Parallel Recombinative Reinforcement Learning (Extended Abstract)

Aristidis Likas, Konstantinos Blekas and Andreas Stafylopatis

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

1 Introduction

This paper presents a population-based technique that is suitable for function optimization in high-dimensional binary domains. The method allows an efficient parallel implementation and is based on the combination of genetic algorithms and reinforcement learning schemes. More specifically, a population of probability vectors is considered, each member corresponding to a reinforcement learning optimizer. Each probability vector represents the adaptable parameters of a team of stochastic units whose binary outputs provide a point of the function state space. At each step of the proposed technique the population members are updated according to a reinforcement learning rule and then recombined in a manner analogous to traditional genetic algorithm operation. Special care is devoted to ensuring the desirable properties of sustained exploration capability and sustained population diversity. We shall denote the proposed population-based approach as *Parallel Recombinative Reinforcement Learning (PRRL).*

2 Genetic Algorithms and Reinforcement Learning

Genetic algorithms in their traditional formulation assume a population of binary strings and at each generation step new members are created by applying genetic operators to appropriately selected strings. Following the principle of 'survival of the fittest', strings are selected for reproduction with probability proportional to their corresponding fitness (function value). The selected strings (usually two) can be combined using the single-point crossover or the uniform crossover operator. In addition, the basic genetic algorithm employs a mutation operator which introduces randomness in the search process by randomly flipping some bit values in the population strings. The main problem with simple genetic algorithms is that they exhibit a fast convergence behavior, mainly due to the effects of the selection scheme and the crossover operator. Population diversity can be introduced only through mutation but its effectiveness is rather limited [?]. This property of gradual decrease of diversity in the population limits the sustained exploration capabilities of simple genetic algorithms, since it inhibits continuing search which is necessary in solving difficult problems.

In the reinforcement learning approach to function optimization [?, ?] a point in the function space is generated according to a probabilistic distribution, and the corresponding function value which is called *reinforcement* is provided to the system. The parameters of the distribution are updated so as to direct the search towards the generated point (high reinforcement), or make the point less probable to be sampled again in the upcoming trials (low reinforcement). In applications to problems defined on binary domains, the simplest reinforcement learning scheme considers that the point $y = (y_1, \ldots, y_n)$ $(y_j \in \{0, 1\})$ to be evaluated at each step is generated by a team of *Bernoulli units*. Each Bernoulli unit j determines the component y_j of the output vector through a Bernoulli selection with probability $p_j = f(w_j)$, where $W = (w_1, \ldots, w_n)$ is the vector of adjustable parameters (weights) and f is a sigmoid function of the form

$$p_j = f(w_j) = 1/(1 + \exp(-w_j)) \tag{1}$$

REINFORCE algorithms [?] constitute an important class of reinforcement learning algorithms. We shall consider the application of a REINFORCE algorithm to a team of Bernoulli units. At each step, the weights are updated as follows:

$$\Delta w_j = \alpha_j (r - \overline{r})(y_j - p_j) - \delta w_j \tag{2}$$

where α_j is the learning rate factor, r is the reinforcement signal delivered by the environment and $\overline{r}$ is a standard of comparison. The latter is computed as a trace of past reinforcement values $\overline{r}(t) = \gamma \overline{r}(t-1) + (1-\gamma) r(t)$, the parameter γ being a decay rate positive and less than 1 (taken equal to 0.9 in our experiments). The decay term $-\delta w_j$ $(0 < \delta < 1)$ in (2) provides sustained exploration capability.

In our scheme the vectors of probabilities $P_i = (p_{i1} \ldots, p_{in})$ constitute the members of the population. For this reason the weights w_j have been discarded and the necessary updates are performed directly on the probability values, according to the following equation which is derived from (1) and (2):

$$p_{ij} := f\left((1-\delta) \ln \frac{p_{ij}}{1-p_{ij}} + \alpha_{ij}(r_i - \overline{r}_i)(y_{ij} - p_{ij})\right) \tag{3}$$

3 The Recombinative Scheme

According to the PRRL approach, each population member i is a vector P_i of probabilities p_{ij} $(i = 1, \ldots, p,\ j = 1, \ldots, n)$ that constitute the state of a reinforcement learning optimizer. At each step, first a reproduction procedure combines the probability vectors and a new generation of vectors is created. Then a sampling procedure is performed and p points $Y_i = (y_{i1}, \ldots, y_{in})$ $(i = 1, \ldots, p)$ of the function space are generated with Bernoulli selection using the corresponding probabilities p_{ij}. The fitness r_i of each point Y_i is evaluated and a reinforcement update of the probabilities takes place using equation (3) so that the search is guided towards promising regions of the space. At each generation step, the p new population members are created as follows: for each current member i we apply crossover with probability p_c. In this case, another member

k is randomly selected and a single-point crossover is performed between the two probability vectors. The crossover point t ($1 \leq t \leq n-1$) is randomly selected. In order to retard the decrease of population diversity, the new probability vector P_l is constructed so as to remain as close as possible to the vector P_i: if $t \leq \lceil n/2 \rceil$, then we set $p_{lj} = p_{kj}$ for $j = 1, \ldots, t$ and $p_{lj} = p_{ij}$ for $j = t+1, \ldots, n$, otherwise, we set $p_{lj} = p_{ij}$ for $j = 1, \ldots, t$ and $p_{lj} = p_{kj}$ for $j = t+1, \ldots, n$.

Reproduction is followed by the sampling (based on the new probability vectors) and evaluation phase for the p population members. Then, the reinforcement update takes place for each population member according to the learning rule of equation (3). The reinforcement comparison $\overline{r}_i$ is computed seperately for each population slot i as a weighted average of prior reinforcement values r_i. At the beginning of the genetic search, all the components of the probability vectors are set equal to 0.5, i.e., no initial knowledge is provided to the learning system. Sustained exploration is achieved with the introduction of the decay term, but we still have a lack of *sustained diversity* due to the effects of crossover.

In order to avoid *genetic drift* we have not employed the 'survival of the fittest' principle, but a uniform random selection scheme concerning all population members. Also, as already mentioned, each current population member is replaced by the closest of the two generated children, so as to avoid a serious disruption of the states of local optimizers. A third technique that has proved very effective in maintaining population diversity is based on the notion of *apathy* [?]. According to the latter approach, some population members remain apathetic for some generations, in the sense that they cannot be selected for recombination. Apathetic members cannot change their state through crossover, but can be chosen for crossover by other members of the population. We put a member into apathy whenever it generates a point of the state space of higher fitness than the best value achieved so far. If for a specified number of steps no better solution is obtained the member is brought back to the active state.

4 Experimental Results

The test problems that were selected for evaluating the effectiveness of our approach are versions of the order-3 deceptive problem, the same as in [?]. The problems considered assume eight subfunctions of the type described in the table below, thus the dimension of the binary space is 24. The fitness of each 24-bit string results from the sum of the corresponding values of the eight subfunctions.

x	000	001	010	011	100	101	110	111
$f(x)$	28	26	22	0	14	0	0	30

The first of the examined problems (called the *tight problem*) is a concatenation of eight subfunctions, where the first three bits of the string are the domain of the first subfunction, the next three bits are the domain of the second subfunction and so on. In the second problem (called the *loose problem*) bits 0, 8, and 16 constitute the domain of the first subfunction, bits 1, 9 and 17 are the domain

of the second subfunction and so on. Both problems have 255 local maxima and only one global maximum with value 240. In all experiments the algorithm was terminated when the global maximum was found or when 5000 generations had been performed concidering different sizes of the population p. In the latter case the result was the best solution found during the search. For each population size 30 experiments were performed using different seed values for the random number generator. The values of the parameters were $\alpha = 0.05$ and $\delta = 0.02$, whereas the crossover probability was $p_c = 1$ and the maximum allowed value for the apathy counter was set equal to 150.

		PRRL		PRL	
	p	*Success (%)*	*Avg. nb. steps*	*Success (%)*	*Avg. nb. steps*
Tight problem	8	46.6	1900	20.0	
	16	80.0	1019	46.6	1133
	32	100.0	640	93.3	1097
	64	100.0	530	100.0	1143
Loose problem	8	33.3	2746	28.0	3287
	16	62.0	1896	49.3	3010
	32	88.3	1465	75.0	2559
	64	100.0	1249	93.3	1973

The results summarized in the table illustrate the performance of PRRL in comparison with a Parallel Reinforcement Learning (PRL) scheme, that considers a population of independent reinforcement learning optimizers operating without crossover. The results represent the percentage of cases in which the global maximum was found, as well as the average number of generation steps required to find it. Experiments were also carried out using a simple genetic algorithm that was successful in very few cases. Our results show that the proposed technique offers clear advantage over both conventional genetic algorithms and parallel reinforcement learning. It is clear that the PRRL algorithm is characterized by a high degree of parallelism since all operations can be performed simultaneously for all members of the population.

References

1. Ackley, D. H., *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publishers, 1987.
2. Kontoravdis, D., Likas, A. and Stafylopatis, A. *A Reinforcement Learning Algorithm for Networks of Units with Two Stochastic Levels*, Proceedings ICANN-92, vol. I, pp. 143-146, Brighton, United Kingdom, 1992.
3. Mahfoud, S. W. and Goldberg, D. E., *Parallel Recombinative Simulated Annealing: A Genetic Algorithm*, IlliGAL Tech. Report No. 92002, Univ. of Illinois at Urbana-Champaign, July 1993.
4. Williams, R.J. and Peng, J., *Reinforcement Learning Algorithms as Function Optimizers*, Proceedings of the International Joint Conference on Neural Networks, Washington DC, vol. II, pp. 89-95, 1989.