A Study of the Lamarckian Evolution of Recurrent Neural Networks

Kim W. C. Ku, *Member, IEEE*, Man Wai Mak, *Member, IEEE*, and Wan-Chi Siu, *Senior Member, IEEE*

***Abstract*—Many frustrating experiences have been encountered when the training of neural networks by local search methods becomes stagnant at local optima. This calls for the development of more satisfactory search methods such as evolutionary search. However, training by evolutionary search can require a long computation time. In certain situations, using Lamarckian evolution, local search and evolutionary search can complement each other to yield a better training algorithm. This paper demonstrates the potential of this evolutionary–learning synergy by applying it to train recurrent neural networks in an attempt to resolve a long-term dependency problem and the inverted pendulum problem. This work also aims at investigating the interaction between local search and evolutionary search when they are combined. It is found that the combinations are particularly efficient when the local search is simple. In the case where no teacher signal is available for the local search to learn the desired task directly, the paper proposes introducing a related local task for the local search to learn, and finds that this approach is able to reduce the training time considerably.**

***Index Terms*—Evolutionary computation, Lamarckian evolution, recurrent neural networks.**

I. Introduction

USING gradient-based local search methods to train neural networks has difficulties in: 1) escaping from local optima when the search surface is rugged, 2) finding better solutions when the surface has many plateaus (gradient is zero, for example), and 3) deciding the search direction when gradient information is not readily available (lack of teacher signals, for example). To alleviate the above deficiencies, Tang and Koehler [28] proposed a global optimization algorithm that subdivides the search space into subregions, and the subregions not containing the global optimum are excluded from searching. However, the complexity of the algorithm grows exponentially with the number of nodes in the networks. Another global optimization algorithm [27] uses a user-defined trace function to lead the search away from local optima, but it is doubtful that a suitable trace can easily be found. Gradient-based algorithms with multistarts [12] can also be used, but the appropriate number of restarts is difficult to derive *a priori*, and the computation time can be very long. It is widely believed that evolutionary search (see [3], [10], [32], [37], and [38] for a review) and simulated annealing [7] can overcome the above difficulties, but these algorithms may require a large number of iterations in order to obtain an acceptable solution.

Recurrent neural networks (RNN's) have closed paths in their topology that enable them to preserve their past states. Therefore, RNN's have the capability of dealing with spatiotemporal tasks that have been found to be difficult for feedforward networks [23]. Bianchini *et al.* [6] observed that the cost function of a feedforward network for any learning task is closely related to that of an equivalent RNN.[1] As a result, any occurrence of local optima in the feedforward network can also be found in the equivalent RNN case. However, an RNN could have additional local optima that may not exist in the feedforward network. Therefore, Bianchini *et al.* [6] argue that local optima occur more frequently in RNN's and that the training of RNN's is more difficult. However, this difficulty could be overcome by combining the efforts of local search (learning) and evolutionary search as they could complement each other.

There are two approaches to embedding learning in an evolutionary search, namely Lamarckian evolution [1], [34] and evolution based on the Baldwin effect [4], [29]. In this paper, we focus on the former approach since we found that it outperforms the latter approach in our previous studies [16], [18].[2] We conjecture that the inefficiency of the latter approach is due to the fact that too many weights in the networks can be changed by learning, and the changes can be substantial [18]. As a result, it is difficult for the evolutionary operations to produce the genotypic changes that match the phenotypic changes due to learning. Further evidence for supporting this argument can be found in [17].

Lamarckian evolution is based on the inheritance of acquired characteristics—an individual can pass the characteristics (observed in the phenotype) acquired through lifetime learning to its offspring genetically (encoded in the genotype). As learning takes place in phenotype space, Lamarckian evolution requires an inverse mapping from the phenotype space to the genotype space, which is impossible in biological systems. However, when there is a simple relationship between the genotypes and the phenotypes (as in our case of evolving RNN's) such that the new phenotypes acquired through learning can be mapped onto the corresponding genotypes, Lamarckian evolution is possible and potentially beneficial.

When Lamarckian learning is embedded in evolutionary search, the change in phenotypes by learning is transformed

Manuscript received November 14, 1997; revised February 16, 1998 and April 20, 1999. This work was supported by the Hong Kong Polytechnic University under Grant A/C A-PA58 and by the Centre for Multimedia Signal Processing, The Hong Kong Polytechnic University.

The authors are with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: enmwmak@polyu.edu.hk).

Publisher Item Identifier S 1089-778X(00)00623-8.

[1]An RNN can be unfolded in time to form a feedforward network with equivalent temporal characteristics [30].

[2]The performance comparisons were based on evolving neural networks for sequence recognition.

to the corresponding change in genotypes. The transformed genotypes are used in subsequent reproduction. When the same concept is applied to the training of neural networks, the inborn weights (weights as a result of evolutionary operations) are replaced by the weights obtained through learning for further evolutionary operations. Therefore, the acquired knowledge through learning is coded directly in the genotypes, resulting in a transfer of knowledge to the offspring.

This paper proposes and compares different approaches to embedding Lamarckian learning in a special type of evolutionary search method, referred to as the cellular genetic algorithm (GA). The resulting hybrid algorithms were applied to train RNN's. The performance of these networks has been evaluated through a sequence recognition problem and a neurocontrol problem. The results show that the hybrid algorithms are not only able to reduce training time, but also are able to improve solution quality significantly. The paper also provides detailed analyses on the interaction between the local search and evolutionary search. These analyzes help to explain why the hybrid algorithms achieve better performance.

The rest of this paper is organized as follows. The cellular genetic algorithm is described in Section II. In Section III, the local search methods used in our experiments are explained. Performance evaluation and analysis of these evolutionary–learning synergy approaches in the sequence recognition problem and the neurocontrol problem are provided in Sections IV and V. The implications of these experiments are summarized and discussed in Section VI.

II. Cellular Genetic Algorithm

Cellular GA's [8], [9], [31] are a special form of GA in which the population of chromosomes are organized as a toroidal, two-dimensional square grid, with each grid point representing a chromosome. In cellular GA's, reproduction can only occur between neighboring chromosomes. This local reproduction has the effect of reducing selection pressure to achieve more exploration of the search space [20]. Therefore, cellular GA's have been used in our experiments in order to reduce the risk of getting stuck in local optima, especially when Lamarckian learning is embedded [1]. In this study, each weight in an RNN is encoded as a gene of a chromosome in the form of a floating-point number. A chromosome, in which the number of genes is equal to the number of weights, represents an RNN. The simple relationship between the phenotype and the genotype makes Lamarckian learning possible. Fig. 1 illustrates the procedure of the cellular GA used in our experiments.

III. Local Search

Local methods search for better solutions in the neighborhood of the current solution. Most of them rely on the availability of gradient information to find better solutions. Their major drawback is that they are easily trapped in local optima. Despite this drawback, they have been applied widely to train RNN's. Typical examples include the real-time recurrent learning algorithm [36] and the backpropagation-through-time algorithm [30].

A. Real-Time Recurrent Learning (RTRL)

The real-time recurrent learning (RTRL) algorithm [36] calculates the instantaneous error gradient $\nabla_{\boldsymbol{w}} J(t)$ by

$$\frac{\partial J(t)}{\partial w_{ij}} = -\sum_k \left(d_k(t) - y_k(t)\right) \frac{\partial y_k(t)}{\partial w_{ij}} \tag{1}$$

where $J(t) = 1/2(d_k(t) - y_k(t))^2$ is the instantaneous squared error at time step t, $y_k(t)$ and $d_k(t)$ are, respectively, the actual output and the desired output of output node k at time step t, and w_{ij} is the weight connecting node j to node i. The sensitivity $(\partial y_k(t)/\partial w_{ij})$ is obtained by the recursion

$$\frac{\partial y_k(t+1)}{\partial w_{ij}} = f_k'(s_k(t)) \left\{ z_j(t)\delta_{ki} + \sum_q w_{kq} \frac{\partial y_q(t)}{\partial w_{ij}} \right\} \tag{2}$$

with $(\partial y_k(0)/\partial w_{ij}) = 0$, where $z_k(t)$ is either the signal applied to input node k at time step t or the actual output of processing node k at time step t, $s_k(t)$ is the net input to processing node k: $s_k(t+1) = \sum_j w_{kj} z_j(t)$, $f_k'(\,)$ is the derivative of the sigmoidal function $f_k(\,)$, and δ_{ki} is the Kronecker delta.

The RTRL algorithm is a gradient-based algorithm in which all of the weights are changed at every time step in a direction opposite to the instantaneous error gradient. It is computationally intensive because it has a computational complexity of $O(n^4)$ for each time step, where n is the number of processing nodes.

B. Delta Rule

The running time of the RTRL algorithm scales poorly with the network size. In order to reduce computational complexity, we propose to update only the weights that connect to the output nodes. Specifically, we only compute the gradient $(\partial J(t)/\partial w_{ij})$ in (1) whenever node i is an output node. Therefore, (2) is simplified to

$$\frac{\partial y_i(t+1)}{\partial w_{ij}} = \begin{cases} f_i'(s_i(t)) z_j(t), & \text{when } i \text{ is an output node} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

This is equivalent to the delta rule for feedforward networks. The dynamics of the network remain unchanged; however, the updates of weights are based on a feedforward architecture. The philosophy behind this approach is to lower the computational complexity by eliminating the term $\sum_q w_{kq}(\partial y_q(t)/\partial w_{ij})$ in (2).

C. Limitations of Local Search

If gradient-based algorithms are able to train RNN's, there will be little incentive to use evolutionary search methods. However, there are situations in which gradient-based algorithms have difficulties in finding an appropriate neural network. The problems used in our experiments are typical examples. The first one is a sequence recognition problem where gradient-based algorithms frequently become stuck in undesirable regions in the search space. The second is a neurocontrol problem, where gradient-based algorithms are not appropriate because teacher signals are not available. The cellular GA was found to be successful in finding an acceptable solution to these two problems.

procedure cellularGA

c_k : a chromosome at position (x_k, y_k) in the grid
c_{new} : a newly produced chromosome
l : length of random walk
M : total number of chromosomes in the population
$w_{ij}^{c_k}$: weights w_{ij} of the network corresponding to c_k
$f(c_k)$: fitness of c_k
$\mathcal{I}$: the set of indexes representing the input nodes (including the bias)
$\mathcal{U}$: the set of indexes representing the processing nodes

begin
Initialize a population of M chromosomes c_k, and evaluate the corresponding fitness $f(c_k)$ where $k = 1, 2, \ldots, M$

// Generate a new chromosome for each reproduction cycle
repeat
Randomly select c_0 at (x_0, y_0) in the grid

// Choose parent c_a along a random walk originating from (x_0, y_0)
Create a random walk set $\{c_1$ at $(x_1, y_1), c_2$ at $(x_2, y_2), \ldots, c_l$ at $(x_l, y_l)\}$ such that
$|x_{k+1} - x_k| \leq 1$ and $|y_{k+1} - y_k| \leq 1$, $k = 0, 1, 2, \ldots, l-1$
Select c_a such that $f(c_a)$ is the best along the random walk

// Choose parent c_b along another random walk originating from (x_0, y_0)
Create a random walk set $\{c'_1$ at $(x'_1, y'_1), c'_2$ at $(x'_2, y'_2), \ldots, c'_l$ at $(x'_l, y'_l)\}$ such that
$|x'_{k+1} - x'_k| \leq 1$ and $|y'_{k+1} - y'_k| \leq 1$, $k = 1, 2, \ldots, l-1$ and
$|x'_1 - x_0| \leq 1$ and $|y'_1 - y_0| \leq 1$
Select c_b such that $f(c_b)$ is the best along the random walk

// Apply crossover to c_a and c_b to produce c_{new}
for all $i \in \mathcal{U}$, $j \in \mathcal{U} \cup \mathcal{I}$ **do**

$$w_{ij}^{c_{new}} := \begin{cases} w_{ij}^{c_a} & \text{with a probability of 0.5} \\ w_{ij}^{c_b} & \text{with a probability of 0.5} \end{cases}$$

endloop

// Apply mutation to c_{new} by randomly selecting a processing node in the network, and each
// weight connected to the input part of the node is changed by exponentially distributed mutation
Randomly select $i \in \mathcal{U}$
for all $j \in \mathcal{U} \cup \mathcal{I}$ **do**

$$w_{ij}^{c_{new}} := \begin{cases} w_{ij}^{c_{new}} + \delta & \text{with a probability of 0.5} \\ w_{ij}^{c_{new}} - \delta & \text{with a probability of 0.5} \end{cases}$$

// δ is a positive number randomly generated from an exponential
// distribution with density function of the form e^{-x}, $x > 0$
endloop

// Replace c_0 by c_{new} if the latter has better fitness
Evaluate $f(c_{new})$
if $f(c_{new}) < f(c_0)$ **then** $c_0 := c_{new}$
until termination condition reached
endproc cellularGA

Fig. 1. Procedure of the cellular GA.

Further improvement in training speed and solution quality can also be obtained by embedding local search in the cellular GA.

IV. The Long-Term Dependency Problem

Many sequence recognition tasks such as speech recognition, handwriting recognition, and grammatical inference involve long-term dependencies—the output depends on inputs that occurred a long time ago. The sequences involved in these tasks are characterized typically by different time scales. In terms of short time scales, they can be characterized by the dynamics that generate the sequences, while in terms of long time scales, they may have syntactic and semantic structures. For example, speech recognition involves the processing of short-term speech signals, as well as the processing of phonemic features spanning a much longer interval. In grammatical inference [19], a single word at the beginning of a sentence may affect the grammatical correctness or alter the meaning of the sentence.

The performance of these applications depends mainly on whether or not the long-term dependencies can be represented accurately; however, extracting these dependencies from data is not an easy task. While recurrent neural networks provide a promising solution to this problem, previous research [5] has shown that the commonly used gradient descent algorithms have difficulty in learning the long-term dependencies. To overcome this difficulty, we propose to combine the cellular GA's and local search for training RNN's.

The long-term dependency problem is defined as follows. It is required to learn a temporal relationship from a sequence of symbols such that the output at time $t+k$ depends on the inputs from time t to time $t+k-1$. The input sequences contain symbols drawn from a symbol set, and each symbol is represented by a binary number. There are only two possible input sequences: $\{x, a_1, a_2, a_3, \ldots, a_k\}$ and $\{y, a_1, a_2, a_3, \ldots, a_k\}$. The first symbol in an input sequence can be either x or y, but the next k input symbols are fixed. When the first input symbol is x at time t, the output at time $t+k$ is x'; when the first input

TABLE I
EXAMPLE OF INPUT AND OUTPUT SEQUENCES FOR THE LONG-TERM DEPENDENCY PROBLEM

time step	$t-1$	t	$t+1$	$t+2$	$t+3$	$t+4$	$t+5$	$t+6$	$t+7$	$t+8$	$t+9$	$t+10$	$t+11$
input symbol	...	x	a_1	a_2	a_3	a_4	a_5	y	a_1	a_2	a_3	a_4	a_5
input pattern	...	001	010	011	100	101	110	000	010	011	100	101	110
output symbol	...	a_1	a_2	a_3	a_4	a_5	x'	a_1	a_2	a_3	a_4	a_5	y'
output pattern	...	01000	01100	10000	10100	11000	00010	01000	01100	10000	10100	11000	00001

Note: For the output patterns, the last two bits determine whether the output symbol after five time steps is x' or y'.

symbol is y at time t, the output at time $t+k$ is y'. For other time intervals, the output predicts the next input. Table I shows an example of input and output sequences with k equal to 5.

In this study, an RNN with three input nodes (a three-bit binary coding was used) and 12 processing nodes (five of them were used as the output nodes) was used to learn the long-term dependency problem with k equal to 5. Therefore, there are a total of $12 \times 12 + 12 \times (3+1) = 192$ weights required to be optimized. The population size of the cellular GA is 100, and each random walk has four steps. The fitness of a chromosome is determined by the network error function, which is the mean-squared error (MSE) between the actual outputs and the desired outputs for the whole training set. In this case, the better the input sequence is recognized, the smaller is the MSE between the actual outputs and the desired outputs.

A. Applying Local Search or Cellular GA's Alone

A set of control experiments has been performed to train the RNN's by local search alone. The limitation of using the gradient-based algorithms (RTRL and delta rule) to solve the long-term dependency problem is demonstrated clearly in Fig. 2, where the MSE's were quickly stuck at 0.08, and no improvement could be obtained by further training. This suggests that there are difficult regions (around an MSE of 0.08) in the search space where the gradient-based algorithms are likely to be trapped, and that applying these local search methods alone (especially the delta rule) is not able to solve the long-term dependency problem.

On the other hand, Fig. 2 shows that the cellular GA is more capable of solving the problem. The average MSE attained after 4 min of simulation (i.e., 20 000 generations) is 0.0303, which is lower than that of the local search methods. Further improvement to the cellular GA might be achieved when local search is embedded. In the following experiments, different approaches to embedding Lamarckian learning in cellular GA's are compared and analyzed.

B. Low Frequency of Learning

One approach to combining local search and cellular GA's is to apply the local search at regular generation intervals. In this experiment, a chromosome was randomly chosen for learning at every 20 generations, while the reproduction process remained unchanged. During learning, the chosen RNN (chromosome) was trained by the RTRL algorithm or the delta rule for one epoch, where an epoch is a complete presentation of all training patterns. The learning rate was set to 0.9 for both algorithms.

The results based on 200 simulation runs are plotted in Fig. 2. When the delta rule is embedded in the cellular GA, the average MSE's attained after 4 min of simulations are smaller than that of the pure cellular GA. This indicates that the evolution of RNN's is improved by the application of the delta rule. On the other hand, Fig. 2 shows clearly that embedding the RTRL algorithm degrades the performance rather than improves it. However, the CGA–RTRL hybrid algorithm achieves an average MSE of 0.0136 (with a variance of 0.0010) after 20 000 generations, which is lower $(p < 0.01)$ than that of the pure cellular GA.[3] This suggests that combining RTRL and the cellular GA has merits, provided that computation time is not an issue. Although RTRL may provide some benefit, the corresponding increase in computation time may not provide a sufficient payoff.

C. Interaction Between Evolutionary Search and Local Search

The above experiments show that an appropriate combination of evolutionary search and local search can reduce the overall training time and improve the solution quality. In order to understand how evolutionary search and local search interact with and complement each other, another set of experiments was performed. Here, in each simulation run, we measured the fitness improvement frequency as a result of the application of local search. The average MSE's of the chromosomes in the population were also recorded. The frequency was obtained by dividing the number of times that a chromosome was improved as a result of learning by the total number of times that learning was applied within a fixed period.

Fig. 3 plots the fitness improvement frequency as a function of generations in a typical simulation run where local search was applied at every 20 generations. At the beginning of the evolution, the MSE's of the population are relatively large, suggesting that they are far away from the optima. During this period (before 3000 generations), the fitness improvement frequency is close to 1 and the average MSE's decrease rapidly. This suggests that the local search is able to complement the evolutionary search.

Fig. 3 also shows that, between 5000 and 6000 generations, the frequency drops to low values, and the average MSE's are in

[3]It is also lower than that of the CGA–delta rule hybrid algorithm, but the difference is not significant $(p > 0.05)$.

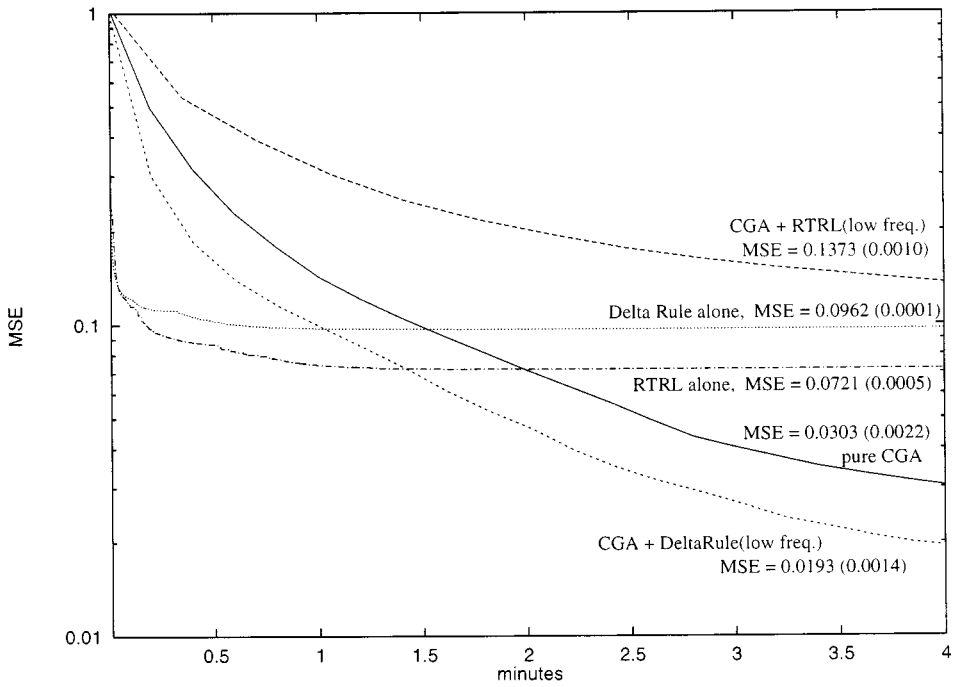


Fig. 2. MSE (based on the average of 200 simulations of the long-term dependency problem) of the best network found by the cellular GA and its hybrid algorithms where local search (RTRL or delta rule) was applied at every 20 generations (i.e., low-frequency learning). The average MSE's after 4 min of simulation and their variances (inside parentheses) are also shown. The performance of gradient-based algorithms alone is also illustrated. All differences in means are statistically significant ($p < 0.01$, calculated by Student's t tests).

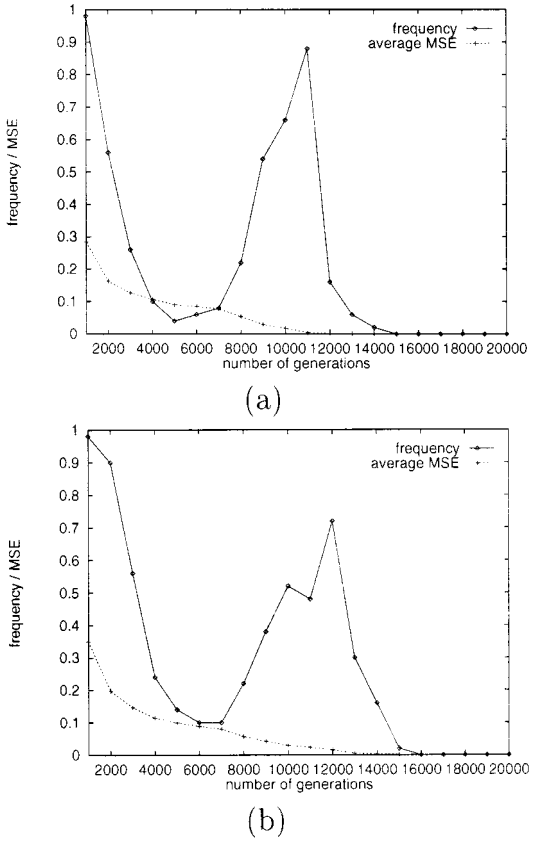


Fig. 3. Graphs showing the variations of fitness improvement frequency (in a typical simulation run) as a result of embedding (a) RTRL and (b) delta rule in the cellular GA. Local search was applied at every 20 generations. The average MSE's of the chromosomes in the population (instead of the best individual) are also plotted.

the range 0.08–0.09. Experiments in Section IV-A have demonstrated that there are difficult regions in the search space corresponding to an MSE of 0.08 where local search is incapable of finding better solutions. Therefore, the fitness improvement frequency during this period becomes very low. However, Fig. 3 shows that the frequency starts to increase after 6000 generations. One reason for this phenomenon is that the evolutionary search helps the networks to move out of the difficult regions. In other words, evolutionary search is able to complement the local search. Beyond 12 000 generations, it becomes increasingly difficult to reduce the MSE's by applying local search because most of the networks are already very close to the optimal solution. Therefore, the frequency decreases as the average MSE approaches zero.

D. Lifetime Learning

Learning is expensive because it takes time. Therefore, learning was applied at a regular generation interval in the previous experiments. However, when the computational complexity of the local search methods is low, it might be beneficial if learning were applied more frequently. In this section, we consider the case where learning is applied to the newly born offspring at every generation. We refer to this approach as "lifetime learning."

Fig. 4 shows that lifetime learning is not appropriate for the CGA–RTRL hybrid algorithm as it results in very poor performance. This agrees with the result obtained in Section IV-B: the RTRL algorithm is so computationally intensive that the gain obtained from learning cannot compensate for the loss in computation time. On the other hand, performance is greatly improved when lifetime learning is applied to the CGA–delta-rule

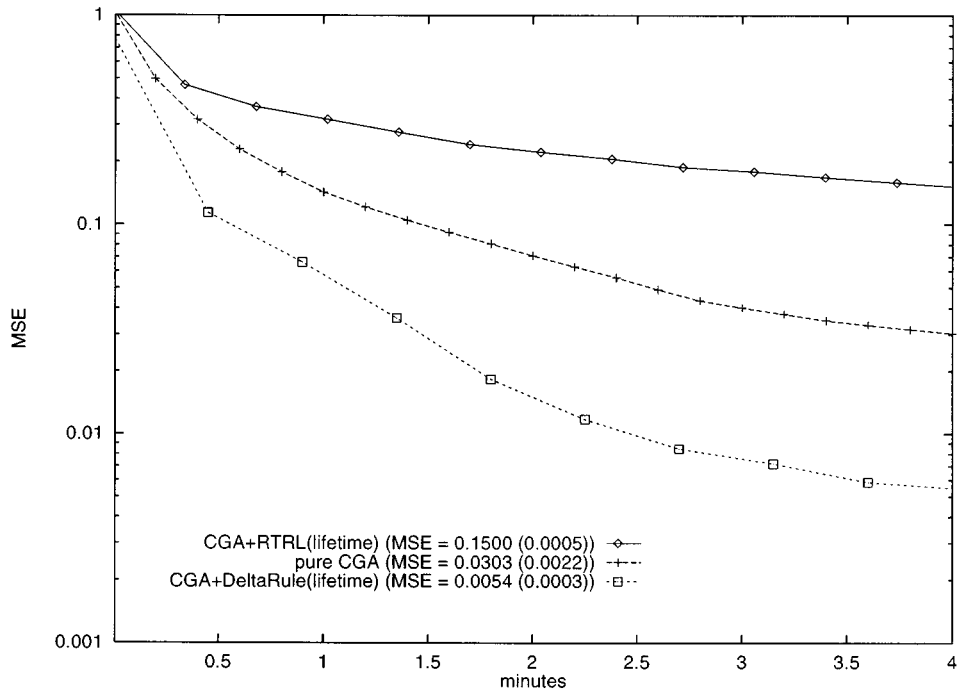


Fig. 4. MSE (based on the average of 200 simulations of the long-term dependency problem) of the best network found by the lifetime learning approach where local search (RTRL or delta rule) was applied to the offspring generated at every generation. The average MSE's after 4 min of simulation and their variances (inside parentheses) are also shown. All differences in means are statistically significant ($p < 0.01$, calculated by Student's t tests).

hybrid algorithm, and the average MSE attained after 4 min is only 18% of that attained by the pure cellular GA.

The computational complexity of the delta rule is low because the RNN is considered as a feedforward network when the error gradient is computed. However, the delta rule is so simple that the error gradient obtained by this algorithm may be inaccurate. As a result, the fitness of a chromosome could deteriorate after the application of the delta rule. Despite this deficiency, the low computational complexity of the delta rule can shorten the overall training time when lifetime learning is embedded in the cellular GA. Among all of the hybrid algorithms, combining the cellular GA and the delta rule achieves the lowest MSE for a given CPU time. The benefit of this approach is further demonstrated in the next section where RNN's are applied to solve the inverted pendulum problem.

V. The Inverted Pendulum Problem

A pendulum of fixed length is hinged at the top of a cart which is free to travel along a horizontal track with fixed length. It is required to balance the pendulum in a vertical plane and to keep the cart within the track boundaries. The dynamics of the system are governed by a system of differential equations:

$$\ddot{\theta} = \frac{mg\sin\theta - \cos\theta(F + m_p l\dot{\theta}^2\sin\theta)}{(4/3)ml - m_p l\cos^2\theta} \tag{4}$$

and

$$\ddot{h} = \frac{F + m_p l(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta)}{m} \tag{5}$$

where h is the cart position, $\dot{h}$ is the cart velocity, θ is the angle of the pendulum, $\dot{\theta}$ is the angular velocity of the pendulum, m_p is the mass of the pendulum (0.1 kg), m is the total mass of the system (1.1 kg), l is the length of the pendulum (0.5 m), F is the force applied, and g is the acceleration of gravity (9.8 m/s^2).

The system dynamics were approximated using the Euler method (i.e., $\theta(t+1) = \theta(t) + \tau\dot{\theta}(t)$) with a time step of $\tau = 0.02$ s. The system is considered to be out of balance when the pendulum falls beyond 12° from the vertical position or the cart runs beyond ±2.4 m from the center.

Previous approaches [2], [22], [33] to tackling the inverted pendulum problem employed a feedforward neural network using $h, \dot{h}, \theta$, and $\dot{\theta}$ as inputs, and the output was interpreted as the force applied to the cart. While the trained networks are able to balance the pendulum, four input variables are required to represent the system status. In real applications, practitioners may find difficulty in acquiring system information such as the cart velocity and the angular velocity. Therefore, it is important to obtain a neurocontroller with as few input variables as possible.

This problem can be tackled, as in [35], by using an RNN in which h and θ are the only inputs. In our experiments, an RNN with two inputs and six processing nodes (one of them also serves as an output node), as shown in Fig. 5(a), was used to balance the pendulum.[4] At each time step, θ and h were applied as inputs to the RNN. If the output were less (larger) than 0.5, the cart was pushed to the left (right) by a force of $10N$. A cellular GA, with a population size of 100 and a random walk of four steps, was used to train the RNN's. The fitness of an RNN is defined as the number of time steps for which the system is still

[4]Note that there are a total of $6 \times 6 + 6 \times (2+1) = 54$ weights required to be optimized.

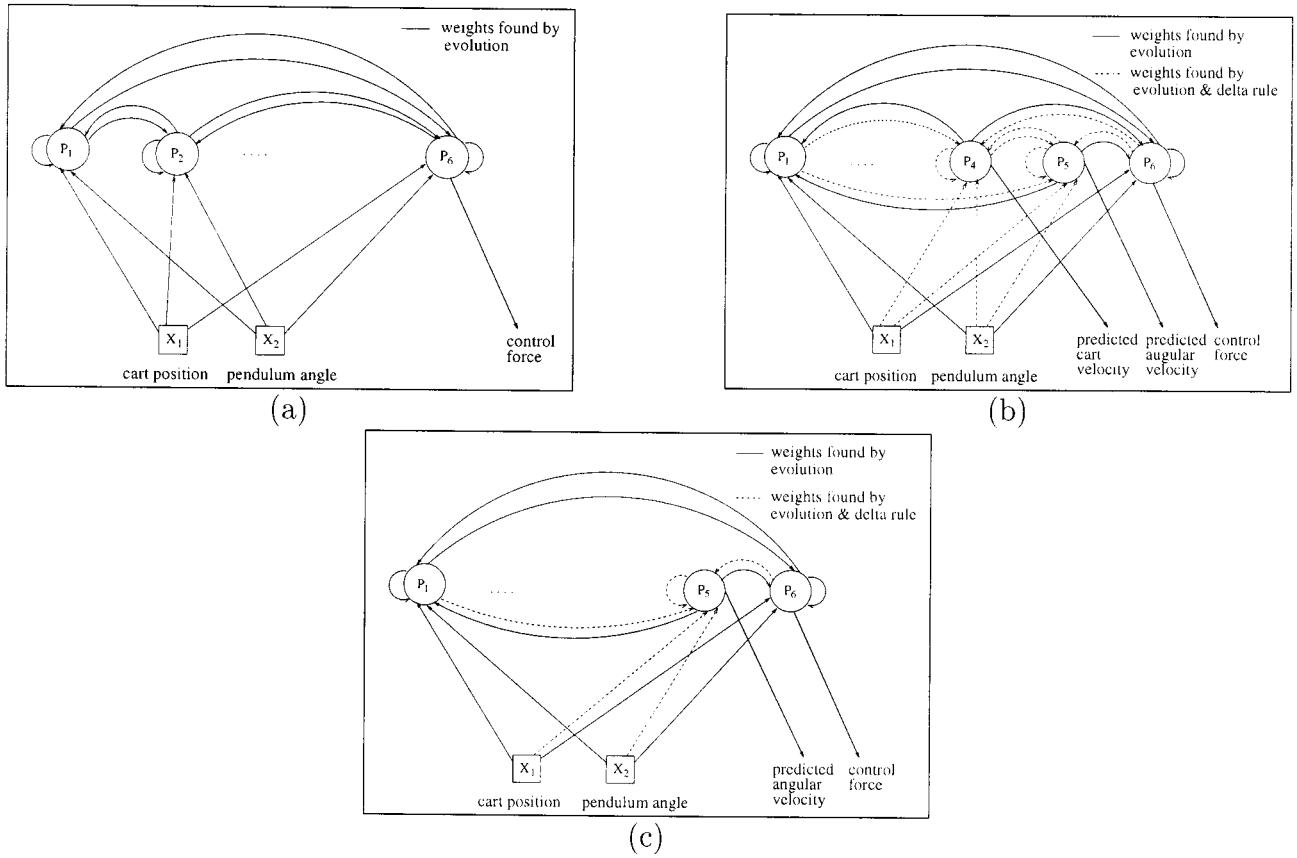


Fig. 5. RNN's for balancing the inverted pendulum. The output node(s) represent(s): (a) the control force, (b) the control force, predicted cart velocity, and predicted angular velocity, and (c) the control force and predicted angular velocity. Note that all weights were trained by evolutionary search, but some of them were also trained by local search. These special weights are labeled in the figures.

balanced. The system was started from a random angle in the range of ±12° and a random cart position in the range of ±2.4 m.

A. Learning a Related Local Task

In the long-term dependency problem, the evolutionary search has been used to find an RNN that matches the actual output with the desired output at every time step. This goal has been the same in the local search methods (RTRL and delta rule). Therefore, both the evolutionary search and the local search aim at finding a solution for the same global task, i.e., to match the actual network outputs with the desired outputs. In the inverted pendulum problem, however, no teacher signal (except for the failure signal at the end indicating that the system is out of balance) and no local search method are available to solve the global task (i.e., balancing the pendulum) directly. Therefore, a simple combination of evolutionary search and local search as in the previous approaches cannot solve this problem. In this work, we propose to overcome this difficulty by allowing the local search to solve a different but "related" local task, i.e., to predict the cart velocity $\dot{h}$ and the angular velocity $\dot{\theta}$ at the next time step. A local task is considered to be "related" to a global task if a completely resolved local task helps to accomplish the global task. The relationship between the global task and the local task in the inverted pendulum problem is examined in the next section.

The idea of using a different task for the local search to solve was first introduced by Nolfi *et al.* [24]. In their experiments, feedforward neural networks were evolved for food hunting (the global task), and the fitness of a feedforward network was determined by the amount of food gathered during its life span. Nolfi *et al.* found that the food-hunting ability will be improved if the feedforward networks are also trained by backpropagation [26] to predict the location of the nearest food. The rationale of introducing a different task (e.g., predicting the food location) for local search to solve is illustrated in Fig. 6. Finding a neural network for a specific task can be considered as searching for the optimum of a fitness function. If the fitness function of a local task is similar to that of the global task, moving a solution toward the optimal regions of the local task may assist the search for the optimal solution of the global task.

Motivated by the work of Nolfi *et al.*, we adopt a similar strategy for the inverted pendulum problem, i.e., introducing a local task to predict $\dot{h}$ and $\dot{\theta}$ (cf. predicting the food location in [24]) in order to help solve the global task of balancing the system (cf. food hunting in [24]). The neural network used in the experiment is shown in Fig. 5(b), where two processing nodes are used to predict the velocities. Although velocity information

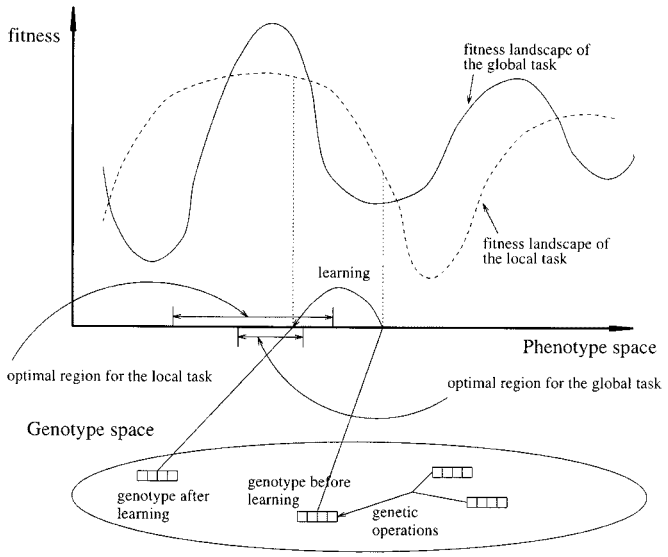


Fig. 6. Diagram depicting the hypothetical relationship (based on the arguments of Nolfi *et al.* [24] and Parisi *et al.* [25]) between a global task and a "related" local task. The former is optimized by evolutionary search, while the latter is optimized by local search. There is an overlap between the optimal regions of these tasks. The local search (learning) progressively brings the solution of the local task closer to its optimum. This may assist the evolutionary search in finding the optimal solution of the global task.

is required during training, it is not required after training, and the trained RNN's only require two inputs to balance the system.

Due to the success of the lifetime learning approach in the long-term dependency problem, it was used in the inverted pendulum problem. The delta rule was used as the local search method. The RNN generated at the end of each reproduction cycle was applied to balance the pendulum, and during the first 1000 time steps,[5] the delta rule was applied to learn the local task. After that, no more local search was applied. The number of steps for which the pendulum was balanced was used as the fitness value for subsequent reproductions. This process was repeated for every generation. Table II lists the training time required to evolve an RNN to keep the system balance for 120 000 time steps. It shows that allowing the RNN's to predict $\dot{h}$ and $\dot{\theta}$ can reduce the training time (compare Algorithms I and V). Table II also shows that, although the training time and the number of generations required to balance the pendulum depend on the training algorithms, they exhibit a large variation. This is because some starting conditions [e.g., $\theta(0) = 12°$ and $h(0) = 2.4$ m] could lead to a training time much longer than the average training time. Therefore, we use a nonparametric statistical test (Wilcoxon–Mann–Whitney) [15] which is applicable to data with continuous distributions. The tests suggest that the difference in average training time and average number of generations in Table II are significant ($p < 0.01$).

B. Empirical Analysis

According to the above results, we conclude that the time taken to find an RNN to balance the inverted pendulum is reduced when local search is applied to learn the prediction of cart velocity and angular velocity. In order to understand why learning a local task can help improve the evolution process, further experiments were performed.

1) Perfect Performance in the Local Task Helps Evolution: In this section, the relationship between the local task (predicting velocities) and the global task (balancing the pendulum) is examined. This is achieved by determining whether or not the evolution time for finding a solution for the global task can be reduced when the local task is assumed to be completely resolved. The architecture of the RNN's remains unchanged, as shown in Fig. 5(b); however, no weights were allowed to be changed by the local search, and the outputs of the two processing nodes were forced to $\dot{h}$ and $\dot{\theta}$ at every time step. Therefore, all RNN's appear to have an innate ability to predict the velocities precisely. In the experiment, evolutionary search alone (without learning) was used to train the RNN's using $h, \dot{h}, \theta$, and $\dot{\theta}$ as inputs.

Table II shows that the above experimental setup (Algorithm III) results in very short training time[6] as compared to evolutionary search alone (Algorithm I). This result suggests that the completely resolved local task helps solve the global task. This also indicates that the local task is not arbitrary; rather it is "related" to the global task. The experiments in Section V-B-3 further demonstrate that the way in which a local task and a global task are "related" could affect the evolution process that solves the global task.

2) Good Performers in the Local Task Helps Evolution: In the previous experiment, all RNN's have an innate ability to model the dynamics of the pendulum system. However, errors cannot be avoided when the ability has not been completely acquired. Here, we demonstrate that even though the RNN's have not learned the local task completely, those with good performance (as a result of learning rather than innate ability) in the local task are also likely to achieve good performance in the global task. In the experiment, the performance (measured in terms of the MSE's between the actual and predicted velocities) in solving the local task by the application of local search was recorded. The performance (measured in terms of the number of steps for which the pendulum is still balanced) of the global task was also recorded. The results are illustrated in Fig. 7 (the ⟐-lines). It shows that the performance of both local and global tasks is poor in the first 1000 generations. However, after 1000 generations, the performance of the global task starts to improve, and the MSE's reduce to a low level, indicating that the networks have learned the system dynamics to some extent. Fig. 7 also shows that the RNN's that are good at balancing the pendulum are also good at predicting the velocities. This suggests that there is an overlap, as hypothesized in Fig. 6, between the optimal regions of the local task and the global task in the inverted pendulum problem.

[5]The system may be out of balance before reaching the 1000th time step.

[6]Algorithm III was created for analytical purposes. Although its training time is very short, it has little practical value as it assumes that every offspring has an innate ability to resolve the local task.

TABLE II
TRAINING TIME AND NUMBER OF GENERATIONS (WITHOUT CONSIDERING THE TIME SPENT ON LEARNING) REQUIRED TO EVOLVE AN RNN TO BALANCE THE PENDULUM FOR UP TO 120 000 TIME STEPS

Training Algorithm	Local Task	Innate Ability	Training time (sec)		No. of generations	
			Average	Std. Dev.	Average	Std. Dev.
(I) pure CGA	Nil	Nil	301	317	5307	3022
(II) pure CGA	Nil	$\dot{\theta}$	112	100	1064	562
(III) pure CGA	Nil	$\dot{h}$ & $\dot{\theta}$	31	27	627	406
(IV) CGA + learning	$\dot{\theta}$	Nil	366	262	3965	1973
(V) CGA + learning	$\dot{h}$ & $\dot{\theta}$	Nil	140	143	2448	1723
(VI) CGA + reverse-learning	$\dot{h}$ & $\dot{\theta}$	Nil	1133	1688	9156	6770

Note: Results are based on 100 simulations running on a Pentium-Pro 200 MHz processor. All differences in means are statistically significant ($p < 0.01$) according to the Wilcoxon–Mann–Whitney test.

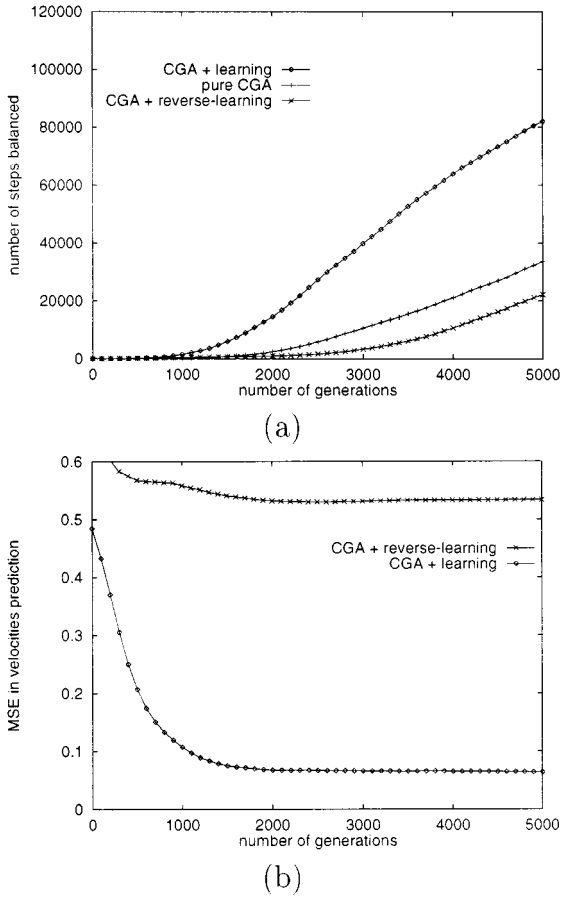


Fig. 7. Graphs showing: (a) the average performance of the global task (number of steps for which the pendulum is still balanced), and (b) the average performance of the local task (MSE in velocity prediction) achieved by the chromosomes in the population. Learning (predicting $\dot{h}$ and $\dot{\theta}$) and reverse learning (maximizing the error in velocity prediction) are performed by applying the delta rule. The results are based on the average of 100 simulations.

In order to further validate that improving the performance of the local task can really accelerate the evolution of RNN's, another experiment was performed. In the experiment, the performance of the local task was degenerated, rather than improved, by the local search. More specifically, the weights were adjusted by "reverse learning," where they were changed in a direction that maximizes MSE, i.e., using gradient ascent to maximize MSE's instead of using gradient descent to minimize MSE's. The results in Fig. 7 (the $\asymp$ lines) indicate that, when the MSE's are high (maintained by "reverse learning"), finding a solution for the global task becomes very difficult. Table II (Algorithm VI) shows that applying reverse learning leads to a very long training time as compared to other hybrid algorithms. These results suggest that improving the performance of the local task by local search is beneficial to the evolution of RNN's.

We must stress that good performance in a related local task does not necessarily mean that the performance in the global task is also good. More specifically, a network that can predict the velocities of the pendulum precisely does not imply that it is able to balance the pendulum. We have illustrated in Section V-B-1 that RNN's with an innate ability to predict velocities have to be fine tuned by evolutionary search before they can balance the system. On the contrary, a network that is able to balance the system does not imply that it can predict the velocities perfectly. For example, Algorithm VI of Table II illustrates that it is possible for networks with high MSE's in velocity prediction, maintained by gradient ascent, to balance the system, although it takes a much longer time and needs many more generations to achieve this goal.

3) Preferable Local Tasks: In the previous experiments, the local task is to learn the system dynamics by predicting the angular and cart velocities. It would be interesting to investigate how the interaction between the local and global tasks affects the evolution process.

The experiments in Section V-B-1 show that having an innate ability to model the dynamics of the whole system can improve the evolution process. The improvement is likely to be small if only part of the system dynamics (angular velocity, for example) is known to the RNN's. This argument is supported by the experimental results in Table II, where evolutionary search with an innate ability to predict the angular velocity only (Algorithm II) takes a longer time as compared to the one with an innate ability to predict both velocities (Algorithm III). Therefore, the task of predicting only the angular velocity is considered as being related loosely to the global task.

Here, the loosely related local task was learned by local search. As shown in Fig. 5(c), one of the processing nodes was assigned to predict the angular velocity; therefore, the

time spent on learning was shortened. Table II shows that, when the local task is changed from predicting both velocities (Algorithm V) to predicting the angular velocity (Algorithm IV), the overall time taken becomes longer, even though the time spent on learning is shortened.[7] This suggests that, when the local task is loosely related to the global task, the ability to evolve solutions for the global task might be reduced. If the local task is "unrelated" to the global task, it is likely that no improvement can be obtained, and it may deteriorate the evolution process in the worst case. One can easily find a local task, forcing the output of a processing node to be a constant, for example, that cannot improve the evolution of solutions for the global task. Therefore, we suggest that one should choose a local task that is closely related to the global task.

VI. Discussion and Conclusion

Local search methods are usually fast, but they have difficulties in avoiding local optima. Evolutionary search methods have the capability to escape from local optima, but their computation time could be long. We have found that it is possible for these search methods to complement each other, and to yield a better training algorithm for RNN's. In the long-term dependency problem and the inverted pendulum problem, embedding the delta rule in the form of lifetime learning is able to speed up and improve the accuracy of the training process.

When evolutionary search is combined with local search, the computational complexity of the local search determines the optimal frequency at which it should be applied. When the complexity is low, it is possible to apply local search at every generation, leading to the lifetime learning approach. Another factor that affects the frequency of applying local search is the evolutionary search's ability to eliminate the regions not containing the global optimum. Hart [12] found that the frequency should be reduced when the fitness distribution of the population reliably indicates the possible locations of the global optimum; otherwise, little benefit can be obtained from local search.

Between the two local search methods that we have investigated, the delta rule is the simplest, although it is not able to find a good solution on its own. However, its low computational complexity makes it suitable for being embedded in the cellular GA. We have demonstrated that good performance can be achieved by embedding it in evolutionary search, suggesting that local search methods need not be sophisticated in order to obtain the benefit of combining evolutionary search and local search. Other researchers [11], [21] also demonstrate that the combination of "simple" local search and evolutionary search can yield better performance as compared to evolutionary search alone. However, the local search method adopted in [11] has limitations, as it is only applicable to simple Boolean networks with binary rather than floating-point weights. As the parity problem used in [11] is simple enough to be solved by local search alone, good results were expected when it was combined with evolutionary search. In contrast, we have shown in Section IV-A that applying the delta rule alone is not able to solve the long-term dependency problem, and that the problem is solved whenever the delta rule is embedded in evolutionary search. Although improvement in evolutionary search was observed in [21], the author did not consider the time spent on learning. Our results, however, clearly demonstrate the benefit of embedding local search in evolutionary search, even if the learning time is taken into account.

For the inverted pendulum problem, local search aims at producing better predictions, while evolutionary search aims at balancing the system. We found that learning the local task is able to improve the evolution of solutions for the global task. While Nolfi *et al.* [24] observed this phenomenon, they did not consider the learning time involved. It is therefore uncertain whether the improvement was achieved at the expense of longer training time. Only limited comparisons between the performance of the global task and the local task were provided in [24]; in particular, not much information regarding the performance of the local task during the course of evolution was given. Without these comparisons and information, the reasons why learning a local task can improve the evolution process remain unclear. Would it be the case that learning a local task, or any arbitrary tasks, is a kind of mutation that increases the exploration of the search space, which in turn improves the evolution process? Our comparisons between the performance of the global task and that of the local task during the course of evolution, as shown in Fig. 7, provide a clearer answer to this question. We show that the evolution process is improved because the performance of the local task (achieved by Lamarckian learning) is good.

We have also found that the interaction between the local task and the global task can affect the evolution process. We suggest that when the two tasks are related, especially when there is an overlap between the regions of optimal solution for the local task and for the global task, learning the local task is able to improve the evolution of solutions for the global task. We have shown that, when the local task is not directly related to the global task, the local task has a smaller contribution to the evolution process. Therefore, one of the criteria for choosing a local task is that it should be related to the global task, where the term "related" means that it is easier to accomplish the global task when the local one is resolved.

Other reports [13], [14] demonstrate that the performance of a target task can be improved even if an arbitrary task rather than the target task is learned. We have not found any evidence in our experiments to support this argument. Neither have we found evidence to disprove this claim. In fact, one of our experiments shows that learning an unsuitable local task deteriorates the evolution process, and one can easily find a local task

[7]Comparing Algorithms IV and I, the former requires a smaller number of generations, but takes a longer training time. This is because the reduction in the number of generations cannot compensate for the additional time spent on learning.

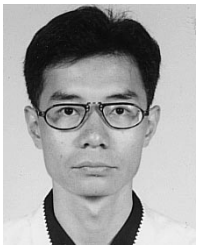
that cannot improve the evolution process. It has also been remarked in [14] that only under very restrictive conditions (e.g., real-valued genotypes, a small amount of learning), can a benefit be obtained by learning an arbitrary task. Therefore, for practical purposes, it is unlikely to obtain any improvement by learning an "unrelated" task.

ACKNOWLEDGMENT

The authors would like to thank D. B. Fogel and the anonymous reviewers for their careful reading of this paper, and for their constructive comments which significantly improved the quality of the contribution.

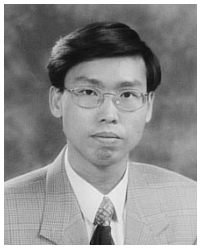
REFERENCES

[1] D. H. Ackley and M. L. Littman, "A case for Lamarckian evolution," in *Artificial Life*, C. G. Langton, Ed. Reading, MA: Addison-Wesley, 1994, pp. 3–10.
[2] C. W. Anderson, "Learning to control an inverted pendulum using neural networks," *IEEE Contr. Syst. Mag.*, vol. 9, pp. 31–37, 1989.
[3] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.
[4] J. M. Baldwin, "A new factor in evolution," *Ameri. Naturalist*, vol. 30, pp. 441–451, 1896.
[5] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.
[6] M. Bianchini, M. Gori, and M. Maggini, "On the problem of local minima in recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 167–177, 1994.
[7] B. Cohen, D. Saad, and E. Marom, "Efficient training of recurrent neural network with time delays," *Neural Networks*, vol. 10, no. 1, pp. 51–59, 1997.
[8] R. J. Collins and D. R. Jefferson, "Selection in massively parallel genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 249–256.
[9] Y. Davidor, "A naturally occurring niche & species phenomenon: The model and first results," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 257–262.
[10] D. B. Fogel, "Using evolutionary programming to create neural networks that are capable of playing tic-tic-toe," in *Proc. Int. Conf. Neural Networks*, 1993, pp. 875–880.
[11] F. Gruau and D. Whitley, "Adding learning to the cellular development of neural networks: Evolution and the Baldwin effect," *Evol. Comput.*, vol. 1, no. 3, pp. 213–233, 1993.
[12] W. E. Hart, "Adaptive global optimization with local search," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. California, San Diego, 1994.
[13] I. Harvey, "Unicycling helps your French: Spontaneous recovery of associations by learning unrelated tasks," *Neural Comput.*, vol. 8, pp. 697–704, 1996.
[14] ——, "Is there another new factor in evolution?," *Evol. Comput.*, vol. 4, no. 3, pp. 313–329, 1997.
[15] G. K. Kanji, *100 Statistical Tests*. Bevery Hills, CA: Sage, 1993.
[16] K. W. C. Ku and M. W. Mak, "Exploring the effects of Lamarckian and Baldwinian learning in evolving recurrent neural networks," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1997, pp. 617–621.
[17] ——, "Empirical analysis of the factors that affect the Baldwin effect," in *Parallel Problem Solving from Nature—PPSN V*, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 481–490.
[18] K. W. C. Ku, M. W. Mak, and W. C. Siu, "Adding learning to cellular genetic algorithms for training recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 10, no. 2, pp. 239–252, 1999.
[19] T. Lin, B. G. Horne, P. Tiňo, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 7, no. 6, pp. 1329–1338, 1996.
[20] B. Manderick and P. Spiessens, "Fine-grained parallel genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithm*, 1989, pp. 428–433.
[21] V. Maniezzo, "Genetic evolution of the topology and weight distribution of neural networks," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 39–53, 1994.
[22] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Mach. Learn.*, vol. 22, pp. 11–32, 1996.
[23] M. C. Mozer, "A focus backpropagation algorithm for temporal pattern recognition," *Complex Syst.*, vol. 3, pp. 349–381, 1989.
[24] S. Nolfi, J. L. Elman, and D. Parisi, "Learning and evolution in neural networks," *Adaptive Behavior*, vol. 3, pp. 5–28, 1994.
[25] D. Parisi and S. Nolfi, "The influence of learning on evolution," in *Adaptive individuals in Evolving Populations: Models and Algorithms*, R. K. Belew and M. Mitchell, Eds. Reading, MA: Addison-Wesley, 1996, pp. 419–428.
[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distribution Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundation*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: M.I.T. Press, 1986.
[27] Y. Shang and B. W. Wah, "Global optimization for neural networks training," *IEEE Computer*, vol. 29, no. 3, pp. 45–54, 1996.
[28] Z. Tang and G. J. Koehler, "Deterministic global optimal FNN training algorithms," *Neural Comput.*, vol. 7, no. 2, pp. 300–311, 1994.
[29] P. Turney, "Myths and legends of the Baldwin effect," in *Proc. Workshop Evol. Comput. and Machine Learning, 13th Int. Conf. Machine Learning*, 1996, pp. 135–142.
[30] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, Oct. 1990.
[31] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, 1994.
[32] ——, "Genetic algorithms and neural networks," in *Genetic Algorithms in Engineering and Computer Science*, G. Winter, J. Periaux, M. Galan, and P. Cuesta, Eds. New York: Wiley, 1995, pp. 191–201.
[33] D. Whitley, S. Dominic, and R. Das, "Genetic reinforcement learning with multilayer neural networks," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 562–569.
[34] D. Whitley, V. S. Gordon, and K. Mathias, "Lamarckian evolution, the Baldwin effect and function optimization," in *Parallel Problem Solving from Nature—PPSN III*, Y. Davidor, H.-P. Schwefel, and R. Manner, Eds. Berlin, Germany: Springer-Verlag, 1994, pp. 6–15.
[35] A. Wieland, "Evolving neural network controllers for unstable systems," in *Proc. Int. Joint Conf. Neural Networks*, 1991, pp. 667–673.
[36] R. J. Williams and D. Zipser, "Experimental analysis of the real-time recurrent learning algorithm," *Connect. Sci.*, vol. 1, pp. 87–111, 1989.
[37] X. Yao, "Evolutionary artificial neural networks," *Int. J. Neural Syst.*, vol. 4, no. 3, pp. 203–222, 1993.
[38] ——, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, pp. 1423–1447, Sept. 1999.



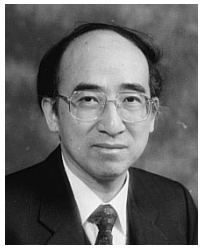
Kim W. C. Ku (M'95) received the B.Sc.(Hons) degree in computer studies from the City Polytechnic of Hong Kong in 1993, the M.Sc. degree in information technology (Knowledge Based Systems) from the University of Edinburgh in 1994, and the Ph.D. degree from The Hong Kong Polytechnic University in 1999.

Presently, he is a Lecturer in the Department of Computer Science, City University of Hong Kong.



Man Wai Mak (S'91–M'93) received the B.Eng.(Hons) degree in electronic engineering from Newcastle Upon Tyne Polytechnic in 1989 and the Ph.D. degree in electronic engineering from the University of Northumbria at Newcastle in 1993.

He was a Research Assistant at the University of Northumbria at Newcastle, from 1990 to 1993. He joined the Department of Electronic Engineering at the Hong Kong Polytechnic University as a Lecturer in 1993 and as an Assistant Professor in 1995. Since 1995, he has been an executive committee member of the IEEE Hong Kong Section Computer Chapter. He is currently the Secretary of the IEEE Hong Kong Section Computer Chapter. His research interests are neural networks, genetic algorithms, and speaker recognition.



Wan-Chi Siu (S'77–M'77–SM'90) received the Associateship from the Hong Kong Polytechnic University (formerly called Hong Kong Polytechnic), the M.Phil. degree from the Chinese University of Hong Kong and the Ph.D. degree from the Imperial College of Science, Technology & Medicine, London, in 1975, 1977 and 1984, respectively. He was with the Chinese University of Hong Kong between 1975 and 1980. He then joined the Hong Kong Polytechnic University as a Lecturer in 1980 and was Chair Professor & Associate Dean of Engineering Faculty between 1992–94. Prof. Siu has been Chair Professor & Head of Department of Electronic and Information Engineering of the same university since 1994. Professor Siu has published over 180 research papers. His research interests include digital signal processing, fast computational algorithms, transforms, video coding, computational aspects of image processing and pattern recognition, and neural networks.

Professor Siu is a Member of the Editorial Board of the Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology, and an overseas member of the Editorial Board of the IEE Review. He was a Guest Editor of the Special Issue on ISCAS'97 of the IEEE Transactions on Circuits and Systems, Pt.II, published in May 1998. He was also an Associate Editor of the IEEE Transactions on Circuits and Systems, Pt.II between 1995–97. Professor Siu was the General Chairman of the International Symposium on Neural Networks, Image and Speech Processing (ISSIPNN'94), and a Co-Chair of the Technical Program Committee of the IEEE International Symposium on Circuits and Systems (ISCAS'97) which were held in Hong Kong in April 1994 and June 1997, respectively. Prof. Siu is the General Chair of the 2003 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) which is to be held in Hong Kong. Between 1991 and 1995, Prof. Siu was a member of the Physical Sciences and Engineering Panel of the Research Grants Council (RGC), Hong Kong Government, and in 1994 he chaired the first Engineering and Information Technology Panel to assess the research quality of 19 Cost Centers (departments) from all universities in Hong Kong.

Professor Siu is a Chartered Engineer, a Fellow of the IEE and the HKIE, and a Senior Member of the IEEE, and has also been listed in Marquis Who's Who in the World, Marquis Who's Who in Science and Engineering and other citation biographies.