An Adaptive Pursuit Strategy for Allocating Operator Probabilities

Dirk Thierens
Department of Computer Science
Utrecht University
The Netherlands
dirk.thierens@cs.uu.nl

ABSTRACT

Learning the optimal probabilities of applying an exploration operator from a set of alternatives can be done by self-adaptation or by adaptive allocation rules. In this paper we consider the latter option. The allocation strategies discussed in the literature basically belong to the class of probability matching algorithms. These strategies adapt the operator probabilities in such a way that they match the reward distribution. In this paper we introduce an alternative adaptive allocation strategy, called the adaptive pursuit method. We compare this method with the probability matching approach in a non-stationary environment. Calculations and experimental results show the superior performance of the adaptive pursuit algorithm. If the reward distributions stay stationary for some time, the adaptive pursuit method converges rapidly and accurately to an operator probability distribution that results in a much higher probability of selecting the current optimal operator and a much higher average reward than with the probability matching strategy. Yet most importantly, the adaptive pursuit scheme remains sensitive to changes in the reward distributions, and reacts swiftly to non-stationary shifts in the environment.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*
; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Self-modifying machines*

General Terms

Algorithms, Performance

Keywords

adaptive operator allocation, adaptive pursuit, non-stationary operator probabilities, non-stationary environment, multi-armed bandit

GECCO'05, June 25–29, 2005, Washington, DC, USA.

1. INTRODUCTION

Genetic algorithms usually apply their exploration operators with a fixed probability. There are however no general guidelines to help determine an optimal value for these probability values. In practice the user simply searches for a reasonable set of values by running a series of trial-and-error experiments. Clearly, this is a computationally expensive procedure and since genetic algorithms are often applied to computational intensive problems, the number of probability values tried has to remain limited. To make matters worse, the problem is compounded by the fact that there is no optimal fixed set of values for a particular problem instance. Depending on the current state of the search process the optimal probability values continuously change.

This problem has long been recognized and different adaptation methods have been proposed to solve it [4][5][11]. In general two classes of adaptation methods can be found:

1. **Self-adaptation.** The values of the operator probabilities are directly encoded in the representation of the individual solutions. These values basically hitchhike with the solutions that are being evolved through the regular search process. The idea is that the operator probability values and problem solutions co-evolve to (near)-optimal settings.

2. **Adaptive allocation rule.** The values of the operator probabilities are adapted following an 'out-of-the-evolutionary-loop' learning rule according to the quality of new solutions created by the operators.

Self-adaptation is particularly applied within Evolutionary Strategies and Evolutionary Programming for numerical optimization problems. When applied to discrete optimization or adaptation problems using genetic algorithms, its success is somewhat limited as compared to the adaptive allocation rule method. In this paper we will focus on the latter class. Looking at the literature it becomes clear that most adaptive allocation rules used belong to the probability matching type [2][3][6][7][8][9][10][14]. Here we propose an adaptive pursuit method as allocation rule and compare it to the probability matching strategy. Results indicate that the adaptive pursuit method appears to be a better adaptive operator allocation rule than the traditionally used probability matching algorithm.

The paper is organized as follows. Section 2 describes the probability matching method and specifies an implementation particularly suited for non-stationary environments.

Section 3 introduces the adaptive pursuit algorithm. Section 4 shows experimental results of both adaptive allocation techniques, which is followed by the Conclusion.

2. PROBABILITY MATCHING

An adaptive operator allocation rule is an algorithm that iteratively chooses one of its operators to apply to an external environment [12]. The environment returns a reward - possibly zero - and the allocation rule uses this reward and its internal state to adapt the probabilities with which the operators are chosen. It is crucial to note that the environment we consider here is non-stationary, meaning that the probability distribution according to which a reward is generated for a given operator changes during the runtime of the allocation algorithm. Unfortunately the non-stationarity requirement excludes the use of a large number of adaptive strategies that have been developed for the well-known multi-armed bandit problem [1].

Formally, we have a set of K operators $\mathcal{A} = \{a_1, \ldots, a_K\}$, and a probability vector $\mathcal{P}(t) = \{\mathcal{P}_1(t), \ldots, \mathcal{P}_K(t)\}$ $(\forall t : 0 \leq \mathcal{P}_i(t) \leq 1; \sum_{i=1}^{K} \mathcal{P}_i(t) = 1)$. The adaptive allocation rule selects an operator to be executed in proportion to the probability values specified in $\mathcal{P}(t)$. When an operator a is applied to the environment at time t, a reward $\mathcal{R}_a(t)$ is returned. Each operator has an associated reward, which is a non-stationary random variable. All rewards are collected in the reward vector $\mathcal{R}(t) = \{\mathcal{R}_1(t), \ldots, \mathcal{R}_K(t)\}$. In addition to the operator probability vector $\mathcal{P}(t)$ the adaptive allocation rule maintains a quality vector $\mathcal{Q}(t) = \{\mathcal{Q}_1(t), \ldots, \mathcal{Q}_K(t)\}$ that specifies a running estimate of the reward for each operator. Whenever an operator is executed his current estimate $\mathcal{Q}_a(t)$ is adapted. The allocation algorithm is run for a period of T time steps. The goal is to maximize the expected value of the cumulative reward $\mathcal{E}[\mathcal{R}] = \sum_{t=1}^{T} \mathcal{R}_a(t)$ received by the adaptive allocation rule. Since the environment is non-stationary, the estimate of the reward for each operator is only reliable when the rewards received are not too old. An elegant, iterative method to compute such a running estimate is the exponential, recency-weighted average that updates the current estimate with a fraction of the difference of the target value and the current estimate:

$$\mathcal{Q}_a(t+1) = \mathcal{Q}_a(t) + \alpha[\mathcal{R}_a(t) - \mathcal{Q}_a(t)] \quad (1)$$

with the adaptation rate $\alpha : 0 < \alpha \leq 1$. The basic probability matching allocation rule computes each operator's selection probability $\mathcal{P}_a(t)$ as the proportion of the operator's reward estimate $\mathcal{Q}_a(t)$ to the sum of all reward estimates $\sum_{i=1}^{K} \mathcal{Q}_i(t)$. However, this may lead to the loss of some operators. Once a probability $\mathcal{P}_a(t)$ becomes equal to 0, the operator will no longer be selected and its reward estimate can no longer be updated. This is an unwanted property in a non-stationary environment because the operator might become valuable again in a future stage of the search process. We always need to be able to apply any operator and update its current value estimate. To ensure that no operator gets lost we enforce a minimal value $P_{min} : 0 < P_{min} < 1$ for each selection probability. As a result the maximum value any operator can achieve is $P_{max} = 1 - (K-1)P_{min}$ where K is the number of operators. The rule for updating the probability vector $\mathcal{P}_a(t)$ now becomes:

$$\mathcal{P}_a(t+1) = P_{min} + (1 - K \cdot P_{min}) \frac{\mathcal{Q}_a(t)}{\sum_{i=1}^{K} \mathcal{Q}_i(t)}. \quad (2)$$

It is easy to see in the above equation that when an operator does not receive any reward for a long time its value estimate $\mathcal{Q}_a(t)$ will converge to 0 and its probability of being selected $\mathcal{P}_a(t)$ converges to P_{min}. It is also clear that when only one operator receives a reward during a long period of time - and all other operators get no reward - then its selection probability $\mathcal{P}_a(t)$ converges to $P_{min} + 1 - K \cdot P_{min} = P_{max}$. Furthermore, $0 < \mathcal{P}_a(t) < 1$ and their sum equals 1:

$$\sum_{a=1}^{K} \mathcal{P}_a(t) = K \cdot P_{min} + \frac{1 - K \cdot P_{min}}{\sum_{i=1}^{K} \mathcal{Q}_i(t-1)} \sum_{a=1}^{K} \mathcal{Q}_a(t-1) = 1.$$

Finally, our probability matching algorithm is specified as:

ProbabilityMatching($\mathcal{P}, \mathcal{Q}, K, P_{min}, \alpha$)
1 **for** $i \leftarrow 1$ **to** K
2 **do** $\mathcal{P}(i) \leftarrow \frac{1}{K}; \mathcal{Q}(i) \leftarrow 1.0$
3 **while** NotTerminated?()
4 **do** $a^s \leftarrow$ ProportionalSelectOperator($\mathcal{P}$)
5 $R_{a^s}(t) \leftarrow$ GetReward(a^s)
6 $\mathcal{Q}_{a^s}(t+1) = \mathcal{Q}_{a^s}(t) + \alpha[R_{a^s}(t) - \mathcal{Q}_{a^s}(t)]$
7 **for** $a \leftarrow 1$ **to** K
8 **do** $\mathcal{P}_a(t+1) = P_{min} + (1 - K \cdot P_{min}) \frac{\mathcal{Q}_a(t)}{\sum_{i=1}^{K} \mathcal{Q}_i(t)}$

The probability matching allocation rule as specified in equation 2 is able to adapt to non-stationary environments. Unfortunately, it pays a heavy price for this in terms of reward maximization. This is most obvious when we consider a non-stationary environment. Suppose we have only two operators a_1 and a_2 with constant reward values $\mathcal{R}_1$ and $\mathcal{R}_2$. From equation 2 it follows that

$$\frac{\mathcal{P}_1(t) - P_{min}}{\mathcal{P}_2(t) - P_{min}} = \frac{\mathcal{R}_1}{\mathcal{R}_2}.$$

Assume that $\mathcal{R}_1 > \mathcal{R}_2$. An ideal adaptive allocation rule should notice in this stationary environment that the operator a_1 has a higher reward than operator a_2. The allocation rule should therefore maximize the probability of applying operator a_1 and only apply a_2 with the minimal probability P_{min}. However, the closer the rewards $\mathcal{R}_1$ and $\mathcal{R}_2$ are, the less optimal the probability matching rule behaves. For instance, when $P_{min} = 0.1$, $\mathcal{R}_1 = 10$, and $\mathcal{R}_2 = 9$ then $\mathcal{P}_1 = 0.52$ and $\mathcal{P}_2 = 0.48$, which is far removed from the desired values of $\mathcal{P}_1 = 0.9$ and $\mathcal{P}_2 = 0.1$.

Matching the reward probabilities is not an optimal strategy for allocating operator probabilities in an optimizing algorithm. In the next section we propose the adaptive pursuit strategy as an alternative allocation method that is far better at maximizing the rewards received while still maintaining the ability to swiftly react to any changes in a non-stationary environment.

3. ADAPTIVE PURSUIT ALGORITHM

Pursuit algorithms are a class of rapidly converging algorithms for learning automata proposed by Thathachar and Sastry [13]. They represent adaptive allocation rules that adapt the operator probability vector $\mathcal{P}(t)$ in such a way that the algorithm pursues the operator a^* that currently has the maximal estimated reward $\mathcal{Q}_{a^*}(t)$. To achieve this, the pursuit method increases the selection probability $\mathcal{P}_{a^*}(t)$ and decreases all other probabilities $\mathcal{P}_a(t), \forall a \neq a^*$. Pursuit algorithms originated from the field of learning automata.

However, they are designed for stationary environments for which it can be proved that they are ϵ-optimal. The ϵ-optimality property means that in every stationary environment, there exists a learning rate $\beta^* > 0$ and time $t_0 > 0$, such that for all learning rates $0 < \beta \leq \beta^* \leq 1$ and for any $\delta \in [0 \dots 1]$ and $\epsilon \in [0 \dots 1]$:

$$\text{Prob}[\mathcal{P}_a^{optimal}(t) > 1 - \epsilon] > 1 - \delta \ , \qquad \forall t > t_0.$$

In practice this means that if the the learning rate β is small enough as a function of the reward distribution correct convergence is assured.

As in the probability matching allocation rule, the pursuit algorithm proportionally selects an operator to execute according to the probability vector $\mathcal{P}(t)$, and updates the corresponding operator's quality or estimated reward $\mathcal{Q}_a(t)$. Subsequently, the current best operator is chosen (ties are broken at random)[1], $a^* = \text{argmax}_a[\mathcal{Q}_a(t+1)]$ and its selection probability is increased

$$\mathcal{P}_{a^*}(t+1) = (1-\beta)\mathcal{P}_{a^*}(t) + \beta,$$

while the other operators have their selection probability decreased

$$\forall a \neq a^* : \mathcal{P}_a(t+1) = (1-\beta)\mathcal{P}_a(t).$$

It is clear from $(1-\beta)\mathcal{P}_{a^*}(t) + \beta = \mathcal{P}_{a^*}(t) + \beta(1 - \mathcal{P}_{a^*}(t))$ that if a particular operator is repeatedly the best operator its selection probability will converge to 1, while the selection probabilities of the other operators will converge to 0 and they will no longer be applied. Consequently, the pursuit algorithm cannot be used in a non-stationary environment. To make the method suitable for non-stationary environments we change the probability updating scheme. The modified update rule ensures that the probability vector is still pursuing the current best operator at the same rate as the standard method, but now the exponential, recency-weighted averages of the operator probabilities are enforced to stay within the interval $[P_{min} \dots P_{max}]$ with $0 < P_{min} < P_{max} < 1$. Calling $a^* = \text{argmax}_a[\mathcal{Q}_a(t+1)]$ the current best operator, we get:

$$\mathcal{P}_{a^*}(t+1) = \mathcal{P}_{a^*}(t) + \beta[P_{max} - \mathcal{P}_{a^*}(t)] \tag{3}$$

and

$$\forall a \neq a^* : \mathcal{P}_a(t+1) = \mathcal{P}_a(t) + \beta[P_{min} - \mathcal{P}_a(t)] \tag{4}$$

under the constraint:

$$P_{max} = 1 - (K-1)P_{min} \ . \tag{5}$$

The constraint ensures that if $\sum_{a=1}^{K} \mathcal{P}_a(t) = 1$ the sum of the updated probabilities remains equal to 1:

$$\begin{aligned}
&\sum_{a=1}^{K} \mathcal{P}_a(t+1) = 1 \\
&\quad \Leftrightarrow \quad \text{rhs. eqt.(3)} + \text{rhs. eqt.(4)} = 1 \\
&\quad \Leftrightarrow \quad (1-\beta)\sum_{a=1}^{K} \mathcal{P}_a(t) + \beta[P_{max} + (K-1)P_{min}] = 1 \\
&\quad \Leftrightarrow \quad P_{max} = 1 - (K-1)P_{min}.
\end{aligned}$$

[1]The tie breaking ensures that if two operators consistently receive the same reward, they will also be applied with the same probability.

Note that since $P_{min} < P_{max}$ the constraint can only be fulfilled[2] if $P_{min} < \frac{1}{K}$. An interesting value for the minimal probability is $P_{min} = \frac{1}{2K}$ which results in the maximum probability $P_{max} = \frac{1}{2} + \frac{1}{2K}$. An intuitive appealing way to look at these values is that the optimal operator will be selected half the time, while the other half of the time all operators have an equal probability of being selected.

Finally, we can now specify more formally the adaptive pursuit algorithm:

AdaptivePursuit($\mathcal{P}, \mathcal{Q}, K, P_{min}, \alpha, \beta$)
1 $P_{max} \leftarrow 1 - (K-1)P_{min}$
2 **for** $i \leftarrow 1$ **to** K
3 **do** $\mathcal{P}(i) \leftarrow \frac{1}{K}$; $\mathcal{Q}(i) \leftarrow 1.0$
4 **while** NotTerminated?()
5 **do** $a^s \leftarrow$ ProportionalSelectOperator($\mathcal{P}$)
6 $R_{a^s}(t) \leftarrow$ GetReward(a^s)
7 $\mathcal{Q}_{a^s}(t+1) = \mathcal{Q}_{a^s}(t) + \alpha[R_{a^s}(t) - \mathcal{Q}_{a^s}(t)]$
8 $a^* \leftarrow$ Argmax$_a(\mathcal{Q}_a(t+1))$
9 $\mathcal{P}_{a^*}(t+1) = \mathcal{P}_{a^*}(t) + \beta[P_{max} - \mathcal{P}_{a^*}(t)]$
10 **for** $a \leftarrow 1$ **to** K
11 **do if** $a \neq a^*$
12 **then** $\mathcal{P}_a(t+1) = \mathcal{P}_a(t) + \beta[P_{min} - \mathcal{P}_a(t)]$

Consider again the 2-operator stationary environment at the end of the previous section with $P_{min} = 0.1$, $\mathcal{R}_1 = 10$, and $\mathcal{R}_2 = 9$. As opposed to the probability matching rule, the adaptive pursuit method will play the better operator a_1 with maximum probability $P_{max} = 0.9$. It also keeps playing the poorer operator a_2 with minimal probability $P_{min} = 0.1$ in order to maintain its ability to adapt to any change in the reward distribution.

4. EXPERIMENTAL RESULTS

To get an idea of the dynamic behavior of these adaptive allocation rules we compare the probability matching algorithm, the adaptive pursuit method, and the non-adaptive, equal-probability strategy, on the following non-stationary environment. We consider an environment with 5 operators (or arms in the multi-bandit problem terminology). Each operator a receives a uniformly distributed reward $\mathcal{R}_a$ between the respective boundaries $\mathcal{R}_5 = \mathcal{U}[4 \dots 6], \mathcal{R}_4 = \mathcal{U}[3 \dots 5], \mathcal{R}_3 = \mathcal{U}[2 \dots 4], \mathcal{R}_2 = \mathcal{U}[1 \dots 3]$, and $\mathcal{R}_1 = \mathcal{U}[0 \dots 2]$. After a fixed time interval ΔT these reward distributions are randomly reassigned to the operators, under the constraint that the current best operator-reward association effectively has to change to a new couple. Specifically, the non-stationary environment in the simulation switches 10 times with the following pattern: 01234 → 41203 → 24301 → 12043 → 41230 → 31420 → 04213 → 23104 → 14302 → 40213, where each sequence orders the operators in descending value of reward. For instance '41203' means that operator a_4 receives the highest reward $\mathcal{R}_5$, operator a_1 receives the second highest reward $\mathcal{R}_4$, operator a_2 receives reward $\mathcal{R}_3$, operator a_0 receives reward $\mathcal{R}_2$, and finally operator a_3 receives the lowest reward $\mathcal{R}_1$. If we had full knowledge of the reward distributions and their switching pattern we

[2]Strictly speaking, P_{min} can be equal to $\frac{1}{K}$. This happens in the case of only 2 operators ($K = 2$) and a lower bound probability of $P_{min} = 0.5$. However, now P_{max} also equals 0.5 so there is no adaptation possible.

could always pick the optimal operator a^* and achieve an expected reward

$$\mathcal{E}[\mathcal{R}^{Opt}] = \mathcal{R}_{a^*} = 5 \ .$$

Clearly, this value can never be obtained by any adaptive allocation strategy since it always needs to pay a price for exploring the effects of alternative actions. Nevertheless, it does represent an upper bound of the expected reward. When operating in a stationary environment the allocation strategies converge to a fixed operator probability distribution. Using this distribution we can compute the maximum achievable expected reward for each allocation rule, which is preferably close to the theoretical upper bound. In a non-stationary environment, we aim to achieve this value as quick as possible, while still being able to react swiftly to any change in the reward distributions. In the experiments we have taken the value $P_{min} = \frac{1}{2K} = 0.1$ for the minimum probability each operator will be applied in the adaptive allocation schemes. For a stationary environment - this is, when the assignment of reward distributions to the arms are not switched - we can compute the expected reward and the probability of choosing the optimal operator once the operator probability vectors have converged.

1. **Non-adaptive, equal-probability allocation rule.**

   This strategy simply selects each operator with equal probability. The probability of choosing the optimal operator a^*_{Fixed} is

   $$\begin{aligned} \text{Prob}[a^s = a^*_{Fixed}] &= \frac{1}{K} \\ &= 0.2 \ . \end{aligned}$$

   The expected reward becomes

   $$\begin{aligned} \mathcal{E}[\mathcal{R}^{Fixed}] &= \sum_{a=1}^{K} \mathcal{E}[\mathcal{R}_a]\text{Prob}[a^s = a] \\ &= \frac{\sum_{a=1}^{K} \mathcal{E}[\mathcal{R}_a]}{K} \\ &= 3 \ . \end{aligned}$$

2. **Probability matching allocation rule.**

   For the probability matching updating scheme the probability of choosing the optimal operator $a^*_{ProbMatch}$ is

   $$\begin{aligned} &\text{Prob}[a^s = a^*_{ProbMatch}] \\ &= P_{min} + (1 - K.P_{min}) \frac{\mathcal{E}[\mathcal{R}_{a^*}]}{\sum_{a=1}^{K} \mathcal{E}[\mathcal{R}_a]} \\ &= 0.2666\ldots \ . \end{aligned}$$

   The expected reward becomes

   $$\begin{aligned} &\mathcal{E}[\mathcal{R}^{ProbMatch}] \\ &= \sum_{a=1}^{K} \mathcal{E}[\mathcal{R}_a]\text{Prob}[a^s = a] \\ &= \sum_{a=1}^{K} a[P_{min} + (1 - K \cdot P_{min}) \frac{\mathcal{E}[\mathcal{R}_a]}{\sum_{a=1}^{K} \mathcal{E}[\mathcal{R}_a]}] \\ &= 3.333\ldots \ . \end{aligned}$$

3. **Adaptive pursuit allocation rule.**

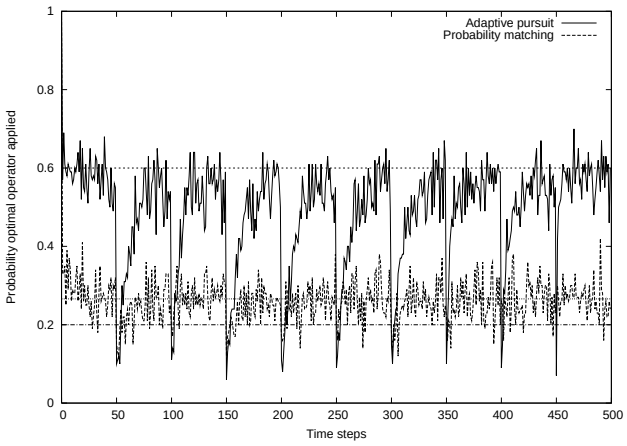

Figure 1: The probability of selecting the optimal operator at each time step in the non-stationary environment with switching interval $\Delta T = 50$ time steps (learning rates $\alpha = 0.8$; $\beta = 0.8$; $Pmin = 0.1$; $K = 5$; results are averaged over 100 runs). The horizontal lines show the expected values for the non-switching, stationary environment for resp. adaptive pursuit (0.6), probability matching (0.27), and random selection (0.2).

For the adaptive pursuit updating scheme the probability of choosing the optimal operator $a^*_{AdaPursuit}$ is

$$\begin{aligned} \text{Prob}[a^s = a^*_{AdaPursuit}] &= 1 - (K-1) \cdot P_{min} \\ &= 0.6 \ . \end{aligned}$$

The expected reward becomes

$$\begin{aligned} &\mathcal{E}[\mathcal{R}^{AdaPursuit}] \\ &= \sum_{a=1}^{K} \mathcal{E}[\mathcal{R}_a]\text{Prob}[a^s = a] \\ &= P_{max}\ \mathcal{E}[\mathcal{R}_{a^*}] + P_{min} \sum_{a=1, a \neq a^*}^{K} \mathcal{E}[\mathcal{R}_a] \\ &= 4 \ . \end{aligned}$$

The computed expected rewards and probabilities of applying the optimal operator show that both adaptive allocation rules have a better performance than the non-adaptive strategy that simply selects each operator with equal probability. More interesting, they also show that - after convergence - the adaptive pursuit algorithm has a significantly better performance than the probability matching algorithm in the stationary environment. The probability matching algorithm will apply the optimal operator in only 27% of the trials while the pursuit algorithm will be optimal in 60% of the cases. Similarly, the probability matching algorithm has an expected reward of 3.3 versus an expected reward of 4 for the pursuit method. Of course, this assumes that both adaptive strategies are able to converge correctly and rapidly. For non-stationary environments it is vital that the adaptive allocation techniques converge quickly and accurately, and at the same time maintain the flexibility to swiftly track any

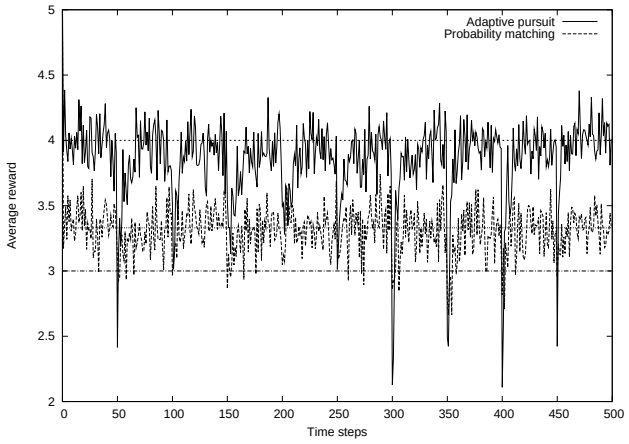

Figure 2: The average reward received at each time step in the non-stationary environment with switching interval $\Delta T = 50$ time steps (learning rates $\alpha = 0.8$; $\beta = 0.8$; $Pmin = 0.1$; $K = 5$; results are averaged over 100 runs). The horizontal lines show the expected values for the non-switching, stationary environment for resp. adaptive pursuit (4), probability matching (3.33), and random selection (3).

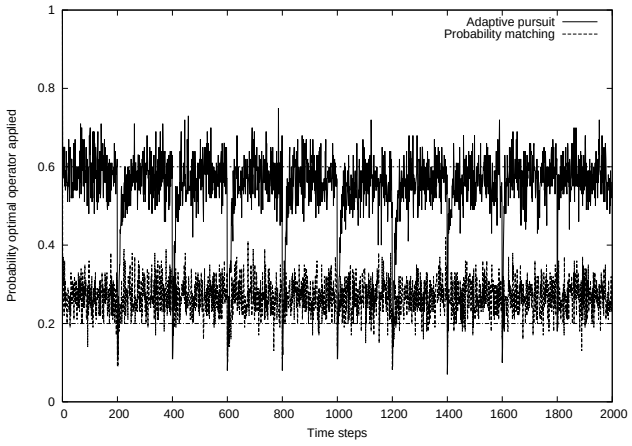

Figure 3: The probability of selecting the optimal operator at each time step in the non-stationary environment with switching interval $\Delta T = 200$ time steps (learning rates $\alpha = 0.8$; $\beta = 0.8$; $Pmin = 0.1$; $K = 5$; results are averaged over 100 runs). The horizontal lines show the expected values for the non-switching, stationary environment for resp. adaptive pursuit (0.6), probability matching (0.27), and random selection (0.2).

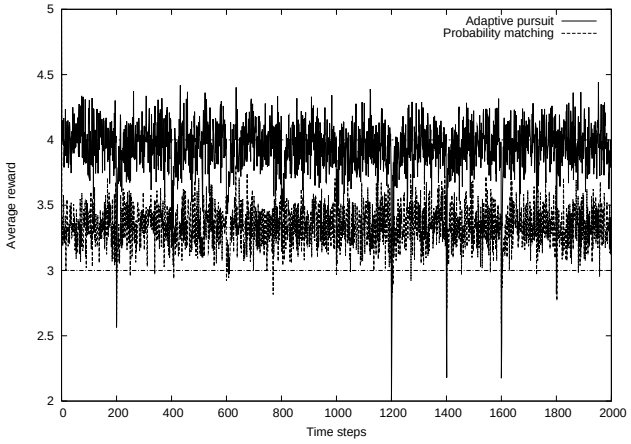

Figure 4: The average reward received at each time step in the non-stationary environment with switching interval $\Delta T = 200$ time steps (learning rates $\alpha = 0.8$; $\beta = 0.8$; $Pmin = 0.1$; $K = 5$; results are averaged over 100 runs). The horizontal lines show the expected values for the non-switching, stationary environment for resp. adaptive pursuit (4), probability matching (3.33), and random selection (3).

changes in the reward distributions. Experimental results on the above specified non-stationary environment show that the adaptive pursuit method does indeed possess these capabilities. In our first simulation we have taken a switching interval $\Delta T = 50$ time steps. The results shown are all averaged over 100 independent runs. Figures 1 and 2 clearly show that the adaptive pursuit algorithm is capable of accurate and fast convergence. At the same time it is very responsive to changes in the reward distribution. Whenever the operator-reward associations are reassigned the performance of the adaptive pursuit algorithm plunges since it is now pursuing an operator that is no longer optimal. It does not take long though for the strategy to correct itself, and to pursue the current optimal operator again. This is in contrast with the probability matching algorithm where the differences between the operator selection probabilities are much smaller and the changes in the reward distributions cause only minor adaptations. Of course a more significant reaction would be observed for the probability matching method if the rewards would have a much large difference between them. The key point though is that in practice one usually will have to deal with reward differences of a few percent, not an order of magnitude.

In a second experiment we have increased the switching interval ΔT to 200 times steps (Figures 3 and 4). Given more time to adapt one can see that both adaptive allocation strategies approach the values that where computed above for the stationary environment.

The results in the Figures 1, 2, 3 and 4 were obtained for a learning rate $\alpha = 0.8$ when updating $Q_a(t)$ in Equation 1, and a learning rate $\beta = 0.8$ when updating $P_a(t)$ in Equations 3 and 4. These values gave the best performance for this particular problem instance. Tables 1, 2, 3, and 4 show the performance for different settings of the learning rates. For low values of the learning rates the adaptive schemes

do not react swiftly enough to the rapidly changing reward distributions. Naturally, the high learning rates are only possible because at each time step an actual reward is given by the environment. If the rewards would only be given with a probability less than 1, the learning rates would necessarily be small to ensure meaningful running estimates that are exponentially, recency-weighted. It should be noted though that whatever the values of the learning rates the adaptive pursuit method keeps outperforming the probability matching scheme.

5. CONCLUSION

Adaptive allocation rules are often used for learning the optimal probability values of applying a fixed set of exploration operators. Traditionally, the allocation strategies adapt the operator probabilities in such a way that they match the distribution of the rewards. In this paper, we have introduced an adaptive pursuit allocation rule and compared it with the probability matching algorithm in a non-stationary environment. Calculations and experimental results show the superior performance of the adaptive pursuit method. The strategy converges accurately and rapidly, yet remains able to swiftly react to any change in the reward distributions.

6. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM j. Computing* Vol.32, No.1, pp.48–77, 2002.

[2] D.W. Corne, M.J. Oates, and D.B. Kell. On fitness distributions and expected fitness gain of mutation rates in parallel evolutionary algorithms. *Proc. 7th Intern. Conf. on Parallel Problem Solving from Nature*. LNCS Vol. 2439, pp.132–141, 2002.

[3] L. Davis. Adapting operator probabilities in genetic algorithms. *Proc. Third Intern. Conf. on Genetic Algorithms and their Applications*. pp. 61–69, 1989.

[4] A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124-141, 1999.

[5] D.E. Goldberg. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Machine Learning*. Vol.5, pp. 407–425, 1990.

[6] T.P. Hong, H.S. Wang, and W.C. Chen. Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of Heuristics*. Vol.6, pp.439–455, 2000.

[7] C. Igel, and M. Kreutz. Operator adaptation in evolutionary computation and its application to structure optimization of neural networks. *Neurocomputing* Vol.55, pp.347–361, 2003.

[8] B. Julstrom. What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. *Proc. Sixth Intern. Conf. on Genetic Algorithms*. pp. 81–87, 1995.

[9] F. G. Lobo, and D. E. Goldberg. Decision making in a hybrid genetic algorithm. *Proc. IEEE Intern. Conf. on Evolutionary Computation*. pp. 122–125, 1997.

[10] D. Schlierkamp-Voosen, and H. Mühlenbein. Strategy adaptation by competing subpopulations. *Proc. Intern. Conf. of Parallel Problem Solving from Nature* pp.199–208, 1994.

[11] J.E. Smith, and T.C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing* No.1, pp.81–87, 1997.

[12] R.S. Sutton, and A.G. Barto. *Reinforcement Learning: an introduction*. MIT Press, 1998.

[13] M.A.L. Thathachar, and P.S. Sastry. A Class of Rapidly Converging Algorithms for Learning Automata. *IEEE Transactions on Systems, Man and Cybernetics*. Vol.SMC-15, pp. 168-175, 1985.

[14] A. Tuson, and P. Ross. Adapting operator settings in genetic algorithms. *Evolutionary Computation* Vol.6, No.2, pp.161–184, 1998.

α	*Probab. Match.*	*Adaptive Pursuit:* (β)								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.10	0.218	0.248	0.272	0.281	0.276	0.290	0.284	0.289	0.288	0.287
0.20	0.229	0.281	0.298	0.315	0.321	0.327	0.329	0.332	0.329	0.340
0.30	0.243	0.313	0.356	0.373	0.381	0.388	0.392	0.386	0.393	0.397
0.40	0.249	0.352	0.401	0.411	0.429	0.427	0.434	0.439	0.436	0.445
0.50	0.254	0.381	0.423	0.443	0.451	0.456	0.448	0.459	0.467	0.471
0.60	0.259	0.392	0.447	0.461	0.474	0.477	0.484	0.484	0.492	0.488
0.70	0.259	0.404	0.448	0.477	0.480	0.490	0.492	0.496	0.496	0.493
0.80	0.257	0.408	0.462	0.478	0.482	0.491	0.495	0.499	0.507	0.502
0.90	0.262	0.404	0.455	0.468	0.476	0.492	0.497	0.493	0.502	0.495

Table 1: The average probability of selecting the optimal operator in the non-stationary environment with switching interval $\Delta T = 50$ time steps for different adaptation rates α and β ($Pmin = 0.1;\ K = 5$; results are averaged over 100 runs).

α	*Probab. Match.*	*Adaptive Pursuit:* (β)								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.10	3.115	3.269	3.392	3.434	3.442	3.472	3.478	3.489	3.478	3.474
0.20	3.166	3.428	3.515	3.568	3.597	3.607	3.621	3.612	3.625	3.635
0.30	3.221	3.489	3.619	3.669	3.689	3.703	3.713	3.710	3.717	3.730
0.40	3.243	3.553	3.685	3.715	3.746	3.751	3.767	3.777	3.778	3.788
0.50	3.270	3.589	3.715	3.765	3.787	3.791	3.787	3.803	3.812	3.825
0.60	3.276	3.612	3.742	3.791	3.807	3.822	3.831	3.839	3.848	3.844
0.70	3.286	3.634	3.740	3.808	3.815	3.840	3.839	3.856	3.846	3.842
0.80	3.288	3.627	3.758	3.808	3.829	3.830	3.853	3.859	3.871	3.862
0.90	3.308	3.627	3.743	3.789	3.815	3.844	3.845	3.840	3.861	3.851

Table 2: The average reward received in the non-stationary environment with switching interval $\Delta T = 50$ time steps for different adaptation rates α and β ($Pmin = 0.1;\ K = 5$; results are averaged over 100 runs).

α	*Probab. Match.*	*Adaptive Pursuit:* (β)								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.10	0.247	0.399	0.414	0.416	0.422	0.423	0.427	0.422	0.423	0.429
0.20	0.257	0.491	0.498	0.508	0.508	0.509	0.515	0.514	0.511	0.516
0.30	0.260	0.520	0.530	0.537	0.537	0.538	0.542	0.540	0.543	0.547
0.40	0.264	0.534	0.546	0.550	0.551	0.554	0.556	0.555	0.559	0.558
0.50	0.265	0.539	0.553	0.557	0.557	0.559	0.559	0.561	0.561	0.562
0.60	0.264	0.537	0.552	0.556	0.558	0.561	0.562	0.565	0.564	0.563
0.70	0.264	0.538	0.552	0.555	0.556	0.560	0.560	0.561	0.560	0.561
0.80	0.267	0.528	0.541	0.549	0.550	0.552	0.557	0.554	0.556	0.560
0.90	0.266	0.521	0.537	0.538	0.546	0.547	0.547	0.549	0.550	0.553

Table 3: The average probability of selecting the optimal operator in the non-stationary environment with switching interval $\Delta T = 200$ time steps for different adaptation rates α and β ($Pmin = 0.1;\ K = 5$; results are averaged over 100 runs).

α	*Probab. Match.*	*Adaptive Pursuit:* (β)								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.10	3.233	3.719	3.757	3.767	3.768	3.775	3.778	3.780	3.776	3.789
0.20	3.287	3.834	3.853	3.877	3.879	3.879	3.893	3.891	3.887	3.892
0.30	3.302	3.873	3.896	3.916	3.912	3.914	3.922	3.921	3.923	3.934
0.40	3.315	3.886	3.915	3.926	3.932	3.933	3.939	3.942	3.948	3.938
0.50	3.320	3.891	3.925	3.940	3.939	3.945	3.940	3.946	3.946	3.950
0.60	3.323	3.890	3.926	3.936	3.941	3.949	3.947	3.956	3.955	3.951
0.70	3.322	3.894	3.928	3.936	3.943	3.948	3.948	3.947	3.947	3.951
0.80	3.333	3.878	3.912	3.934	3.937	3.934	3.946	3.940	3.945	3.951
0.90	3.329	3.881	3.916	3.913	3.933	3.933	3.933	3.938	3.936	3.944

Table 4: The average reward received in the non-stationary environment with switching interval $\Delta T = 200$ time steps for different adaptation rates α and β ($Pmin = 0.1;\ K = 5$; results are averaged over 100 runs).