Enhancing Stochasticity in Reinforcement Learning Schemes: Application to the Exploration of Binary Domains

D. KONTORAVDIS, A. LIKAS AND A. STAFYLOPATIS

Department of Electrical and Computer Engineering
National Technical University of Athens
157 73 Zographou, Athens, Greece

CONTENTS

ABSTRACT

The paper develops reinforcement algorithms for networks of stochastic units which select their output based on a distribution whose dependence on the controllable parameters (weights) of the network is not deterministic. A special case of the proposed schemes concerns those applied to Normal/Bernoulli units, which are binary units with two stochastic levels. Both REINFORCE algorithms as well as algorithms not belonging to the REINFORCE class have been developed. All schemes are designed to exploit the two parameters of a normal distribution in order to explore discrete domains. The ability of the proposed algorithms to perform efficient exploration is tested in a number of optimization problems concerning the maximization of a set of functions defined on binary domains. Particular emphasis has been given on deriving schemes having the property of sustained exploration. Obtained results indicate the superiority of the reinforcement schemes applied to Normal/Bernoulli units over reinforcement schemes applied to single-parameter Bernoulli units.

INTRODUCTION

A necessary property for any form of reinforcement learning algorithm is its exploratory behavior. Exploration is required in all areas where reinforcement learning is applicable, ranging from function optimization to learning control in an unknown environment. The ultimate goal of exploration is to select actions that improve the performance of the learning agent over time. A common way to achieve the desired exploratory behavior is based on randomness, i.e., actions are generated randomly according to a probability distribution. Many reinforcement learning algorithms are based on the above operation principle /2,4,7,14,17/. While such algorithms are known to have some weaknesses /10/, they are simple to implement and surprisingly effective in many cases. Furthermore, other exploration strategies available are either limited to finite domains, because they require a variable associated with each state-

action pair /4,9,11/, or they assume the existence of a learned model of the environment /8,12/ which might be difficult to obtain.

Since costs are usually assigned to each experience faced by the learning agent, in order for the exploration to be effective, these costs must be taken into account through modification of the probability distribution with which actions are selected. A useful property of a searching algorithm is its ability to continue exploration even after the action selection probabilities have been biased towards selecting the best action. This is necessary in order for the learning system to avoid local maxima and be able to adapt to time-varying, dynamic environments. The above property, which has been termed *sustained exploration* /1/, in fact determines two opposing principles that should be combined for efficient learning, namely: acting to gain new information and acting to avoid punishments /10/.

Many reinforcement learning algorithms apply to networks of stochastic units which draw their output from some probability distribution, employing either a single or multiple parameters /2,3,17/. Moreover, the parameters of the distribution can generally be expressed as a *deterministic* function of the variables representing the actual adaptable quantities of the network (e.g. weights). A significant part of the research in reinforcement learning has been on problems with discrete action spaces, in which the learning system chooses one of a finite number of possible actions. The above problems have been commonly tackled by using stochastic units which draw their output from a single parameter distribution, such as the Bernoulli distribution /3,17/.

It has been pointed out in /5,17/ that, for real-valued functions, the independent control of the mean and the variance might lead to interesting search properties. Clearly, there is no way of independently controlling the mean and the variance of the random output generated using a single parameter distribution. Motivated by the idea of applying the above principle to discrete-valued functions, we introduce in this paper a new type of processing unit which is characterized by the fact that the

probability distribution determining its output, depends *stochastically* on the adjustable network parameters. Assuming that the parameters that control exploration can be made available to the learner for adaptation, we present a general learning scheme that is consistent with the REINFORCE framework proposed in /17/. As is the case with any REINFORCE algorithm, the proposed scheme can be shown to statistically climb the gradient of the expected reinforcement in *immediate* reinforcement tasks, i.e., tasks where the payoff provided to the learner is determined by the most recent output (or input-out pair).

As an instantiation of the above general scheme we consider a two layered Normal/Bernoulli unit, in which the first layer is characterized by a normal distribution, while the second one is characterized by a Bernoulli distribution whose parameter is computed deterministically from the output of the first layer. Based on the above type of doubly stochastic unit we have developed several reinforcement learning schemes, including a REINFORCE algorithm as well as algorithms not belonging to the REINFORCE class (the determination of the standard deviation σ not being based on the derivatives suggested by REINFORCE). These schemes allow the exploration of discrete output spaces via the modification of the mean and the standard deviation of the normal distribution. The superiority of the proposed schemes over the REINFORCE algorithm with single-parameter Bernoulli units is illustrated by means of simulation experiments involving the optimization of complex functions defined on high-dimensional binary domains. Without loss of generality, the formulation of the algorithms assumes teams of units, such that each unit does not receive any input (external or from other units). Therefore, units are not interconnected and operate as independent *bit generators* rather than in the conventional network context.

In the developed reinforcement schemes, although a larger degree of randomness is introduced, the compromise between gaining information and gaining rewards was successful only to a limited extent. To this end, we further examine a number of variants with the purpose of achieving

active exploration and avoiding false maxima of the reinforcement/reward function.

The paper is organized as follows: Section 2 presents the general class of REINFORCE algorithms and introduces a novel type of unit, whose output is drawn from a distribution having a stochastic relationship with the adaptable parameters of the unit. Moreover, a general reinforcement learning scheme applying to networks of such units is presented and is shown to belong to the general class of REINFORCE algorithms. A characteristic example of the new type of units, the Normal/Bernoulli unit, is also described. Section 3 describes various reinforcement learning algorithms applying to the Normal/Bernoulli unit, and investigates their efficiency by considering various optimization problems. In addition, some variants of the designed algorithms are presented, and their effectiveness in achieving sustained exploration is examined. Finally, the main conclusions are summarized in Section 4.

2. REINFORCE ALGORITHMS APPLIED TO DOUBLY STOCHASTIC UNITS

REINFORCE learning algorithms have been analyzed by Williams /17/ for tasks that are characterized as *associative* reinforcement learning tasks (i.e., the learning system is required to form the correct input-output associations).

Any REINFORCE algorithm prescribes weight increments of the form

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})\frac{\partial \ln g_i}{\partial w_{ij}} \qquad (1)$$

where a_{ij} is a learning rate factor, r is the reinforcement signal delivered by the environment and b_{ij} is a reinforcement baseline. The quantity $\partial \ln g_i / \partial w_{ij}$ is the *characteristic eligibility* of the weight parameter w_{ij}, where $g_i\left(\xi; w^i, x^i\right)$ is the probability mass function (in case of a discrete distribution) or the probability density function (in case of a continuous

distribution) which determines the value ξ of the output y_i of the ith unit as a function of the parameter vector w^i of the unit (consisting of all the parameters w_{ij}) and its input pattern x^i. An important result proved in /13,17/ is that for any REINFORCE algorithm the average update direction in weight space lies in a direction for which the performance measure $E\{r|W\}$is increasing, where W is the matrix of the current network weights (*stochastic hillclimbing* property).

The type of unit commonly employed in REINFORCE algorithms is a *stochastic semi-linear* unit /17/. This type of unit is also encountered in reinforcement learning networks employing the *associative reward-penalty* (A_{R-P}) algorithm /2/. The operation of such type of unit consists of a deterministic computation, the result of which is then used as input to a stochastic computation. The deterministic computation involves a summation and a differentiable squashing function, while the output of the unit is drawn from some *single* parameter distribution. A characteristic example is the Bernoulli semilinear unit depicted in Figure 1, whose stochastic component consists of a Bernoulli random number generator with parameter p_i which is produced by the deterministic component as a function of the unit's adjustable parameters. Bernoulli semilinear units have been used in both associative and non-associative reinforcement learning tasks, with the output/action space being *discrete* /3,15,16/. These

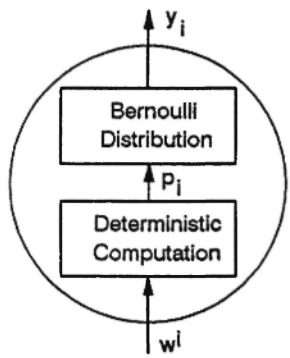

Fig. 1: Bernoulli semilinear unit

studies have demonstrated a strong tendency of the REINFORCE algorithms to converge at a local maximum of the reinforcement function. Moreover, even when the algorithms succeeded in finding the global maximum they were usually slow.

REINFORCE algorithms have also been employed in the case of units that determine their output stochastically from *multiparameter* distributions, such as the normal distribution /17/. In general, random units using multiparameter distributions have the potential to control the degree of exploratory behavior independently of where they choose to explore. For example, in the case of a Gaussian unit we can determine both the location being searched, via the mean μ_i, and the breadth of the search around that location, via the standard deviation σ_i. Because of the above useful property, multiparameter distributions (e.g. Gaussian) have been used as the stochastic component of units producing real-valued outputs, in order to learn functions with *continuous* values /5,14/.

In order to exploit the merits of a multiparameter distribution in the case of *discrete* domains, a new type of stochastic unit is introduced in the next section. In contrast with the stochastic units presented above (e.g. Bernoulli, Gaussian), such a type of unit is characterized by the fact that the parameters determining the output probability distribution, depend stochastically on the set of parameters that can be modified during exploration. The incorporation of larger stochasticity in the computation of the unit's output introduces a larger degree of variation. Thus, an improved exploratory behavior can be expected, in terms of elapsed time to find the optimal solution.

2.1 The General Case

Consider a stochastic unit whose output is determined by a distribution, such that there is no deterministic relationship between the probability of producing an output and the adaptable parameters of the unit

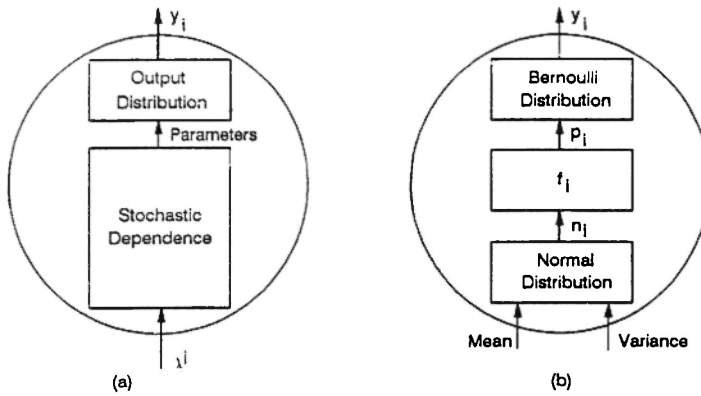

Fig. 2: Units with two stochastic levels: (a) General case, (b) Normal/Bernoulli case

(Figure 2(a)). Let y_i denote the output of such a unit, and also let λ^i represent the vector of all adaptable parameters affecting the input-output behavior of that unit. For the sake of simplicity, we have assumed that there is no input to the unit. It is obvious that for such a unit, the probability $g_i\left(\xi;\lambda^i\right)=\Pr\left\{y_i=\xi|\lambda^i\right\}$ of producing a certain output ξ is a random variable (the notation used is appropriate for the case where the set of possible output values y_i is discrete, however, the results to be presented also apply to continuous-valued units). We assume that the stochastic dependence of g_i on the parameters λ^i is expressed in terms of a random experiment, depending on the vector λ^i, which produces an outcome $o_i = \omega$ from a set of possible outcomes Ω_i. Each possible outcome may be regarded as equivalent to assigning some specific values to the parameters controlling the distribution from which the output y_i is drawn.

We assume that the sample space Ω_i is discrete. Moreover, $p\left(o_i=\omega|\lambda^i\right)$ indicates the probability of the event $\{\omega\}$, provided that the vector of the adjustable parameters of unit i is λ^i. Then, the expected value of the probability $g_i(\xi;\lambda^i)$ of providing an output $y_i = \xi$ given that the parameter vector is λ^i can be written:

$$E[g_i(\xi;\lambda^i)] = \sum_{\omega\in\Omega_i} p(y_i = \xi|\omega)p(o_i = \omega|\lambda^i) \qquad (2)$$

where the probability $p(yi = \xi|\omega)$ depends on ω in a deterministic way.

The expected value $E\{g_i(\xi;\lambda^i)\}$ appearing above actually represents the overall probability that $y_i = \xi$ given the adaptable parameter vector λ^i. This quantity can be used to determine the characteristic eligibility required by the REINFORCE theorem /17/ in order to develop a rule for updating each parameter λ_{ik} (the kth component of λ^i). We show in Appendix 1 that the use of the expected value $E\{g_i(\xi;\lambda^i)\}$ is compatible with the general REINFORCE framework and yields an update rule exhibiting the stochastic hillclimbing property.

Consider now a team of stochastic units such as the one described above. As already mentioned, we consider no input to the units. However, this is not actually a restriction since in the case of a network of interconnected units there exists a deterministic functional dependence of the parameters λ^i on the actual parameters (weights) and inputs of the network. Consequently, the increments of the weights can be directly derived through application of the chain rule from the increments of λ^i which are prescribed by the following proposition.

Proposition 1. *If each parameter* λ_{ik} *is adjusted according to equation*

$$\Delta\lambda_{ik} = \alpha_{ik}(r - b_{ik})\frac{\partial \ln E\{g_i|\lambda^i\}}{\partial\lambda_{ik}} \tag{3}$$

then the following holds

$$E\{\Delta\lambda_{ik}|\Lambda\} = \alpha_{ik}\frac{\partial E\{r|\Lambda\}}{\partial\lambda_{ik}} \tag{4}$$

where Λ *denotes the set of parameters* $\{\lambda_{ik}\}$ *for all* i, k.

In Appendix 1 we provide a brief derivation of the above result proceeding in a similar manner to the original proof of Williams /17/.

In essence, the REINFORCE rule given above, prescribes the adjustment of the adaptable parameters of a *doubly stochastic* unit, i.e., a

stochastic unit that computes its output using a two-stage random generation process.

2.2 The Normal/Bernoulli Case

In /6/, a characteristic example of the general type of doubly stochastic processing elements introduced above has been considered. Such a unit is depicted in Figure 2(b). At the first stochastic level the output n_i is drawn from a normal distribution with parameters μ_i and σ_i, while at the second level the output y_i is a Bernoulli random variable with parameter p_i, the latter being a deterministic function f_i of the output n_i of the first level. We shall consider that f_i is the logistic function, i.e., $p_i = 1/(1 + \exp(-\kappa n_i))$, where the parameter κ determines its slope (we have used $\kappa = 0.5$ in all our experiments involving Normal/Bernoulli units).

It is obvious that the probability $g_i(\xi;\mu_i,\sigma_i) = \Pr\{y_i=\xi|\mu_i,\sigma_i\}$ is a random variable since it is a deterministic function of the random variable n_i. Considering a team of binary stochastic units such as the one shown in Figure 2(b), the REINFORCE algorithm prescribing the update of the parameters μ_i and σ_i of each unit i, can be easily derived following the discussion of the previous section:

$$\Delta\mu_i = \alpha_{i\mu}(r - b_{i\mu})\frac{\partial \ln E\{g_i|\mu_i,\sigma_i\}}{\partial\mu_i} \qquad (5)$$

$$\Delta\sigma_i = \alpha_{i\sigma}(r - b_{i\sigma})\frac{\partial \ln E\{g_i|\mu_i,\sigma_i\}}{\partial\sigma_i} \qquad (6)$$

In practice, application of the above learning rules requires the computation of estimates of the partial derivatives appearing in equations (5) and (6). However, when the output of the random number generator at the first stochastic level can be written as a differentiable function of its parameters, it is possible to obtain such estimates, since the chain rule can be applied through the random generator, as if we were backpropagating through deterministic components /14,17/. The normal random number generator which constitutes the first level of the unit shown in Figure 2(b),

satisfies the aforementioned property and the technique for backpropagating through random components can be applied.

3. REINFORCEMENT SCHEMES EMPLOYING NORMAL/ BERNOULLI UNITS

We have developed several reinforcement learning algorithms for the new type of Normal/Bernoulli unit, and we have tested their efficiency in a number of optimization problems. The problems actually involved the search of high-dimensional binary spaces in order to find the point where a given function takes its maximum value.

3.1 A REINFORCE Scheme

A REINFORCE scheme is obtained if the parameters μ_i and σ_i are adjusted according to equations (5) and (6). Taking into account the way in which we can backpropagate through a normal random number generator /14,17/, we considered the quantity $\partial \ln g_i / \partial n_i$ as an estimate of $\partial \ln E\{g_i|\mu_i,\sigma_i\} / \partial \mu_i$ and the quantity $\partial \ln g_i / \partial n_i (n_i - \mu_i)/\sigma_i$ as an estimate of $\partial \ln E\{g_i|\mu_i,\sigma_i\} / \partial \sigma_i$ (n_i being the output of the first level). Since g_i is a deterministic function of n_i (through its dependence on p_i), using the chain rule we compute:

$$\frac{\partial \ln g_i}{\partial n_i} = \frac{\partial \ln g_i}{\partial p_i} \frac{\partial p_i}{\partial n_i} \tag{7}$$

where from the Bernoulli distribution and the logistic function we have, respectively,

$$\frac{\partial \ln g_i}{\partial p_i} = \frac{(y_i - p_i)}{p_i(1 - p_i)} \tag{8}$$

$$\frac{\partial p_i}{\partial n_i} = \kappa p_i(1 - p_i) \tag{9}$$

Moreover, we used as reinforcement baseline the adaptive quantity $\bar{r}$, which is computed as an exponentially weighted average of prior reinforcement values

$$\bar{r}(t) = \gamma r(t-1) + (1-\gamma)r(t) \tag{10}$$

where γ is a decay rate positive and less than 1, which in all our experiments was set equal to 0.9. (This technique is referred to in the literature as *reinforcement comparison*.)

Thus, the update of the parameters μ_i and σ_i of the ith unit is performed using the following learning rules (identical learning rates α_μ and α_σ were used for all units)

$$\Delta\mu_i = \alpha_\mu(r-\bar{r})(y_i - p_i) \tag{11}$$

$$\Delta\sigma_i = \alpha_\sigma(r-\bar{r})(y_i - p_i)\frac{(n_i-\mu_i)}{\sigma_i} \tag{12}$$

We will refer to the above reinforcement scheme as N/B.

In the following we will discuss some issues concerning the adaptation of the mean and the standard deviation when reinforcement comparison and the above mentioned estimators are used. For the ease of presentation we will omit the subscripts indexing each unit. It is clear that the sign of the quantity $\partial \ln g/\partial n$ (which is the sign of y - p in the case of Normal/Bernoulli units) determines the sign of the derivative of g at the sampled point n. Considering the adaptation of the mean, if the sampled point y gives rise to a higher function value than expected, then μ moves in the direction indicated by the gradient of the probability function g at the point n. Thus, if the unit outputs 1 then μ will increase, while μ will decrease in case the unit outputs 0. There is a similar behavior, i.e., μ simply moves in the opposite direction, if the sampled point y leads to a lower function value. In fact, the change made to μ corresponds to that required to make the re-occurrence of y more likely if a better point is found, and make it less likely if a worse point is discovered.

As far as σ is concerned, the update is performed analogously, i.e., the probability of sampling a better point is increased and the probability of sampling an inferior one is decreased. To understand the behavior of the algorithm in terms of search, assume that the sampled point y is better than expected. In this case the search is narrowed around μ if, based on the sign of the derivative of the probability g at n, it is implied that the probability of producing y (evaluated at n) is lower than the corresponding probability evaluated at μ. Moreover, the search around μ is broadened if the opposite holds. As a consequence, for a certain output y, if μ is at the top of a hill (local maximum of the probability function g), then σ will be decreased. The decrease will be made to the point where it is not likely to sample worse points (worse in the sense that they have a lower probability of producing the specific output y). Accordingly, for a certain output y, if μ is at the bottom of a valley (local minimum of the probability function g), then σ will be increased to obtain points with higher probability of producing the specific output y.

3.2 Alternative Ways for Adapting σ

In the REINFORCE scheme presented above, σ is updated locally at every unit based on its impact on the probability of producing a certain output y. In order to further understand the utility of σ in the exploration of the output space we also tested two other reinforcement schemes, in which μ is adapted according to equation /11/, but the rules for updating σ (which is considered the same for all units) do not follow equation /12/. Therefore, the algorithms presented below do not comply with the REINFORCE framework.

Ideally, a unit that explores an output/action space should have the following properties /5/:

- It should be able to improve its performance in cases where it is doing poorly be exhibiting a greater amount of exploratory behavior.

- It must not degrade its performance in cases it is doing well by exhibiting in those cases a very random behavior.

Following the above guidelines, a reinforcement algorithm can be derived by requiring that, at each time step, σ be proportional to the expected heuristic reinforcement $\bar{h}(t)$:

$$\sigma(t) = \alpha_\sigma \bar{h}(t) \tag{13}$$

where α_σ is a positive constant and $\bar{h}(t)$ is a trace of the past absolute values $|r-\bar{r}|$

$$\bar{h}(t) = \gamma \bar{h}(t-1) + (1-\gamma)|r(t) - r(t-1)| \tag{14}$$

The parameter γ is the same decay factor used in the computation of $\bar{r}$. In fact, $\bar{h}$ represents information regarding the state of the learning system. If the system is performing well and explores around a local maximum, then $\bar{h}$ (and hence σ) tends to become zero. On the contrary, when the system is far from the neighborhood of a local maximum, then $\bar{h}$ takes on larger values, indicating that a larger amount of exploratory behavior is needed. It should be noted that $|r-\bar{r}|$ could have been used instead of $\bar{h}$ as an indicator of how well the reinforcement learning system is performing. However, the latter quantity changes more 'smoothly' over time than the former, hence large modifications to the values of σ are avoided. In the following discussion, the reinforcement scheme just described will be distinguished by the name N/B_1.

The two properties stated above indicate that standard deviation should be small when the current expected output is close to the optimal. Since the quality of the expected output is quantified by the value of the expected reinforcement, we can deduce that if the expected reinforcement is high, then the learning system is performing well and thus σ should be small. Conversely, if the expected reinforcement is low, σ should be large so that a wider area is explored. The above rationale has been used in /5/ as

part of a stochastic reinforcement learning algorithm for learning functions with continuous outputs. However, in that case the maximum attainable reinforcement was known *a priori*. Since this is not the case in our optimization problems, the following alternative is adopted. We keep track of the maximum value of reinforcement attained so far, and use this as a standard of performance against which the performance of the system is compared. Thus, a reinforcement scheme is obtained in which σ is updated based on the following equation

$$\sigma(t) = \alpha_\sigma(r_{\max}(t) - \bar{r}(t)) \qquad (15)$$

where α_σ is a positive constant, r_{max}(t) is the maximum attained reinforcement until time t and $\bar{r}$ (t) is the expected reinforcement at that time instant. In what follows, we will refer to the above scheme as N/B_2.

The reinforcement schemes N/B_1 and N/B_2 actually decrease σ when search has reached the area of a local (or global) maximum, thus allowing convergence to such a maximum. Since it is true that the reward function is generally convex in the area of a local maximum, it is implied that σ is decreased when the reinforcement function is convex. On the contrary, when search is far from the neighborhood of a local maximum (e.g. in an area where the reward function is concave) then σ is increased.

3.3. Comparative Results

We have tested the efficiency of the reinforcement schemes presented in the last sections, by considering a number of optimization problems originally studied by Ackley /1/ and also examined in /15,16/. The problems we have tested were the *One-Max*, the *Two-max*, the *Porcupine*, the *Plateaus* and a specific combinatorial optimization problem, the *minimum-cut graph partitioning* problem. A brief description of the problems can be found in Appendix 2.

In all our experiments we have used a team of Normal/Bernoulli units

(Figure 2(b)) with no interconnections among them. The size of the team is determined by the dimensionality N of the binary space to be searched. The value of N is the only information about the function that is assumed to be available before the computation begins. At each trial the network generates a sample point y which is a bit vector of length N. The function value at that point is used as the reinforcement signal r which is delivered to every unit of the network. The latter should adapt its behavior so that during the next trial a point of higher function value is generated.

Williams and Peng /15,16/, performed experiments on the same optimization problems using a team of Bernoulli logistic units, i.e., units of the type shown in Figure 1 having the logistic as the squashing function. As was the case with their experiments, we found that our algorithms generally converge faster if we replaced p_i, with $\overline{y}_i$ in the eligibility factor of equation (11). Similarly to $\overline{r}$, the quantity $\overline{y}_i$ is an average of past values of y_i and is updated by

$$\overline{y_i}(t) = \gamma \overline{y_i}(t-1) + (1-\gamma) y_i(t) \tag{16}$$

where the decay rate γ has the same value as that used for updating $\overline{r}$. All the reported results have been obtained using this form of eligibility, i.e., the parameter μ_i of each unit i was updated according to

$$\Delta \mu_i = \alpha_\mu (r - \overline{r})(y_i - \overline{y_i}) \tag{17}$$

It should be noted that the above rule for updating the mean is not a member of the class of REINFORCE algorithms in the strict sense. However, the proposed schemes are compared with a single-parameter REINFORCE variant adopting the same form of eligibility /15,16/. Moreover, as experiments indicate, the conclusions drawn carry over to the case where pure members are considered.

Another issue concerning the REINFORCE scheme N/B is that, in order to guarantee that the σ_i's will not become negative, we actually consider $\theta_i = \ln \sigma_i^2$ as the adaptable parameter in place of σ_i.

Table 1 (first row) presents the average number of steps (over 50 runs) required by the scheme N/B to find the global maximum of each function. At the start of each run, all the parameters μ_i were set equal to zero, while the standard deviations σ_i were all initialized to the same value σ_{init}. Generally, small values of σ_{init} in the range (2.0,3.0) were most appropriate. A larger value of σ_{init} was needed as the value of α_μ was decreased, in order to retain the same learning speed. Table 1 also shows the values of the learning rates α_μ and α_σ used for all the runs. It should be pointed out that the choice of the value of α_μ was critical for the convergence of the algorithm. On the other hand, the method was relatively robust with respect to α_σ, although a gradual degradation in learning time and quality of the result was observed, as α_σ increased.

The performance of the reinforcement schemes N/B_1 and N/B_2 is also shown in Table 1 (second and third row, respectively). Finally, for comparison purposes, Table 1 (fourth row) also gives the results reported by Williams and Peng /15,16/ on the same optimization problems. The results concern the average time to find the maximum using a REINFORCE variant (denoted by B) obtained from equation (1) for the case of a team of Bernoulli logistic units by replacing p_i with $\bar{y}_i$ in the eligibility factor:

$$\Delta w_i = \alpha(r - \bar{r})(y_i - \bar{y_i}) \tag{18}$$

As indicated by our experiments, all the examined reinforcement schemes have a strong tendency to converge. It should be pointed out that the results reported concern the number of steps until the first generation of the global maximum, and not the number of steps until convergence (which generally occurs a few steps after the first generation of the optimum). Furthermore, all algorithms failed to find the global maximum in more than half of the runs for the task of graph partitioning. The graph instances considered were the two hierarchically structured graphs, MLC-32 and MLC-64 presented in /1/. More specifically, the success or failure of the algorithms in these instances depends strongly on the initial direction of

the search, in the sense that either the global maximum is rapidly attained or the system is trapped for ever in some other local maximum. Therefore, the mean time to find the maximum for the cases of success is relatively short, but these results cannot be considered significant.

Observation of Table 1 indicates superiority of the proposed two-stochastic level (Normal/Bernoulli) reinforcement schemes over the single level (Bernoulli) scheme. This is a consequence of the advantages offered by the use of two individually controllable parameters during the exploration process (mean and standard deviation), rather than a single one. This results in a more effective search of the output space, thus increasing the speed of reaching a solution. In fact, the performance of the algorithms was similar when searching for the maximum of One-Max or Porcupine. The reason is that when the reward function is linear (One-Max) then changing σ does not necessarily improve the expected reward on the next step. Moreover, Porcupine is essentially the same as One-Max, since reinforcement learning algorithms are able to handle noisy reinforcement signals and treat the porcupine 'quills' as if they were merely noise added to some underlying function. The superiority of the

TABLE 1
Comparative results

	One Max			*Two Max*		
Algorithm	α_μ	α_σ	*Steps*	α_μ	α_σ	*Steps*
N/B	0.05	0.02	7	0.03	0.02	10
N/B_1	0.05	0.06	6	0.03	0.06	8
N/B_2	0.05	0.06	7	0.05	0.06	9
B			9			102

	Porcupine			*Plateaus*		
Algorithm	α_μ	α_σ	*Steps*	α_μ	α_σ	*Steps*
N/B	0.05	0.02	7	0.010	0.02	290
N/B_1	0.05	0.06	7	0.010	0.06	279
N/B_2	0.05	0.06	6	0.010	0.06	295
B			9			435

Normal/Bernoulli schemes was apparent in the Two-Max and the Plateaus functions which actually represent more difficult tasks.

As already noted, in the graph partitioning problem, which is the most difficult among those examined, all reinforcement schemes sometimes failed since they converged to a local maximum. In /15/ a REINFORCE variant that incorporated a decay term $-\delta w_{ij}$ ($\delta < 1$) in the weight update rule (18), was proposed in order to achieve sustained exploration. The use of the decay term led to a tendency for random restarts coupled with the tendency for stochastic gradient following. In what follows, we consider some techniques that enhance the sustained exploration capabilities of our reinforcement schemes, based on alternative forms for the update rules.

3.4 Achieving Sustained Exploration

The reinforcement schemes presented in the previous sections represent efficient search strategies that are based on the ability to focus the search into promising regions of the output space. The latter is achieved by using the function values received in the past, as a guidance for the future directions of the search. As experiments indicate, the quick discovery of promising regions is more successfully achieved when using two distinct parameters during the search, instead of a single one. However, in order to deal more effectively with the problem of local maxima existing in various function landscapes, as well as the problem of time-varying functions, an exploration algorithm must also have the ability to broaden search at certain times /1/. This property of sustained exploration actually emphasizes divergence, i.e., return to global searching, but without completely forgetting what has been learned. In the following, our purpose is to devise some techniques in which the divergence mechanism is not *external* to the learning system (e.g., a temperature T that increases after the detection of convergence), but rather an *internal* event triggered when the system tends to settle on a certain state.

In order to control the convergence of the Normal/Bernoulli reinforcement schemes presented previously one has to consider ways of intervening in the update rules of the mean and the standard deviation. Considering the way in which the mean is updated in all schemes (equation (17)), we have developed a variant involving the incorporation of a decay term in the increment. In fact, this variant constitutes an adaptation of the idea presented in /15/. In our scheme, a decay term was included in the rule determining the adjustment of μ_i, which was then performed according to the following equation:

$$\Delta\mu_i = \alpha_\mu(r - \bar{r})(y_i - \overline{y_i}) - \delta\mu_i \qquad (19)$$

where δ is a decay rate. In fact, divergence relies on an adaptation mechanism that allows the output probabilities p_i to move away from the values to which they have converged, i.e., move towards the value 0.5. Therefore, in order for the decay term to be effective, it must suggest a displacement of μ_i towards the value ζ for which the sigmoid yields the output 0.5. In our case $\zeta = 0$, but in general the decay term should have the form $-\delta(\mu_i - \zeta)$.

As far as the standard deviation σ is concerned, which is the other parameter controlling the output of each unit, a number of observations can be made. Although standard deviation actually determines the amount of exploratory behavior exhibited by one unit, it is difficult to devise a technique for sustained exploration based on σ. One of our attempts, for example, consisted in trying to keep σ constant over time and equal to a relatively large value. However, that approach sometimes failed to converge. Moreover, its effectiveness reduced over time due to the fact that the mean was eventually becoming very large in magnitude. Another alternative that we have examined involved the incorporation of an 'increase' term $+\delta\sigma_i$ in equation (12). The purpose of that term was to trigger an increase of σ whenever the value $r - \bar{r}$ tended to become zero, i.e., the learning system tended to converge to a certain output. However, the method failed because in many cases it led the system to undesired

situations (e.g., huge values of σ). The justification for the last observation lies in the fact that it is always equiprobable to select a value either on the right or on the left of the mean of a normal distribution. Thus, the increase of σ cannot guarantee a suitable 're-organization' of the learning system, and may lead to unexpected states.

The Normal/Bernoulli reinforcement schemes N/B, N/B_1 and N/B_2 were modified giving rise to three other schemes exhibiting the property of sustained exploration. The resulting algorithms update the parameter μ according to equation (19), while the update of σ is performed in the same way as in the original schemes. In what follows the three variants will be referred to as SN/B, SN/B_1 and SN/B_2, respectively.

Experiments were conducted using the three schemes SN/B, SN/B_1 and SN/B_2 described above. We have also performed experiments using the REINFORCE variant (denoted by SB) examined in /15,16/, which employs Bernoulli logistic units and incorporates a decay term in the update rule of equation (18). As in the previous series of experiments, we adopt the strategy of running each algorithm until the optimum is obtained for the first time. The number of function evaluations until the first generation of the global optimum, is used as a performance measure assessing the various schemes. The problem considered was that of graph partitioning in which the respective algorithms N/B, N/B_1, N/B_2 and B typically failed. In particular, we restricted attention to the MLC-32 and MLC-64 graphs /1/. For those specific instances of the graph partitioning problem there are two global maxima as well as a number of local maxima.

In Table 2 the average number of steps (over 50 runs) that the algorithms took to find the global maximum are shown, as well as the values of the learning rates α_μ and α_σ and the decay δ used in all runs. The results suggest that all reinforcement schemes exploiting the two parameters of the normal distribution were remarkably faster than the REINFORCE variant controlling search via the single parameter of the Bernoulli distribution. Moreover, experiments using different learning rates

TABLE 2
Comparative results on graph partitioning: average steps

	MLC-32				*MLC-64*			
Algorithm	α_μ	α_σ	δ	*Steps*	α_μ	α_σ	δ	*Steps*
SN/B	0.44	0.02	0.022	2140	0.24	0.003	0.01	7140
SN/B_1	0.6	0.2	0.03	2525	0.3	0.02	0.01	7930
SN/B_2	1.0	0.15	0.03	2190	0.3	0.03	0.01	7810
SB	$\alpha = 0.04$		0.01	4925	$\alpha = 0.2$		0.01	11650

have shown that the SN/B_1 scheme is more robust than the other schemes with respect to changes in the parameter values.

Table 3 shows for each learning algorithm, the standard deviation of the time to find a global maximum over the 50 runs, as well as the longest time for any run until a global maximum was discovered. As can be observed, all the examined schemes are characterized by relatively large standard deviation. This is a consequence of the probabilistic nature of reinforcement algorithms. From the results displayed in the table it is clear that, in addition to improving the average number of required steps, the employment of two parameters leads also to better behavior concerning the worst case and the standard deviation.

Figure 3 shows a plot of the value of the reinforcement received as a function of trial number. The plot concerns a typical run of the algorithm SN/B on the MLC-32 problem (similar plots can be obtained for algorithms SN/B_1 and SN/B_2). It should be noted that the reinforcement received at a

TABLE 3
Comparative results on graph partitioning: deviation and worst case

	MLC-32		*MLC-64*	
Algorithm	*Deviation*	*Worst*	*Deviation*	*Worst*
SN/B	1870	8520	4590	22160
SN/B_1	1847	7880	5130	21490
SN/B_2	1690	5714	5970	23500
SB	3780	14690	9078	30550

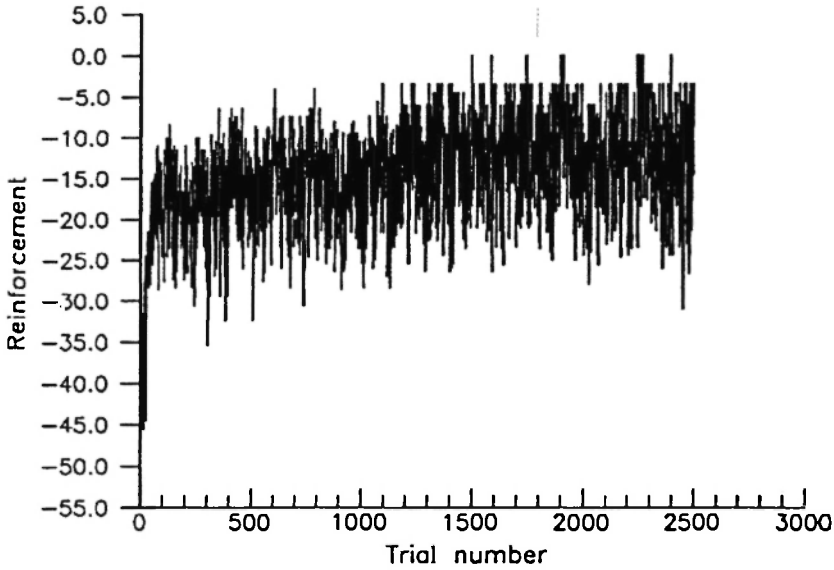

Fig. 3: A typical run for SN/B

global maximum is 0 for the problem considered. For illustrative purposes the run was not terminated after a global maximum was discovered. The plot shows that the global maximum was first found at trial 1490 and since then it was generated again several times (sustained exploration). In order to better illustrate the ability of the algorithm to generate points with higher payoff (on the average) as the number of trials increases, the reinforcement received per trial was averaged over bins of 100 trials. The obtained smoothed curve is plotted in Figure 4.

4. CONCLUSIONS

We have developed reinforcement algorithms applying to networks of stochastic units having the characteristic that the output distribution of each unit depends stochastically on the adjustable network parameters. A REINFORCE algorithm is presented for the general type of the above doubly stochastic units, which updates the adaptable parameters of the

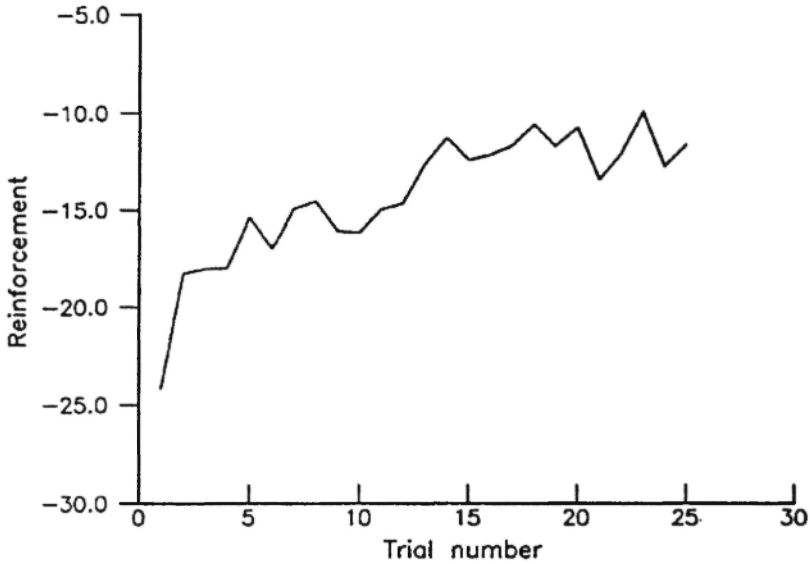

Fig. 4: Averaged reinforcement over bins of 100 trials

network in a way that the stochastic hillclimbing property is satisfied (with respect to the expected value of the upcoming reinforcement signal).

As a special case, we have derived a REINFORCE algorithm for Normal/Bernoulli units, i.e., doubly stochastic units employing a normal and a Bernoulli distribution. Moreover, by considering alternative ways for adapting the standard deviation, several other reinforcement schemes applied to Normal/Bernoulli units have been developed. All these algorithms possess the ability of exploring discrete domains by exploiting the two parameters of the normal distribution employed at the first level.

Experimental results concerning a number of discrete optimization problems, indicate that the proposed reinforcement schemes exhibit better exploration capabilities and are more powerful than a single-parameter scheme in terms of elapsed time to locate the global maximum.

Finally, emphasis has been given in developing techniques that

achieve a good compromise between the two complementary goals of efficient searching, i.e., acting to acquire new information and acting to obtain rewards based on past information. To this end, the algorithms have been adapted so that the above goals can be dealt with in a unified way. This makes possible the exploration of complex and dynamic environments. Experiments on instances of the graph partitioning problem have revealed that the proposed algorithms, which incorporate an internal mechanism for switching between local and global searching, constitute a very powerful scheme for exploring unknown domains.

APPENDIX 1

In this Appendix we present the proof of Proposition 1.

Let Y_i denote the set of possible output values ξ of unit i, which is considered to be discrete, although this is not a critical assumption.

If every parameter λ_{ik} is adjusted according to equation (3), we have that

$$
\begin{aligned}
E[\Delta\lambda_{ik}|\Lambda] &= \sum_{\omega\in\Omega_i} E[\Delta\lambda_{ik}|\omega,\Lambda]p(o_i=\omega|\lambda^i) \\
&= \sum_{\omega\in\Omega_i}\sum_{\xi\in Y_i} E\left[\frac{\alpha_{ik}(r-b_{ik})}{E[g_i(\xi;\lambda^i)]}\frac{\partial E[g_i(\xi;\lambda^i)]}{\partial\lambda_{ik}}|y_i=\xi,\Lambda\right] p(y_i=\xi|\omega)p(o_i=\omega|\lambda^i) \\
&= \alpha_{ik}\sum_{\xi\in Y_i} E\left[(r-b_{ik})\frac{\partial E[g_i(\xi;\lambda^i)]}{\partial\lambda_{ik}}|y_i=\xi,\Lambda\right] \qquad (20)
\end{aligned}
$$

This leads to the following equation

$$
\begin{aligned}
E[\Delta\lambda_{ik}|\Lambda] &= \alpha_{ik}\sum_{\xi\in Y_i} E[r|y_i=\xi,\Lambda]\frac{\partial E[g_i(\xi;\lambda^i)]}{\partial\lambda_{ik}} \\
&- \alpha_{ik}\sum_{\xi\in Y_i} E[b_{ik}|y_i=\xi,\Lambda]\frac{\partial E[g_i(\xi;\lambda^i)]}{\partial\lambda_{ik}} \qquad (21)
\end{aligned}
$$

Assuming that the reinforcement baseline b_{ik} is conditionally independent on y_i when Λ is given, the second term in the right-hand side of the above

equation can be written:

$$\alpha_{ik} E[b_{ik}|\Lambda] \sum_{\xi \in Y_i} \frac{\partial E[g_i(\xi;\lambda^i)]}{\partial \lambda_{ik}} \tag{22}$$
$$= \alpha_{ik} E[b_{ik}|\Lambda] \frac{\partial}{\partial \lambda_{ik}} \sum_{\omega \in \Omega_i} \sum_{\xi \in Y_i} p(y_i = \xi|\omega) p(o_i = \omega|\lambda^i)$$
$$= 0$$

Moreover, the expected value of r can be written as follows:

$$E[r|\Lambda] = \sum_{\omega \in \Omega_i} E[r|\omega, \Lambda] p(o_i = \omega|\lambda^i)$$
$$= \sum_{\omega \in \Omega_i} \sum_{\xi \in Y_i} E[r|y_i = \xi, \Lambda] p(y_i = \xi|\omega) p(o_i = \omega|\lambda^i)$$
$$= \sum_{\xi \in Y_i} E[r|y_i = \xi, \Lambda] E[g_i(\xi;\lambda^i)] \tag{23}$$

When the value of y_i is specified, the expected value of r is independent of λ_{ik}. Therefore

$$\frac{\partial E[r|\Lambda]}{\partial \lambda_{ik}} = \sum_{\xi \in Y_i} E[r|y_i = \xi, \Lambda] \frac{\partial}{\partial \lambda_{ik}} E[g_i(\xi;\lambda^i)] \tag{24}$$

From equations (22)-(24) we finally find that when the elements of vector λ^i are adjusted according to equation (3), it holds that

$$E[\Delta\lambda_{ik}|\Lambda] = \alpha_{ik} \frac{\partial E[r|\Lambda]}{\partial \lambda_{ik}} \tag{25}$$

i.e., the stochastic hillclimbing property is satisfied.

APPENDIX 2

This Appendix presents the suite of functions used to test the performance of the proposed reinforcement algorithms. Let N denote the dimensionality of the binary space to be searched. For the first four problems $N = 20$, while in the fifth problem we have considered $N = 32$ and

$N = 64$. For any output vector y produced by the network, let $N1$ represent the number of 1 bits in the vector, and let $N0$ represent the number of 0 bits. The following problems have been considered in our simulations. A more detailed description of the functions to be optimized can be found in /1/.

1. *One-Max*. This is a linear function defined as $f(y) = 10N1$. The global maximum is located at the point designated by all 1's in the vector y.

2. *Two-Max*. The function is defined as $f(y) = |18N1 - 8N|$, and has one global maximum and one local maximum. The global maximum (with value $10N$) is the bit vector containing all 1's, while the local maximum (with value $8N$) is the bit vector having all 0's.

3. *Porcupine*. The function to be maximized is $f(y) = 10N1 - 15(N1 \bmod 2)$. It is just like the One-Max function except that we subtract 15 when $N1$ is odd. The global maximum is again at the point y containing all 1's, but every point whose Hamming distance from the global maximum is even, constitutes a local maximum.

4. *Plateaus*. The function is defined as follows: Divide the bits into four equal-sized groups. For each group, if all bits are 1 compute a score which is $2.5N$, otherwise the score is 0. Return as a value of the function the sum of the scores for the four groups. This function has only five possible values and its main characteristic is the existence of large regions (plateaus) in which all points have the same value. Like the first three functions, the global maximum of the Plateaus is at the point y having all 1's.
5. The last optimization problem considered is the *minimum cut graph partitioning* problem. This is a member of the class of NP-complete problems and concerns the separation of the nodes of a graph (with even number of nodes N) into two groups having the same number of nodes, such that the number of edges connecting nodes of different

groups is as small as possible. A partition of a graph can be represented by a vector of N bits by assigning 0 to all the nodes in one group and 1 to all the nodes in the other group. Thus, following Ackley /1/, graph partitioning can be regarded as the maximization of the function $f(y) = -c(y) - 0.1(N1 - N0)^2$, where $c(y)$ is the number of edges crossing the partition for the particular vector y.

REFERENCES

1. Ackley, D.H., *A Connectionist Machine for Genetic Hillclimbing*, Norwell, MA: Kluwer, 1987.
2. Barto, A.G. and Anandan, P., Pattern Recognizing Stochastic Learning Automata, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, 360-375, 1985.
3. Barto, A.G. and Jordan, M.I., Gradient following without back-propagation in layered networks, in: M. Caudill and C. Butler, eds, *Proceedings of the IEEE First Annual Conference on Neural Networks*, vol. II, 629-636, San Diego, 1987.
4. Barto, A.G., Bradtke, S.J. and Singh, S.P., Real-time learning and control using asynchronous dynamic programming, *Technical Report* 91-57, University of Massachusetts, Amherst, MA, 1991.
5. Gullapalli V., A stochastic reinforcement learning algorithm for learning real-valued functions, *Neural Networks*, vol. 3, 671-692, 1990.
6. Kontoravdis, D., Likas, A. and Stafylopatis, A., A reinforcement learning algorithm for networks of units with two stochastic levels, *Proceedings ICANN-92*, vol. I, 143-146, Brighton, UK, 1992.
7. Lin, L., Self-improving reactive agents based on reinforcement learning, planning and teaching, *Machine Learning*, vol. 8, 293-321, 1992.
8. Munro, P., A dual backpropagation scheme for scalar-reward learning, *Ninth Annual Conference of the Cognitive Science Society*, 165-176, Hillsdale, NJ, 1987.

9. Sutton, R.S., Integrated modeling and control based on reinforcement learning and dynamic programming, *Advances in Neural Information Processing Systems 3*, R.P. Lippmann, J.E. Moody, and D.S. Touretzky, eds, 471-478, San Mateo, CA: Morgan Kaufmann, 1991.
10. Thrun, S.B., The role of exploration in learning control, in: *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, D.A. White and D.A. Sofge, eds, Van Nostrand Reinhold, 1992.
11. Thrun, S.B. and Moller K., Active exploration in dynamic environments, *Advances in Neural Information Processing Systems 4*, J.E. Goody, S.J. Hanson and R.P. Lippmann, eds, San Mateo, CA: Morgan Kaufmann, 1992.
12. Werbos, P.J., A menu of designs for reinforcement learning over time, in: *Neural Networks for Control*, W.T. Miller, R.S. Sutton and P.J. Werbos, eds, MIT Press, 1990.
13. Williams, R.J., Toward a Theory of reinforcement learning connectionist systems, *Technical Report NU-CCS-88-3*, Boston, MA, 1988.
14. Williams, R.J., On the use of backpropagation in associative reinforcement learning, *Proceedings of the Second Annual International Conference on Neural Networks*, vol. I, 263-270, San Diego, CA, 1988.
15. Williams, R.J. and Peng, J., Reinforcement learning algorithms as function optimizers, *Proceedings of the International Joint Conference on Neural Networks*, vol. II, 89-95, Washington, D.C., 1989.
16. Williams, R.J., Peng, J., Function optimization using connectionist reinforcement learning networks, *Connection Science*, vol. 3, 241-268, 1991.
17. Williams, R.J., Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine Learning*, vol. 8, 229-256, 1992.