Efficient PAC Learning for Episodic Tasks with Acyclic State Spaces

Spyros Reveliotis and Theologos Bountourelis
School of Industrial & Systems Engineering
Georgia Institute of Technology
{spyros,tbountou}@isye.gatech.edu

Abstract

This paper considers the problem of computing an optimal policy for a Markov Decision Process (MDP), under lack of complete *a priori* knowledge of (i) the branching probability distributions determining the evolution of the process state upon the execution of the different actions, and (ii) the probability distributions characterizing the immediate rewards returned by the environment as a result of the execution of these actions at different states of the process. In addition, it is assumed that the underlying process evolves in a repetitive, episodic manner, with each episode starting from a well-defined initial state and evolving over an acyclic state space. A novel efficient algorithm for this problem is proposed, and its convergence properties and computational complexity are rigorously characterized in the formal framework of computational learning theory. Furthermore, in the process of deriving the aforementioned results, the presented work generalizes Bechhofer's "indifference-zone" approach for the Ranking & Selection problem, that arises in statistical inference theory, so that it applies to populations with bounded general distributions.

Keywords: Markov Decision Processes, Computational Learning Theory, Machine Learning, Efficient Probably Approximately Correct (PAC) Algorithms, Ranking & Selection, Uncertainty Management

1 Introduction

The problem considered in this work can be stated as follows: A certain task is executed repetitively in an episodic manner. Each episode starts from a well-defined initial state and evolves sequentially over an acyclic state space. At each state, the task evolution is the result of an action that is selected by a controlling agent; the execution of this action determines probabilistically the subsequent state, but it also results in a certain reward for the controlling agent. The returned reward can be of arbitrary sign,[1] and it is a random quantity drawn from some bounded[2] general distribution that is dependent on the current state and the commanded action. The agent's objective is to select the actions to be commanded at the different states of the underlying task in a way that maximizes the expected total reward to be collected over any single episode. However, the initial knowledge of the controlling agent about the underlying task and its operational environment is limited to (i) the set of the environmental states, (ii) the available actions at each state, (iii) an upper bound for the magnitude of the

[1] A negative reward can be considered as a cost.
[2] i.e., a distribution with bounded support

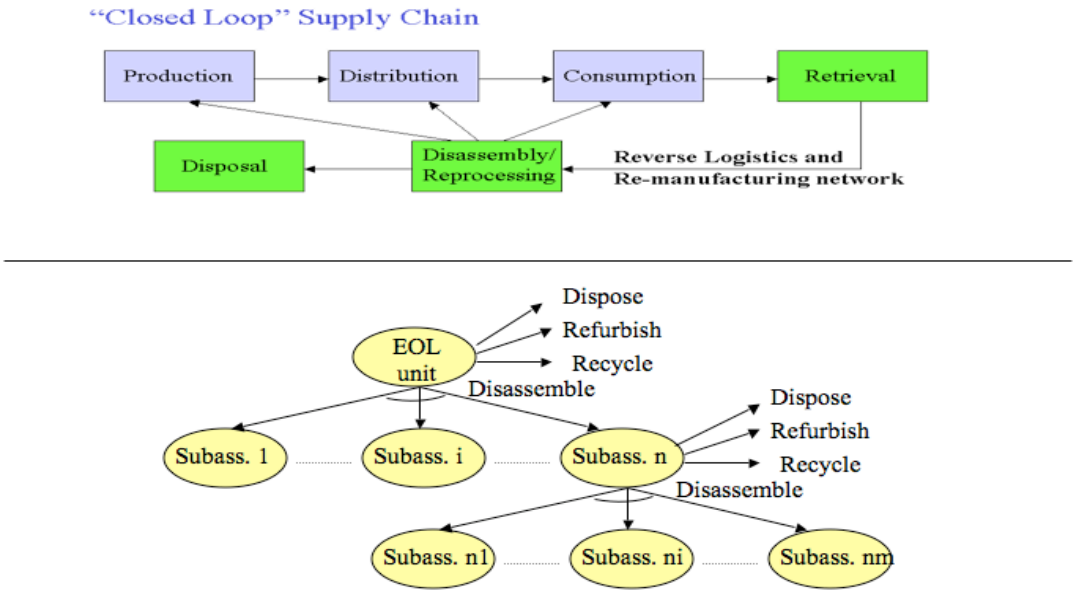

Figure 1: The Optimal Disassembly Planning Problem

experienced rewards, and (iv) a set of action sequences that can lead with positive probability to each of the environmental states, together with a lower bound for the corresponding state-reaching probabilities. The agent knows neither (v) the branching probability distributions that determine the state evolution as a result of the different actions, nor (vi) the type and the moments of the distributions determining the experienced rewards. Hence, the agent must compute a (near-)optimal policy for the aforestated objective, while observing and appropriately compiling the results of its decisions on the operational environment.

From an application standpoint, the aforestated problem arises frequently in sequential decision making contexts with a *"decision-tree"* structure [7] and with incomplete information regarding the underlying problem parameters. An application that has been the primary motivation for our work is the *Optimal Disassembly Planning (ODP)* problem [14, 15, 16]. This problem is depicted in Figure 1 and it can be briefly described as follows: Presently, under the pressure of environmental, safety, and other societal and economic concerns, a number of industrial sectors have started deploying additional operations that seek to retrieve the corresponding product units at the end of their functional life, extract from them any possible value, and dispose the remaining material in an environmentally friendly manner. Hence, the retrieved product units are brought to designated facilities, known as *re-manufacturing centers*, where those in fairly good condition are refurbished and re-introduced in the supply chain,[3] the remaining are disassembled in order to retrieve potentially re-usable components and/or raw material, like glass, metals and plastics, and any remaining unusable material is forwarded to landfills and/or incineration. The decision-making dynamics underlying this disassembly

[3]typically directed to a secondary market

process are depicted in the lower part of Figure 1: Starting with the initially retrieved unit, the decision maker must select a disposition option for this item, among those depicted in the figure. Typically, this decision will be made without complete knowledge of the quality status of the various components in this artifact and their salvage value. However, the missing information is partially regained through a number of measurements that are performed on the considered artifact and classify it to a certain quality category, that will eventually influence the decision making process. In case that the selected option is "disassembly", a similar testing and classification procedure will be applied to all the derived components, and this process will repeat itself, until all the obtained artifacts have been directed to a certain disposition venue. Hence, it is clear from this description that the considered decision making process is sequential and it evolves in *episodes* over a *finite, acyclic* state space, with each episode corresponding to the complete disposition of a particular product unit. Furthermore, in an optimized setting, the decision taken at each stage must maximize the value extracted from the corresponding artifact. However, the characterization of such an *optimal policy* is constrained by (i) the stochastic nature of the quality status of the extracted items and their salvage value, and also (ii) the stochasticity in the involved operational costs. Even worse, the *a priori* modeling of this stochasticity is not possible, since it is determined by consumer behavior and market forces that are not readily observable by the considered process. Hence, the process decision maker must resort to techniques that will enable her to derive the optimal policy through the observation and processing of the implications of her decisions during the process operation.

Problems and techniques that concern the incremental computation of (near-)optimal policies in lack of complete *a priori* information about the problem parameters fall in a broader scientific area that is known as *machine learning* [13]. In fact, researchers in the area of machine learning should recognize the problem outlined above as a typical *reinforcement learning (RL)* [18] problem, and therefore amenable to the available RL algorithms. For instance, under the aforestated assumptions and some standard regularity conditions, the classical, by now, *Q-learning* algorithm [20] can be applied with guaranteed convergence to optimality. Indeed, references [15, 16] detail an implementation of the Q-learning algorithm for the aforementioned ODP problem and establish conditions for its convergence to an optimal policy. However, these results are of an *asymptotic* nature, and they do not provide any further information regarding the *rate of convergence*. In general, RL – or as it is otherwise known, *Neuro-Dynamic Programming (NDP)* – theory has currently focused primarily on (i) the development of algorithms that will converge asymptotically to an optimal policy, and (ii) the pertinent representation and compression of the information that is necessary for the effective and efficient learning of the target (near-)optimal policies, especially in the case of tasks with very large discrete state spaces; the reader is referred to [2, 18] for some excellent expositions of the relevant results. On the other hand, the study of the complexity of the RL problem and of the convergence rate of RL learning algorithms to an optimal policy falls into the realm of *computational learning theory (CLT)* [11]. Currently, the CLT results on the RL problem are rather limited, and they pertain primarily to algorithmic variations that facilitate the analysis of the problem complexity rather than practical implementations, c.f., for instance, the works of [9, 10, 5, 6].

In the light of the above discussion, the main objective of the work presented in this paper is to establish that in the case of repetitive tasks evolving episodically over acyclic state spaces, as is the case with the aforementioned ODP problem, one can obtain customized learning algorithms that are computationally efficient and (yet!) effectively implementable. More specifically, for any given $\varepsilon > 0$ and $\delta \in (0, 1/2)$, a straightforward implementation of these

algorithms returns an ε-optimal policy with probability $1-\delta$,[4] while the resulting computational complexity is significantly lower than the algorithmic complexity identified in [9, 10] for more generally structured RL problems. This reduction in computational complexity and the suggested implementational efficiency of our algorithms stem from (i) the ability to pertinently characterize and exploit, in the considered problem context, the flow of the information necessary for the computation of the optimal policy, and (ii) results coming from the area of statistical inference – in particular, the area known as *"ranking & selection" (R&S)* [12] – that enable the systematic and rigorous resolution of the action selection problem arising at the different problem states.[5] Furthermore, another attribute that characterizes our algorithms and facilitates their straightforward implementation is their *"direct"* nature, i.e., the fact that they proceed to the computation of the target policy while avoiding the explicit estimation of the missing problem parameters. We mention, for completeness, that the work of [5] develops an alternative *"indirect"* algorithm for the considered problem that first estimates the missing problem parameters to a certain level of accuracy, and subsequently computes an "apparently optimal" policy through a standard *Dynamic Programming (DP)* computation. From a computational standpoint, both algorithms end up having comparable computational complexity, but, as mentioned above, we believe that the direct nature of the algorithm(s) presented herein enables a more straightforward and unobtrusive implementation in the underlying operational context.[6]

The rest of the paper is organized as follows: Section 2 provides a rigorous characterization of the learning problem considered in this work, and Section 3 presents a notion of computational efficiency that is pertinent to machine learning algorithms. Subsequently Section 4 proposes a methodology for developing efficient learning algorithms for the learning task under consideration, and finally, Section 5 concludes the paper by summarizing the key contributions and by outlining the remaining research tasks that are necessary for the completion of the proposed research and its implementation in the ODP context.[7]

2 A formal characterization of the considered learning problem

We begin our discussion of the learning problem of interest in this work, by providing a formal characterization of the *"environment"* in which the considered learning task will take place. This environment essentially constitutes a *discrete-time Markov Decision Process (DT-MDP)* [2], the structure of which is completely characterized by a quadruple:

$$\mathcal{E} = (X, \mathcal{A}, \mathcal{P}, \mathcal{R}) \tag{1}$$

where the components X, $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{R}$ are further defined as follows:

- X is the finite set of the process *states*, and it is partitioned into a sequence of *"layers"*, X^0, X^1, ..., X^L. $X^0 = \{x^0\}$ and defines the *initial state* of the process, while states $x \in X^L$ are its *terminal states*.

[4] As discussed in Section 3, such algorithms are characterized as *probably approximately correct (PAC)*.

[5] At the same time, our work extends the past results on the R&S problem – in particular, Bechhofer's "indifference-zone" approach [1] – so that they apply to the more general statistical assumptions underlying our work.

[6] An interesting future research issue would be the experimental comparative study of the *"empirical"* performance of these two algorithms.

[7] We also notice, for completeness, that an abridged version of the presented results, which outlines the key ideas but foregoes some of the technical derivations and the discussion provided in this manuscript, can be found in [17].

- $\mathcal{A}$ is a set function defined on X, that maps each state $x \in X$ to the finite, non-empty set $\mathcal{A}(x)$, comprising all the decisions / actions that are feasible in x. It is further assumed that for $x \neq x'$, $A(x) \cap A(x') = \emptyset$. For subsequent development, we also define $|\bar{A}| \equiv max_{x \in X} |\mathcal{A}(x)|$.

- $\mathcal{P}$ is the *state transition function*, defined on $\bigcup_{x \in X} \mathcal{A}(x)$, that associates with every action a in this set a discrete probability distribution $p(\cdot; a)$, that is unknown to the learning agent. The support sets, $\mathcal{S}(a)$, of the distributions $p(\cdot; a)$ are subsets of the state set X that satisfy the following property: For any given action $a \in \mathcal{A}(x)$ with $x \in X^i$ for some $i = 0, \ldots, L-1$, $\mathcal{S}(a) \subseteq \bigcup_{j=i+1}^{L} X^j$; for $a \in \mathcal{A}(x)$ with $x \in X^L$, $\mathcal{S}(a) = X^0$. In words, the previous assumption implies that the environment operates in an *episodic* fashion, where each episode is an *acyclic* traversal of the underlying state space from the initial state to a terminal state. Furthermore, it is assumed that every state $x \in X$ can be reached from the initial state x^0 with positive probability, through some sequence of actions, and that the learning agent knows (i) at least one such sequence of actions for every state, and also (ii) a lower bound, $\underline{q}$, for the corresponding state-reaching probabilities.[8]

- $\mathcal{R}$ is the *immediate reward function*, defined on $\bigcup_{x \in X} \mathcal{A}(x)$, that associates with each action a in this set a probability distribution, $\mathcal{D}(\mu(a), v(a))$, characterizing the *immediate reward* experienced by the *learning agent* every time that action a is selected and executed. The parameters $\mu(a)$ and $v(a)$ denote respectively the *mean* and the *maximum possible magnitude* of the rewards drawn from the distribution $\mathcal{D}(\mu(a), v(a))$, and they take finite values for every a. On the other hand, the only information initially available to the learning agent about $\mathcal{R}()$ is an upper bound $\bar{v}$ for the quantities $v(a),\ a \in \bigcup_{x \in X} \mathcal{A}(x)$.

The learning agent controls the selection of the action to be executed at every state of the environment. More specifically, starting from the initial state x^0 at period $t = 0$, and at every consecutive period $t = 1, 2, 3, \ldots$, the agent (i) observes the current state of the environment, x_t, (ii) selects an action $a_t \in \mathcal{A}(x_t)$ and commands its execution upon the environment, and subsequently (iii) it experiences a reward r_t, where the latter is a random sample drawn from the distribution $\mathcal{D}(\mu(a_t), v(a_t))$. Hence, at the end of some period t, the agent has experienced an entire *"history"*

$$x_0, a_0, r_0, x_1, a_1, r_1, \ldots, x_t, a_t, r_t \tag{2}$$

The ultimate objective of this agent is to determine an action selection scheme – or, in the relevant terminology, a *policy* – π^*, that maps each state $x \in X$ to an action $\pi^*(x) \in \mathcal{A}(x)$ in a way that maximizes the expected total reward experienced over any single episode. Letting M denote the (random) duration of any single episode in terms of number of periods, t, the aforestated objective can be formally expressed as follows:

$$\pi^* = \arg\max_{\pi} E_{\pi}[\sum_{t=0}^{M} r_t | x_0 = x^0] \tag{3}$$

[8] The reader should notice that this characterization of the state transition function ignores the *multi-threading* effect that results from the disassembly operation in the ODP problem. We have opted to omit this particular feature from the basic positioning of the problem considered in this work, since it would complicate the exposition of the main ideas without contributing substantially to the underlying analysis. The extension of the developed results in order to accommodate this particular feature is very straightforward and it is briefly outlined in the concluding section.

In the above equation, the expectation $E_\pi[\cdot]$ is taken over all the possible episode realizations under policy π. It is easy to see that, in the considered operational context, an optimal policy π^* can be obtained through the following simple recursion:

$\forall x \in X^L$,

$$\pi^*(x) := \arg\max_{a \in \mathcal{A}(x)} \{\mu(a)\} \tag{4}$$

$$V^*(x) := \mu(\pi^*(x)) \tag{5}$$

$\forall x \in X^i,\ i = L-1, \ldots, 0$,

$$\pi^*(x) := \arg\max_{a \in \mathcal{A}(x)} \{\mu(a) + \sum_{x' \in \mathcal{S}(a)} p(x'; a) \cdot V^*(x')\} \tag{6}$$

$$V^*(x) := \mu(\pi^*(x)) + \sum_{x' \in \mathcal{S}(\pi^*(x))} p(x'; \pi^*(x)) \cdot V^*(x') \tag{7}$$

The quantity $V^*(x)$ appearing in the above recursion is known as the *(optimal) value* of the corresponding state x, and it expresses the expected return to be collected by the learning agent in a single episode, when it starts from state x and follows the optimal policy π^* until the completion of the episode.

Yet, despite the fact that it provides a pertinent characterization of the optimal policy, the algorithm defined by Equations 4–7 is not directly applicable in the considered problem context, since the quantities $\mu(a)$ and $p(x; a)$, $x \in X,\ a \in \mathcal{A}(x)$, are not initially known to the agent. Furthermore, as discussed in the introductory section, the application of standard RL algorithms, like Q-learning [20], guarantees only asymptotic convergence to optimality, and to the best of our knowledge, currently there are no formal results characterizing the convergence rate of the "standard" Q-learning algorithm to an optimal policy. At the same time, Q-learning is notorious for rather slow convergence. Hence, in the rest of this document, we seek to exploit the special structure of the considered problem in order to derive customized learning algorithms with proven convergence and complexity properties. However, before delving into this discussion, we shall formalize the notions of computational complexity and efficiency to be employed in the considered problem context.

3 Efficient PAC learnability

In computational learning theory [11], a learning algorithm is characterized as *probably approximately correct (PAC)*, if, upon its completion, it provides with probability at least $(1 - \delta)$, an approximation, $\hat{h}$, of the target concept h^*, that differs from h^* by an *"error"*, $\texttt{err}(\hat{h})$, less than or equal to ε. In this definition, both δ and ε are externally specified parameters, and the quantity $\texttt{err}(\hat{h})$ is appropriately specified from the attributes of the target concept h^*. In addition, a PAC algorithm is said to be *efficient*, if it executes in a number of steps that is a polynomial function of $1/\delta$, $1/\varepsilon$, and some additional parameters that characterize the complexity of the learning task and the *"size"* of the target concept h^*.

In the context of the learning problem considered in this work, the target concept is any optimal policy π^*, and a natural solution space is the set Π consisting of all the *deterministic*

policies π that map each state $x \in X$ to a unique action $\pi(x) \in \mathcal{A}(x)$. For these policies, we define:

$$\texttt{err}(\pi) = E_{\pi^*}[\sum_{t=0}^{M} r_t | x_0 = x^0] - E_{\pi}[\sum_{t=0}^{M} r_t | x_0 = x^0] \tag{8}$$

On the other hand, the complexity of the considered learning problem is measured by the magnitude of the environmental parameters $|X|$, $|\bar{A}|$, L, $\bar{v}$, and $1/\underline{q}$, respectively characterizing the size of the task state space, the extent of choice at each state, the length of the decision sequences, the magnitude of the collected rewards, and the difficulty of accessing the various states of the task state space.

In the light of the above characterizations, *an efficient PAC algorithm for the learning problem considered in this work* is defined as follows:

Definition 1: An *efficient PAC* algorithm for the RL problem considered in this work is an algorithm that, for any environment $\mathcal{E} = (X, \mathcal{A}, \mathcal{P}, \mathcal{R})$ and parameters $\delta \in (0, 1/2)$ and $\varepsilon > 0$,

i. will execute in a finite number of steps, that is polynomial with respect to $1/\delta$, $1/\varepsilon$, $|X|$, $|\bar{A}|$, L, $\bar{v}$ and $1/\underline{q}$, and

ii. upon its completion, will return, with probability at least $1-\delta$, a policy $\hat{\pi}$ with $\texttt{err}(\hat{\pi}) \leq \varepsilon$. $\diamond$

The next section discusses the special structure of the considered RL problem that enables the development of an efficient PAC algorithm for it.

4 Developing an efficient PAC learning algorithm for the considered RL problem

The PAC algorithm proposed in this work for the RL problem of Section 2 is strongly based upon the following fundamental observation:

Observation 1: According to Equations 4–7, that characterize the structure of an optimal policy, π^*, for the RL problem considered in this work, we can assess the optimal value $V^*(x)$ of any state $x \in X$ only when we have already computed the optimal values $V^*(x')$ for all the states $x' \in X$ that constitute successor states of x through some action $a \in \mathcal{A}(x)$. Therefore, any learning agent that will base the identification of an optimal policy on the computation of the optimal value function V^*, must acquire its knowledge proceeding from the terminal states, $x \in X^L$, of the underlying state space to the initial state, x^0. This observation subsequently suggests the following basic structure for the proposed algorithm:

- Starting with the set of terminal states, X^L, the proposed algorithm maintains a *"frontier set of actively explored states"*, for which it tries to learn the optimal action $\pi^*(x)$.
- An actively explored state x is assigned an *"apparent optimum action"*, $\hat{\pi}(x)$, when it satisfies a criterion to be defined in the following. At that point, x is declared as *"fully explored"*, and it leaves the "frontier" set, while action $\hat{\pi}(x)$ is the action to be executed at state x, any time that this state is visited until the completion of the algorithm.
- On the other hand, a state $x \in \bigcup_{i=0}^{L-1} X^i$ becomes an actively explored state as soon as all the states $x' \in X$ that constitute successor states of x through some action $a \in \mathcal{A}(x)$, become fully explored.

- The algorithm pursues the above exploration pattern for a pre-defined number of episodes, N, and it terminates either upon the selection of an action $\hat{\pi}(x^0)$, for the initial state x^0, or upon the depletion of the episode budget, N. In the first case, the algorithm returns the computed policy $\hat{\pi}(x),\ \forall x \in X$, while in the second case it reports failure. $\diamond$

Notice that the algorithm defined in Observation 1 can fail either (i) because it did not manage to determine a complete policy $\hat{\pi}(x),\ \forall x \in X$, within the specified episode budget, N, or (ii) because the chosen policy $\hat{\pi}(x),\ \forall x \in X$, had an error $\mathtt{err}(\hat{\pi}) > \varepsilon$. Letting δ^I and δ^{II} denote the respective probabilities of failure according to the modes (i) and (ii), we obtain, by the Bonferroni inequality, that

$$\delta \leq \delta^I + \delta^{II} \tag{9}$$

where δ denotes the total probability of failure of the considered algorithm. Therefore, we can guarantee a success probability of at least $1 - \delta$ for this algorithm, by picking δ^I and δ^{II} such that $\delta^I + \delta^{II} = \delta$. For expository purposes, in the following we shall assume that $\delta^I = \delta^{II} = \delta/2$.

Generally speaking, the proposed algorithm will fail according to mode (ii) only because it was not able to assess adequately the consequences of its various actions upon the environment, which further translates to inadequate observation and exploration of these consequences. Hence, the ability of the proposed algorithm to satisfy a particular PAC requirement, expressed in terms of the tolerated error ε and the failing probability $\delta^{II} = \delta/2$, will depend on the establishment of a pertinent and adequate exploration scheme. On the other hand, in order to prevent failure according to mode (i), the proposed exploration scheme must be *efficient*, i.e., there must exist an episode budget, N, that is polynomially related to $1/\delta$, $1/\varepsilon$, $|X|$, $|\bar{A}|$, L, $\bar{v}$ and $1/\underline{q}$, and will permit the execution of the aforementioned exploration scheme with probability at least $1 - \delta^I = 1 - \delta/2$. We address each of these two issues below.

4.1 Establishing the PAC capability of the proposed algorithm

In this section we establish that the aforestated PAC requirement for the algorithm of Observation 1 – i.e., the requirement that the policy $\hat{\pi}(x),\ \forall x \in X$, returned by the algorithm defined in Observation 1, will have an error $\mathtt{err}(\hat{\pi}) \leq \varepsilon$ with probability at least $1 - \delta^{II} = 1 - \delta/2$ – can be replaced by the following more local requirement: At every actively explored state, x, the algorithm must be able to identify, with a certain probability of success, $1 - \delta(x)$, an action $\hat{\pi}(x) \in \mathcal{A}(x)$ with an expected total reward that differs by at most $\varepsilon(x)$ from the *maximal* expected total reward that can be collected by performing some action $a \in \mathcal{A}(x)$, while in state x, and subsequently following the pre-determined policy, $\hat{\pi}$, until the environment resets itself to the initial state x^0. This localized version of the PAC policy resolution problem can be addressed through results from statistical inference. One particular approach is that presented in the next theorem, which constitutes a generalization of Bechhofer's *"indifference-zone" (IZ)* approach for the "ranking-and-selection" (R&S) problem [1], to populations with bounded general distributions.

Theorem 1 *Suppose that there are k populations distributed according to some bounded general distributions with respect to some attribute of interest, and that $\bar{v}$ constitutes a known uniform absolute bound for this attribute. Furthermore, suppose that the means μ_i of these k populations are unknown, and that it is desired to determine which population has the largest mean μ_i. In particular, the experimenter specifies a* confidence level $1 - \delta$ *and an* "indifference" parameter ε *with the requirement that*

$$\mu_{[k]} - \mu_{[k-1]} \geq \varepsilon \implies PCS \geq 1 - \delta \tag{10}$$

where $\mu_{[1]} \leq \ldots \leq \mu_{[k]}$ *are the ordered population means and* PCS *is the* probability for correct selection*, i.e., the probability of correctly identifying the population with the largest mean* μ_i*.*

Then, this problem can be resolved by:

1. *taking a sample from each of the* k *populations, of* size

$$n = \lceil \frac{4\bar{v}^2}{\varepsilon^2} \ln(\frac{k-1}{\delta}) \rceil \tag{11}$$

2. *computing the corresponding sample means* $\bar{X}_i,\ i = 1, \ldots, k$*, and*

3. *selecting the population with the largest sample mean.*

Proof: The probability for correct selection, PCS, can be bounded as follows:

$$\begin{aligned} PCS &= \Pr\{\text{select } k\} && (12)\\ &= \Pr\{\bar{X}_k > \bar{X}_i,\ \forall i \neq k\} && (13)\\ &= \Pr\{\bar{X}_i - \bar{X}_k - (\mu_i - \mu_k) < -(\mu_i - \mu_k),\ \forall i \neq k\} && (14)\\ &\geq \Pr\{\bar{X}_i - \bar{X}_k - (\mu_i - \mu_k) < \varepsilon,\ \forall i \neq k\} && (15) \end{aligned}$$

Equation 12 above holds by the definition of PCS and the assumption that $\mu_{[k]} = \mu_k$ and $\mu_k - \mu_i \geq \varepsilon,\ \forall i \neq k$, Equation 13 holds from the definition of the selection procedure provided in Theorem 1 (c.f. Step 3), and Equation 15 holds from the fact that $\mu_k - \mu_i \geq \varepsilon,\ \forall i \neq k$. Setting

$$E_i \equiv \{\bar{X}_i - \bar{X}_k - (\mu_i - \mu_k) < \varepsilon\},\ \forall i \neq k \tag{16}$$

and applying the Bonferroni inequality, we get:

$$\begin{aligned} PCS &\geq \Pr(\bigcap_{i=1}^{k-1} E_i) && (17)\\ &\geq 1 - \sum_{i=1}^{k-1} [1 - \Pr(E_i)] && (18)\\ &= 1 - \sum_{i=1}^{k-1} \Pr\{\bar{X}_i - \bar{X}_k - (\mu_i - \mu_k) \geq \varepsilon\} && (19) \end{aligned}$$

Recognizing that each single observation X_i will belong in the interval $[-\bar{v}, \bar{v}]$, for all $i \in \{1, \ldots, k\}$, and using the relevant Hoeffding inequality for the difference of two sample means (c.f. [8], Eq. 2.7) , we get:

$$\Pr\{\bar{X}_i - \bar{X}_k - (\mu_i - \mu_k) \geq \varepsilon\} \ \leq \ e^{-n\varepsilon^2/(2\bar{v})^2} \tag{20}$$

When combined with Equations 17–19, Equation 20 implies that

$$PCS \ \geq \ 1 - (k-1)e^{-n\varepsilon^2/(2\bar{v})^2} \tag{21}$$

Setting

$$1 - (k-1)e^{-n\varepsilon^2/(2\bar{v})^2} \geq 1 - \delta \tag{22}$$

and solving for n, we get:

$$n \geq \frac{4\bar{v}^2}{\varepsilon^2} \ln(\frac{k-1}{\delta}) \tag{23}$$

Equation 23 implies the selection of the sample size n stated in Theorem 1, and concludes the proof. $\diamond$

In order to completely characterize the application of Theorem 1 in the context of the algorithm outlined in Observation 1, we need to specify the parameters $\delta(x)$, $\varepsilon(x)$ and $\bar{v}(x)$ to be employed at each state $x \in X$. The pricing of $\bar{v}(x)$ is a direct consequence of the acyclic structure presumed for the underlying task state space.

Observation 2: For any given state $x \in X^l$, $l = 0, 1, \ldots, L$, any action sequence leading the environment from state x to the initial state x^0 will contain at most $L - l + 1$ actions. As a result, the magnitude of the total reward collected by the execution of any such action sequence will be bounded from above by $(L - l + 1)\bar{v}$. $\diamond$

The pricing of the remaining parameters $\varepsilon(x)$ and $\delta(x)$ is performed through the following two lemmas:

Lemma 1 *Under the assumption that $PCS = 1$ at every state $x \in X$, the policy $\hat{\pi}$, returned by the algorithm defined in Observation 1, will have $\mathtt{err}(\hat{\pi}) \leq \varepsilon$, if the "indifference" parameter $\varepsilon(x)$, employed during the implementation of Theorem 1 at each state $x \in X$, is set to a value $\varepsilon(x) \leq \varepsilon/(L+1)$.*

Proof: Let $V^{\hat{\pi}}(x)$ denote the value of state x under policy $\hat{\pi}$, i.e., the expected total reward to be obtained by starting from state $x \in X$ and following policy $\hat{\pi}$ until the environment resets itself to the initial state x^0. We shall prove Lemma 1 by establishing the stronger result that

$$\forall l = 0, \ldots, L, \ \forall x \in X^l, \quad V^*(x) - V^{\hat{\pi}}(x) \ \leq \ \frac{L - l + 1}{L + 1} \cdot \varepsilon \tag{24}$$

This last result is proven with induction on l, starting from $l = L$ and proceeding to $l = 0$. The satisfaction of the base case for $l = L$ is immediately implied by the definition of $V^*(x)$ and $V^{\hat{\pi}}(x)$ for $x \in X^L$ (e.g., c.f. Equation 5) and the proposed value for the "indifference" parameter, $\varepsilon/(L+1)$. Next, suppose that the inequality of Equation 24 holds true for $x \in \bigcup_{i=l}^{L} X^i$. We shall show that it also holds true for $x \in X^{l-1}$. For this, consider a state $x \in X^{l-1}$ and let $\hat{\pi}(x)$ denote the action selected by policy $\hat{\pi}$. Also, let $Q^{\hat{\pi}}(x, a)$ (resp., $Q^*(x, a)$) denote the expected total reward obtained by initializing the environment at state x, executing the action $a \in \mathcal{A}(x)$ in that state, and following the policy $\hat{\pi}$ (resp., an optimal policy π^*) thereafter, until the environment resets itself to state x^0. Finally, let $\hat{a} = \arg\max_{a \in \mathcal{A}(x)}\{Q^{\hat{\pi}}(x, a)\}$ and $a^* = \arg\max_{a \in \mathcal{A}(x)}\{Q^*(x, a)\}$. Then,

$$\begin{aligned}
V^*(x) - V^{\hat{\pi}}(x) &= Q^*(x, a^*) - Q^{\hat{\pi}}(x, \hat{\pi}(x)) && (25)\\
&= Q^*(x, a^*) - Q^{\hat{\pi}}(x, \hat{a}) + \\
&\quad Q^{\hat{\pi}}(x, \hat{a}) - Q^{\hat{\pi}}(x, \hat{\pi}(x)) && (26)\\
&\leq Q^*(x, a^*) - Q^{\hat{\pi}}(x, \hat{a}) + \frac{\varepsilon}{L+1} && (27)\\
&= Q^*(x, a^*) - Q^{\hat{\pi}}(x, a^*) + Q^{\hat{\pi}}(x, a^*) \\
&\quad - Q^{\hat{\pi}}(x, \hat{a}) + \frac{\varepsilon}{L+1} && (28)\\
&\leq Q^*(x, a^*) - Q^{\hat{\pi}}(x, a^*) + \frac{\varepsilon}{L+1} && (29)
\end{aligned}$$

Equation 25 is an immediate consequence of the definitions of $V^*(x)$, $V^{\hat{\pi}}(x)$, $Q^*()$, $Q^{\hat{\pi}}()$, a^*, and $\hat{\pi}(x)$. Equation 27 results from the definition of the "indifference" parameter $\varepsilon(x)$ for state x and the assumption that $PCS = 1$ at every node $x \in X$. Finally, Equation 29 results from the definition of $\hat{a}$. We also have that:

$$\begin{aligned} Q^*(x,a^*) - Q^{\hat{\pi}}(x,a^*) &= \sum_{x' \in \mathcal{S}(a^*)} p(x';a^*) \cdot \\ &\quad [V^*(x') - V^{\hat{\pi}}(x')] \qquad (30) \\ &\leq [\sum_{x' \in \mathcal{S}(a^*)} p(x';a^*)] \cdot \\ &\quad \frac{L-l+1}{L+1} \cdot \varepsilon \qquad (31) \\ &= \frac{L-l+1}{L+1} \cdot \varepsilon \qquad (32) \end{aligned}$$

Equation 30 results from the definition of $Q^*()$ and $Q^{\pi}()$, Equation 31 results from the induction hypothesis, and Equation 32 results from the fact that $\mathcal{S}(a)$ is the support set for the discrete distribution $p(\cdot;a)$. The combination of Equations 29 and 32 gives:

$$\begin{aligned} V^*(x) - V^{\hat{\pi}}(x) &\leq \frac{L-l+1}{L+1} \cdot \varepsilon + \frac{1}{L+1} \cdot \varepsilon \qquad (33) \\ &= \frac{L-(l-1)+1}{L+1} \cdot \varepsilon \qquad (34) \end{aligned}$$

and completes the inductive argument for the proof of Equation 24.

Finally, the proof of Lemma 1 is established by applying Equation 24 for $l = 0$. $\diamond$

Lemma 2 *The policy $\hat{\pi}$, returned by the algorithm defined in Observation 1, will have $\mathtt{err}(\hat{\pi}) \leq \varepsilon$, with probability at least $1 - \delta/2$, if, during the implementation of Theorem 1 at each node $x \in X$,*

1. *the "indifference" parameter $\varepsilon(x)$ is set to $\varepsilon(x) = \varepsilon/(L+1)$, and*
2. *the PCS parameter $\delta(x)$ is set to $\delta(x) = \delta/(2|X|)$.*

Proof: The validity of Lemma 2 is an immediate consequence of the proof of Lemma 1, when noticing that, for a nodal PCS value of $1-\delta(x)$, the conditions of Lemma 1 will hold with probability $[1-\delta(x)]^{|X|}$. Hence, setting $1-\delta/2 \leq [1-\delta(x)]^{|X|}$, we obtain $\delta(x) \leq 1-(1-\delta/2)^{1/|X|}$. The result of Lemma 2 is implied from this last inequality, when noticing that, for $\delta \in (0,1)$, $(1-\delta/2)^{1/|X|} \leq 1 - (1/|X|) \cdot \delta/2$. $\diamond$

4.2 Establishing the efficiency of the proposed algorithm

In order to establish the efficiency of the proposed algorithm, we need to show that, for any environment $\mathcal{E} = (X, \mathcal{A}, \mathcal{P}, \mathcal{R})$ and parameters $\delta \in (0, 1/2)$ and $\varepsilon > 0$, the sampling scheme defined by Observation 1, the results of Theorem 1, Observation 2 and Lemma 2 can be executed, with probability of success at least $1 - \delta^I = 1 - \delta/2$, within an episode budget, N, that is polynomially related to $1/\delta$, $1/\varepsilon$, $|X|$, $|\bar{A}|$, L, $\bar{v}$ and $1/\underline{q}$. This result is established in the rest of this section. In the subsequent discussion, $\Psi(x,a)$ denotes an observation of the total reward

obtained by the learning agent when selecting action $a \in \mathcal{A}(x)$, while in state $x \in X$, and subsequently following the pre-determined policy $\hat{\pi}$ until the end of the running episode.

As a first step, we argue that focusing on the number of the budgeted episodes in order to establish the polynomial complexity of the proposed algorithm is justifiable, since the execution of each single episode is of polynomial complexity with respect to the parameters of interest. This result is formally stated in the following observation:

Observation 3: The combined computational cost experienced by the proposed algorithm with respect to (i) the action selection and (ii) the collection and processing of a single observation, $\Psi(x, a)$, during any single episode, is $O(L|\bar{A}|)$. Also, for any actively explored state x that has completed the sampling requirements of Theorem 1, the determination of the apparent optimum action $\hat{\pi}(x)$ is of complexity $O(|\bar{A}|)$. Hence, the overall computational cost experienced by the proposed algorithm during any single episode is $O((L+1)|\bar{A}|)$. $\diamond$

On the other hand, the total number of observations $\Psi(x, a)$ that must be taken across all the state-action pairs, (x, a), of the environment, is $\sigma = \sum_{x \in X} |\mathcal{A}(x)| \cdot n(x)$, where $n(x)$ is obtained from Equation 11, by substituting: (i) k with $|\mathcal{A}(x)|$; (ii) $\bar{v}$ with $(L - l + 1)\bar{v}$, where l is the level of node x; (iii) ε with $\varepsilon(x) = \varepsilon/(L+1)$; and (iv) δ with $\delta(x) = \delta^{II}/|X| = \delta/(2|X|)$. Hence, we have:

Observation 4: The total number of observations, $\Psi(x, a)$, that must be taken across all the state-action pairs, (x, a), of the environment, is

$$\sigma = O(\frac{|X||\bar{A}|(L+1)^4\bar{v}^2}{\varepsilon^2} \ln(\frac{|X||\bar{A}|}{\delta})) \tag{35}$$

i.e., σ is a polynomial function of $1/\varepsilon$, $1/\delta$ and the parameters $|\bar{A}|$, $|X|$, L and $\bar{v}$, that characterize the problem "size". $\diamond$

However, the stochastic nature of the state transitions taking place in the considered environment, implies that the number of episodes $n(\sigma)$ required to collect these σ observations will be, in general, larger than σ; in fact, $n(\sigma)$ can be infinitely large, in the worst case. A systematic characterization of the statistics of $n(\sigma)$ can be facilitated by the following observation.

Observation 5: Under the assumption that, at every unexplored state x, the proposed algorithm selects the exercised action $a \in \mathcal{A}(x)$ in a way that maintains a positive probability for accessing the set of actively explored states, the materialization of an observation $\Psi(x, a)$ at any single episode constitutes a *Bernoulli trial* [4], with its probability of success bounded from below by $\underline{q}$. $\diamond$

Next, we show that the combination of Observations 4 and 5 enables the determination of an episode budget, N, that is polynomially related to the problem parameters and will suffice for the collection of the σ requested observations, with probability at least $1 - \delta^I = 1 - \delta/2$. For this, we make the very conservative but simplifying assumption that the requested observations, $\Psi(x, a)$, are pursued one at a time; i.e., at every single episode, the algorithm focuses on a particular observation $\Psi(x, a)$ that it tries to achieve, and it ignores any other potentially available observations. Focusing on this particular algorithmic implementation enables the determination of the required budget, N, according to the decomposing scheme described below in Observation 6. Furthermore, the derived result remains applicable to the more practical algorithmic implementations where actively explored states are sampled in parallel, since these more realistic operational schemes do increase the probability of reaching an actively explored state at any single episode.

Observation 6: An episode budget, N, that is adequate for the collection of the σ observations of Equation 35, with probability at least $1 - \delta^I = 1 - \delta/2$, and is polynomially related to the problem parameters $1/\delta$, $1/\varepsilon$, $|X|$, $|\bar{A}|$, L, $\bar{v}$, $1/\underline{q}$, can be obtained by:

a. first determining an episode budget, $N(x,a)$, that (i) will enable the considered algorithm to obtain a single observation $\Psi(x,a)$ with probability of success $1-\delta^I/\sigma = 1-\delta/(2\sigma)$ and (ii) it is polynomially related to σ and the aforementioned parameters of interest, and

b. subsequently setting $N = \sigma \cdot N(x,a)$.

$\diamond$

The result of Observation 6 is an immediate consequence of the application of the Bonferroni inequality to the particular algorithmic implementation described above. The next lemma determines an episode budget $N(x,a)$ that satisfies the requirements of Observation 6.

Lemma 3 *An episode budget, $N(x,a)$, that guarantees the collection of a single observation, $\Psi(x,a)$, with probability at least $1-\delta/(2\sigma)$, is*

$$N(x,a) = \lceil \frac{1}{\underline{q}} \ln(\frac{2\sigma}{\delta}) \rceil \tag{36}$$

Proof: It is well known from basic probability theory [4], that the number of failures, y, before the first success, in a sequence of independent Bernoulli trials with success probability q, follows a geometric distribution with cdf

$$F(y) = \begin{cases} 1-(1-q)^{\lfloor y \rfloor + 1} & \text{if } y \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

In the context of Lemma 3, we are essentially requesting that the number of failures experienced in the involved sequence of Bernoulli trials, does not exceed $N(x,a)-1$ with probability at least $1-\delta/(2\sigma)$. Hence, according to Equation 37, the allocated budget, $N(x,a)$, must satisfy:

$$1-(1-\underline{q})^{N(x,a)} \geq 1-\delta/(2\sigma) \iff \tag{38}$$

$$(1-\underline{q})^{N(x,a)} \leq \delta/(2\sigma) \tag{39}$$

Notice that, since Equation 38 applies for any state-action pair (x,a), we have used the minimal success probability $\underline{q}$. From the well-known inequality $1-y \leq e^{-y}$, it follows that Equation 39 can be satisfied by picking $N(x,a)$ such that:

$$e^{-N(x,a)\underline{q}} \leq \delta/(2\sigma) \tag{40}$$

Solving Equation 40 for $N(x,a)$, we obtain:

$$N(x,a) \geq \frac{1}{\underline{q}} \ln(\frac{2\sigma}{\delta}) \tag{41}$$

which proves the validity of the lemma. $\diamond$

The above discussion can be recapitulated as follows: Observation 3 establishes that the execution of a single episode under the proposed algorithm has a computational cost that is a polynomial function of the problem-defining parameters, $|X|$, $|\bar{A}|$, L, $\bar{v}$ and $1/\underline{q}$. Furthermore, Observation 4 establishes that the total number of observations, σ, that must be collected across all state-action pairs (x,a), in order to guarantee the PAC performance of the proposed algorithm, is polynomially related to the parameters $1/\varepsilon$, $1/\delta$, $|X|$, $|\bar{A}|$, L, $\bar{v}$ and $1/\underline{q}$. Observation 5 establishes that the acquisition of a single observation, $\Psi(x,a)$, can be perceived as a Bernoulli

trial with its success probability bounded from below by $\underline{q}$, and Lemma 3 exploits this result in order to determine an episode budget, $N(x,a)$, that enables the acquisition of $\Psi(x,a)$ with probability at least $1-\delta/(2\sigma)$ and is polynomially related to the parameters $1/\varepsilon$, $1/\delta$, $|X|$, $|\bar{A}|$, L, $\bar{v}$ and $1/\underline{q}$. Finally, Observation 6 establishes that the episode budget $N=\sigma\cdot N(x,a)$ is adequate for collecting the requested σ observations with probability $1-\delta^I=1-\delta/2$ and it remains a polynomial function of $1/\varepsilon$, $1/\delta$, $|X|$, $|\bar{A}|$, L, $\bar{v}$ and $1/\underline{q}$. Hence, the proposed algorithm is *efficient*. A detailed description of this algorithm, according to its basic characterization in Observation 1 and its further parametrization through the results of Sections 4.1 and 4.2, is provided in Figure 2. Furthermore, the next theorem is an direct consequence of all the previous developments.

Theorem 2 *The algorithm of Figure 2 is an efficient PAC algorithm for the RL problem considered in this work.*

5 Conclusions

In this paper we investigated the RL problem for the case that the underlying target task evolves in an "episodic" manner over a state space with a well-defined initial state and acyclic structure. The departing point of our analysis was the observation that the acyclic structure of the underlying state space implies a certain information flow for the overall learning process, which admits a natural and effective translation to an exploration strategy. Subsequently, the main body of our results combined this exploration strategy with further relevant results from the area of statistical inference, in order to design a RL algorithm customized to the special structure of the considered problem. The derived algorithm is computationally simple, and therefore, easily implementable in "real-world" applications. It was also shown to be efficient, according to the formal characterizations of efficiency provided by computational learning theory. By taking advantage of the admittedly simpler structure of the RL problem considered in this work, the proposed algorithm presents significantly lower computational complexity than the efficient PAC learning algorithm proposed in [9, 10] for more general RL problems, and it also compares favorably, in terms of computational complexity and implementational feasibility, with the algorithm developed in [5], which also addresses episodic RL problems evolving over acyclic state spaces. Finally, an additional contribution of the presented work was the extension of the applied statistical theory itself, through Theorem 1, which establishes a new criterion for Ranking & Selection that is applicable to populations with general distributions.

We should also notice, at this point, that the developed algorithm is immediately extensible to the more complex version of the problem where, at every episode, a task is splitting to a number of subtasks that execute in parallel and contribute to the total reward collected by the learning agent. This is actually the case with the ODP problem described in the introductory section, where a single episode involves the concurrent processing of all the items extracted at the different stages of the disassembly process. The only necessary modification for accommodating this additional problem element concerns the appropriate evaluation of the quantity $\bar{v}(x)$ in a way that it applies to the notion of observations $\Psi(x,a)$ experienced by the learning agent in this new operational context, and we leave the resolution of this technical detail to the reader. Furthermore, the proposed methodology is also extensible to the case of RL problems where the immediate rewards can be drawn from unbounded distributions, which, however, possess bounded mean and variance. Under the assumption that the learning agent possesses a uniform upper bound, $\bar{v}$, for the variances of the aforementioned distributions, and using the Chebychev instead of the Hoeffding inequality in the relevant derivation, one can establish

Input: L; $X^l,\ l = 0, \ldots, L$; $\mathcal{A}(x), \forall x \in X$; $\bar{v}$; $\underline{q}$; ε; δ
Output (under successful completion): $\hat{\pi}(x), \forall x \in X$

I. Initialization

(a) Compute $X \equiv \bigcup_{l=0}^{L} X^l$; $|X|$; $|\mathcal{A}(x)|, \forall x \in X$;

(b) Set
$\bar{v}(x) := (L - l + 1)\bar{v},\ \forall l = 0, \ldots, L,\ \forall x \in X^l$;
$\varepsilon(x) := \varepsilon/(L+1),\ \forall x \in X$;
$\delta(x) := \delta/(2|X|),\ \forall x \in X$;
$n(x) := \lceil \frac{4\bar{v}(x)^2}{\varepsilon(x)^2} \ln(\frac{|A(x)|-1}{\delta(x)}) \rceil, \forall x \in X$;
$\sigma := \sum_{x \in X} |\mathcal{A}(x)| n(x)$;
$N := \sigma \lceil (1/\underline{q}) \ln(2\sigma/\delta) \rceil$;
$Q(x,a) := 0, \forall x \in X, \forall a \in \mathcal{A}(x)$;
$O(x,a) := 0, \forall x \in X, \forall a \in \mathcal{A}(x)$;
$AE := X^L$; $UE := \bigcup_{l=0}^{L-1} X^l$;
$i := 1$

II. Policy Computation
while $(AE \neq \emptyset \wedge i \leq N)$ do

(a) Initiate a new episode by placing a token at the initial state, x^0, and try to route this token to an actively explored state, $x \in AE$, by picking actions that maintain a positive probability to reach such a state;

(b) If successful

i. select an action $a \in \mathcal{A}(x)$ for which $O(x,a) < n(x)$;

ii. obtain an observation $\Psi(x,a)$, by accumulating the total reward obtained by exercising action a at state x, and subsequently following the pre-computed policy $\hat{\pi}$ until the termination of the current episode;

iii. $Q(x,a) := Q(x,a) + \Psi(x,a)$; $O(x,a) := O(x,a) + 1$;

iv. If $(O(x,a) = n(x))$

- $Q(x,a) := Q(x,a)/n(x)$;
- If $(\forall a' \in \mathcal{A}(x), O(x,a') = n(x))$
  - $\hat{\pi}(x) := \arg\max_{a \in \mathcal{A}(x)} \{Q(x,a)\}$;
  - remove state x from AE;
  - Remove from UE every state $x' \in UE$ for which all the immediately successor states are not in $AE \cup UE$, and add them to AE.

(c) $i := i + 1$;

endwhile

III. Exit
If $(AE = \emptyset)$ return $\hat{\pi}(x), \forall x \in X$, else report failure

Figure 2: The proposed PAC algorithm for the RL problem considered in this work

a R&S criterion similar to that of Theorem 1, with the new sample size being equal to $n = \lceil \frac{2(k-1)\bar{v}}{\varepsilon^2\delta} \rceil$, and with the parameters k, ε and δ having the same interpretation with that stated in Theorem 1. A third important aspect of the results developed herein is that they are directly applicable to *partially observable (PO-)* MDP's. This capability stems from the direct, on-line nature of the proposed algorithm, which enables it to forego the explicit characterization of the internal system dynamics, and to work exclusively on the space induced by the measurement / observation sequences. This situation is exemplified by the state space definition of the ODP problem discussed in the introductory section, and it is reminiscent of the class of *Augmented MDP (AMDP)* algorithms in the emerging PO-MDP literature (cf. [19], Chpt. 16).

When viewed from a more practical implementational standpoint, the PAC nature of the proposed algorithm implies that it should be perceived as the *"Phase I"* of a broader learning process, during which the agent tries to learn a (near-)optimal policy fast and with very high probability. Once the execution of this algorithm has been completed, the agent will switch to *"Phase II"*, where a more standard – e.g., the Q-learning – RL algorithm will be employed, in a way that incorporates and exploits the information obtained in Phase I. Notice that maintaining some active exploration in Phase II is important, since (i) this exploration can counter-balance the potential of error tolerated, through the error probability δ, in Phase I, and (ii) it enables the reaction of the learning agent to any non-stationarity of the environmental parameters. On the other hand, the above interpretation of the presented algorithm as a "Phase I" computation in a broader learning process naturally raises the question of how much effort should be expended on it. Clearly, this effort depends on the "tightness" of the PAC requirement, as expressed by the values of the parameters ε and δ, and while the resolution of this issue will be context-specific, in general, some relevant observations are in order. First of all, it should be clear from the above developments that the proposed algorithm is more *"exploration"* than *"exploitation"*-oriented. More specifically, during the algorithm execution, the primary concern underlying the applied action selection policy is the coverage of the necessary sampling requirements that will lead to the identification of a target ε-optimal policy, rather than the maximization of the value accumulated during that period. Such a strategy is consistent with the implicit *stationarity* assumption underlying the problem statement, since in that case, the earlier an optimized strategy is identified, the higher the long-run profitability of this strategy will be. On the other hand, things are different in a non-stationary operational context. In that case, expending an extensive effort to find an optimized policy for the prevailing conditions might be futile, since this policy will be rendered irrelevant by the future evolution of the system dynamics. In fact, for highly non-stationary environments, the execution of the considered algorithm might not be even feasible, since the system sojourn in any particular parametric regime might not be long enough in order to perform the necessary sampling. Hence, in the case of non-stationary operational environments, one should compromise for a rapidly obtainable policy with a decent performance and maintain a high level of exploration in the algorithm implementing the "Phase II" computation.[9] From a more technical standpoint, the selection of a pertinent value for the performance parameter ε, that characterizes the suboptimality of the derived policy, should be relativized with respect to the magnitude of the expected immediate rewards. In fact, this relativistic interpretation of the value of ε is applied automatically by the algorithm of Figure 2, since in the calculation of $n(x)$ – the only place where the parameter ε is actually involved during the algorithm implementation – the factor $4\bar{v}(x)^2/\varepsilon(x)^2$ can be rewritten as $1/(\varepsilon(x)/(2\bar{v}(x)))^2$. At the same time, the formulae determining the episode budget, N, in Figure 2, also reveal

[9] More generally, there is an obvious trade-off between the "tightness" expressed by the parameters ε and δ, and the level of exploration that must be maintained in Phase II.

that this quantity is affected by the second performance parameter, δ, only through the value of $\ln(1/\delta)$. Hence, one can afford to be more demanding regarding the *success* of the computation performed in Phase I rather than the *quality* of the result of this computation. This last remark corroborates the above suggestion that, in many applications, a pertinent implementation of the developed algorithm should seek to compute with high success an initial near-optimal policy, rather than expend a very large amount of effort for getting (very close) to the optimal policy.

Future work will seek to enhance the efficiency of the developed algorithm and to detail its implementation from a more practical standpoint. It is clear from all the previous discussion that the main focus of this work was the establishment of the PAC nature of the proposed algorithm and its efficiency, where this last concept was interpreted according to the definitions provided by computational learning theory. In particular, all the presented developments sought to explicitly establish the ability of the proposed algorithm to guarantee the PAC requirement, within a number of episodes that is polynomially related to the parameters of interest, rather than provide the tightest possible bound for such an episode budget. Hence, in a first line of our future research, we shall seek to identify a tighter bound for the episode budget, N, that will guarantee the PAC performance of the algorithm. Such an immediate improvement can be achieved, for instance, by replacing the calculation $N = \sigma N(x, a) = \sigma\lceil(1/\underline{q})\ln(2\sigma/\delta)\rceil$, in the algorithm of Figure 2, with another calculation that computes the required episode budget, N, by inverting the binomial distribution for any given triplet of σ, $\underline{q}$ and $1 - \delta^I$. In a similar vein, we shall consider the possibility of replacing the R&S criterion of Theorem 1 with other R&S criteria that will employ sampling techniques of more sequential nature, e.g., similar to those discussed in [3, 12]. Finally, two additional issues that concern the further detailing of the implementation of the algorithm outlined in Figure 2, are (i) the design of more pertinent strategies to be followed by the algorithm when trying to obtain the required samples $\Psi(x, a)$ for the different state-action pairs (x, a), and (ii) the organization of the information provided in the collected rewards in a concise set of data structures that will enable the more expedient application of the applied R&S criteria. Obviously, all the above issues are extensively interrelated and constitute part of our current investigations.

Acknowledgement

This work was partially supported by NSF grants DMI-MES-0318657 and CMMI-0619978. The first author would like to thank Anton Kleywegt for his comments that instigated this line of research, and Seong-Hee Kim for providing the insightful technical report [12] on the R&S problem.

References

[1] R. E. Bechhofer. A single-sample multiple decision procedure for ranking means of normal populations with known variances. *Annals of Mathematical Statistics*, 25:16–39, 1954.

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

[3] E. Even-Dar, S. Mannor, and Y. Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *Proceedings of COLT'02*, pages 255–270, 2002.

[4] W. Feller. *An Introduction to Probability Theory and its Applications, Vol. II (2nd. ed.).* Wiley, N.Y., 1971.

[5] C. N. Fiechter. Efficient reinforcement learning. In *Proceedings of COLT'94*, pages 88–97. ACM, 1994.

[6] C. N. Fiechter. Expected mistake bound model for on-line reinforcement learning. In *Proceedings of ICML'97*, pages 116–124. AAAI, 1997.

[7] J. Heizer and B. Render. *Operations Management (7th ed.)*. Pearson/Prentice Hall, Upper Saddle River, N.J., 2004.

[8] W. Hoeffding. Probability inequalities for sum of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[9] M. Kearns and S. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. *Neural Information Processing Systems*, 11:996–1002, 1999.

[10] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

[11] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, Cambridge, MA, 1994.

[12] S.-H. Kim and B. L. Nelson. Selecting the best system. Technical report, School of Industrial & Systems Eng., Georgia Tech, 2004.

[13] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[14] S. A. Reveliotis. Uncertainty management in optimal disassembly planning through learning-based strategies. In *Proceedings of the NSF–IEEE–ORSI Intl. Workshop on IT-enabled Manufacturing, Logistics and Supply Chain Management*, pages 135–141. NSF/IEEE/ORSI, 2003.

[15] S. A. Reveliotis. Modelling and controlling uncertainty in optimal disassembly planning through reinforcement learning. In *IEEE Intl. Conf. on Robotics & Automation*, pages 2625–2632. IEEE, 2004.

[16] S. A. Reveliotis. Uncertainty management in optimal disassembly planning through learning-based strategies. *IIE Trans.*, 39:645–658, 2007.

[17] S. A. Reveliotis and T. Bountourelis. Efficient learning algorithms for episodic tasks with acyclic state spaces. In *Proceedings of the 2006 IEEE Intl. Conf. on Automation Science and Engineering*, pages 421–428. IEEE, 2006.

[18] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 2000.

[19] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, MA, 2005.

[20] C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, UK, 1989.