Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 196 (2008) 913–922
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems

D.E. Koulouriotis *, A. Xanthopoulos

Department of Production and Management Engineering, School of Engineering, Democritus University of Thrace, Greece

Abstract

Multi-armed bandit tasks have been extensively used to model the problem of balancing exploitation and exploration. A most challenging variant of the MABP is the non-stationary bandit problem where the agent is faced with the increased complexity of detecting changes in its environment. In this paper we examine a non-stationary, discrete-time, finite horizon bandit problem with a finite number of arms and Gaussian rewards. A family of important ad hoc methods exists that are suitable for non-stationary bandit tasks. These learning algorithms that offer intuition-based solutions to the exploitation–exploration trade-off have the advantage of not relying on strong theoretical assumptions while in the same time can be fine-tuned in order to produce near-optimal results. An entirely different approach to the non-stationary multi-armed bandit problem presents itself in the face of evolutionary algorithms. We present an evolutionary algorithm that was implemented to solve the non-stationary bandit problem along with ad hoc solution algorithms, namely action-value methods with e-greedy and softmax action selection rules, the probability matching method and finally the adaptive pursuit method. A number of simulation-based experiments was conducted and based on the numerical results that we obtained we discuss the methods' performances.

Keywords: Decision-making agents; Action selection; Exploration–exploitation; Multi-armed bandit; Genetic algorithms; Reinforcement learning

1. Introduction

Consider the problem of an agent that is operating in an environment of incomplete information. The agent is frequently faced with the choice between maximizing its expected profits based on its current knowledge of its environment and trying to learn more about the environment in order to improve the quality of its decisions. This situation is widely known as the exploitation–exploration trade-off. The multi-armed bandit problem [1,2] is possibly the most generic setting in which this trade-off can be modeled. A decision-maker is repeatedly faced with n different choices or actions. Depending on the choice it made, it receives a stochastic reward. The agent's goal is to maximize the sum or some other function of the series of rewards. Over the

* Corresponding author.
E-mail address: jimk@pme.duth.gr (D.E. Koulouriotis).

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.07.043

years, the original multi-armed bandit problem was extended with a number of variations appearing in the literature. Gittins' Markovian model was extended to a general continuous-time setting [3,4]. Banks and Sundaram in [5] suggested that a cost should be incurred when the controller switches from an alternative to another. Here the arms of the bandit are considered to be on-going projects. The incorporation of switching costs to the multi-armed bandit problem makes its solution much more complicated. Jun [6] surveys the literature on bandits with switching costs. Auer et al., [7] introduced the adversarial bandit problem. In this problem the rewards collected by the agent do not follow some probability distribution but are generated by an adversary in order to make things more complicated for the agent. The multi-armed bandit framework provided the way to model a wide range of real world problems such as dynamic pricing of a Web-store [8], and job assignment [9]. Azoulay-Schwartz et al. in [10] study the problem of optimally selecting a supplier in the context of electronic commerce from a bandit perspective. McCall and McCall [11] cast the problem of a person seeking employment at one of n different firms into the multi-armed bandit framework. Chulkov and Desai [12] use the bandit problem as a tool to aid their analysis of IT project failures. The optimal solution of the classical bandit problem was given by Gittins [2], who presented a method that required the calculation of complicated functions called *dynamic allocation indices* or more simply Gittins' indices. Gittins proved that the optimal strategy is to select the choice with the highest index value assuming that the rewards are discounted exponentially, there are no switching costs and that each alternative has a particular distribution of rewards.

A most challenging variant of the MABP is the non-stationary bandit problem where the agent is faced with the increased complexity of detecting changes in its environment. In this problem formulation the reward generating process of the bandit's arms do not remain the same throughout the task but undergo changes. A family of ad hoc methods well-suited for the non-stationary bandit problem exists that is of particular interest for numerous reasons. These learning algorithms do not balance exploration and exploitation in a sophisticated way but have the advantage of not relying on strong theoretical assumptions while in the same time can be fine-tuned to produce near-optimal results. Four well-known algorithms of this type and namely *e-greedy action selection*, *softmax action selection*, *probability matching and adaptive pursuit method* (see [13–15]) will be presented in subsequent sections. An entirely different approach to the multi-armed bandit problem presents itself in the face of evolutionary algorithms. We present an evolutionary algorithm that we implemented to solve non-stationary the multi-armed bandit problem. A number of simulation-based experiments were conducted and based on the numerical results that we obtained we discuss the ad hoc methods' along with the evolutionary algorithm's performances. The remaining material of this paper is organized as follows. In Section 2 we formally state the non-stationary multi-armed bandit problem that we study in this paper. Sections 3–3.5 are devoted to the presentation of the ad hoc techniques that we are going to use to solve the bandit problem. We present the evolutionary algorithm that we implemented in Sections 4–4.2. Numerical results from the experiments that we carried out for both the ad hoc methods and the evolutionary algorithm are located in Sections 5–5.3. These sections also contain the analysis and interpretation of our findings. Finally, Section 6 contains our concluding remarks.

2. The non-stationary multi-armed bandit problem definition

The non-stationary multi-armed bandit problem that we examined can be formally defined as follows. A decision-making agent is presented with n different actions, or controls a_i, $i = 1, 2, 3, \ldots, n$. The agent is repeatedly asked to select one of the available actions for a finite number of decision-making epochs or episodes t_j, $j = 1, 2, 3, \ldots, k$. The decision-maker is allowed to select only one action per episode. Immediately after each selection in a decision-making epoch, the agent is given a numerical reward $r_{t_j}(a_i)$ dependent on the selected action. The reward on the jth episode, $j < c$ and $c \in (1, k)$, given that action a_i was selected, is chosen from a probability distribution with mean or expected reward $Q_s^*(a_i)$. The actions' expected rewards, $Q_s^*(a_i)$, $i = 1, 2, 3, \ldots, n$, are initially unknown to the agent. At some point t_c the actions' expected rewards take on new values $Q_f^*(a_i)$ and remain unchanged until the end of the task. The decision-maker's objective is to maximize the sum of collected rewards J

$$\max J = \max \sum_{j=1}^{k} r_{t_j}. \tag{1.1}$$

3. Ad hoc techniques

In Sections 3.2–3.5 we present four ad hoc techniques that can cope with non-stationarity in multi-armed bandit problems. These methods lack strong theoretical foundations but can be fine-tuned in order to produce near-optimal results, a quality that makes them very interesting, especially from a practitioner's point of view. These methods are (i) the probability matching algorithm, (ii) the e-greedy action selection, (iii) the softmax action selection and finally, (iv) the adaptive pursuit method. The whole of these algorithms maintain *estimates* of the actions' actual expected rewards $Q^*(a_i)$ that are updated every time when action a_i is tried. For the rest of the paper we will call these estimates *action value estimates*. The following section presents a fundamental equation, the action value update rule.

3.1. Action value estimates update rule

For every available action a_i, $i = 1, 2, 3, \ldots, n$, the decision-making agent maintains an action value estimate $Q_q(a_i)$, where q is the number of episodes that action a_i was selected. The action value estimates are initialized arbitrarily. After action a_i is selected for the qth time in episode t_j and the corresponding reward is observed, the action value estimate for this particular action is updated according to rule (1.2) while all the other action value estimates remain unchanged

$$Q_q(a_i) = Q_{q-1}(a_i) + \beta[r_{t_j} - Q_{q-1}(a_i)]. \tag{1.2}$$

β is a positive parameter, $0 < \beta < 1$, that can be either constant, or a function of time, $\beta = \beta_i(t) = \frac{1}{q}$. The latter is called a *sample-average method* for estimating action values because each estimate is an average of the sample of rewards received from selecting action a_i. As q reaches ∞, by the law of large numbers $Q_q(a_i)$ converges to $Q^*(a_i)$, and thus makes the sample-average method suitable for stationary problems.

Constant β parameter method is called an *exponentially-weighted average method* and is well-tailored for use in non-stationary bandit tasks as the sequence $\{Q_0, Q_1, \ldots, Q_q\}$, never converges completely but continues to display fluctuations in response to the most recently received rewards. For the case of constant β Eq. (1.2) can be written in the form

$$Q_q(a_i) = (1-\beta)^q Q_0 + \sum_{j=1}^{q} \beta(1-\beta)^{q-j} r_j. \tag{1.3}$$

From (1.3) one can observe that the estimate $Q_q(a_i)$ is a weighted average of the initial guess Q_0 and the rewards that were collected in the past. The weight given to r_j decreases exponentially as the age of the observation increases. All the algorithms that will be presented in Sections 3.2–3.5 make use of the exponentially-weighted average method.

3.2. Probability matching technique

Perhaps the most straight-forward way to assign selection probabilities to the available actions is to make the selection probability $P(a_i)$ of action a_i proportional to its current value estimate

$$P_{t_j}(a_i) = \frac{Q_{t_j}(a_i)}{\sum_n Q_{t_j}(a)}. \tag{1.4}$$

However, this approach does not exclude the possibility that at some point the probability with which a certain action is being selected reaches near 0, rendering this action practically unavailable to the agent for the rest of the task. That is exactly what needs to be avoided in a non-stationary bandit task. This problem can be dealt with simply by setting a minimum value $P_{\min}$ for all the selection probabilities. As a direct consequence, the highest value that any selection probability can take up now is: $P_{\max} = 1 - (n-1)P_{\min}$, where n is the number of the agent's alternatives. Bearing all of that in mind we derive the *probability matching rule* (see [15])

$$P_{t_j}(a_i) = P_{\min} + (1 - nP_{\min}) \frac{Q_{t_j}(a_i)}{\sum_n Q_{t_j}(a)}. \tag{1.5}$$

According to this technique *all* the selection probabilities are updated as shown in (1.5) in every episode t_j. Despite the fact that the probability matching technique has the ability to detect changes made in non-stationary multi-armed bandit environments, it exhibits poor performance in terms of maximizing J for reasons that we will elaborate on in Section 5.1.

3.3. E-greedy action selection

The *e-greedy action selection rule* can be seen as a reasonable extension of the greedy action selection method, a well-known strategy of selecting actions in stationary bandit tasks. According to the greedy action selection rule the agent simply selects the action with the highest action value estimate at this time. This action is called the *greedy action*. The e-greedy action selection method incorporates the element of the explicit search of the decision space in random time intervals. To be more specific, in decision-making epoch t_j the *e-greedy agent*

- selects the greedy action with probability $1 - e$

or

- with probability e, selects an action at random: $P\{a = a_i\} = \frac{1}{n},\ i = 1, 2, 3, \ldots, n$.

The e-greedy action selection method is characterized by e, a small positive parameter, $0 < e < 1$. The value of e is selected by trial and error, usually in the space $\{0.01, 0.1\}$. It is obvious that setting $e = 0$ results in the greedy action selection rule. High values of e will force the agent to make exploratory choices very often and as a result will prevent the agent from concentrating its choices to the optimal action, while on the other hand, give it the ability to react rapidly to changes that take place in its environment.

3.4. Softmax action selection

As it was illustrated in the former section, when the e-greedy action selection method explores, all available actions have equal probabilities to be chosen. In some bandit tasks, it might be better if actions with higher action values were made more likely to be selected, that is, if the probability of an action a_i to be selected in episode t_j was a function of its action value estimate $Q_{t_j}(a_i)$ in that episode. Action selection methods of this type are called *softmax action selection methods*. The most popular softmax action selection method uses the Boltzmann distribution. In episode t_j the agent selects action a_k with probability

$$P\{a = a_k\} = \frac{e^{Q_{t_j}(a_k)/T}}{\sum_{i=1}^{n} e^{Q_{t_j}(a_i)/T}}. \tag{1.6}$$

T is a positive parameter called *temperature*. With proper adjustment of parameter T, the softmax action selection method can be effectively applied to non-stationary bandit tasks. Relatively high temperatures result in nearly equiprobable actions. The lower the temperature is, the greater the difference in selection probability among actions with different action values is.

3.5. Adaptive pursuit method

Pursuit methods is a family of learning algorithms for the multi-armed bandit problem that were initially proposed by Thathachar and Sastry [17]. Thierens [15], offers a variant suitable for non-stationary environments called the *adaptive pursuit algorithm*. Similarly to the probability matching technique, all selection probabilities cannot fall under the threshold $P_{\min}$ or exceed the maximum value $P_{\max} = 1 - (n - 1)P_{\min}$. According to the adaptive pursuit method, the probabilities are updated *before* the agent makes its selection. Let a_g be the greedy action in episode t_j . The probability of selecting a_g is incremented towards $P_{\max}$, while all the remaining probabilities are decremented towards $P_{\min}$, ensuring that $\sum_{i=1}^{n} P_{t_j}(a_i) = 1$:

$$P_{t_j}(a_g) = P_{t_{j-1}}(a_g) + \alpha(P_{\max} - P_{t_{j-1}}(a_g)), \tag{1.7}$$

$$P_{t_j}(a_i) = P_{t_{j-1}}(a_i) + \alpha(P_{\min} - P_{t_{j-1}}(a_i)), \quad \forall i \neq g. \tag{1.8}$$

Again, α is a small positive parameter, $0 < \alpha < 1$. The action value estimates $Q_{t_j}(a_i)$ are updated after the selection of an action and the observation of the reward. We remind that the action value estimates are exponentially-weighted averages of the corresponding rewards, as we saw in Section 3.1.

4. Evolutionary algorithm

Based on the fact that evolutionary algorithms have been successfully applied to a wide range of optimization problems, the multi-armed bandit problem appears to be a good candidate problem to be solved via an appropriately designed EA. Each individual of the population of such an application-specific algorithm would normally be a vector containing the action selection probabilities. In Sections 4–4.2 we present the evolutionary algorithm that was implemented in order to solve the non-stationary multi-armed bandit problem that we are facing. Related work can be found in the paper by Fogel and Beyer [16], who solve a kind of a stationary two-armed bandit problem with Gaussian payoffs for both arms by means of an evolutionary algorithm. Their intention was to determine whether evolution favors the maximization of expected gains or not.

The structure of the algorithm that we implemented is illustrated in Fig. 1. The algorithm's population is a $s * n$ matrix with elements $P_{qi} \in \Re$, $0 \leqslant P_{qi} \leqslant 1$, $\sum_{i=1}^{n} P_{qi} = 1 \ \forall q = 1, 2, \ldots, s$, where n is the number of available actions and s the population size. Each row of the matrix corresponds to an individual, a vector whose elements are the selection probabilities of the available actions. After each individual has sampled from one of the n actions, all individuals are ranked in order of decreasing reward. The first e individuals in the ranking pass to the next generation with probability 1. We will call parameter *e elite count*. The observed rewards are then normalized and the selection operator is applied. In our algorithm we used stochastic uniform selection. The standard evolutionary algorithm operators were implemented as follows.

4.1. Crossover

With probability p_c, $0 < p_c < 1$, select two individuals a and b with elements P_{ai} and P_{bi} respectively, $i = 1, 2, \ldots, n$. p_c denotes the crossover probability.

- Create crossover mask m, where m is a random reordering of the elements of a.
- Create new individual c:
  $P_{ci} = P_{bi} + r(P_{mi} - P_{bi}), \ \forall i \in [1, n]$, where r is a random number uniformly distributed between 0 and 1: $r \sim U(0, 1)$.

4.2. Mutation

With probability p_m select individual a with elements p_{ai}, $i = 1, 2, \ldots, n$, where p_m is the mutation probability.

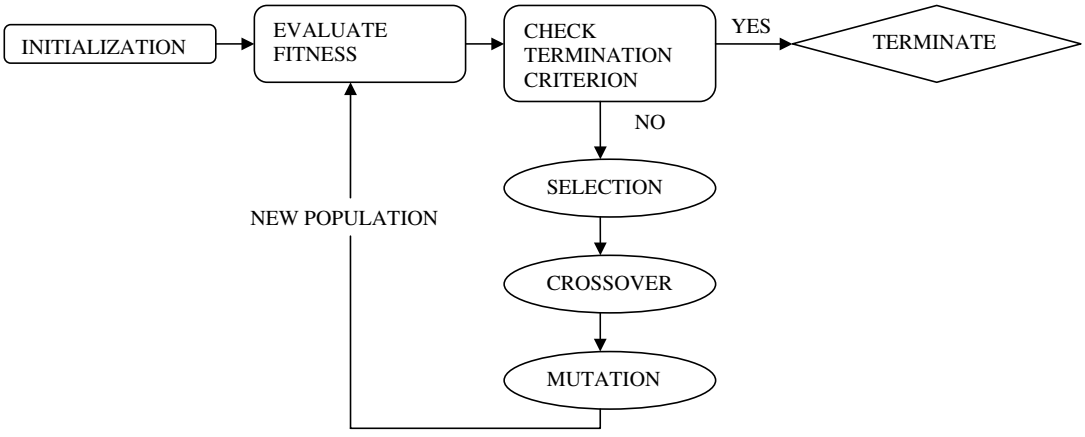

Fig. 1. Evolutionary algorithm structure.

- Mutate individual a:

  $p_{ak} \leftarrow p_{ak} + r(1 - p_{ak})$.

  $p_{ai} \leftarrow p_{ai} + r(0 - p_{ai}),\ \forall i \in [1, n],\ i \neq k$, where k is a randomly selected element and r uniformly distributed random variable between 0 and 1.

The parameters of the evolutionary algorithm are the population size s, the elite count e and the crossover and mutation probabilities p_c and p_m respectively.

5. Results and discussion

We used a setting with 15 available actions, $n = 15$, in order to measure both the ad hoc methods' and the evolutionary algorithm's performance. The time horizon was set to 1200 episodes, $k = 1200$. All actions when selected in the time interval from $t = 0$ to $t = 499$ produce rewards that are normally distributed with a standard deviation of 1.0, and means that are shown in Table 1, $Q_s^*(a_i) \sim N(\mu_i, 1.0)$. The change point of our non-stationary bandit task is $t_c = 500$. From that point on the actions' expected rewards are shifted to the values shown in Table 2 whereas their standard deviations remain unaltered.

The agent's goal is to maximize the sum of rewards $J = \sum_{j=1}^{1200} r_{t_j}$. As one may notice, the optimal strategy for this bandit task is to select a_{13} with a mean reward of 9.0 units for 500 episodes and then switch to a_3 with a mean reward of 10.0 units. In order for a learning algorithm which is applied to this problem to behave optimally, the algorithm should be able to waste as less time as possible in detecting the change in the bandit's arms, and concentrate its pulls to the new optimal action. Things get more complicated for the agent by the existence of a relatively large number of available actions with mean rewards that are close to each other.

5.1. Ad hoc methods results

In this section we present and comment the numerical results we obtained by means of simulation-based experiments for the ad hoc methodologies. We are interested in two measures; the value of the objective function J and the % fraction of simulation time that the agent behaved optimally. These two measures are not necessarily fully related as there are several sub-optimal actions in our example that yield relatively high rewards.

The probability matching algorithm is characterized by two parameters, the minimum selection probability $P_{\min}$ and the parameter β from Eq. (1.2). We tested four parameter configurations for this algorithm which are given in Table 3 along with the corresponding performance measures. As we can see from Table 3, the "short-sighted" ($P_{\min} = 0.0001$, $\beta = 0.5$) probability matching algorithm had the best results. The agent managed to accumulate 7428 reward units with 17.16% optimality. This percentage reveals an inherent weakness of the probability matching algorithm. This method cannot excel in maximizing reward when the differences between the available actions' values are relatively small something which is the case in our example. In general, we could argue that the probability matching algorithm will exhibit similar poor performance in a wide range of non-stationary bandit settings.

Table 1
Mean rewards μ_i of actions a_i for t_0 to t_{499}

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
μ_i	1.0	2.0	4.0	3.0	1.0	1.0	7.0	8.0	6.0	1.0	4.0	1.0	9.0	8.0	5.0

Table 2
Mean rewards μ_i of actions a_i for t_{500} to t_{1200}

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
μ_i	2.0	6.0	10.0	2.0	1.0	2.0	7.0	1.0	6.0	6.0	1.0	1.0	5.0	8.0	4.0

Table 3
Probability matching – parameters and results

	J	% Optimality
$P_{\min} = 0.0001$, $\beta = 0.3$	7044	15.75
$P_{\min} = 0.001$, $\beta = 0.3$	6900	15.25
$P_{\min} = 0.01$, $\beta = 0.3$	6732	15.91
$P_{\min} = 0.0001$, $\beta = 0.5$	7428	17.16

Table 4
Softmax selection – parameters and results

	J	% Optimality
$T = 0.3$, $\beta = 0.3$	10,980	81.08
$T = 0.3$, $\beta = 0.1$	10,428	75.66
$T = 0.5$, $\beta = 0.1$	10,284	70.0
$T = 0.5$, $\beta = 0.3$	10,800	78.83

The parameters to be set for the e-greedy method are e, the probability with which the agent explicitly explores in an episode, and β from the action value update rule. We carried out four trials of the e-greedy method with the parameter sets that are shown in Table 5. The best parameter set was found to be $e = 0.01$ and $\beta = 0.1$. With these parameters the e-greedy algorithm collected 10,464 reward units by selecting the optimal action approximately 8 times out of ten. By setting e to the value of 0.01 we manage to balance the exploitation–exploration trade-off adequately as the agent has the ability to track the changes that take place in its environment quickly without spending much time in costly exploratory moves.

The parameters (T and β) that we examined for the softmax selection method can be found in Table 4. It is reminded that high temperatures cause the agent to select actions nearly equiprobably while in the limiting case of $T \to 0$, the agent uses simple greedy action selection. Selecting the temperature is, in general, not as straight-forward as the selection of e in the e-greedy method as we must take into consideration the values of the actions' expected rewards. Despite this somehow increased complexity of the softmax selection algorithm, it can be effectively fine-tuned in order to behave in a highly desirable manner. In the problem that we examined in this paper the softmax algorithm with $T = 0.3$ and $\beta = 0.3$ displayed 81.08% optimal action selection and $J = 10{,}980$. Note that setting β to 0.3 results in the rapid decay of the effect that past rewards have on the computation of the actions' expected rewards.

The parameters that we used to examine the performance of the adaptive pursuit algorithm together with the corresponding results are given in Table 6. The best results that we observed were $J = 10{,}896$ and 84%

Table 5
E-greedy selection – parameters and results

	J	% Optimality
$e = 0.01$, $\beta = 0.1$	10,464	78.91
$e = 0.01$, $\beta = 0.3$	10,668	73.33
$e = 0.05$, $\beta = 0.3$	10,668	77.41
$e = 0.1$, $\beta = 0.1$	9756	67.25

Table 6
Adaptive pursuit – parameters and results

	J	% Optimality
$P_{\min} = 0.001$, $\beta = 0.3$ $\alpha = 0.3$	10,896	84.0
$P_{\min} = 0.001$, $\beta = 0.1$ $\alpha = 0.3$	10,250	72.83
$P_{\min} = 0.001$, $\beta = 0.3$ $\alpha = 0.1$	10,200	63.25
$P_{\min} = 0.01$, $\beta = 0.3$ $\alpha = 0.1$	9888	66.66

Table 7
EA – parameters and results

	J	% Optimality
$par = [10 \quad 1 \quad 0.1 \quad 0.1]$	9936	73.5
$par = [10 \quad 1 \quad 0.2 \quad 0.1]$	9830	72.66
$par = [10 \quad 2 \quad 0.1 \quad 0.1]$	10,112	78.33
$par = [15 \quad 2 \quad 0.2 \quad 0.1]$	9796	70.5
$par = [15 \quad 3 \quad 0.3 \quad 0.1]$	9862	74.83
$par = [15 \quad 5 \quad 0.3 \quad 0.1]$	9600	66.58
$par = [20 \quad 8 \quad 0.1 \quad 0.2]$	9420	64.25
$par = [20 \quad 8 \quad 0.1 \quad 0.3]$	9624	68
$par = [30 \quad 8 \quad 0.1 \quad 0.1]$	9600	68.66
$par = [30 \quad 8 \quad 0.1 \quad 0.2]$	9036	61.66
$par = [30 \quad 8 \quad 0.2 \quad 0.1]$	9144	60.25

Table 8
Ad hoc methods best parameters – total reward and % optimality

	J	% Optimality
E-greedy, $e = 0.01$, $\beta = 0.1$	10,464	78.9167
Softmax, $T = 0.3$, $\beta = 0.3$	10,980	81.08
Pursuit, $P_{\min} = 0.001$, $a = 0.3$, $\beta = 0.3$	10,896	84
Probability matching, $P_{\min} = 0.0001$, $\beta = 0.5$	7428	17.16

optimality, for the parameter set $P_{\min} = 0.001$, $a = 0.3$ and $\beta = 0.3$. Note that parameter $P_{\min}$ can only take values less than $\frac{1}{n}$. In practice much smaller values of $P_{\min}$ are required in order for the algorithm to work properly. Parameter a is the rate with which the greedy action's selection probability converges to $P_{\max}$. Bandit tasks with very noisy reward signals generally require small values of α. In our example this is not the case, so we can allow a relatively high α in order to increase the intensity of the exploitation.

The performances of the ad hoc methods with the best parameters are shown in a more compact form in Table 8.

5.2. Evolutionary algorithm results

Before we proceed to the presentation of the results obtained from the evolutionary algorithm we should point out some characteristics of the multi-armed bandit framework. The multi-armed bandit problem provides a setting where learning algorithms can be tested *on-line*. The term on-line indicates that the algorithm's performance is not evaluated solely in terms of how good the final solution was, but also based on the quality of the solutions it derived *during* the learning process. Evolutionary algorithms typically maintain large populations of candidate solutions that are evaluated one by one. However, the knowledge that the algorithm acquires by trying actions in a bandit problem can only be exploited to maximize the objective function only *across* generations and not *within* them, as the individuals that yield relatively small rewards will not be removed from the population until the next generation of individuals. Therefore, anything but a *small* population of individuals would be grossly sub-optimal for the multi-armed bandit problem.

Operators such as crossover and mutation introduce random changes to the population and are therefore responsible for the explicit search of the decision space. However, it is also important to prevent the random modification of individuals that yield high rewards. That is why the number of individuals that pass deterministically to the next generation, e (elite count), also plays an important role to the evolutionary algorithm's performance. This parameter guarantees that individuals with large rewards will survive to the next generation. The parameter sets that we used for the evolutionary algorithm along with the corresponding performance measures are shown in Table 7. The best parameter set for the evolutionary algorithm appears to be population size 10, elite count 2, $p_c = 0.1$ and $p_m = 0.1$. These parameters achieve a total reward of 10,104 units with 78.33% optimality.

Table 9
Ad hoc methods–EA best parameters – total reward and % optimality

	J	% Optimality
E-greedy, $e = 0.01$, $\beta = 0.1$	10,464	78.9167
Softmax, $T = 0.3$, $\beta = 0.3$	10,980	81.08
Pursuit, $P_{\min} = 0.001$, $a = 0.3$, $\beta = 0.3$	10,896	84
Probability matching, $P_{\min} = 0.0001$, $\beta = 0.5$	7428	17.16
EA, $s = 10$, $e = 2$, $p_c = 0.1$, $p_m = 0.1$	10,104	78.33

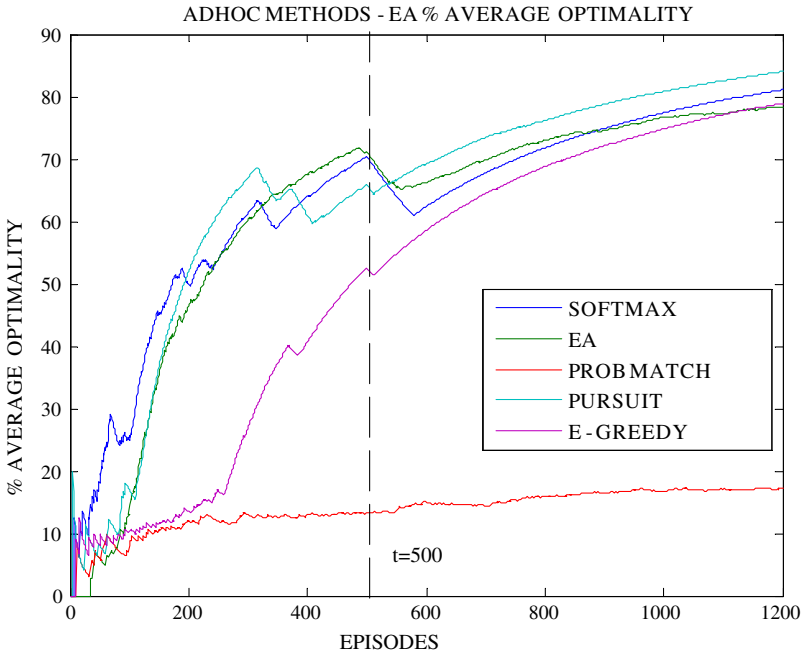

Fig. 2. Ad hoc methods – EA% average optimality chart.

5.3. Ad hoc methods – evolutionary algorithm results

We present the results for both the ad hoc techniques and the evolutionary algorithm with the best parameters in Table 9. The methods' measures of performance are presented graphically in Fig. 2. The first thing that someone notices is that all the methods except the probability matching algorithm performed very well. The adaptive pursuit algorithm ranks first with a 84.0% optimal action selection. The numerical results that we obtained indicate that despite the absence of theoretical guarantees of optimality, the ad hoc methods that we tested can excel in solving non-stationary multi-armed bandit problem of this type. The evolutionary algorithm that we presented also proved to be well-behaved for this problem as it succeeded in choosing the optimal action approximately 78 times out of 100. However, compared to the ad hoc methods we must note that the relatively large number of parameters complicates the process of fine-tuning the algorithm in order to obtain the best possible results for the non-stationary multi-armed bandit problem.

6. Conclusions

We examined a 15-armed non-stationary bandit problem with Gaussian rewards. The time horizon was set to 1200 epochs, while the change in the arms' reward distributions takes place in episode number 500. Four important ad hoc methods for solving the non-stationary bandit problem were presented: probability

matching, e-greedy action selection, softmax selection and finally, adaptive pursuit method. We also implemented an evolutionary algorithm that was applied in the same problem. In order to study the algorithms' performance we conducted a series of experiments. All ad hoc methodologies besides the probability matching algorithm achieved optimality at the level of 80%, with the adaptive pursuit technique ranking first. The evolutionary algorithm performed very well, displaying 78.3% optimality in selecting actions. The most important issue when applying these methodologies in a non-stationary bandit task is determining the best parameters for each method, as the performance of each method is greatly affected by parameter selection.

References

[1] D.A. Berry, B. Fristedt, Bandit Problems: Sequential Allocation of Experiments, Chapman and Hall, London, 1985.
[2] J.C. Gittins, Multi-armed Bandit Allocation Indices, Wiley, New York, 1989.
[3] P.P. Varaiya, J.C. Walrand, C. Buyukkoc, Extensions of the multi armed bandit problem: the discounted case, IEEE Transactions on Automated Control 30 (5) (1985) 426–439.
[4] H. Kaspi, A. Mandelbaum, Multi-armed bandits in discrete and continuous time, Annals of Applied Probability 8 (4) (1998) 1270–1290.
[5] J.S. Banks, R.K. Sundaram, Switching costs and the Gittins index, Econometrica 62 (1994) 687–694.
[6] Tackseung Jun, A survey on the bandit problem with switching costs, De Economist 152 (2004) 513–541.
[7] P. Auer, N. Cesa-Bianchi, Y. Freund, R.E. Schapire, Gambling in a rigged Casino: the adversarial multi-armed bandit problem, in: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, 1995, pp. 322–331.
[8] B. Leloup, L. Deveaux, Dynamic pricing on the Internet: theory and simulations, Electronic Commerce Research 1 (2001) 265–276.
[9] I. Valsecchi, Job assignment and bandit problems, International Journal of Manpower 24 (7) (2003) 844–866.
[10] R. Azoulay-Schwartz, S. Kraus, J. Wilkenfeld, Exploitation vs. exploration: choosing a supplier in an environment of incomplete information, Decision Support Systems 38 (2004) 1–18.
[11] B.P. McCall, J.J. McCall, Systematic search, belated information and the Gittins' index, Economics Letters 8 (1981) 327–333.
[12] D.V. Chulkov, M.S. Desai, Information technology project failures. Applying the bandit problem to evaluate managerial decision making, Information Management and Computer Security 13 (2) (2005) 135–143.
[13] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
[14] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, Journal of Artificial Intelligence Research 4 (1996) 237–285.
[15] D. Thierens, An adaptive pursuit strategy for allocating operator probabilities, in: Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2005), 2005, pp. 1539–1546.
[16] D.B. Fogel, H.G. Beyer, Do evolutionary processes minimize expected losses? Journal of Theoretic Biology 207 (2000) 117–123.
[17] M.A.L. Thathachar, P.S. Sastry, A class of rapidly converging algorithms for learning automata, IEEE Transactions on Systems, Man and Cybernetics SMC-15 (1985) 168–175.