf{z} \end{bmatrix}, \tilde{\mathbf{G}}_{\boldsymbol{\theta}}(\mathbf{y}) = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{z}/M & \mathbf{F}_{\boldsymbol{\theta}}(\mathbf{y}) \end{bmatrix},$$

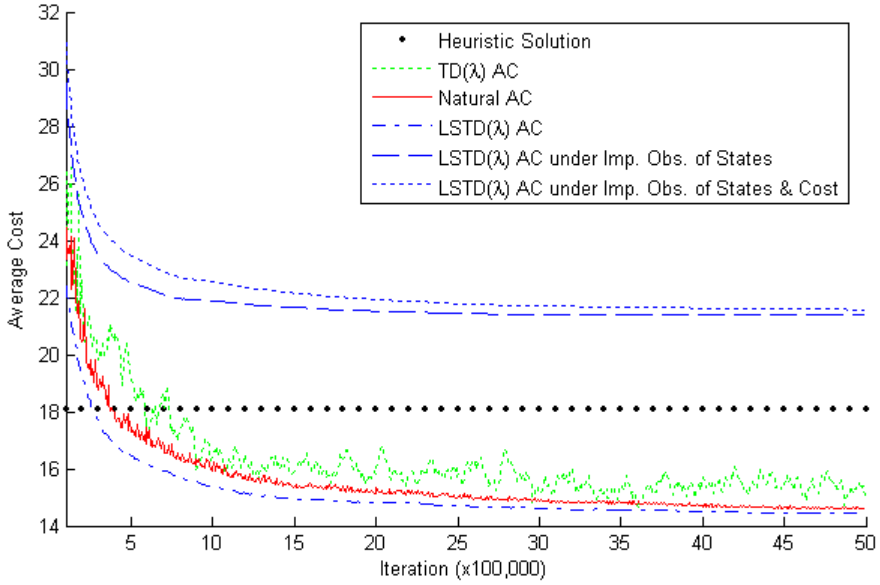

Figure 10: Results obtained by algorithms for the larger-scale problems.

and

$$\tilde{\Xi}_k = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{z}_k(\boldsymbol{\psi}'_{\boldsymbol{\theta}_k}(\mathbf{x}_{k+1}, u_{k+1}) - (P_{\boldsymbol{\theta}}\boldsymbol{\psi}_{\boldsymbol{\theta}})'(\mathbf{x}_k, u_k)) \end{bmatrix}$$

satisfy Condition 1.(3)-1(8). In our case, (16) can be rewritten as

$$\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{y}) = \begin{bmatrix} \tilde{\mathbf{h}}_{\boldsymbol{\theta}}(\mathbf{y}) \\ -\mathbf{F}_{\boldsymbol{\theta}}(\mathbf{y}) \\ 1 \end{bmatrix}, \ \mathbf{G}_{\boldsymbol{\theta}}(\mathbf{y}) = \begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{z}/M & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \text{ and } \quad \Xi_k = \begin{bmatrix} \tilde{\Xi}_k \\ \mathbf{0} \end{bmatrix}. \tag{19}$$

Note that $\mathbf{h}_{\boldsymbol{\theta}}$, $\mathbf{G}_{\boldsymbol{\theta}}$ and Ξ are linear functions of $\tilde{\mathbf{h}}_{\boldsymbol{\theta}}(\mathbf{y})$, $\mathbf{F}_{\boldsymbol{\theta}}(\mathbf{y})$, $\mathbf{z}$, and $\tilde{\Xi}$, thus share their convergence properties. It is clear then that they also satisfy satisfy Conditions 1.(3)-1(8). So, $\mathbf{h}_{\boldsymbol{\theta}}(\cdot), \mathbf{G}_{\boldsymbol{\theta}}(\cdot)$ and Ξ_k also satisfy Condition 1.(3)-1(8). Meanwhile, the step-size $\{\gamma_k\}$ satisfies Condition 1.(1), and the step-size $\{\beta_k\}$ satisfies Eq. (17) (which is as explained above implies Condition 1.(2)). Now, only Condition 1(9) remains to be checked. To that end, note that all diagonal elements of $\mathbf{G}_{\boldsymbol{\theta}}(\mathbf{y})$ equal to one, and the only off-diagonal block is a bounded matrix divided by M. So, $\mathbf{G}_{\boldsymbol{\theta}}(\mathbf{y})$ is positive definite if M, which can be set arbitrarily, is large enough. Thus, Condition 1(9) also holds. This proves the convergence.

Last, to show that Equation (18) is satisfied, one only needs to invoke the same correspondence and the result in [25].

■

Theorem 3.1 follows by setting $\boldsymbol{\phi}_{\boldsymbol{\theta}} = \boldsymbol{\psi}_{\boldsymbol{\theta}}$ and following the proof in Section 6 of [25].

B Proof of Theorem 4.1

Under the ergodicity conditions mentioned earlier, $\{\mathbf{x}_k\}$ and $\{\mathbf{x}_k, u_k\}$ are Markov chains with stationary distributions under any policy $\boldsymbol{\theta}$. Let $\pi_{\boldsymbol{\theta}}(\mathbf{x})$ and $\eta_{\boldsymbol{\theta}}(\mathbf{x}, u)$ denote the stationary distributions of $\{\mathbf{x}_k\}$ and $\{\mathbf{x}_k, u_k\}$, respectively. Since the observations are generated directly from $\{\mathbf{x}_k\}$, $\{\mathbf{y}_k\}$ and $\{\mathbf{y}_k, u_k\}$ also have stationary distributions (denoted by $\tilde{\pi}_{\boldsymbol{\theta}}(\mathbf{y})$ and $\tilde{\eta}_{\boldsymbol{\theta}}(\mathbf{y}, u)$, respectively). The average cost proxy can be rewritten as

$$\begin{aligned}\tilde{\alpha}_{\boldsymbol{\theta}} &= \sum_{\mathbf{y},u}\left(\sum_{\mathbf{x}}\pi_{\boldsymbol{\theta}}(\mathbf{x})\mu_{\boldsymbol{\theta}}(u|\mathbf{y})p(\mathbf{y}|\mathbf{x})\right)\tilde{g}(\mathbf{y},u)\\ &= \sum_{\mathbf{y},u,\mathbf{x}}\left(\pi_{\boldsymbol{\theta}}(\mathbf{x})\mu_{\boldsymbol{\theta}}(u|\mathbf{y})p(\mathbf{y}|\mathbf{x})\sum_{l=1}^{M}\sum_{\boldsymbol{\xi}^{(l)}}\nu^{(l)}_{\boldsymbol{\xi}^{(l)},\mathbf{y}^{(l)}}g_l(\boldsymbol{\xi}^{(l)})\right) + \sum_{\mathbf{y},u,\mathbf{x}}\pi_{\boldsymbol{\theta}}(\mathbf{x})\mu_{\boldsymbol{\theta}}(u|\mathbf{y})p(\mathbf{y}|\mathbf{x})g_U(u). \quad (20)\end{aligned}$$

Let $P_{\boldsymbol{\theta}}(u|\mathbf{x}) = \sum_{\mathbf{y}}\mu_{\boldsymbol{\theta}}(u|\mathbf{y})p(\mathbf{y}|\mathbf{x})$. Then the second term of Eq. (20) reduces to

$$\sum_{u,\mathbf{x}}\pi_{\boldsymbol{\theta}}(\mathbf{x})P_{\boldsymbol{\theta}}(u|\mathbf{x})g_U(u) = \sum_{\mathbf{x},u}\eta_{\boldsymbol{\theta}}(\mathbf{x},u)g_U(u),$$

which is a term readily found in the average true cost. Since $\sum_u \mu_{\boldsymbol{\theta}}(u|\mathbf{y}) = 1$, the first term of Eq. (20) equals

$$\begin{aligned}&\sum_{\mathbf{y},\mathbf{x}}\left(\pi_{\boldsymbol{\theta}}(\mathbf{x})p(\mathbf{y}|\mathbf{x})\sum_{l=1}^{M}\sum_{\boldsymbol{\xi}^{(l)}}\nu^{(l)}_{\boldsymbol{\xi}^{(l)},\mathbf{y}^{(l)}}g_l(\boldsymbol{\xi}^{(l)})\right)\\ &= \sum_{\mathbf{x},\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(M)}}\left(\pi_{\boldsymbol{\theta}}(\mathbf{x})\cdot\left(\prod_{m=1}^{M}p(\mathbf{y}^{(m)}|\mathbf{x})\right)\cdot\sum_{l=1}^{M}\sum_{\boldsymbol{\xi}^{(l)}}\nu^{(l)}_{\boldsymbol{\xi}^{(l)},\mathbf{y}^{(l)}}g_l(\boldsymbol{\xi}^{(l)})\right)\\ &= \sum_{\mathbf{x}}\left(\pi_{\boldsymbol{\theta}}(\mathbf{x})\sum_{l=1}^{M}\sum_{\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(M)}}\left(\left(\prod_{m=1}^{M}p(\mathbf{y}^{(m)}|\mathbf{x})\right)\cdot\sum_{\boldsymbol{\xi}^{(l)}}\nu^{(l)}_{\boldsymbol{\xi}^{(l)},\mathbf{y}^{(l)}}g_l(\boldsymbol{\xi}^{(l)})\right)\right). \quad (21)\end{aligned}$$

Note that for the marginal distribution of $\mathbf{y}^{(l)}$ given $\mathbf{x}$ it holds

$$\sum_{i=1,\,i\neq l}^{M}\sum_{\mathbf{y}^{(i)}}\left(\prod_{m=1}^{M}p(\mathbf{y}^{(m)}|\mathbf{x})\right) = p(\mathbf{y}^{(l)}|\mathbf{x}) = p(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}).$$

The right hand side of Eq. (21) becomes

$$\begin{aligned}&\sum_{\mathbf{x}}\left(\pi_{\boldsymbol{\theta}}(\mathbf{x})\sum_{l=1}^{M}\sum_{\mathbf{y}^{(l)}}p(\mathbf{y}^{(l)}|\mathbf{x}^{(l)})\sum_{\boldsymbol{\xi}^{(l)}}\nu^{(l)}_{\boldsymbol{\xi}^{(l)},\mathbf{y}^{(l)}}g_l(\boldsymbol{\xi}^{(l)})\right)\\ &= \sum_{\mathbf{x}}\left(\pi_{\boldsymbol{\theta}}(\mathbf{x})\sum_{l=1}^{M}\sum_{\boldsymbol{\xi}^{(l)}}g_l(\boldsymbol{\xi}^{(l)})\sum_{\mathbf{y}^{(l)}}p(\mathbf{y}^{(l)}|\mathbf{x}^{(l)})\nu^{(l)}_{\boldsymbol{\xi}^{(l)},\mathbf{y}^{(l)}}\right). \quad (22)\end{aligned}$$

From the definition of $\nu^{(l)}_{\mathbf{x}^{(l)},\mathbf{y}^{(l)}}$,

$$\sum_{\mathbf{y}^{(l)}} p(\mathbf{y}^{(l)}|\mathbf{x}^{(l)})\nu^{(l)}_{\boldsymbol{\xi}^{(l)},\mathbf{y}^{(l)}} = \begin{cases} 1, & \text{if } \boldsymbol{\xi}^{(l)} = \mathbf{x}^{(l)}, \\ 0, & \text{otherwise,} \end{cases}$$

thus, the right hand side of (22) equals

$$\sum_{\mathbf{x}} \pi_{\boldsymbol{\theta}}(\mathbf{x}) \sum_{l=1}^{M} g_l(\mathbf{x}^{(l)}).$$

Now both terms in Eq. (20) have been calculated and we conclude

$$\begin{aligned} \tilde{\alpha}_{\boldsymbol{\theta}} &= \sum_{\mathbf{x}} \pi_{\boldsymbol{\theta}}(\mathbf{x}) \sum_{l=1}^{M} g_l(\mathbf{x}^{(l)}) + \sum_{\mathbf{x},u} \eta_{\boldsymbol{\theta}}(\mathbf{x},u) g_U(u) \\ &= \sum_{\mathbf{x},u} \eta_{\boldsymbol{\theta}}(\mathbf{x},u) g(\mathbf{x},u) = \bar{\alpha}_{\boldsymbol{\theta}}. \end{aligned}$$

■

References

[1] D. Aberdeen. *Policy-Gradient Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, The Australian National University, 2003.

[2] D. Aberdeen and J. Baxter. Internal state policy-gradient algorithms for infinite-horizon POMDPs. Technical report, RSISE, Australian National University, 2002.

[3] J. Bai, J. Dongarra, A. Ruhe, and H. van der Vorst. Templates for the solution of algebraic eigenvalue problems: A practical guide. In *Society for Industrial and Applied Mathematics*, 2000.

[4] A. Barto, R. Sutton, and C. Anderson. Neuron like elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man and Cybernetics*, 13, 1983.

[5] J. Baxter and P. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 41–48, 2000.

[6] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[7] H. Berenji and D. Vengerov. A convergent actorcritic-based frl algorithm with application to power management of wireless transmitters. *IEEE Transactions on Fuzzy Systems*, 11(4):478–485, 2003.

[8] D. Bertsekas and S. Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical Report 2349, LIDS REPORT, 1996. MIT.

[9] D. P. Bertsekas, V. Borkar, and A. Nedic. Improved temporal difference methods with linear function approximation. In A. Barto, W. Powell, and J. Si (Eds.), editors, *Learning and Approximate Dynamic Programming*, pages 231–255. IEEE Press, 2004.

[10] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[11] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Incremental natural actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 20, pages 105–112, 2008.

[12] V. Borkar. An actor-critic algorithm for constrained Markov decision processes. *Systems and Control Letters*, 54:207–213, 2005.

[13] J. Boyan. Least-squares temporal difference learning. In *Proceeding of the 16th International Conference on Machine Learning*, 1999.

[14] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[15] S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(2):33–57, 1996.

[16] A. Cassandra. A survey of POMDPs applications. In *American Association for Artificial Intelligence Symposium*, 1998.

[17] H. Chang, M. Fu, J. Hu, and S. Marcus. *Simulation-based Algorithms for Markov Decision Processes (Communications and Control Engineering)*. Springer-Verlag, New York, 2007.

[18] R. Crites and A. Barto. An actor/critic algorithm that is equivalent to Q-learning. *Advances in Neural Information Processing Systems*, 7:401–408, 1994.

[19] J. Dongarra. Sparse matrix storage formats. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 2000.

[20] G. Gajjar, S. Khaparde, P. Nagaraju, and S. Soman. Application of actor-critic learning algorithm for optimal bidding problem of a GenCo. *IEEE Transactions on Power Engineering Review*, 18(1):11–18, 2003.

[21] M. Ghavamzadeh and Y. Engel. Bayesian actor-critic algorithms. In *ACM International Conference Proceeding Series*, volume 227, pages 297–304, 2007.

[22] T. Jaakkola, S. Singh, and M. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. *Advances in Neural Information Processing Systems*, 7, 1995.

[23] M. Khamassi, L. Lachze, B. Girard, A. Berthoz, and A. Guillot. Actor-critic models of reinforcement learning in the basal ganglia: From natural to artificial rats. *Adaptive Behavior*, 13(2):131–148, 2005.

[24] V. Konda. *Actor-critic Algorithms*. PhD thesis, MIT, Cambridge, MA, 2002.

[25] V. Konda and J. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[26] F. Melo and M. Ribeiro. Convergence of classical reinforcement learning algorithms and partial observability. Technical report, Technical Report, 2006.

[27] E. Mizutani and S. Dreyfus. Two stochastic dynamic programming problems by model-free actor-critic recurrent-network learning in non-Markovian settings. In *Proceeding of IEEE International Joint Conference on Neural Networks*, volume 2, pages 1079–1084, 2004.

[28] A. Nedic and D. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems: Theory and Applications*, 13:79–110, 2003.

[29] C. Niedzwiedz, I. Elhanany, Z. Liu, and S. Livingston. A consolidated actor-critic model with function approximation for high-dimensional POMDPs. In *AAAI 2008 workshop for Advancement in POMDP Solvers (part of the AAAI 2008 Conference)*, Chicago, 2008.

[30] I. Ch. Paschalidis, K. Li, R. Moazzez-Estanjini, Y. Lin, and D. Guo. Intelligent forklift dispatching in warehouses using a sensor network. In *Proceedings of the 17th Mediterranean Conference on Control and Automation (MED 09)*, pages 112–114, Thessaloniki, Greece, June 24–26 2009.

[31] J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[32] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71:1180–1190, 2008.

[33] W.B. Powell. *Approximate Dynamic Programming*. John Wiley and Sons, 2007.

[34] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994.

[35] M. Rosenstein and A. Barto. Supervised actor-critic reinforcement learning. In J. Si, A. Barto, W. Powell, and D. Wunsch, editors, *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*, pages 359–380. John Wiley and Sons, Inc., New York, 2004.

[36] Y. Saad. Krylov subspace methods on supercomputers. *SIAM Journal on Scientific and Statistical Computing*, 1989.

[37] K. Samejima and T. Omori. Adaptive internal state space construction method for reinforcement learning of a real-world agent. *Neural Networks*, 12:1143–1155, 1999.

[38] S. Singh, T. Jaakkola, and M. Jordan. Learning without state-estimation in partially observable Markovian decision processes. In *Proceedings of the Eleventh International Conference on Machine Learning*, 1994.

[39] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[40] H. Yu. A function approximation approach to estimation of policy gradient for POMDP with structured policies. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[41] H. Yu and D. Bertsekas. Discretized approximations for POMDP with average cost. In *The 20th Conference on Uncertainty in Artificial Intelligence*, pages 619–627, 2004.

[42] H. Yu and D. P. Bertsekas. Convergence results for some temporal difference methods based on least squares. *IEEE Trans. Automat. Contr.*, 54(7):1515–1531, 2009.