stats

MDPI

Article

An ϵ-Greedy Multiarmed Bandit Approach to Markov Decision Processes †

Isa Muqattash [1] and Jiaqiao Hu [2,*]

[1] Independent Researcher, Stony Brook, NY 11794-3600, USA; ismuqattash@gmail.com
[2] Department of Applied Mathematics and Statistics, The State University of New York at Stony Brook, Stony Brook, NY 11794-3600, USA
* Correspondence: jiaqiao.hu.1@stonybrook.edu
† This work was completed while the author was a PhD student at the Department of Applied Mathematics and Statistics, SUNY at Stony Brook, Stony Brook, NY 11794-3600, USA.

Abstract: We present REGA, a new adaptive-sampling-based algorithm for the control of finite-horizon Markov decision processes (MDPs) with very large state spaces and small action spaces. We apply a variant of the ϵ-greedy multiarmed bandit algorithm to each stage of the MDP in a recursive manner, thus computing an estimation of the "reward-to-go" value at each stage of the MDP. We provide a finite-time analysis of REGA. In particular, we provide a bound on the probability that the approximation error exceeds a given threshold, where the bound is given in terms of the number of samples collected at each stage of the MDP. We empirically compare REGA against another sampling-based algorithm called RASA by running simulations against the *SysAdmin* benchmark problem with 2^{10} states. The results show that REGA and RASA achieved similar performance. Moreover, REGA and RASA empirically outperformed an implementation of the algorithm that uses the "original" ϵ-greedy algorithm that commonly appears in the literature.

Keywords: multiarmed bandits; epsilon-greedy method; Markov decision process (MDP); sampling; optimization under uncertainties

Citation: Muqattash, I.; Hu, J. An ϵ-Greedy Multiarmed Bandit Approach to Markov Decision Processes. *Stats* **2023**, *6*, 99–112. https://doi.org/10.3390/stats6010006

Academic Editor: Stéphane Mussard

Received: 19 November 2022
Revised: 22 December 2022
Accepted: 23 December 2022
Published: 1 January 2023

1. Introduction

Consider a finite-horizon Markov decision process (MDP) with non-negative rewards. Let A and S denote finite action and state spaces, respectively, and let H denote the horizon of the MDP. For any stage $i = 0, \ldots, H-1$ and state $s_i \in S$, choosing a control $a_i \in A$ (stochastically) leads to a future state s_{i+1}. The system dynamics can be captured by $s_{i+1} = f(s_i, a_i, w_i)$, where $w_i \sim U(0,1)$ is a uniformly distributed random variable representing the uncertainly of the system (*c.f.* [1,2]). Let $R(s_i, a_i, w_i)$ denote the immediate reward, and let Π be the set of all possible deterministic nonstationary Markovian policies $\pi = \{\pi_i | \pi_i : S \to A, i = 0, \ldots, H-1\}$. To simplify our notation, let $w = (w_i, w_{i+1}, \ldots, w_{H-1})$. Given a policy $\pi \in \Pi$, the reward-to-go value function is given by $V_i^\pi(s_i) = E_w^\pi\left[\sum_{t=i}^{H-1} R(s_t, \pi_t(s_t), w_t)\right]$, with $V_H^\pi(s) = 0 \, \forall \, s \in S$. Let the optimal reward-to-go value function be denoted by $V_i^*(s) = \max_{\pi \in \Pi} V_i^\pi(s)$, with $V_H^*(s) = 0 \, \forall \, s \in S$, then $V_i^*(s)$ can be computed recursively using

$$V_i^*(s) = \max_{a \in A} Q_i^*(s,a)$$
$$Q_i^*(s,a) = E_w^\pi\left[R(s,a,w) + V_{i+1}^*(f(s,a,w))\right].$$

In this paper, we state our results for the objective of maximizing the total reward-to-go value for a fixed initial state s_0. When clear from the context, we omit the subscript for the initial horizon and state, and use $V^*(s)$ instead of $V_0^*(s_0)$. The results can be trivially restated for minimization problems.

Recent years have seen theoretical progress in the area of simulation-based methods for solving large finite-horizon MDPs. Two algorithms closely related to the method that we propose in this paper are the recursive automata sampling algorithm (RASA) of [3] and the adaptive multistage sampling algorithm (AMS) of [4]. RASA recursively applies a learning automata algorithm at each stage of the MDP in order to calculate the reward-to-go value function. When applied to the initial state of the MDP, the algorithm produces an approximated value for the MDP. The objective is to maximize the total reward criterion. In [3], the authors give two probabilistic bounds on the performance of RASA as a function of the number of samples at each stage. In particular, a lower bound on the probability that the algorithm will sample the optimal action, and an upper bound on the probability that the error of the algorithm exceeds a given threshold.

The AMS algorithm is very similar to RASA, except that learning automata are replaced with an upper confidence bounds (UCBs) multiarmed bandit (MAB) algorithm to drive the simulation at each stage of the MDP. In [4], the authors showed that the MDP value function estimator produced by AMS is asymptotically unbiased. This asymptotic convergence result is weaker than the convergence in probability result shown for RASA. The value function estimator required to show the theoretical results for AMS is based on weighted averaging of the Q-function estimates of each sampled action, where the weights are taken on the basis of the number of times each action is chosen for simulation by the underlying MAB algorithm. Thus, the resultant value function estimator is biased below the true value function of the MDP.

Other simulation-based approaches for solving MDPs include stochastic annealing ([5]), neurodynamic programming ([6]), stochastic neural networks ([7]), and Q-learning ([8,9]).

In this paper, we propose a new algorithm called the recursive epsilon-greedy algorithm (REGA), which combines ideas from RASA and AMS. Like AMS, the sampling strategy at each stage of the MDP is based on a multiarmed bandit model. In particular, we use a generalized variant of Auer et al.'s ϵ-greedy algorithm (*c.f.* [10]). On the other hand, our theoretical analysis and the value function estimator that we use are similar to those of RASA. We give a finite-time analysis of REGA following the same ideas as those found in [3]. In particular, we give an upper bound on the probability that the approximated reward-to-go generated by REGA has an absolute error larger than a tolerated threshold $\epsilon > 0$. We provide a convergence rate in terms of the number of samples captured at each stage of the MDP. Much credit goes to [3] for the ideas behind our theoretical analysis.

The rest of the paper is organized as follows: The REGA algorithm is introduced in Section 2. Section 3 provides a finite-time analysis of REGA. Empirical results in comparison to the RASA algorithm appear in Section 4. We conclude with some remarks in Section 5.

This article is based on Chapter 2 of the first author's PhD thesis [11].

2. The Algorithm

MAB problems are well-studied in the literature (*c.f.* [10,12–14]), and provide a mechanism for the trade-off between exploration and exploitation. In this paper, we consider finite-horizon MDPs with finite action and state spaces A and S and finite horizon H. Suppose that the system dynamics of the MDP are not known. The objective is to use a simulation-based technique to approximate the total reward of the MDP. Instead of simulating all permissible actions an equal number of times, it is reasonable to allocate more simulations towards actions with a likely higher reward to go. This becomes more important in settings where simulation resources are limited.

Consider a single stage $i = 0, 1, \ldots, H-1$ and current state $s \in S$, and let $c_i \geq 1$ be tuneable parameters that control the amount of pure exploration that takes place at each stage i of the algorithm. Let $\hat{Q}_{i,M_i}(s,a)$ and $\hat{V}_{i,M_i}(s) = \underset{a \in A(s)}{\operatorname{argmax}}\, \hat{Q}_{i,M_i}(s,a)$ denote estimates of $Q_i^*(s,a)$ and $V_i^*(s)$, respectively, when M_i simulations are allocated to stage i. When clear from the context, we drop the suffix M_i, and use simplified notations $\hat{Q}_i(s,a)$ and $\hat{V}_i(s)$. By convention, we set $\hat{V}_H(s) = 0$. Moreover, let $M_a^i(s)$ denote the number of times action a

is sampled at stage i. It always holds that $M_i = \sum_{a \in A(s)} M_a^i(s)$. Lastly, for a non-negative integer m and a state s, let $\{\epsilon_m^s\}_m$ be the nonincreasing sequence given by

$$\epsilon_m^s = \min\left\{1, \frac{c_i|A(s)|}{\sqrt{m}}\right\}, \tag{1}$$

where $A(s)$ is the set of available actions at state s, and $|A(s)|$ is the size of $A(s)$. When the state s is clear from the context, we drop the superscript and use the simplified notation ϵ_m instead of ϵ_m^s. The REGA algorithm is outlined in Algorithm 1.

The widely cited ϵ-greedy method of [10] defines ϵ_m^s differently, where denominator $\sqrt{m}$ in Equation (1) is replaced with m. See Section 4 for a comparison of empirical results.

Algorithm 1: Bytecode runtime measurement

Input: Stage $i < H$, state $s \in S$, $c_i \geq 1$, and $M_i \geq 1$.
Initialization: $\hat{V}_H(s) = 0$, $\hat{Q}_i(s,a) = 0$ and $M_a^i(s) = 0 \quad \forall\, a \in A(s)$.
Loop (For $m = 1, 2, \ldots, M_i$)**:**
{
1. Update the current optimal action $\hat{a}_i$.

$$\hat{a}_i = \underset{a \in A(s)}{\operatorname{argmax}}\, \hat{Q}_i(s,a)$$

2. Pick the next action to sample

$$\bar{a}_i = \begin{cases} \hat{a}_i, & w.p.\ (1-\epsilon_m) \\ \text{(uniformly) random arm}, & w.p.\ \epsilon_m \end{cases}$$

3. Generate $w_m^i \sim U(0,1)$ then simulate the next state $f(s, \bar{a}_i, w_m^i)$ and immediate reward $R(s, \bar{a}_i, w_m^i)$.
4. Update $\hat{Q}_i(s, \bar{a}_i) =$

$$\frac{M_{\bar{a}_i}^i(s)\hat{Q}_i(s,\bar{a}_i) + \hat{V}_{i+1}\left(f(s,\bar{a}_i,w_m^i)\right) + R(s,\bar{a}_i,w_m^i)}{M_{\bar{a}_i}^i(s)+1}. \tag{2}$$

5. Update $M_{\bar{a}_i}^i(s) = M_{\bar{a}_i}^i(s) + 1$.
}
Output: $\hat{V}_i(s) = \underset{a \in A}{\operatorname{argmax}}\, \hat{Q}_i(s)$.

The REGA algorithm treats each available action at the current state as an arm in a MAB setting. As such, the next arm to sample is chosen in accordance with the ϵ-greedy MAB method. Let m be the number of simulations run at this stage thus far. At first, all arms are uniformly sampled up to a certain total number of purely exploratory/training simulations. After that, with ties broken arbitrarily, the arm believed to be optimal is sampled with increasing probability $1 - \epsilon_m$. For each simulation and chosen action, the MDP advances (stochastically) to a future stage s_{i+1} according to the system dynamics $s_{i+1} = f(s_i, a_i, w_i)$ (*c.f.* [1,2]). The next state s_{i+1} is once again treated as a MAB problem. We continue to apply the ϵ-greedy method recursively until the maximal horizon H is reached. Via backward induction, the optimal reward to go at each stage is estimated, and when applied to the initial state of the MDP, we obtain an estimation of the optimal value function of the MDP. The main advantage of REGA is that the reward to go need not be computed for every state of the MDP, as the algorithm only considers the states visited during sampling. As such, REGA is well-suited for MDPs with very large state spaces.

3. Theoretical Analysis

In this section, we provide a theoretical treatment of the convergence properties of REGA. Before introducing the main result, the following lemma is needed to justify some algebraic steps. The proof can be found in Appendix A.

Lemma 1. *Let $c \geq 1$ be a fixed real number, and let $M \geq 1$ and $r > \left(c^2 + c\right)^2$ be fixed integers. Then, the following inequalities hold:*

$$\frac{cM}{\sqrt{r+M+1}} < \sum_{i=r+1}^{r+M} -\log\left(1 - \frac{c}{\sqrt{i}}\right) < \frac{(c+1)M}{\sqrt{r}} \tag{3}$$

$$\sum_{i=r+1}^{r+M} \log^2(1 - \frac{c}{\sqrt{i}}) < (c+1)^2 \log\left(1 + \frac{M}{r}\right). \tag{4}$$

The discrete probability distribution of the sum of independent Bernoulli trails that are not identically distributed is known as the Poisson binomial distribution. The following theorem gives a bound on the error of estimating the Poisson binomial distribution CDF with the "usual" Poisson distribution CDF (*c.f.* [15], Thm. 5.1).

Theorem 1. *Let $X_1, \ldots, X_n$ be independent Bernoulli variables with respective success probabilities $p_1, \ldots, p_n$. Assume that $0 < p_i < 1$ and let $\lambda_i = -\log(1 - p_i)$ for all $1 \leq i \leq n$. Let $Y(\lambda)$ denote a random variable having the Poisson distribution with mean parameter λ. Then, for each integer $m \geq 1$, we have*

$$P\left[Y\left(\sum_{i=1}^{n} \lambda_i\right) \leq m\right] \leq P\left[\sum_{i=1}^{n} X_i \leq m\right] \leq P\left[Y\left(\sum_{i=1}^{n} \lambda_i\right) \leq m\right] + \frac{1}{2}\sum_{i=1}^{n} \lambda_i^2.$$

Before analyzing REGA, we first consider a simpler version of the algorithm in which $\hat{V}_{i+1}$ is replaced with V^*_{i+1} in Equation (2). This removes the recursion from REGA under the assumption that the true reward to go at the next stage is known and thus reduces the problem from a multistage problem to a single-stage problem. We refer to this algorithm as the nonrecursive epsilon-greedy algorithm (NREGA). Later on, we derive results pertaining to REGA.

In what follows, we use the following additional notation. Note that $\xi_\epsilon > 0$ for $\epsilon > 0$.

$$\Lambda = \max\left\{\left(c^2 + c\right)^2, (c|A(s)|)^2\right\},$$

$$\xi_\epsilon = \frac{R_{\max}H + \epsilon}{R_{\max}H} \log\left(\frac{R_{\max}H + \epsilon}{R_{\max}H}\right) - \frac{\epsilon}{R_{\max}H}.$$

The next lemma gives a finite time bound on the estimation errors of the Q-functions induced by NREGA at the exit of the algorithm as a function of the number of allocated simulations at each stage. The result is analogous to Lemma 3.2 in [3].

Lemma 2. *Let $\epsilon > 0$ and $\alpha \in (1, 2)$ be fixed. Consider a single stage of NREGA applied to a MDP using parameters $c \geq 2\sqrt{3}$ and a large enough total number of allocated simulations M, such that $M \geq \Lambda^{1/\alpha}$ and $M^\alpha \geq M + 2$. Then, for any action a and state s, we have*

$$P\left[\left|\hat{Q}_{i,M}(s,a) - Q^*_i(s,a)\right| \geq \epsilon\right] < \frac{(c+1)^2}{2} \log\left(1 + M^{1-\alpha}\right) \tag{5}$$

$$+ ec\left(c \exp\left[\frac{\sqrt{2} - c}{\sqrt{2}}\right]\right)^{M^{1-\alpha/2}} + 2\exp\left[-\xi_\epsilon M^{1-\alpha/2}\right].$$

Proof. Consider a fixed but arbitrary stage i of the algorithm, state s, and action a of the MDP. Denote by $a_i(t)$ the arm played at iteration t of the algorithm. Let $I\{a_i(t) = a\}$ be a random variable indicating whether or not the chosen arm $a_i(t)$ is in fact arm a, and let $I^U\{a_i(t) = a\}$ denote whether or not the arm was chosen (independently) per the uniformly random rule in step 2 of the algorithm. Let $N_a^{i,M}(s) = \sum_{t=1}^{M} I\{a_i(t) = a\}$ be a random variable denoting the total number of times action a is played during the first M iterations of the algorithm at stage s. Similarly, let $U_a^{i,M}(s) = \sum_{t=1}^{M} I^U\{a_i(t) = a\}$.

In an argument identical to the one in the proof of Lemma 3.2 in [3], it can be shown for any positive integer N that

$$P\left[\left|\hat{Q}_{i,M}(s,a) - Q_i^*(s,a)\right| \geq \epsilon\right] < 2\exp[-\xi_\epsilon N] + P\left[N_a^{i,M}(s) \leq N\right]. \tag{6}$$

We first bound the term $P\left[N_a^{i,M}(s) \leq N\right]$ and then replace N with an appropriate function of M.

Indeed, $N_a^{i,M}(s) \geq U_a^{i,M}(s)\, \forall a \in A(s)$ and $N_a^{i,M}(s) > U_a^{i,M}(s)$ for some $a \in A(s)$. Therefore, $N_a^{i,M}(s)$ dominates $U_a^{i,M}(s)$ almost surely. Hence,

$$\begin{aligned} P\left[N_a^{i,M}(s) \leq N\right] &\leq P\left[U_a^{i,M}(s) \leq N\right] \\ &= P\left[\sum_{t=1}^{M} I^U\{a_i(t) = a\} \leq N\right]. \end{aligned} \tag{7}$$

For iteration $t = 1, \ldots, M$ of the algorithm, let ϵ_t be defined as per Equation (1). Per the algorithm description, random variables $I^U\{a_i(t) = a\}$ are independent and have probabilities

$$P\left[I^U\{a_i(t) = a\} = 1\right] = \frac{\epsilon_t}{|A(s)|} = \begin{cases} \frac{c}{\sqrt{t}}, & t \geq (c|A(s)|)^2 \\ \frac{1}{|A(s)|}, & \text{o/w} \end{cases}.$$

Let integer $r \geq \Lambda$ be a constant "shift" variable, and let $X_1, \ldots, X_M$ be independent Bernoulli variables with success probabilities $p_t = \frac{c}{\sqrt{r+t}}$. Since $p_t \leq \frac{\epsilon_t}{|A(s)|}\, \forall\, t = 1, \ldots, M$, then each random variable X_t is (first-order) stochastically dominated by the corresponding random variable $I^U\{a_i(t) = a\}$ for any fixed t. Hence, it holds that

$$P\left[\sum_{t=1}^{M} I^U\{a_i(t) = a\} \leq N\right] \leq P\left[\sum_{t=1}^{M} X_t \leq N\right]. \tag{8}$$

Combining Equations (7) and (8) yields

$$P\left[N_a^{i,M}(s) \leq N\right] \leq P\left[\sum_{t=1}^{M} X_t \leq N\right]. \tag{9}$$

Let $\lambda_t = -\log(1 - \frac{c}{\sqrt{t}})$. We now apply Theorem 1 to Equation (9) which yields

$$P\left[N_a^{i,M}(s) \leq N\right] \leq P\left[Y\left(\sum_{t=r+1}^{r+M} \lambda_t\right) \leq N\right] + \frac{1}{2}\sum_{t=r+1}^{r+M} \lambda_t^2. \tag{10}$$

Applying Chernoff bounds to a Poisson random variable $Y(\lambda)$ yields

$$P(Y \leq y) \leq \frac{e^{-\lambda}(e\lambda)^y}{y^y}, \text{ for } y \leq \lambda. \tag{11}$$

Consider $N \leq \frac{cM}{\sqrt{r+M+1}}$. It follows from Lemma 1 that $N \leq \sum_{t=r+1}^{r+M} \lambda_t$. Hence, we can apply the Chernoff bound from Equation (11) to Equation (10) to obtain

$$P\left[N_a^{i,M}(s) \leq N\right] \leq \frac{1}{2}\sum_{t=r+1}^{r+M} \lambda_t^2 + \frac{\exp\left[-\sum_{t=r+1}^{r+M} \lambda_t\right]\left(e\sum_{t=r+1}^{r+M} \lambda_t\right)^N}{N^N}. \tag{12}$$

We wish to give an elementary upper bound on the right-hand side of (12). Since $\frac{e^{-\lambda}(e\lambda)^y}{y^y}$ is decreasing in λ for $y \leq \lambda$, it follows from Lemma 1 that

$$\begin{aligned} P\left[N_a^{i,M}(s) \leq N\right] &\leq \frac{1}{2}(c+1)^2 \log\left(1+\frac{M}{r}\right) \\ &+ \exp\left[-\frac{cM}{\sqrt{r+M+1}}\right]\left(\frac{ecM}{N\sqrt{r+M+1}}\right)^N. \end{aligned} \tag{13}$$

Now, let $r = \lceil M^\alpha \rceil$ and $N = \lceil M^{1-\alpha/2} \rceil$. For the bound in Equation (13) to converge to zero as $M \to \infty$, it is necessary that $\alpha \in (1,2)$. Moreover, the condition $N \leq \frac{cM}{\sqrt{r+M+1}}$ is indeed satisfied since $c \geq 2\sqrt{3}$ implies that

$$\begin{aligned} M \geq \left(\frac{\sqrt{3}}{c-\sqrt{3}}\right)^{\frac{2}{2-\alpha}} &\Leftrightarrow M^{1-\alpha/2}+1 \leq \frac{cM^{1-\alpha/2}}{\sqrt{3}} \\ &\Rightarrow M^{1-\alpha/2}+1 \leq \frac{cM}{\sqrt{M^\alpha+M+2}} \\ &\Rightarrow \lceil M^{1-\alpha/2} \rceil \leq \frac{cM}{\sqrt{M^\alpha+M+2}} \\ &\Leftrightarrow N \leq \frac{cM}{\sqrt{r+M+1}}, \end{aligned}$$

where the second equation follows from the restriction $M^\alpha \geq M+2$.

Substituting r and N into Equation (13) and relaxing the bound yields

$$\begin{aligned} P\left[N_a^{i,M}(s) \leq N\right] &\leq \frac{1}{2}(c+1)^2 \log\left(1+\frac{M}{M^\alpha}\right) + \\ &\exp\left[-\frac{cM}{\sqrt{M^\alpha+M+2}}\right]\left(\frac{ecM^{\alpha/2}}{\sqrt{M^\alpha+M+1}}\right)^{M^{1-\alpha/2}+1}. \end{aligned} \tag{14}$$

Through condition $M^\alpha \geq M+2$, we have

$$\begin{aligned} &\exp\left[-\frac{cM}{\sqrt{M^\alpha+M+2}}\right]\left(\frac{ecM^{\alpha/2}}{\sqrt{M^\alpha+M+1}}\right)^{M^{1-\alpha/2}+1} \\ &\leq \exp\left[-\frac{cM}{\sqrt{2M^\alpha}}\right](ec)^{M^{1-\alpha/2}+1} \\ &= \exp\left[-\frac{cM^{1-\alpha/2}}{\sqrt{2}}+M^{1-\alpha/2}+1\right]c^{M^{1-\alpha/2}+1} \\ &= ec\left(c\exp\left[\frac{\sqrt{2}-c}{\sqrt{2}}\right]\right)^{M^{1-\alpha/2}}. \end{aligned} \tag{15}$$

Combining Equations (14) and (15) yields

$$P\Big[N_a^{i,M}(s) \leq N\Big] \leq \frac{(c+1)^2}{2}\log\Big(1+M^{1-\alpha}\Big) + ec\left(c\exp\left[\frac{\sqrt{2}-c}{\sqrt{2}}\right]\right)^{M^{1-\alpha/2}}. \quad (16)$$

$N = \lceil M^{1-\alpha/2}\rceil$, and $\xi_\epsilon > 0$ for $\epsilon > 0$. Therefore, Equation (6) can be relaxed to

$$P\big[\big|\hat{Q}_{i,M}(s,a) - Q_i^*(s,a)\big| \geq \epsilon\big] < \frac{(c+1)^2}{2}\log\Big(1+M^{1-\alpha}\Big) + ec\left(c\exp\left[\frac{\sqrt{2}-c}{\sqrt{2}}\right]\right)^{M^{1-\alpha/2}} + 2\exp\Big[-\xi_\epsilon M^{1-\alpha/2}\Big], \quad (17)$$

which completes the proof. □

Corollary 1. *Suppose the conditions of Lemma 2 hold. Let* $\Phi = c\exp\left[\frac{\sqrt{2}-c}{\sqrt{2}}\right] < 1$. *If* M *is large enough, such that* $\Phi^{M^{1-\alpha/2}} \leq M^{1-\alpha}$ *and* $\exp\Big[-\xi_\epsilon M^{1-\alpha/2}\Big] \leq M^{1-\alpha}$, *then it follows that*

$$P\big[\big|\hat{Q}_{i,M}(s,a) - Q_i^*(s,a)\big| \geq \epsilon\big] < \left(\frac{(c+1)^2}{2} + ec + 2\right)M^{1-\alpha}.$$

Proof. It is well-known (*c.f.* [16]) that $\log(1+x) \leq x$ for $x > -1$. The result immediately follows by relaxing Equation (5) of Lemma 2. □

The result in Corollary 1 can be made arbitrarily close to $O(1/M)$ in the limit by selecting α arbitrarily close to 2. However, as α is increased towards 2, the minimal value of M for which the finite time bound applies grows exponentially.

For the purposes of our theoretical analysis, we now assume that the optimal action at each stage of the algorithm is unique, so that the MDP has a unique optimal policy. This assumption is identical to that in [3].

Assumption 1. $\theta = \min_{s\in S, i=0,\ldots H-1} \theta_i(s) > 0$, *where* $\theta_i(s) = \min_{a\neq a^*}(V_i^*(s) - Q_i^*(s,a))\ \forall s \in S, a \in A(s)$ *and* $i = 0,\ldots,H-1$.

A small value of θ means increased difficulty in differentiating between the truly optimal action and the best suboptimal action(s). Therefore, θ gives a measurement on the "size" of the problem.

We are finally ready to leap from NREGA to the recursive REGA algorithm. We start by restating Lemma 3.3 of [3] in a manner that is useful and applicable to REGA. We refer the reader to the original manuscript for the proof but note that equation (11) as it originally appears in Lemma 3.3 of [3] is unnecessarily loose, where $N_a^i(x)$ is bounded from above by K_{i+1} instead of K_i. Using the lemma as-is causes our lower bound on the probability in Theorem 2 to converge to 0 instead of 1 as the number of simulations at each stage of the MDP is increased to infinity, whereas one would expect the bound to improve as the number of samples is increased. We necessarily use the tighter bound.

Lemma 3. *Consider REGA applied with* M_i *simulations allocated to each stage* $i = 0,\ldots,H-1$. *Suppose that Assumption 1 holds, and suppose that* $\delta_i \in (0,1)$ *satisfies* $P\left(\big|\hat{Q}_{i,M_i}(s,a) - Q_i^*(s,a)\big| > \frac{\theta}{2^{i+2}}\right) < \delta_i$ *for all* i *and* $s \in S$. *At the exit step we have*

$$P\left(\big|\hat{V}_{0,M_0}(s_0) - V_0^*(s_0)\big| < \frac{\theta}{2}\right) > (1-\delta_0)\prod_{i=1}^{H-1}(1-\delta_i)^{\Pi_{j=0}^{i-1}M_j}.$$

The following definition is needed in the next theorem:

$$\xi_i = \frac{R_{\max}H + \theta/2^{i+2}}{R_{\max}H} \log\left(\frac{R_{\max}H + \theta/2^{i+2}}{R_{\max}H}\right) - \frac{\theta/2^{i+2}}{R_{\max}H}.$$

We now state our main result pertaining to the REGA algorithm in the form of a finite-time bound on the probability that the approximation error of the MDP value function exceeds half the size of the problem, i.e., $\theta/2$. The bound is given as a function of the number of simulations M_i allocated at each stage.

Theorem 2. *Suppose that Assumption 1 holds and that the requirements of Corollary 1 are satisfied for each stage i of the MDP, where ξ_ϵ is replaced with ξ_i. Let $\phi > 1$ be fixed, and suppose that $M_i = \prod_{j=0}^{i-1} M_j^{\phi/(\alpha-1)}$ for all $i = 1, \ldots, H-1$. Denote $\Psi = \frac{(c+1)^2}{2} + ec + 2$. If M_0 is large enough such that $\Psi M_0^{1-\alpha} < 1$, then at the exit step of REGA we have*

$$P\left(\left|\hat{V}_{0,M_0}(s_0) - V_0^*(s_0)\right| < \frac{\theta}{2}\right) > \left(1 - \Psi M_0^{1-\alpha}\right)^{HM_0^{(\alpha-1)/\phi}}.$$

Proof. Via Corollary 1, it holds for each stage $i = 0, \ldots, H-1$ that

$$P\left(\left|\hat{Q}_{i,M_i}(s,a) - Q_i^*(s,a)\right| > \frac{\theta}{2^{i+2}}\right) < \Psi M_i^{1-\alpha}.$$

Via Lemma 3, we have

$$\begin{aligned}
&P\left(\left|\hat{V}_{0,M_0}(s_0) - V_0^*(s_0)\right| < \frac{\theta}{2}\right) \\
&> \left(1 - \Psi M_0^{1-\alpha}\right) \prod_{i=1}^{H-1} \left(1 - \Psi M_i^{1-\alpha}\right)^{\prod_{j=0}^{i-1} M_j} \\
&= \left(1 - \Psi M_0^{1-\alpha}\right) \prod_{i=1}^{H-1} \left(1 - \Psi M_i^{1-\alpha}\right)^{M_i^{(\alpha-1)/\phi}} \\
&> \left(1 - \Psi M_0^{1-\alpha}\right)^{HM_0^{(\alpha-1)/\phi}},
\end{aligned}$$

where the last inequality follows from the fact that $\left(1 - \Psi x^{1-\alpha}\right)^{x^{(\alpha-1)/\phi}}$ is increasing in x. This completes the proof. □

We conclude by noting that the bound on the probability in Theorem 2 goes to 1 as $M_0 \to \infty$.

4. Empirical Results

Value iteration and policy iteration are classical exact methods for solving MDPs. One iteration of these methods has runtime complexity $O(|S|^2|A|)$ and $O(|S|^2|A| + |S|^3)$, which are nonlinear in the size of the state space $|S|$. On the other hand, REGA and RASA have runtime complexity $O\left(\left(\max_{h=0,\ldots,H-1}\{N_h\}\right)^H\right)$, which is independent of the size of the state space. Although the runtime complexities of REGA and RASA are also independent of the size of the action space, the accuracies of the algorithms are certainly dependent on the size of the action space. As such, REGA and RASA are generally not well-suited for MDPs with large action spaces. Due to the backward induction process, the runtime complexities of REGA and RASA grow exponentially as the sampling horizon H is increased. Thus, REGA and RASA are particularly attractive algorithms for problems with a large state space S, small action space A, and small horizon H.

One solution to the exponential runtime complexity issue pertaining to a large horizon H is to use rolling-horizon control (RH), also known in the literature as *receding/moving horizon control*. RH provides an approximate solution to infinite horizon control problems in an online manner. The basic idea is to consider a fixed horizon length to determine the best policy/action at the current stage, ignoring stages far out into the future in the hope that they have little effect on the decision being made at the current stage. If the length of this "sliding" lookahead window is chosen to be large enough, then the error induced by this approach can be made small. For further information, we refer the reader to [17].

4.1. SysAdmin Problem

In what follows, we demonstrate the performance of REGA when applied to the well-known *SysAdmin* discrete-time problem (*c.f.* [18]). In the *SysAdmin* problem, a system administrator manages B computing machines that are connected via a known network configuration, so that each machine $b = 1, \ldots, B$ is connected to a subset of "neighboring" machines $N(b)$. Let I_b^h denote the state of machine b at horizon $h = 1, \ldots, H$, where $I_b^h = 1$ indicates that the machine is in working state, and $I_b^h = 0$ indicates that the machine is in faulted state. The state of the system at time h can be captured as a binary vector $< I_1^h, I_2^h, \ldots, I_B^h >$, so the size of the state space is $2^{|B|}$, which is exponential to the number of machines. At each time step, the system administrator collects rewards $r(b)$ associated with machines b that are in working state. The objective is to maximize $E\left[\sum_{h=1}^{H}\sum_{b=1}^{B} r(b)\, I_b^h\right]$, the expected total (nondiscounted) reward over the entire horizon of the problem, where the expectation is taken with respect to machine states.

At each time step, the system administrator has to decide to either reboot a particular machine or not reboot any machine at all. Only one machine can be rebooted at each time step. Hence, the action space includes $|B| + 1$ actions. We denote the action taken at each horizon h by a binary vector $< A_1^h, A_2^h, \ldots, A_B^h >$, where $A_b^h = 1$ if machine b is rebooted, and $A_b^h = 0$ otherwise. Since, at each time step, at most one machine can be rebooted, we have $0 \leq \sum_{b=1}^{B} A_b^h \leq 1$.

Machine states are controlled via the following (stochastic) rules: If any machine in $N(b)$ is in faulted state, then b also fails with high probability p_1. On the other hand, if none of the machines in $N(b)$ is in faulted state, then b can still fail with a much smaller probability p_2. When a machine is rebooted, it fails to start up in a working state with a very small probability p_3. The transitional probabilities for each machine can be formally stated as follows:

1. A faulted machine that is not rebooted remains in a faulted state.

$$P\left(I_b^{h+1} = 0 \,|\, I_b^h = 0\,,\, A_b^h = 0\right) = 1.$$

2. A machine in working state that is not rebooted enters a faulted state with different probability depending on the state of neighbor machines. Let statement T denote that $\exists\, \beta \in N(b)$ s.t. $I_\beta^h = 0$.

$$P\left(I_b^{h+1} = 0 \,|\, I_b^h = 1\,,\, A_b^h = 0\right) = \begin{cases} p_1, & T \text{ true} \\ p_2, & \text{o/w} \end{cases}$$

$$P\left(I_b^{h+1} = 1 \,|\, I_b^h = 1\,,\, A_b^h = 0\right) = \begin{cases} 1 - p_1, & T \text{ true} \\ 1 - p_2, & \text{o/w} \end{cases}$$

3. Rebooting a machine can result in it becoming in faulted state (regardless of its current state).

$$P\left(I_b^{h+1} = 0 \,|\, I_b^h \in \{0, 1\}\,,\, A_b^h = 1\right) = p_3.$$

$$P\left(I_b^{h+1} = 1 \,|\, I_b^h \in \{0,1\}\,,\, A_b^h = 1\right) = 1 - p_3.$$

Typically, in the literature, the *SysAdmin* problem is modeled as a finite horizon discrete-time MDP with total (undiscounted) reward criteria. Due to the size of the state space, it is often cited in articles related to *factored MDPs* (*c.f.* [18]).

4.2. Results

We consider two network topologies: a ring configuration and a star configuration. In the ring configuration, each machine $b \in B$ is connected to two neighboring machines $b-1$ and $b+1$ (arithmetic taken modulo $|B|$). In the star configuration, there is one central server, and all other machines are connected only to the central server. The two topologies are depicted in Figure 1.

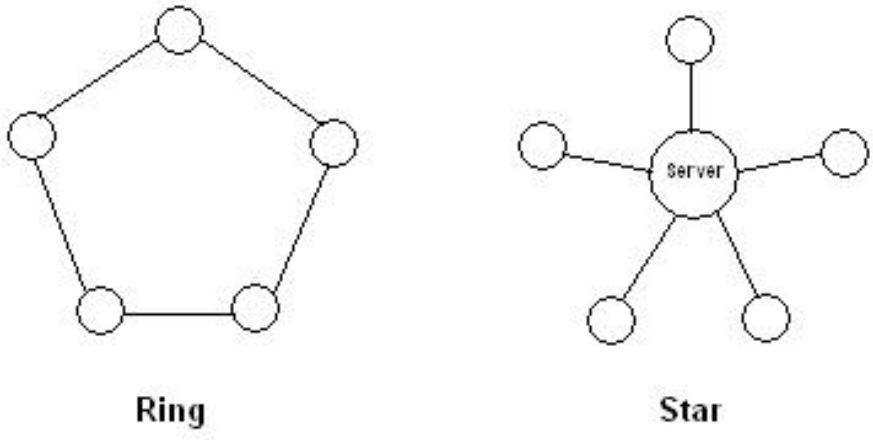

Figure 1. Example network topologies.

In addition to REGA and RASA, we also consider a modification to REGA that is consistent with the original ϵ-greedy method of [10] that is heavily cited in the literature, where denominator $\sqrt{m}$ in Equation (1) is replaced with m. We refer to this method as original REGA (OREGA). We also consider a trivial purely greedy strategy (PGS) where each action is sampled exactly once, after which each iteration of the algorithm plays the machine with highest observed average reward. Using custom C++ code running on a quad-core CPU and 12GB of RAM, and allowing each run to execute for one minute, we simulated REGA, RASA, OREGA, and PGS against the cycle and star network topologies of 10 machines with an initial state of all machines being online. The problems have state spaces with 2^{10} states and action spaces with 11 actions. We set the horizon of the problem to $H = 3$. The number of simulations was set to a constant across all stages (i.e., $M_1 = M_2 = M_3$). The reward of machine i being online was set to i in order to eliminate any simplifications introduced by symmetry where trivial strategies can perform well. The machine failure rates were set to $p_1 = 0.7$, $p_2 = 0.1$ and $p_3 = 0.01$. For REGA and OREGA, we set the parameters c_i to $c_1 = c_2 = c_3 = 6$ (the performance of the ϵ-greedy method, and thus REGA, is generally sensitive to the choice of values c_i. Our choice is not necessarily the best, but performed well). For RASA we set the tuneable parameter $\mu_i = 1 - 2^{-1/M_i}$ as was chosen in the simulations described by the original paper [3]. Each simulation was repeated 30 times in order to calculate the average reward and standard error for each setting. REGA and RASA achieved similar performances and, as expected, both outperformed the trivial PGS policy. Interestingly, OREGA did not perform as well as REGA and RASA, and this will be investigated in more detail in future work. Tables 1 and 2 summarize the results of our simulations. The same results are plotted in Figures 2 and 3. For both networks, one can verify via value iteration that the optimal value is $V^* = 149.93$.

Table 1. Performance of REGA, RASA, OREGA, and PGS against a cycle network configuration of 10 machines with machine failure probabilities $p_1 = 0.7, p_2 = 0.1, p_3 = 0.01$. The optimal value is $V^* = 149.93$.

M_i	REGA	RASA	OREGA	PGS
35	160.8 (0.99)	158.7 (0.85)	161.0 (0.92)	**139.8 (0.77)**
75	**151.0 (0.81)**	**150.5 (0.84)**	**149.6 (1.03)**	135.1 (0.43)
100	148.1 (0.88)	147.1 (0.65)	143.7 (0.58)	132.7 (0.51)
125	144.6 (0.43)	145.0 (0.52)	140.2 (0.62)	132.3 (0.42)
150	144.5 (0.52)	144.5 (0.73)	138.7 (0.49)	131.4 (0.33)
175	142.6 (0.50)	141.8 (0.46)	137.0 (0.36)	131.2 (0.33)
200	142.1 (0.41)	141.9 (0.56)	136.3 (0.31)	131.8 (0.35)
225	140.7 (0.42)	141.5 (0.57)	135.5 (0.29)	130.3 (0.27)
250	140.7 (0.39)	140.8 (0.48)	135.0 (0.27)	131.2 (0.39)

Table 2. Performance of REGA, RASA, OREGA, and PGS against a star network configuration of 10 machines with machine failure probabilities $p_1 = 0.7, p_2 = 0.1, p_3 = 0.01$. The optimal value is $V^* = 149.93$.

M_i	REGA	RASA	OREGA	PGS
35	162.7 (0.58)	159.5 (0.93)	163.6 (0.34)	**145.2 (0.76)**
75	153.6 (0.79)	152.4 (0.73)	**152.9 (0.65)**	139.2 (0.46)
100	**149.6 (0.52)**	**149.9 (0.60)**	146.7 (0.58)	138.4 (0.47)
125	149.4 (0.61)	147.3 (0.57)	143.5 (0.50)	137.8 (0.40)
150	147.8 (0.44)	148.0 (0.47)	142.4 (0.32)	138.0 (0.37)
175	146.1 (0.48)	146.3 (0.41)	141.7 (0.39)	137.9 (0.46)
200	145.8 (0.51)	145.9 (0.50)	141.3 (0.29)	136.4 (0.40)
225	145.4 (0.40)	145.0 (0.49)	140.5 (0.27)	137.0 (0.27)
250	145.1 (0.37)	144.7 (0.48)	140.3 (0.26)	136.5 (0.37)

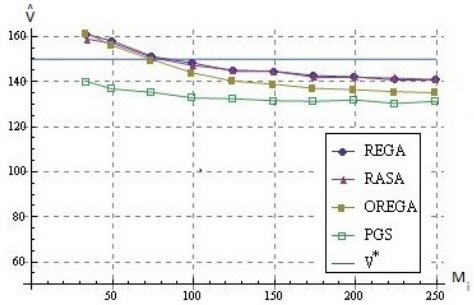

Figure 2. Performance of REGA, RASA, and PGS against a cycle network configuration of 10 machines with machine failure probabilities $p_1 = 0.7, p_2 = 0.1, p_3 = 0.01$.

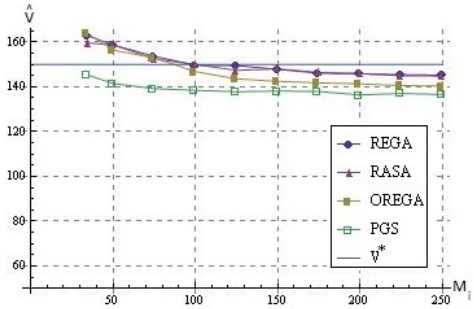

Figure 3. Performance of REGA, RASA, and PGS against a star network configuration of 10 machines with machine failure probabilities $p_1 = 0.7, p_2 = 0.1, p_3 = 0.01$.

5. Discussion

We introduced REGA, a recursive method based on the well-known ϵ-greedy multi-armed bandit algorithm useful for numerically solving MDPs with (possibly large) state spaces, but small action spaces. Indeed, the curse of dimensionality renders the algorithm most suitable for small finite-horizon settings, but techniques such as rolling-horizon control can aid in such situations. Due to the numerical nature of REGA, it can be of particular interest to use the algorithm to train model-free reinforcement learning and deep reinforcement learning problems. Indeed, the algorithm does not impose any restrictions on how the agent interacts with the environment, and so long as the state transition function $f(s, \bar{a}_i, w_m^i)$ and immediate reward $R(s, \bar{a}_i, w_m^i)$ can be computed (or simulated), then the Q-value estimates can be updated as per the REGA algorithm specification. Furthermore, if training the reinforcement learning models can occur offline with less sensitivity to running time, then exploring larger horizon settings may be feasible. We leave the investigation of applying REGA to appropriate popular and/or benchmark prediction and control problems from reinforcement learning as future work.

We provided finite-time analysis of REGA and showed that the bound on the error induced by the algorithm approaches zero as the number of sample points at each stage of the MDP is increased. The algorithm was tested empirically against the *SysAdmin* benchmark problem with up to 2^{10} states. In our experiments, we found that a modification to the exploration/exploitation trade-off model of the ϵ-greedy method commonly cited in the literature yields improved results in practice, and that will be further investigated in future work. Other directions of future study include constructing a counterpart algorithm that is useful for MDPs with large action spaces and small state spaces. It would also be of interest to carry out a formal theoretical analysis comparing the expected performance of REGA and RASA.

Author Contributions: Methodology, I.M. and J.H.; software, I.M.; formal analysis, I.M. and J.H.; writing—original draft preparation, I.M.; writing—review and editing, I.M.; supervision, J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proof of Lemma 1

Let $x > \left(c^2 + c\right)^2$, and consider function $b(x) = -\log\left(1 - c/\sqrt{x}\right)$. We have $\lim_{x\to\infty} b(x) = 0$ and $b'(x) < 0$. Therefore, b is positive and decreasing in x, so the definite integral of b can be bounded from below by the right-hand Riemann sum, and from above by the left-hand Riemann sum, yielding

$$0 < \int_{r+1}^{r+M+1} b(x)dx \leq \sum_{i=r+1}^{r+M} b(i) \leq \int_{r}^{r+M} b(x)dx. \tag{A1}$$

To prove the first part, we bound the function b from below. Identity $\log(1-y) < -y$ holds for $y \in (0,1)$. Since $c/\sqrt{x} \in (0,1)$, we obtain

$$b(x) > \frac{c}{\sqrt{x}}. \tag{A2}$$

Combining Equations (A1) and (A2) yields

$$\begin{aligned}\sum_{i=r+1}^{r+M} b(i) &\geq \int_{r+1}^{r+M+1} b(x)dx \\ &> \int_{r+1}^{r+M+1} \frac{c}{\sqrt{x}}dx \\ &= 2c\left(\sqrt{r+M+1}-\sqrt{r+1}\right) \\ &= \frac{2cM}{\sqrt{r+M+1}+\sqrt{r+1}} \\ &> \frac{cM}{\sqrt{r+M+1}},\end{aligned}$$

which proves the first part. We now turn our attention to the second part of our claim. For brevity, denote $u = \frac{(c^2+c)(\sqrt{x}-c)}{\sqrt{x}-c^2-c}$. Basic algebraic steps show that

$$\begin{aligned}-\log(1-\frac{c}{\sqrt{x}}) &= \log\left(\frac{(u-c)(c+1)}{cu}\right) \\ &= \log(1-\frac{c}{u}) - \log(1-\frac{c}{c^2+c}) = \int_{c^2+c}^{u} \frac{c}{t(t-c)}dt. \\ &\leq \int_{c^2+c}^{u} \frac{c+1}{t^2}dt = \frac{c}{\sqrt{x}-c} < \frac{c+1}{\sqrt{x}}. \end{aligned} \tag{A3}$$

Combining Equations (A1) and (A3) yields

$$\begin{aligned}\sum_{i=r+1}^{r+M} b(i) &\leq \int_{r}^{r+M} b(x)dx \\ &\leq \int_{r}^{r+M} \frac{c+1}{\sqrt{x}}dx \\ &= 2(c+1)\left(\sqrt{r+M}-\sqrt{r}\right) \\ &= \frac{2(c+1)M}{\sqrt{r+M}+\sqrt{r}} \\ &< \frac{(c+1)M}{\sqrt{r}}.\end{aligned}$$

This completes the proof of Equation (3). We now prove Equation (4). Squaring both sides of Equation (A3) yields

$$\log^2(1-\frac{c}{\sqrt{x}}) < \frac{(c+1)^2}{x}.$$

Therefore,

$$\sum_{i=r+1}^{r+M} \log^2(1-\frac{c}{\sqrt{i}}) < \sum_{i=r+1}^{r+M} \frac{(c+1)^2}{i}. \tag{A4}$$

However, $1/x$ is positive and decreasing for $x > 0$; therefore, its definite integral can be bound from below by the function's right-hand Riemann sum. Hence,

$$\sum_{i=r+1}^{r+M} \frac{(c+1)^2}{i} \leq \int_{r}^{r+M} \frac{(c+1)^2}{x}dx = (c+1)^2\log\left(1+\frac{M}{r}\right),$$

which completes the proof.

References

1. Hernández-Lerma, O. *Adaptive Markov Control Processes*; Applied Mathematical Sciences; Springer: New York, NY, USA, 1989.
2. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed.; John Wiley & Sons, Inc.: New York, NY, USA, 1994.
3. Chang, H.S.; Fu, M.C.; Hu, J.; Marcus, S.I. Recursive Learning Automata Approach to Markov Decision Processes. *IEEE Trans. Automat. Control* **2007**, *52*, 1349–1355. [CrossRef]
4. Chang, H.S.; Fu, M.C.; Hu, J.; Marcus, S.I. *Simulation-Based Algorithms for Markov Decision Processes*; Communications and Control Engineering; Springer: London, UK, 2007.
5. Hu, J.; Chang, H.S. An Approximate Stochastic Annealing algorithm for finite horizon Markov decision processes. In Proceedings of the 2010 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, USA, 15–17 December 2010; pp. 5338–5343. [CrossRef]
6. Bertsekas, D.P.; Tsitsiklis, J.N. *Neuro-Dynamic Programming*, 1st ed.; Athena Scientific: Nashua, NH, USA, 1996.
7. Santharam, G.; Sastry, P.S. A Reinforcement Learning Neural Network for Adaptive Control of Markov Chains. *Trans. Syst. Man Cybern.—Part A Syst. Hum.* **1997**, *27*, 588–600. [CrossRef]
8. Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. Thesis, King's College, Cambridge, UK, 1989.
9. Watkins, C.; Dayan, P. Technical Note: Q-Learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
10. Auer, P.; Cesa-Bianchi, N.; Fischer, P. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.* **2002**, *47*, 235–256. [CrossRef]
11. Muqattash, I.M. Multi-Armed Bandits with Applications to Markov Decision Processes and Scheduling Problems. Ph.D. Thesis, Stony Brook University, Stony Brook, NY, USA, 2014.
12. Auer, P.; Cesa-Bianchi, N.; Freund, Y.; Schapire, R.E. The Nonstochastic Multiarmed Bandit Problem. *SIAM J. Comput.* **2003**, *32*, 48–77. [CrossRef]
13. Bubeck, S.; Cesa-Bianchi, N. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Found. Trends Mach. Learn.* **2012**, *5*, 1–122. [CrossRef]
14. Pandey, S.; Chakrabarti, D.; Agarwal, D. Multi-armed Bandit Problems with Dependent Arms. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*; ACM: New York, NY, USA, 2007; pp. 721–728. [CrossRef]
15. Serfling, R. Some Elementary Results on Poisson Approximation in a Sequence of Bernoulli Trials. *SIAM Rev.* **1978**, *20*, 567–579. [CrossRef]
16. Topsøe, F. Some Bounds for the Logarithmic Function. *Res. Rep. Collect.* **2004**, *7*, 1–20.
17. Hernández-Lerma, O.; Lasserre, J.B. Error bounds for rolling horizon policies in discrete-time Markov control processes. *IEEE Trans. Autom. Control* **1990**, *35*, 1118–1124. [CrossRef]
18. Guestrin, C.; Koller, D.; Parr, R.; Venkataraman, S. Efficient Solution Algorithms for Factored MDPs. *J. Artif. Int. Res.* **2003**, *19*, 399–468. [CrossRef]