(0)

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

AD-A281 633

ated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and ollection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including arters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA rwork Reduction Project (0704-0188), Washington, DC 20503.

	2. REPORT DATE May 1994	3. REPORT TYPE AND DATES COVERED Professional Paper
4. TITLE AND SUBTITLE EVOLVING RECURRENT PERCEPTRONS FOR TIME–SERIES MODELING		5. FUNDING NUMBERS PR: ZW67 PE: 0601152N WU: DN303002
6. AUTHOR(S) J. R. McDonnell and D. E. Waagen		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Command, Control and Ocean Surveillance Center (NCCOSC) RDT&E Division San Diego, CA 92152–5001		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Chief of Naval Research Arlington, VA 22217–5000		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE

94-20692

17Pg

13. ABSTRACT (Maximum 200 words)

This paper describes evolutionary programming, a systematic multi-agent stochastic search technique, used to generate recurrent perceptrons (nonlinear IIR filters). A hybrid optimization scheme is proposed that embeds a single-agent stochastic search technique, the method of Solis and Wets, into the evolutionary programming paradigm. The proposed hybrid optimization approach is further augmented by "blending" randomly selected parent vectors to create additional offspring. The first part of this work investigates the performance of the suggested hybrid stochastic search method. After demonstration on the Bohachevsky and Rosenbrock response surfaces, the hybrid stochastic optimization approach is applied in determining both the model order and the coefficients of recurrent perceptron time–series models. An information criterion is used to evaluate each recurrent perceptron structure as a candidate solution. It is speculated that the stochastic training method implemented in this study for training recurrent perceptrons can be used to train perceptron networks that have radically recurrent architectures.

Published in *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 24–38, January 1994.

14. SUBJECT TERMS evolutionary programming neural networks signal detection			15. NUMBER OF PAGES 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAME AS REPORT

94 7 6 192

UNCLASSIFIED

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE (Include Area Code)	21c. OFFICE SYMBOL
J. R. McDonnell	(619) 553-5762	Code 731

Accession For	
NTIS GRA&I	[✓]
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	

Dist	Avail and/or Special
A-1	20

NSN 7540-01-280-5500

Standard form 298 (BACK)

UNCLASSIFIED

Evolving Recurrent Perceptrons for Time-Series Modeling

J. R. McDonnell, *Member, IEEE*, and D. Waagen

***Abstract*—Evolutionary programming, a systematic multi-agent stochastic search technique, is used to generate recurrent perceptrons (nonlinear IIR filters). A hybrid optimization scheme is proposed that embeds a single-agent stochastic search technique, the method of Solis and Wets, into the evolutionary programming paradigm. The proposed hybrid optimization approach is further augmented by "blending" randomly selected parent vectors to create additional offspring. The first part of this work investigates the performance of the suggested hybrid stochastic search method. After demonstration on the Bohachevsky and Rosenbrock response surfaces, the hybrid stochastic optimization approach is applied in determining both the model order and the coefficients of recurrent perceptron time-series models. An information criterion is used to evaluate each recurrent perceptron structure as a candidate solution. It is speculated that the stochastic training method implemented in this study for training recurrent perceptrons can be used to train perceptron networks that have radically recurrent architectures.**

I. Introduction

Artificial neural networks with recurrent connections represent an alternative to feedforward networks for nonlinear models of time-series data. Feedback connections allow for the feedforward information to be distributed back into the network, and may result in increasingly complex nonlinear manifolds with an increasing order of recurrency. This work demonstrates the application of the evolutionary search method in "evolving" simple recurrent perceptrons that may serve as building blocks for more complicated structures. Once feasibility is demonstrated for simple recurrent perceptron structures, the evolutionary search method can then be applied to highly recurrent perceptron networks with complex architectures. Stochastic methods are an attractive training option for complicated architectures because they are not constrained to a specific network topology. This feature allows both the network structure and weights to be determined during the training process.

Simultaneously determining both perceptron weights and structure requires a procedure that is amenable to combinatorial optimization problems. Successful algorithms for these types of problems have generally been stochastic search techniques such as simulated annealing [1], genetic algorithms [2], and evolutionary programming [3]. The evolutionary programming (EP) paradigm has been shown to have the desired attributes: combinatorial optimization capabilities [4], the ability to determine model structure [5], and the ability to train neural networks [6].

Manuscript received March 31, 1993; revised July 7, 1993 and August 9, 1993.

J. R. McDonnell is with NCCOSC, RDT&E Division, San Diego, CA.
D. Waagen is with TRW, System Integration Group, Ogden, UT.

IEEE Log Number 9213943.

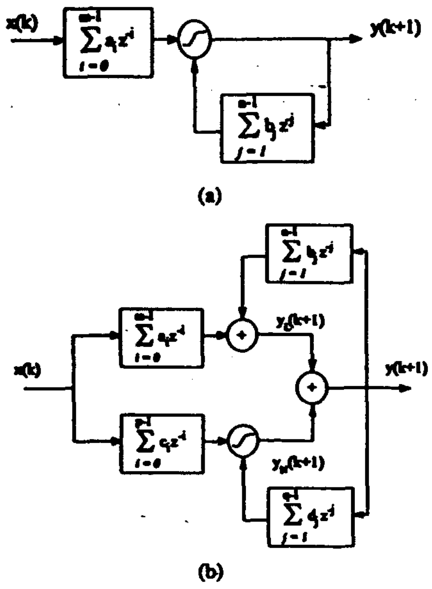


Fig. 1. (a) The nonlinear IIR filter structure. (b) The parallel linear-nonlinear IIR filter structure. Evolutionary programming can adapt any activation function, and, since the model order is determined during the search, transversal structures may result.

The "perceptron" in this study refers to a recursive adaptive filter with an arbitrary output function. Fig. 1(a) shows the proposed perceptron structure or nonlinear IIR filter. Fig. 1(b) shows a linear-nonlinear architecture, although, as will be seen in the later studies, the linear activation function could be replaced by one that is nonlinear. This recurrent perceptron model is inspired by the structure of infinite impulse response (IIR) filters and is postulated for time-series modeling. This work applies an evolutionary or systematic multi-agent stochastic search to determine the order of the recurrent perceptron structure as well as the tapped-delay line weights. Modifications to the perceptron topology are accomplished by either increasing or decreasing the number of tapped delays on the input or feedback lines, respectively. These structural modifications are limited to a random change of plus or minus one tapped delay on a randomly selected line. Since the model is determined during training, a possibility exists that nonlinear finite impulse response (FIR) perceptrons will result should the feedback order become zero.

Williams [7] characterizes recurrency based on its utilization in a connectionist architecture. *Conservative recurrence* corresponds to a tapped-delay input signal and is termed a *transversal filter network*. This approach yields a network that is sensitive to temporal patterns without directly incorporating recurrent units. Transversal filter networks have been widely applied in the field of speech recognition (e.g., Waibel *et al.* [8]). *Liberal recurrence* is the feedback from the output to the input units and corresponds to a nonlinear multi-input, multi-output (MIMO) IIR filter. Williams assigns the term *recursive filter* to transversal filter networks with tapped-delays on the output line feeding back as inputs. *Radical recurrence* refers to recurrent networks that model systems with strongly hidden states. By definition, if a system has a weakly visible state, it can be modeled with either the transversal or recursive filter networks. The radical approach allows coupling effects that are not possible with the more traditional transversal and recursive filter approaches. Both structures shown in Fig. 1 have liberal recurrence with the potential to reduce to conservative recurrence during the evolutionary training process.

Feedforward networks have been successfully used for both time-series prediction and system modeling, generally using tapped-delay, or transversal filter, network structures. However, these networks are not necessarily the typical feedforward configuration trained solely by error backpropagation. For example, a Connectionist Normalized Linear Spline Network (CNLS) has been formulated by the Center for Nonlinear Studies at Los Alamos National Laboratory [9–11]. Pruning connections [12–13] or weight sharing [14] can improve generalization capabilities as well as increase processing throughput, since architecture size is reduced. Using only the most recent observation, Rao and Ramamurti [15] generate a radically recurrent network based upon a cascade-correlation [16] approach.

Saravanan [17] utilizes a purely recurrent structure so that next-step estimates are only a function of previous estimates. This network is trained using evolutionary search methods Recurrent neural network structures have also been successfully trained using EP by McDonnell and Waagen [18] and Angeline *et al.* [19]. Other types of recursive structures that have been evolved include finite state machines [3] and the order and coefficients of ARMA models [5], [20]. While "optimal prediction can be thought of, quite simply, in terms of optimal filtering in absence of measurements" [21], practical applications make use of recent observations. Li and Haykin [22] and McDonnell and Waagen [18] utilize both a window of observations and a window of previous estimates for nonlinear time-series prediction. If an event occurs which precludes making an observation, then substitution of the estimate for past observations may suffice, depending on the accuracy of the model and noise levels.

The combination of more efficient local search methods with global techniques is appealing. As Yao [23] states, "the efficiency of evolutionary training can be improved by incorporating a local search procedure into the evolution." However, this requirement may limit the applicability of the global search method to specific types of architectures since local search techniques are somewhat restrictive. To alleviate this concern and maintain the integrity of the stochastic search, only direct search methods are considered for being embedded into EP. This rationale was successfully used in the development of the *stochastic direction* algorithm by Waagen *et al.* [24].

Before discussing the optimization of recurrent perceptrons, Section II describes and demonstrates a hybrid optimization approach that combines the Solis and Wets random optimization technique and EP. Variants of both methods are applied to finding extrema of an unknown function. A hybrid strategy is subsequently developed that embeds the Solis and Wets technique within EP. Once the hybrid approach has been successfully demonstrated, the recurrent perceptron structure is discussed in Section III. Results are then given in Section IV for evolving recurrent perceptron next-step predictors for a variety of time-series data.

II. Multi-Agent Stochastic Search

A. Benefits of Stochastic Optimization

As a direct search method, stochastic optimization does not require derivatives of the objective function nor continuity of the response surface [25]. The advantages of random search methods include ease of implementation, insensitivity to the type of criterion function, efficiency, flexibility, and the generation of information about the response surface [26]. Efficiency refers to the allocation of resources for evaluating additional points on the response surface versus deciding which point to evaluate next. Of course, this can be detrimental if the criterion function requires an extensive amount of computation. If it is computationally expensive to evaluate the objective function, then the information generated during the course of the stochastic search can be used to direct the search procedure. Pierre [27] stipulates that the following search evaluation criteria should be considered before selecting any particular optimization strategy: "1) How much computational equipment is required? 2) Has the search technique proved to be completely successful on similar types of performance measures? 3) What accuracy is required of the search? 4) What is a fair measure of the cost of the search? 5) How will the time spent in evaluating the performance measure and its derivative, if used, compare with the time spent on other aspects of the search?"

In response to these issues, some generalizations may be made with respect to evolutionary search strategies. 1) The computational resources must provide sufficient memory and processor power to conduct N separate searches, since evolutionary methods are based on multi-agent search strategies. Most implementations occur on serial platforms even though multi-agent search strategies are inherently parallelizable. Considerable computational resources may be required if the problem has an extremely high dimensionality. 2) As previously discussed, evolutionary search strategies are an excellent means to solve combinatorial optimization problems and discover globally optimal solutions. 3) The issue of accuracy has ramifications with respect to *a priori* knowledge of the response surface. If a correct model structure is assumed, evolutionary search strategies will, in general, tend to be slower than traditional optimization schemes. This results from the inefficiency of not using information about the gradient (although gradient methods can be incorporated in parallel with the evolutionary search strategy). However, the time complexity for evolutionary search does not necessarily increase dramatically with increased dimensionality [28] or additional constraints. In sum, evolutionary search strategies are robust across a broad spectrum of problem domains. 4) The number of function evaluations is a useful metric for comparing evolutionary search strategies. Time complexity or accuracy may serve as a useful metric for comparisons with other search methods. 5) The matter of efficiency is discussed in the previous paragraph.

B. Single-Agent Stochastic Search

Random optimization has traditionally been based on single-agent stochastic search (SASS) strategies. Both Rao [25] and Karnop [26] generate a random walk sequence to an extremum by perturbing the search point with a uniform random variable. Rao also exploits the directionality of the randomly generated vectors that continue to yield lower valued objective functions. In an algorithmic formulation similar to that of Rao, Matyas utilizes Gaussian perturbations about the search point along with a bias term to direct the search [29]. Solis and Wets [30] have enhanced this approach by evaluating the objective function at $x - \delta x$ if evaluation at $x + \delta x$ does not improve the current value of the objective function and by incorporating a variable perturbation variance. The bias and additional function evaluation serve as stochastic equivalents to incorporating momentum and gradient information. Baba has successfully applied the method of Solis and Wets to training feedforward networks to predict SO_2 concentrations in air [31].

Algorithm 1 from Solis and Wets [30] was used in the studies presented here. The variance of the perturbation size ξ is controlled by the repeated number of successes, *scnt*, or failures, *fcnt*, in decreasing the objective function f. The contraction ct and expansion ex constants, as well as the upper and lower bounds on standard deviation of the random perturbations σ are set by the user. The Algorithm 1 variant gives the basic Solis and Wets method global optimization capability by increasing the standard deviation of the perturbation when it falls below an arbitrary lower bound (see step 2 below). The formulation is described as follows

1. *Initialize the search vector* $\mathbf{x}_0$ *and bias vector* $\mathbf{b}_0 = 0$. *Set* $k = 0$, *scnt=0, fcnt=0. Fix ex, ct, Scnt, Fcnt,* σ_{ub}, σ_{lb} *and initialize* $\sigma_o = 1$.
2. $$\text{Set } \sigma_k = \begin{cases} ex \cdot \sigma_{k-1} & if \quad scnt > Scnt \\ ct \cdot \sigma_{k-1} & if \quad fcnt > Fcnt \\ \sigma_{ub} & if \quad \sigma_{k-1} < \sigma_{lb} \\ \sigma_{k-1} & \quad otherwise \end{cases}$$
3. *Generate a multivariate Gaussian random vector* $\xi_k \sim N(\mathbf{b}_k, \sigma\mathbf{I})$.

4(a). *If* $f(\mathbf{x}_k + \xi_k) < f(\mathbf{x}_k)$, *then set* $\mathbf{x}_{k+1} = \mathbf{x}_k + \xi_k$ *and* $\mathbf{b}_{k+1} = 0.4\xi_k + 0.2\mathbf{b}_k$, *scnt=scnt+1, fcnt=0.*

4(b). *Otherwise, if* $f(\mathbf{x}_k - \xi_k) < f(\mathbf{x}_k) < f(\mathbf{x}_k + \xi_k)$, *then set* $\mathbf{x}_{k+1} = \mathbf{x}_k - \xi_k$ *and* $\mathbf{b}_{k+1} = \mathbf{b}_k - 0.4\xi_k$, *scnt=scnt+1, fcnt=0.*

4(c). *Otherwise,,* $\mathbf{x}_{k+1} = \mathbf{x}_k$ *and* $\mathbf{b}_{k+1} = 0.5\mathbf{b}_k$, *fcnt=fcnt +1, scnt=0.*

5. *If* k = *maximum number of iterations then stop, else* $k = k + 1$ *and go to Step* 2.

The coefficient values 0.4 and 0.2 are retained from Solis and Wets' results [30]. Note that the conditions in Step 2 are not mutually exclusive. The standard deviation σ specifies the size of the sphere that most likely contains the perturbation vector, and the bias term b locates the center of the sphere based on directions of past success. Step 4(b) implements a reversal strategy seeking a better solution in the direction opposite to that of initial perturbation.

Optimization experiments were conducted to find the point $(x_1, x_2,)$ which minimizes the Bohachevsky [32] function $f(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$. The standard deviation was initialized as $\sigma_0 = 1.0$, and $\mathbf{x}$ was initially sampled in the region $\mathbf{x} \in [-25, 25]^2$ for all the experiments conducted. The transcendental terms generate many local minima within the region $\mathbf{x} \in [-1, 1]^2$ while the quadratic terms dominate the surface structure outside this interval. A unique global minimum exists at $\mathbf{x} = (0, 0)$. The first set of experiments consisted of using the Solis and Wets' algorithm outlined above. The second set of experiments employed the same algorithm, except that the bias term was not used. A third set of experiments employed Gaussian perturbations having a standard deviation proportional to the magnitude of the objective function such that $\xi \sim N(\mathbf{b}, f(\mathbf{x})\mathbf{I})$. The final set of experiments did not incorporate a bias term so that $\xi \sim N(0, f(\mathbf{x})\mathbf{I})$. The variance modification parameters are the same as those reported in [30]: $ex = 2, ct = 0.5, Scnt = 5$, and $Fcnt = 3$. The upper and lower bounds on the standard deviation were set as $\sigma_{ub} = 1.0$ and $\sigma_{lb} = 0.00001$, respectively. The average results of 10 trials are shown in Fig. 2. Based on these experiments of low dimensionality, it appears that the accuracy of the extremum point may be improved significantly if the standard deviation of the random perturbations is allowed to expand and contract independently of the response surface height. Also, roughly an order of magnitude improvement in the cost function was observed using a random perturbation $\xi \sim N(0, \sqrt{f(\mathbf{x})\mathbf{I}})$ as opposed to $\xi \sim N(0, f(\mathbf{x})\mathbf{I})$. Modification of the perturbation size remains an active area of research, as exemplified by work in evolution strategies [33] and meta-EP [34].

As successful as the basic Solis and Wets algorithm appears, search surfaces may be encountered for which global optimization is not practical if σ_{ub} is continually less than some critical standard deviation that guarantees one can tunnel from any point on the search surface to an extremum with reasonable likelihood. To reduce the occurrence of entrapment conditions encountered in SASS strategies, it is suggested that the Solis and Wets random optimization method be embedded in a multi-agent stochastic search such as EP. Even if σ_{ub} is not constrained, a multi-agent search technique will provide a more rigorous search over high-dimensional spaces.

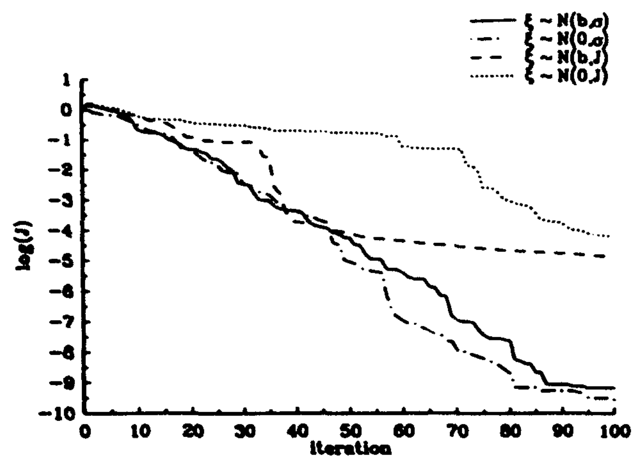


Fig. 2. A comparison of variants of the Solis and Wets random optimization method as applied to the Bohachevsky function and averaged over ten trials. It appears that higher accuracy is attained by allowing the variance of the random perturbations to expand and contract independent of the response surface height.

C. The Evolutionary Programming Paradigm

In 1958, Brooks [35] described a *creeping random method* where k points were generated via Gaussian perturbations about a search point. The best point was kept and the process repeated. Brooks observed that "there are some rather intriguing analogies that can be made between the creeping random method and evolution." This analogy was also apparent to Fogel *et al.* [3] who proposed a population-based random search strategy termed *evolutionary programming* where, instead of keeping the single best point, a population of search points is maintained.

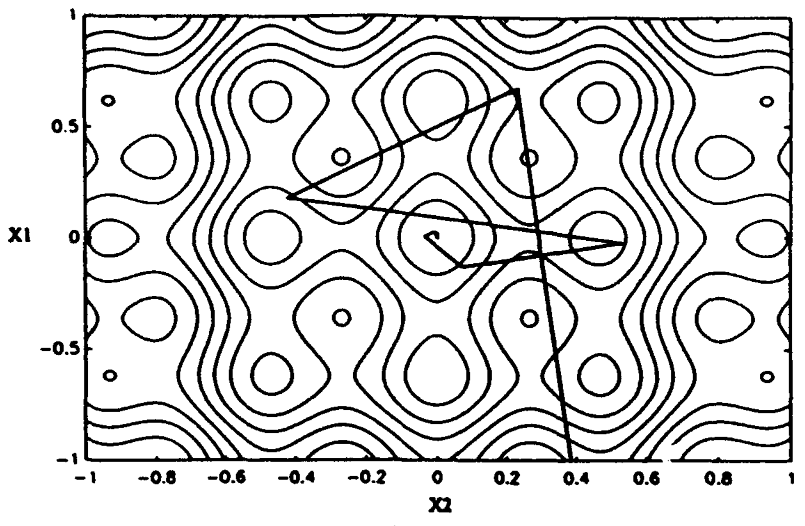
EP is a systematic multi-agent stochastic search (MASS) paradigm that can be used for finding global extrema on response surfaces. Although the EP methodology simulates the evolutionary process found in nature, the mechanisms incorporated in this framework and resulting characteristics may also be found in some of the stochastic optimization techniques previously discussed. Normally distributed perturbations are applied to the j^{th} element x_{ij} in the solution vector $\mathbf{x}_i$ according to $\delta x_{ij} \sim N(0, S_{f,ij} \cdot J_i + \beta_{ij})$ where $S_{f,ij}$ is the scale factor, J_i is the magnitude of the objective or criterion function corresponding to $\mathbf{x}_i$, and β_{ij} is an offset vector [5]. The scale factor can be considered as a probabilistic analog to the step-size used in gradient methods. Similar to the hill-climbing and tunneling ability of simulated annealing relaxation methods, EP employs a competition process that allows less fit organisms (search points) to be retained in the population in a probabilistic fashion. The competition process is viewed as a competitive annealing mechanism. An EP optimization algorithm similar to that in [5] is given below:

1. *Form an initial population* $\mathbf{P} = [\mathbf{x}_0\mathbf{x}_1\mathbf{x}_2 \cdots \mathbf{x}_{2N-1}]$ *of size 2N by randomly initializing each n-dimensional solution vector* $\mathbf{x}_i$. *A user-specified search domain* $\mathbf{x}_i \in [x_{\min}, x_{\max}]^n$ *may be imposed.*
2. *Assign a cost to each element* x_i *in the population based on the associated objective function* $J_i = \Phi(\mathbf{x}_i) s.t. \Phi : R^n \rightarrow R$.
3. *Reorder the population in descending order based on the number of wins generated from a stochastic competition process. Wins are generated by randomly selecting other members in the population* $\mathbf{x}_j$ *and incrementing the win counter* w_i *if* $J_i < J_j$.
4. *Generate offspring* $(\mathbf{x}_N \cdots \mathbf{x}_{2N-1})$ *from the N highest ranked elements* $(\mathbf{x}_0 \cdots \mathbf{x}_{N-1})$ *in the population by modifying each element* $x_{ij} \in \mathbf{x}_i$ *with a random perturbation* $\delta x_{ij} \sim N(0, S_{f,ij} \cdot J_i + \beta_{ij})$ *such that*
$$x_{i+N,j} = x_{ij} + \delta x_{ij}.$$
5. *Loop to Step 2.*

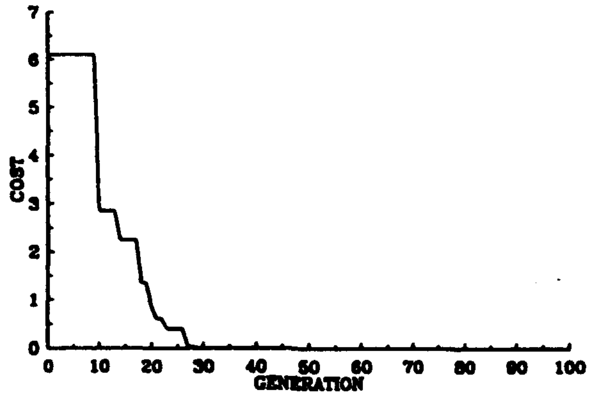
A trajectory of the best population member at each generation during a search on the Bohachevsky surface is superimposed on the Bohachevsky contours as shown in Fig. 3(a). Fig. 3(b) shows best cost in the population at each generation of the EP optimization process. The search is stochastic, so it is expected this trajectory will vary in every trial.

A variety of other techniques may be employed as alternatives to the methods given above. For example, each additional offspring can replace the least fit organism in the population, as is done by [17] and [36]. In a parallel implementation, Yip and Pao [37] generate a multitude of offspring from each parent and replace the parent with the best offspring in a probabilistic manner using simulated annealing. When the offspring are generated with structural modification(s), some level of parameter optimization should occur rapidly to reduce the occurrence of discarding good structures. One approach might be to allow these new, higher-cost members of the population to mature by modification of the objective function according to $J'(x,k) = \left(1 - e^{-(\tau \cdot k + \alpha)}\right) J(x)$ where the maturity level k of a population member could be determined by how many generations it has existed within the population and the parameters τ and α are user-specified. A deterministic means to minimize redundancy might also be employed to delay the potential dominance of a single member in the population. After all, only a single solution is required and convergence of all the members in the population to a single point reduces the effectiveness of the search process in exploring other portions of the search space. Retention of higher-cost search points can be done probabilistically by setting the number of competitions to an arbitrarily low value, thus allowing a more relaxed search. As the number of competitions increases, the retention of the more fit individuals becomes more deterministic. Fogel [34] discusses other variants of the EP paradigm.

The EP search outlined above was augmented with simple bisection search capabilities by averaging, or "blending," randomly chosen vectors according to $\mathbf{x}_0 = 0.5(\mathbf{x}_i + \mathbf{x}_j)$ where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the randomly chosen parent vectors selected from $\{\mathbf{x}_0, \cdots, \mathbf{x}_{N-1}\}$ and $\mathbf{x}_0$ is the offspring vector. This was done for half the offspring while the other half were generated using the perturbation approach $\delta x_i \sim N(0, J_i \cdot \mathbf{I})$ where the covariance matrix is an identity matrix scaled by the value of the objective function. The normal perturbations complement the averaging or bisection search method since it is unlikely that the best solution point exists on the line between two search points. Likewise, blending elements of the population complements the random walk procedure generated by the evolutionary search if it is assumed that the population



(a)



(b)

Fig. 3. Global optimization via evolutionary programming. (a) The trajectory of the best member in the population at each generation is superimposed on the contour plot of the Bohachevsky function with $S_f = 1$, 50 parents, 20 competitions. (b) The cost of the best member in the population at each generation. Since EP is a stochastic optimization process, the trajectory shown in (a) will undoubtedly vary for each trial.

points are distributed about an extremum point. The average results for 10 optimization trials on the Bohachevsky function are shown in Fig. 4. The next experiment decoupled the cost function from the perturbation size so that $\delta x_i \sim N(0, I)$. By decoupling the perturbation variance from the objective function, roughly an order of magnitude improvement in the best member of generation 100 was observed as shown in Fig. 4. Fogel [5] reports requiring an average of 65.5 generations over 20 trials to achieve $\log_{10}(f(x)) < -6$ using the same number of parents (50) and offspring (one per parent). By decoupling the cost function and implementing a simple bisection search, it took less than 30 generations, as averaged over 10 trials, to achieve similar results. The bisection search will not provide an advantage unless the global optimum is bounded by a portion of the population.

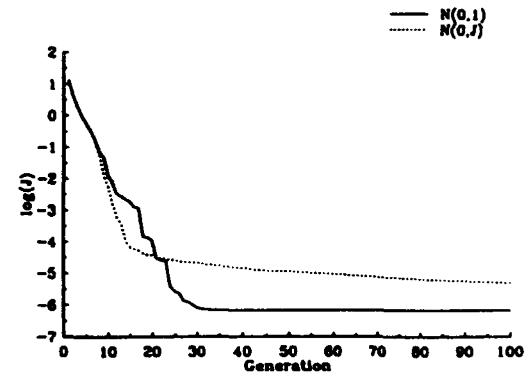


Fig. 4. Augmenting the EP search strategy with bisection search where the offspring are generated by averaging two randomly selected parent vectors. Half of the offspring were generated by mutation, the other half by averaging pairs of randomly selected parent vectors. The bisection search is useful when combined with multi-agent stochastic methods that distribute the population about a global extremum.

D. A Hybrid Approach

Single-agent stochastic search methods can be easily incorporated into EP without sacrificing the integrity of the evolutionary search procedure. A variant of EP is proposed to take advantage of the benefits offered by Solis and Wets and blending methods discussed in the preceding sections. It is speculated that different random optimization methods will prevail on different types of response surfaces. Based on its

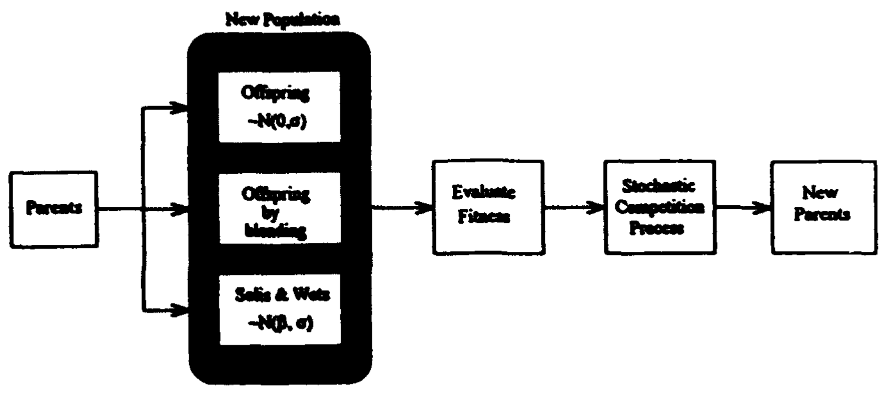


Fig. 5. One generation of the hybrid multi-agent stochastic search. The first set of offspring is generated by perturbing the parent vectors with $N(0, \sigma)$ random variables. The standard deviation can be tied to the height of the response surface so that $\sigma = S_f \cdot \text{cost}$. The second set of offspring is generated by an averaging or blending process. The third set of offspring is generated according to the Solis and Wets algorithm and deterministically replace the parents of a lower cost is achieved. Note that the variances in the offspring are generated by different methods and are not the same.

multi-agent search capabilities, EP is an attractive framework for combining a variety of stochastic search procedures. Although the previous experiments in a two-dimensional search space are not conclusive, the following properties appear potentially beneficial to a hybrid approach: 1) multi-agent search and variance expansion tend to avoid local minima, 2) information garnered during the search process about the response surface can be used to direct the search, and 3) convex optimization and perturbation variance reduction independent of the response surface height may improve accuracy. Making the perturbation variance proportional to the value of the cost function may not always yield optimal performance. Decoupling the perturbation variance from the cost function value may prove beneficial since the shape of the response surface is often not well known and may even take on negative values. A similar decoupling strategy was employed by Waagen *et al.* [24]. If the height of the response surface is known *a priori*, then the offset β (see Step 4 of the EP algorithm) in the standard deviation of the perturbations can be incorporated. Unfortunately, knowledge of the height of the response surface generally corresponds to determining the location of the global extrema. If cost functions for which the optimal value is zero, such as mean-squared error, are employed, then this issue is less significant.

Figure 5 illustrates a hybrid approach that employs different methods for offspring generation within EP. While parents are selected from the whole population in the usual fashion [5], the manner in which the offspring are generated is variable. The first set of offspring results from parent search points that are perturbed by a random vector $\delta \mathrm{x} \sim N(0, \sigma \cdot \mathrm{I})$ where σ can be fixed [24], proportional to the corresponding height of the response surface [5], or conditionally based on search performance [33]. The second set of offspring results from blending the parameters associated with a pair of randomly chosen parents. This may be as simple as averaging all of the elements in the search string. If the model structure is part of the search vector, then both the first and second set of offspring can easily accommodate changes in the model structure as well as the weights. The final set of offspring is generated using the method of Solis and Wets, which acts on the existing parent structures. Each offspring in this third set will replace its parent if the offspring has lower objective function than its predecessor. The type of convex optimization method applied to the second set of offspring may also be applied to other parameters, such as the bias vector b. For example, if the Solis and Wets bias term is included in the search string, then the first set of offspring is instantiated with $\mathrm{b} = 0$, while a member of the second set of offspring will have a bias vector determined using $\mathrm{b}_0 = 0.5 * (\mathrm{b}_i + \mathrm{b}_j)$, thereby taking the average of the bias vectors from two randomly selected parents. The other Solis and Wets parameters are instantiated in a similar manner, as are the structural parameters (i.e., model order, in this study). Although the Solis and Wets method could be repeatedly applied to the offspring, and has been done so with success, it is speculated that the different offspring strategies offer a more robust search as well as help to maintain diversity.

The average cost from 10 trials using the hybrid approach on the Bohachevsky function is shown in Fig. 6. The hybrid technique achieves an accuracy of 10 orders of magnitude within 50 generations (this corresponds to a maximum of 7550 function evaluations). In order to ascertain which optimization procedure was being utilized, a histogram was generated from two sample optimization runs on the Bohachevsky response surface as shown in Fig. 7. These runs contained 10 parent vectors and two sets of 10 offspring vectors generated via normal perturbations and blending, respectively. One run was made with fixed variance perturbation vectors $\xi \sim N(0, \mathrm{I})$; the second trial was conducted with the perturbation variance proportional to the cost function $\xi \sim N(0, J \cdot \mathrm{I})$. When $N(0, J \cdot \mathrm{I})$ perturbations were incorporated, the perturbation technique was the predominant beneficial search mode. When $N(0, \mathrm{I})$ perturbations were used, the Solis and Wets search technique provided most of the optimization capability. The averaging method rarely yielded the lowest cost member in the population.

Before dismissing the blending approach as inadequate, some comments should be made regarding these results. It

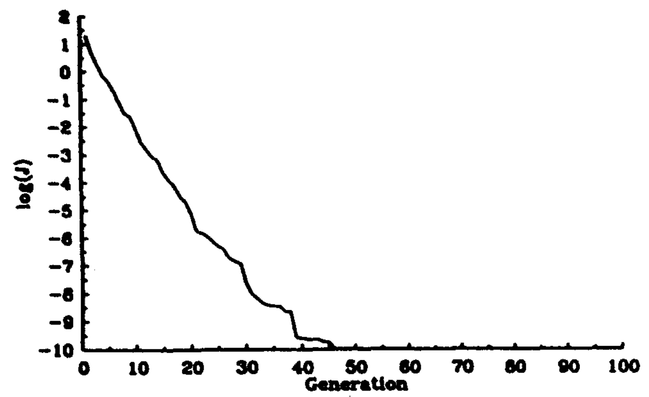


Fig. 6. Optimization of the Bohachevsky function using the hybrid search strategy with 50 parent points. The second set of offspring has a fixed perturbation variance $\sigma = 1$.

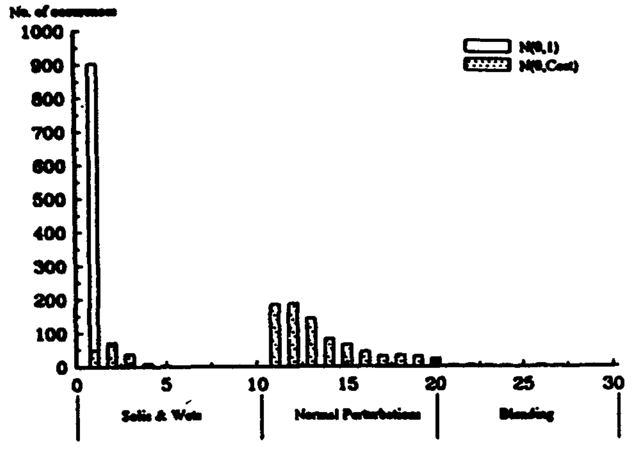


Fig. 7. A histogram that shows the frequency of the occurrence of the generation member with the lowest cost for the Bohachevsky response surface. The $N(0,1)$ perturbations are generally too large for the small global well, and optimization occurs primarily via the Solis and Wets technique. The $N(0,\text{cost})$ perturbations are small enough for optimization to occur within the global well. The blended or averaged set of offspring rarely occurs.

is expected that the top members (say, vectors 1–5) of the population will tend to be the best if, only by default, better solutions are not found. The larger perturbations will generate points outside the small diameter of the global well as observed in the $\xi \sim N(0, \mathrm{I})$ section of the histogram. This is also true in some respect to the Solis and Wets optimization procedure, since it was instantiated with a unit variance and discrete changes occur to the variance based on the performance of the search vector. When the Bohachevsky function was artificially elevated so that the global minimum had the corresponding cost $f(0,0) = 10$, the histograms for the two methods were virtually identical. Since subsequent investigations in evolving perceptron architecture relied heavily on the blending approach, the conclusion is drawn that implementing a variety of stochastic methods within EP provides a robust approach for the optimization of problems whose response surface is not well known. Incidentally, both of the runs that generated the histograms in Fig. 7 yielded nearly identical levels of cost after 1000 generations.

The Rosenbrock [38] function $f(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$ was chosen for comparing the hybrid approach to

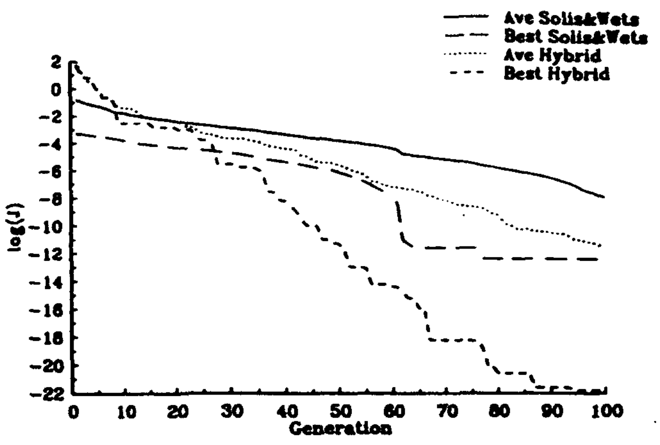


Fig. 8. Comparing optimization methods using Solis and Wets and the hybrid approach on the Rosenbrock response surface. Fifty parent points were used in the hybrid technique. Each generation corresponds to a maximum of 150 function evaluations for each method. The second set of offspring in the hybrid approach have a fixed perturbation variance of $\sigma = 1$.

the Solis and Wets random optimization method. This function has a unique global minimum at $\mathbf{x} = (1,1)$ and is referred to as a "banana valley" because it contains a steep valley along $x_2 = x_1^2$. Figure 8 shows the average and best results from 10 trials for both the Solis and Wets technique and the hybrid approach outlined above. In an effort to compare the search processes based on an equivalent number of maximum function evaluations, each generation equals a maximum of 150 function evaluations for the Solis and Wets method and a maximum of 150 function evaluations for the hybrid approach. Both average curves show optimization was still occurring after the maximum number of generations or iterations had been reached and the experiment was arbitrarily halted. These results compare favorably with generic EP results [5] where it is reported that it took an average of 86 generations (over 20 trials) to achieve an accuracy of $\log_{10}(f(\mathbf{x})) < -4$. It should be noted that the results generated from the Solis and Wets method, by itself, are also comparable. Now that the capabilities of the hybrid stochastic search have been demonstrated on well-known response surfaces. the next step is to determine its effectiveness in evolving simple recurrent perceptron structures similar to those shown in Fig. 1.

III. Evolving Recurrent Perceptrons

A. Motivation

The recursive structures shown in Fig. 1 are referred to as *recurrent perceptrons* because they incorporate nonlinearities on the output of a recursive linear combiner. The recurrent perceptron structure is inspired by recursive adaptive filters and the discrete time equation that models linear time-invariant (LTI) system dynamics

$$y(k+1) = \sum_{i=0}^{m-1} a_i x(k-i) + \sum_{i=0}^{n-1} b_i y(k-i)$$

where x represents the input to the system and y is the system output. While much is known about the stability, controllability, and observability of LTI systems, as well as methods for generating models of such systems, the same cannot be said for nonlinear systems in general [39]. Nonlinear versions of the identification models (motivated by LTI dynamics) given in Narendra and Parthasarathy [39] are described by

nonlinear parallel model:

$$\hat{y}(k+1) = f\left(\sum_{i=0}^{m-1} \hat{a}_i x(k-i) + \sum_{i=0}^{n-1} \hat{b}_i \hat{y}(k-i)\right)$$

nonlinear series-parallel model:

$$\hat{y}(k+1) = f\left(\sum_{i=0}^{m-1} \hat{a}_i x(k-i) + \sum_{i=0}^{n-1} \hat{b}_i y(k-i)\right)$$

where $\hat{y}$ is the estimate of the system output. The nonlinear series-parallel model is a transversal structure that utilizes the actual system outputs. Further, its linear counterpart is preferable for generating stable adaptive laws [39]. As stability is paramount when generating recursive filters, stable filters were always evolved using the hybrid stochastic search method implemented in this investigation.

The recurrent perceptron structures investigated in this study are characterized by the following difference equations

$$\begin{aligned}\textit{Class I:}\quad y(k+1) =& f[x(k), x(k-1), x(k-2), \cdots,\\ & x(k-m+1), y(k), y(k-1),\\ & y(k-2), \cdots, y(k-n+1)]\\ \textit{Class II:}\quad y(k+1) =& f[x(k), x(k-1), x(k-2),\\ & \cdots, x(k-m+1), y(k), y(k-1),\\ & y(k-2), \cdots, y(k-n+1)]\\ & + g[x(k), x(k-1), x(k-2), \cdots,\\ & x(k-m+1), y(k), y(k-1),\\ & y(k-2), \cdots, y(k-n+1)]\end{aligned}$$

where f and g are not necessarily the same mapping. The Class I is similar to the Model IV discrete time plant model given by Narendra and Parthasarathy [39] for nonlinear system identification and control, except that the nonlinear transformation in [39] is implemented with a multi-layer perceptron and the nonlinearity in this paper is accomplished using a single perceptron. Class II is similar to the Model III discrete time plant model given in [39], if the evolutionary search process yields an f that is dependent only on previous outputs $[y(k), y(k-1), \cdots, y(k-n+1)]$ and a g that is dependent only on past inputs $[x(k), x(k-1), \cdots, x(k-m+1)]$. The selection of these models in [39] was "motivated by the models that have been used in the adaptive systems literature for the identification and control of linear systems and can be considered their generalization to nonlinear systems."

B. Formulation

The recurrent perceptron model structure shown in Fig. 1(a) is characterized by the Class I difference equation and can be described by

$$\hat{y}(k+1) = f\left(\sum_{i=0}^{m-1} a_i x(k-i) + \sum_{i=0}^{n-1} b_i \hat{y}(k-i) + \alpha\right)$$

where the search strategy must determine the order of the feedforward terms, m, the order of the feedback terms, n, as well as the feedforward coefficients, a_i, the feedback coefficients, b_j, and the bias α. The IIR synapses proposed by Back and Tsoi [40] and the neurons used by Li and Haykin [22] have the same structure as this nonlinear parallel model.

Recalling that polynomials can also be used to approximate any static mapping $f : R^m \rightarrow R^n$ to an arbitrary degree of accuracy, and that the sigmoid function can be expressed as an inverted polynomial series

$$f(x) = (1+e^{-x})^{-1} \approx \left(1 + \sum_{i=0}^{n} \frac{(-x)^i}{i!}\right)^{-1}$$

leads to the suggestion that other nonlinear mappings that can also be expressed by polynomial series are equally applicable for use as activation functions [41], [42]. The stochastic search method used for training does not explicitly incorporate knowledge of the activation function (just I/O observations), so any activation function can be implemented without regard for continuity constraints. By virtue of their smoothness, continuous activation functions tend to possess good function approximation properties. The search may even be conducted over a set of candidate mapping functions F such that $f \in F$ thereby incorporating the selection of the activation function(s) in the evolutionary optimization process.

The objective function for each perceptron is similar to Akaike's minimum information theoretical criterion (AIC) estimate [43] as employed by Preistley [44] for evaluating autoregressive moving-average (ARMA) models

$$\mathrm{AIC(m,n) = Nln}(\hat{\sigma}_e^2) + 2(\mathrm{m+n}+1)$$

where N is the effective number of observations. An additional factor of 1 is added to the number of fitted parameters $(m+n)$ to account for the bias term α. The MLE of the innovation variance [44] is determined according to

$$\hat{\sigma}_e^2 = \frac{1}{N}\sum_{k=0}^{N-1} \hat{e}^2(k)$$

where the observation error is given by $\hat{e}(k) = y(k) - \hat{y}(k)$. To prevent the search process from driving the number of parameters to zero and stalling at a large MSE, the modification to the model order can take one of three states $(-1, 0, +1)$. Approximately 20% of the time, either of the conditions $(-1, +1)$ existed for a randomly selected tapped-delay line. Thus, if there are four tapped-delay lines, each line is being modified about 5% of the time. If a large number of tapped-delay lines are employed, then the percentage of time that any of the lines are affected may be increased to maintain a similar modification ratio.

If direct linear feedthrough [45] capabilities are desired to be present in parallel with the nonlinear contributions, then the perceptron structure can be reformulated as a combination of linear and nonlinear recurrencies. This combined structure corresponds to the Class II model where g is a linear functional as shown in Fig. 1(b) and is expressed by

$$\hat{y}(k+1) = \hat{y}_L(k+1) + \hat{y}_N(k+1)$$

where

$$\hat{y}_L(k+1) = \sum_{i=0}^{m-1} a_i x(k-i) + \sum_{i=0}^{n-1} b_i \hat{y}(k-i) + \alpha$$

and

$$\hat{y}_N(k+1) = f\left(\sum_{i=0}^{p-1} c_i x(k-i) + \sum_{i=0}^{q-1} d_i \hat{y}(k-i) + \beta\right)$$

If this structure or its variants are used, then the AIC score becomes

$$AIC(m,n,p,q) = N\ln(\hat{\sigma}_e^2) + 2(m+n+p+q+2)$$

Once an acceptable model has been determined, the evolutionary model building procedure can be iteratively applied to the residuals as discussed by PriestLey [44]. Deterministic training can also be applied to the evolved model in an effort to "fine tune" the model coefficients. Deterministic training was usually not applied to the nonlinear models generated in this work, either during or after the training process, and may potentially have yielded slightly better results.

C. Deterministic Training

More deterministic methods may be used to update the perceptron weights while the model structure is evolving, or they may be applied to the results of the stochastic search as a post-processing check to ensure local optimality. The recurrent perceptron model corresponding to Fig. 1(a) can be described by

$$y_{k+1} = f(\mathbf{w}^T \mathbf{z})$$

where

$$\mathbf{w}^T = [a_0\ a_1\ \cdots\ a_{m-1}\ b_0\ b_1\ \cdots\ b_{n-1}\alpha]$$
$$\mathbf{z}^T = [x_k\ x_{k-1}\ \cdots\ x_{k-m+1}\ y_k\ y_{k-1} \cdots y_{k-n+1}\ 1]$$

If the objective function is given by the instantaneous squared-error $E_k = e_k^2 = (z_k - y_k)^2$ where z_k is the desired output, then a straightforward gradient approach yields the well-known stochastic approximation weight update equation

$$\mathbf{w}_{new} = \mathbf{w} + \eta e_k f'(\mathbf{w}^T\mathbf{z})\mathbf{z}$$

Tsoi and Back [40] have derived a multi-layer perceptron version of this update scheme for nonlinear FIR and IIR perceptrons. Williams and Zipser [46] have also formulated a batch update scheme with an arbitrary lag window for recurrent multi-layer perceptrons.

IV. Prediction Results

A. The Simple Pendulum

Consider the equation for a simple pendulum with a velocity-squared damping term

$$J\ddot{\theta} + B\dot{\theta}|\dot{\theta}| + K\sin\theta = \tau$$

where $J = 1$, $B = 0.1$, and $K = 1$. For simplicity, the system was simulated using Euler integration with a stepsize of

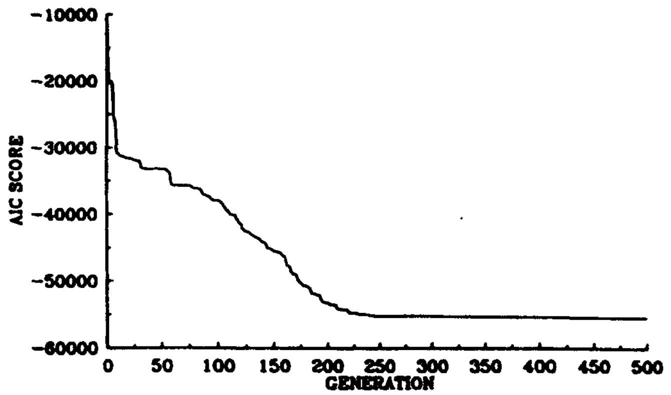


Fig. 9. Evolutionary optimization of the pendulum model. The lowest AIC score in the population is shown at each generation of the evolutionary search process.

0.05. A linear-nonlinear series-parallel model was postulated according to

$$\theta(k+1) = \sum_{i=0}^{m-1} a_i\tau(k-i) + \sum_{i=0}^{n-1} b_i\theta(k-i) + \sin\left(\sum_{i=0}^{p-1} c_i\tau(k-i) + \sum_{i=0}^{q-1} d_i\theta(k-i)\right)$$

where the maximum window size was limited to five samples. A $N(0,1)$ random forcing function was used to generate 50 000 samples, 5000 of which were used for training purposes. The search population consisted of 10 parents, each generating a single offspring using EP perturbations, as well as another set of 10 offspring by averaging randomly chosen parents.

The optimization process took place over 500 generations as shown in Fig. 9. The resulting model does not incorporate the forcing function, but instead relies only on past observations of the pendulum displacement as given by

$$\theta(k+1) = 0.87220\theta(k) + 0.7858\theta(k-1) - 0.4127\theta(k-2) - 0.3190\theta(k-3) + \sin(0.0687\theta(k))$$

Note that this approximation can be reduced to a purely linear system because of the small coefficient on the sin argument. The simulated system is almost linear by virtue of the small displacements and angular velocities. This model has a MSE=$1.54 \cdot 10^{-5}$ for the training data shown in Fig. 10. A test set was generated using $\tau(k) = 0.5\cos(2\pi k/1000)$ with the resulting MSE=$3.69 \cdot 10^{-6}$ on the testing data shown in Fig. 11.

B. The Sunspot Series

The second set of experiments was conducted on Wolf's sunspot series for the years 1700–1988. These numbers are indicative of the average relative number of sunspots observed each day of the year and serve as a standard benchmark for time-series modeling [12], [13], [40] where the objective is to generate a single-step prediction based on past observations. Consistent with Weigend *et al.* [12], the data set was

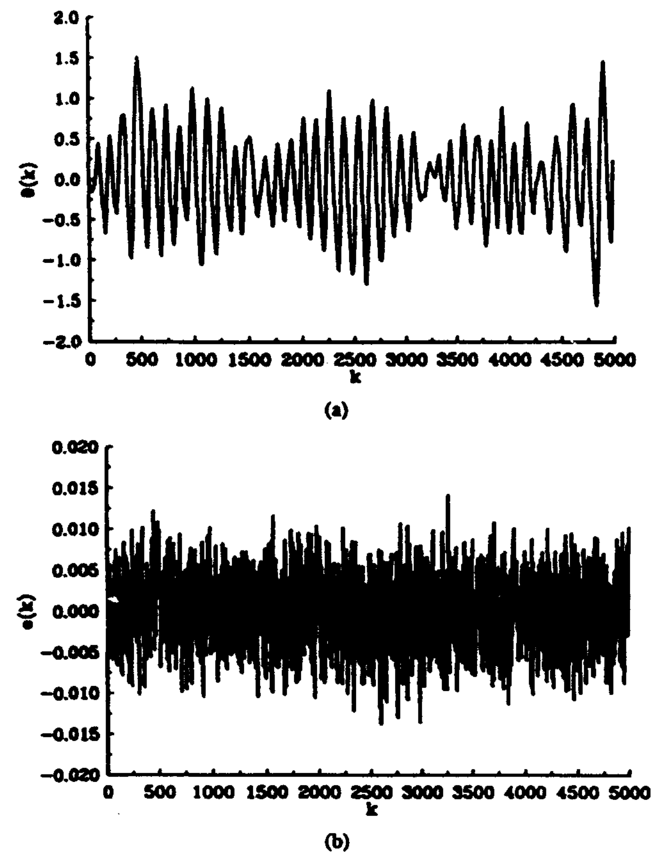


Fig. 10. (a). The desired output used in the training set. (b) The error for each point in the training time-series. The forcing function τ $N(0,1)$ was not incorporated in the resulting model.

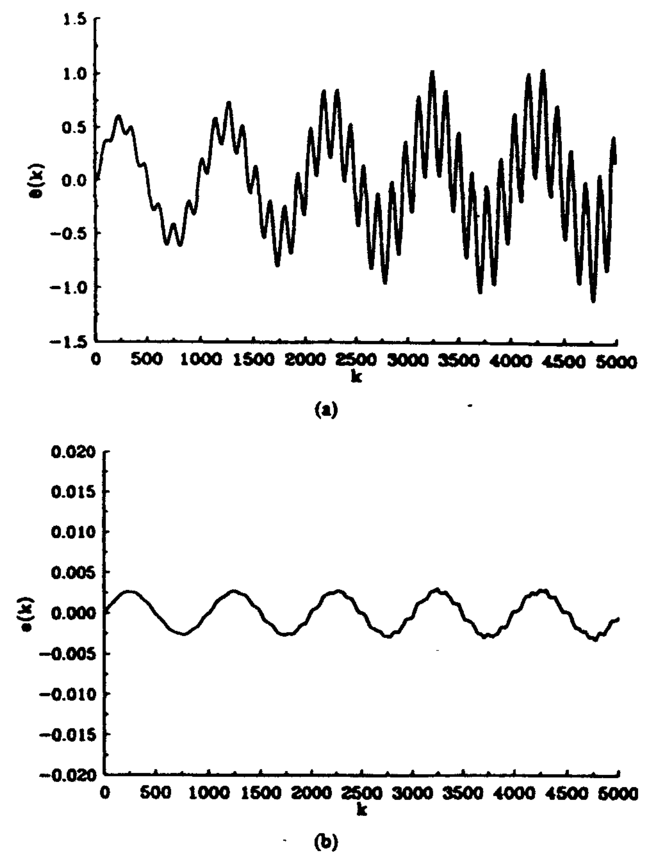


Fig. 11. (a). Test data for the pendulum model generated from $\tau(k) = \cos(2\pi k/1000$. (b). The error for the test time series.

partitioned into a training set over years 1700–1920 and test sets for years 1921–1955 and 1956–1979, respectively, as the latter test set is atypical of the entire time-series [13]. Weigend *et al.* use 12 inputs to 8 hidden units in a 12-8-1 fully-connected feedforward architecture where the number of inputs was chosen to allow direct comparison to the threshold autoregressive (TAR) model of Tong and Lim [47]. Weigend *et al.* subsequently reduce the 12-8-1 network to a 12-3-1 network by weight-elimination. Svarer *et al.* [13] employ the Optimal Brain Damage method of Le Cun *et al.* [48] to generate a pruned network or nonlinear subset model with 5 inputs that are not fully-connected to 3 hidden units in a two layer feedforward network. Priestley [44] describes a variety of more traditional time-series models for the sunspot data set. These include autoregressive (AR), ARMA, TAR, and bilinear models.

Poor results were usually obtained when using simple structures like that shown in Fig. 1(a), and reasonable results were usually found using the Class II type models for a variety of activation functions. Also, nonrecurrent models were evolved by using the evolutionary search process to determine the order of just the tapped-delay input lines. The maximum number of delays was arbitrarily set at $m_{\max} = n_{\max} = p_{\max} = q_{\max} = 20$ for the remaining experiments in this work. The sunspot data set was normalized by a factor of 200. The experiments were run with 10 parents, 20 offspring (10 for the blending process and 10 utilizing normal perturbations where the standard deviation is proportional to the MSE of the network), 10 competitions, $\sigma_{lb} = 0.0001$, and $\sigma_{ub} = 1.0$ and $S_f = 1$. The initial order of the tapped-delay lines was randomly chosen.

To facilitate comparison with previous work done on the sunspot series, the average relative variance was determined for the evolved model. The average relative variance *arv* is given by [12]

$$arv = \frac{\sum_k (x_k - \hat{x}_k)^2}{\sum_k (x_k - \bar{x}_k)^2} = \frac{1}{\hat{\sigma}^2 N} \sum (x_k - \hat{x}_k)^2$$

and provides a normalized mean squared error (NMSE) metric for comparing the performance of different models. The NMSE is independent of the training set size and is unity in the event that the estimate is equivalent to the mean of the data, (*i.e.*, $\hat{x} = \bar{x}$). In the following text, the *arv* set {*arv*1, *arv*2, *arv*3} will refer to the average relative error corresponding to {training set, test set (1921–56), test set (1957–1979)}.

When constrained as a linear system, a second-order transversal filter with a bias term as given by $\hat{y}_{k+1} = 1.2605x_k - 0.4915x_{k-1} - 0.1321x_{k-2} + 0.0831$ was evolved. That is, the order of the feedback lines became zero. The evolved linear model has an $arv = \{0.1494, 0.1732, 0.4512\}$, which compares favorably with the ninth-order AR model given by Priestley with an $arv = \{0.1865, 0.2235, 0.4994\}$. It

TABLE I
WEIGHT SET FOR THE EVOLVED TRANSVERSAL FILTER NETWORK

	Hidden unit 1	Hidden unit 2	Bias
Output	−0.9448	−1.1872	0.5423

	x_k	x_{k-1}	x_{k-2}	x_{k-3}	x_{k-4}	x_{k-5}	x_{k-6}	x_{k-7}	Bias
Hid unit1	−1.5332	1.7664	1.3729	0.3080	−0.2951	0.0252	−0.2966	−0.2740	1.1819
Hid unit2	−1.1093	-	-	-	-	-	-	-	−0.2191

TABLE II
WEIGHTS FOR THE EVOLVED, RECURSIVE, FILTER NETWORK

	Hidden unit1	Hidden unit 2	Input bias	Output bias	Input bias
Output	−0.6550	1.0668	−0.2323	0.3079	−0.2323

	x_k	x_{k-1}	x_{k-2}	x_{k-3}	x_{k-4}	x_{k-5}	x_{k-6}
Hid unit1	−0.6512	0.5742	0.9500	0.0830	−0352	−0.0531	0.0065
Hid unit2	1.3000	0.3824	0.3040	-	-	-	-
	x_{k-7}	x_{k-8}	$\hat{y}_k$	$\hat{y}_{k-1}$	$\hat{y}_{k-2}$	$\hat{y}_{k-3}$	bias
Hid unit 1	−0.2252	−0.1251	0.5443	0.3283	−0.0391	0.3393	0.2483
Hid unit2	-	-	-	-	-	-	0.1042

is interesting to note that the subset model found by Svarer *et al.* also includes the first three terms $\{x(k), x(k-1), x(k-2)\}$ as well as $\{x(k-7), x(k-10)\}$. (Neural network subset models were not investigated in this study but are achievable using EP as demonstrated in [18].) From this model structure, a least-squares estimate can be found by forming the normal equations from $\mathbf{y} = H\mathbf{w}$ or equivalently

$$\begin{bmatrix} y_3 \\ y_4 \\ \vdots \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} y_2 & y_1 & y_0 & 1 \\ y_3 & y_2 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ y_k & y_{k-1} & y_{k-2} & 1 \end{bmatrix} \begin{bmatrix} \mathrm{w}_0 \\ \mathrm{w}_1 \\ \mathrm{w}_2 \\ \mathrm{w}_3 \end{bmatrix}$$

and solving for the weight vector $\mathbf{w}_{LS} = (H^T H)^{-1} H^T \mathbf{y}$. If this is done using the complete data set, a pure follower strategy results, since $\mathbf{w}_{LS} \approx [1\,0\,0\,0]^T$. The follower strategy yields an $arv = \{0.2903, 0.4268, 0.9647\}$. No improvement was found when a gradient search scheme was applied starting from the evolved weight coefficients. This yielded a slightly modified weight coefficient vector $\mathbf{w} = [1.2612 - 0.4899 - 0.1248\ 0.0791]^T$ after a small number of iteration.

A transversal filter network was purposely evolved by disallowing feedback of the previous estimates into the network. The resulting network is equivalent to a single hidden layer network with two hidden units, one of which receives eight inputs and one of which receives only the last observation. The weights and biases for this network are given in Table I. Each node utilizes a tanh activation function. The average relative variance for this network is given in Table III along with *arv* values for deterministically trained models. The better transversal networks that are known [12], [13] have three hidden nodes and utilize observations from an eleventh-order lag that corresponds to the average period of the data. The best model is only partially connected, thereby incorporating a subset of the actual time-series inputs.

A linear-nonlinear model incorporating a *tanh* nonlinearity and a bias term was evolved as given by

$$\begin{aligned}\hat{y}_{k+1} = {} & 0.5788y_k + 0.0176y_{k-1} - 0.1396y_{k-2} + 0.0549y_{k-3} \\ & - 0.0508y_{k-4} + 0.1030y_{k-5} - 0.0239y_{k-6} \\ & + 0.01997y_{k-7} - 0.4076\hat{y}_k - 0.2614\hat{y}_{k-1} \\ & + 0.0475 + \tanh(0.7109y_k - 0.0569y_{k-1} \\ & - 0.0375y_{k-2} - 0.0324y_{k-3} - 0.0532y_{k-4} \\ & - 0.0630y_{k-5} - 0.0180y_{k-6} + 0.0687y_{k-7} \\ & + 0.0964y_{y-8} + 0.0869y_{k-9} + 0.0768\hat{y}_k \\ & + 0.1740\hat{y}_{k-1})\end{aligned}$$

This single-step sunspot predictor has an $arv = \{0.1260, 0.1140, 0.3630\}$.

Using a structure similar to that of the transversal network, except this time with feedback connections to the hidden units, a recurrent network was evolved where the search determined not only the order of the input lines, but of the feedback lines as well. During the training process, the number of tapped-delay lines on one feedback loop became zero, thus resulting in a partially feedforward network and partially recurrent network. The weights for this structure are given in Table II. Better results were not obtained on the sunspot data using this network or any of the other evolved recursive filter networks. The recurrent networks' *arv* values are given in Table III, where the first 20 observations have been substituted for the estimated values. A plot of the single-step estimates generated from the recurrent network is shown in Fig. 12(a), with the error line shown in Fig. 12(b). Although better transversal networks have been generated, it is still suspected that recurrency might be appropriate for this data based upon the results discussed by Priestley. Better results might be obtained in evolving both the transversal and recurrent structures if the representation (number of hidden nodes) is increased and the complexity penalty for the number of terms is relaxed.

TABLE III
COMPARISON OF NORMALIZED ERROR RESULTS OF PREVIOUS WORK ON THE SUNSPOT DATA SET WITH THE SOLUTION FOUND USING A RECURRENT STRUCTURE

Model	Train (1700–1920)	Test (1921–1955)	Test (1956–1979)	Number of parameters
Tong and Lim [44]	0.097	0.097	0.28	16
Weigend {\it et al.} [12]	0.082	0.086	0.35	43
Svarer {\it et al.} [13]	0.090	0.082	0.35	12–16
Transversal Net	0.0987	0.0971	0.3724	14
Recurrent net	0.1006	0.0972	0.4361	22

TABLE IV.
PERFORMANCE OF SINGLE-STEP PREDICTORS ON THE LOGISTIC MAP FOR 100 SAMPLES.

Predictor	x_0	AIC	MSE	ISE	arv
$\hat{x}_{k+1} = \sin(3.1476x_k) + 0.0274$	0.2	−316.7	0.0004	0.0004	0.0032
$\hat{x}_{k+1} = \sin(\pi x_k)$	0.2	−213.7	0.0012	0.0013	0.0091
1-10-1 network	0.2	−117.1	0.0017	-	0.0131
1.15-1 network	0.2	−303.4	0.0002	-	0.0015

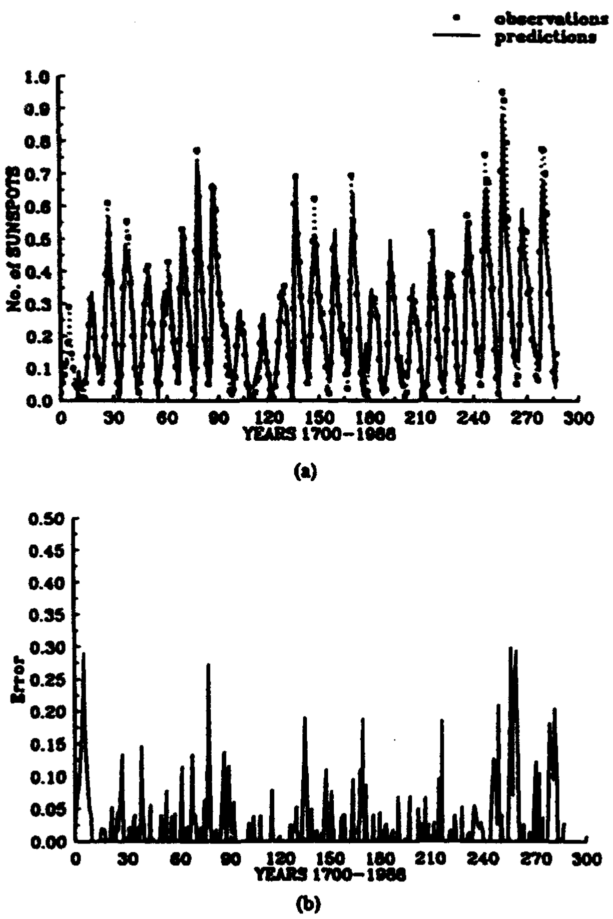


Fig. 12. (a) The evolved recurrent model for the sunspot data. Training was done over years 1700–1920. The test set consists of years 1921–1988. The first 10 data training points were not included in the model evaluation criteria (this corresponds to the maximum model order) of the evolved filter. (b) The error trace for the evolved recurrent single-step sunspot predictor. The data was normalized by a factor of 200.

C. The Logistic Map

Weigend *et al.* [12] use the iterated quadratic or logistic map $x_{k+1} = 4x_k(1 - x_k)$ on the unit interval as an example of deterministic chaos. It can be easily shown that $x_k = \sin^2(2^k\pi\theta)$ is a solution to this equation. A single-step predictor of the form

$$\hat{y}_N(k+1) = \cos\left(\sum_{i=0}^{m-1} a_i x(k-i) + \sum_{i=0}^{n-1} b_i \hat{y}(k-i)\right) + \sin\left(\sum_{i=0}^{p-1} c_i x(k-i) + \sum_{i=0}^{q-1} d_i \hat{y}(k-i)\right) + \alpha$$

was postulated. After 5000 generations, the evolutionary search yielded a predictor of the form $\hat{y}_{k+1} = \sin(3.1476x_k) + 0.0274$, thereby disregarding the cos node and the recurrent terms (the tapped-delay orders of m, n, and q went to zero). Fig. 13 shows the results of the evolved solution on 200 points generated from $x_0 = 0.2$. Upon inspection of the evolved solution, it was observed that $\hat{y}_{k+1} = \sin(\pi x_k)$ might serve as a suitable estimate. Figure 14 shows the performance of this estimate on the same 200 points used in Fig. 13. Figure 15 gives the state space plot for each of the estimates and the actual quadratic mapping function. Table II contains representative AIC and MSE values for a 100 point sequence starting with the given initial condition. The initial error at x_0 was neglected in these calculations because no observations have been made. For a large number of data points, the integral-squared error (ISE) represents a closed form solution to the MSE since

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_{i+1} - \hat{x}_{i+1})^2 \approx \sum_{i=1}^{n}(x_{i+1} - \hat{x}_{i+1})^2 \Delta x_i$$

In the limit, the MSE approximation becomes the ISE that is defined on the unit interval [0, 1] as

$$\text{ISE} = \int_0^1 (x_{k+1} - \hat{x}_{k+1})^2 dx_k$$

which is also reported in Table IV.

This example illustrates the potential ability of the search to find nonrecurrent solutions when recurrent solutions are unnecessary. Weigend *et al.* report using three radial-basis function nodes to achieve this mapping without further elaboration. Investigations using backpropagation showed that a better solution can be found for $10 < N < 15$ hidden units in

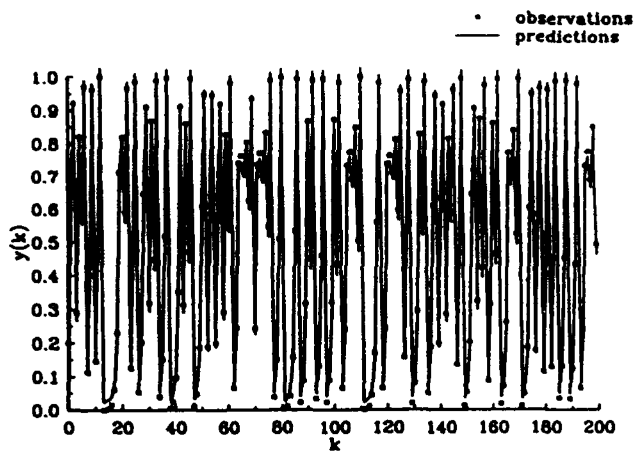


Fig. 13. Logistic map test data and the evolved solution of the form $\hat{x}_{k+1} = \sin(3.1476x_k) + 0.0274$.

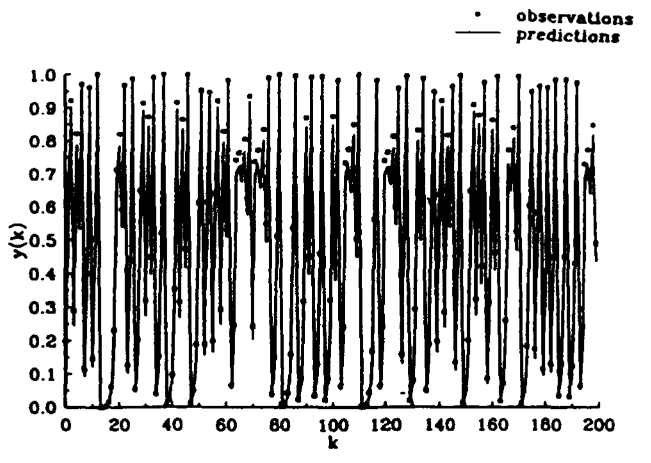


Fig. 14. Logistic map test data and the estimate $\hat{x}_{k+1} = \sin(\pi x_k)$ generated from inspection of the evolved solution.

a 1-N-1 feedforward network. The AIC values given for these networks in Table IV were determined according to

$$\mathrm{AIC}(N_w) = N\ln(\hat{\sigma}_e^2) + 2(N_w)$$

where N_w is the number of weights and biases in the network.

D. *The Mackey-Glass Equation*

The Mackey-Glass equation represents a model for white blood cell production in leukemia patients [49]. This model is complicated by the addition of a time delay τ in the nonlinear differential equation

$$\dot{x}(t) = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t)$$

where the free parameters are selected $a = 0.2$, $b = 0.1$, $c = 10$, and $\tau = 30$ as discussed in Jones *et al.* [10]. The training sets consisted of 500 data points, while the test set was comprised of the subsequent 500 data points. This data was extracted from a longer run so that the initial transients would have no effect. The data were normalized by a factor of 1.4 and observations were made every second that corresponds to the simulation stepsize. These experiments incorporated a parallel sin-cos nodal arrangement as used in the logistic equation example. After 5000 generations of training, the resulting 4-5-2-1 plus bias solution has the form

$$\begin{aligned}\hat{y}_{k+1} = {} & \cos(1.2881y_k + 0.7920y_{k-1} + 1.6431y_{k-2} \\ & - 0.2588y_{k-3} - 0.7175y_{k-4} - 0.3239\hat{y}_k \\ & - 1.0519\hat{y}_{k-1} - 0.0572\hat{y}_{k-2} - 0.2140\hat{y}_{k-3} \\ & - 0.1310\hat{y}_{k-4} - 1.1726\hat{y}_{k-5}) + \sin(2.9475y_k \\ & + 0.8605y_{k-1} + 0.0069y_{k-2} \\ & - 1.6800\hat{y}_k - 1.0020\hat{y}_{k-1}) - 0.9784\end{aligned}$$

Figure 16(a) shows the results of this model on the training and test sets, while Fig. 16(b) shows the error on the same data sets. For the data sample shown in Fig. 16(a), Table V lists the AIC, MSE, and *arv* values of the training and test sets for the evolved single-step predictor. The last 450 points of the training set were used for evaluation purposes of the training data to allow the transient effects due to the perceptron recurrencies to die out. This difference in the number of observations alters the AIC scores. Nevertheless, the *arv* values compare favorably with the results found in the recurrent network training evaluation done by Logar *et al.* [50] for the Mackey-Glass equation with $\tau = 17$. The preditor can be made nonrecursive by letting $\hat{y} \leftarrow y$, which yields a MSE = 0.00088 and an arv = 0.0232 on the test set. However, if a transversal filter is desired, it is suspected that better results would be obtained by training the appropriate structure. This model performs poorly if forward projections are generated by replacing the observations with previous estimates so that $y \leftarrow \hat{y}$. A two-step ahead predictor yields a MSE = 0.0070 on the test set, while a three-step ahead predictor is dramatically worse with a MSE = 5.1125 on the test set.

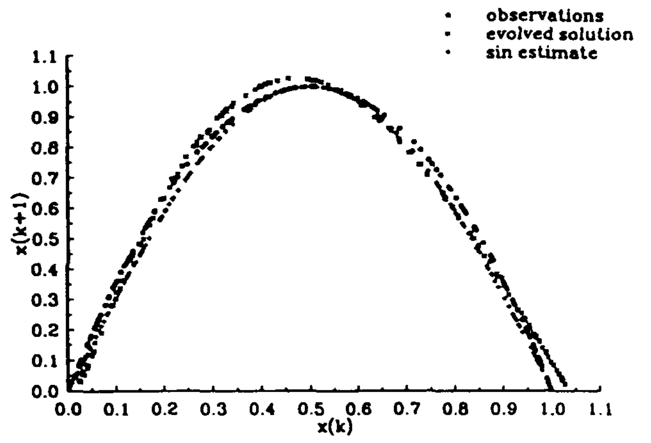


Fig. 15. The state-space plot for the logistic map. Observations generated from $x_{k+1} = 4x_k(1 - x_k)$ are denoted with an "o," the evolved solution estimate $\hat{x}_{k+1} = \sin(3.1476x_k) + 0.0274$ is denoted by an "x," and the estimate generated by inspection, the evolved solution $\hat{x}_{k+1} = \sin(\pi x_k)$ is denoted by a "+."

V. Conclusion

This work has incorporated an efficient single-agent search strategy, the method of Solis and Wets, into the EP framework and augmented it with the simplest convex optimization

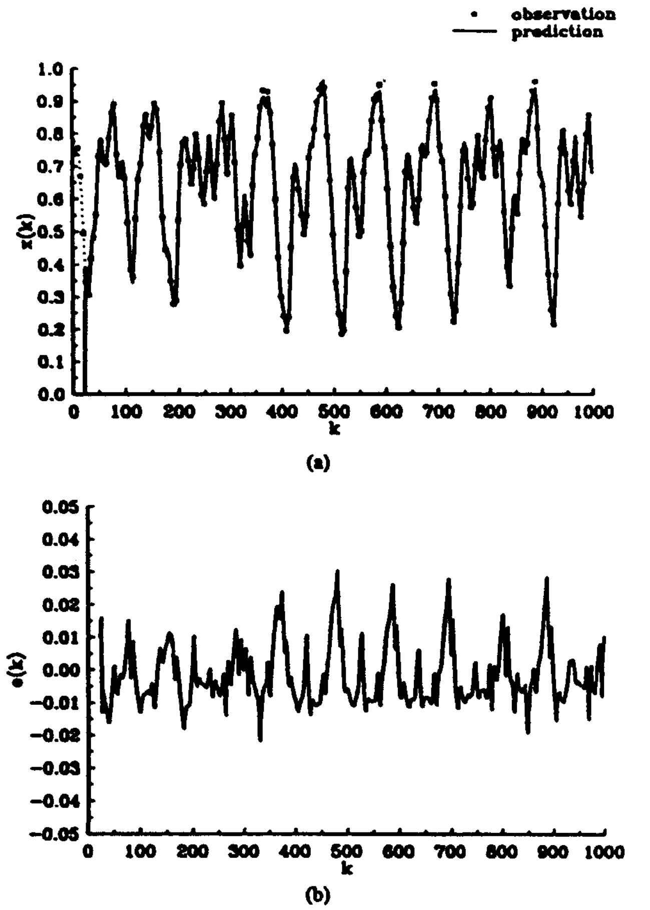


Fig. 16. (a). The evolved solution on the training and test data generated from the Mackey-Glass equation. The first 500 points were in the training set, and the subsequent 500 points were used for the testing set. Every fifth observation point is shown by a solid square. (b). The error between the evolved solution and the Mackey-Glass dynamic equations for the time-series shown in Fig. 14.

TABLE V
THE PERFORMANCE RESULTS FOR THE NEXT-STEP AHEAD PREDICTOR ON THE MACKEY-GLASS EQUATION

Data set	No. of effective observations	AIC	MSE	*arv*
Training	450	−4406.1	0.00005	0.0014
Test	500	−4594.4	0.00009	0.0025

capability, bisection search, to yield a hybrid multi-agent stochastic search technique. This hybrid method can enhance EP optimization efficiency while alleviating local minima problems associated with single-agent search techniques. A myriad of other local methods could also have been easily incorporated with the multiagent EP search procedure.

The hybrid method was applied to nonlinear IIR filters for single-step prediction tasks. Since the model structure was simultaneously determined along with the weighting coefficients, the evolved solutions did not always have recurrent structure (i.e., transversal filter structures sometimes resulted). Good single-step prediction ability was evolved using nonlinear mappings, even though the resulting models were dissimilar to the models (if any) that created the original data. The learning procedure had the most trouble with the noisy sunspot data, indicating that the representation may not be sufficient and/or additional work is warranted for systems with noisy output. Other types of information-based criteria, such as the *minimum description length* (MDL) modeling principle [51], can be used in lieu of the AIC as an objective function, and may yield better results, since the AIC is not a consistent estimator [52].

The simple structures investigated in this work demonstrated a reasonable degree of proficiency for the nonlinear time-series problems studied. Similar AIC values were attained for a varied assortment of models of the same data sets. From these results, it is suspected that the joint parameter-function space of particular data sets may be dense in the number of acceptable solutions, some of which may be found by the relaxation scheme employed in this investigation. While it is evident that the hybrid stochastic optimization scheme provides an effective learning mechanism, the issue of what types of time-series representations can be effectively modeled using this approach has not been addressed in this investigation. By cascading and/or parallelizing recurrent perceptrons to generate multi-unit networks, as in traditional feedforward architectures, additional capabilities for more complex time-series processing tasks may be achieved.

ACKNOWLEDGMENT

The authors thank Dr. Al Gordon, program director for NRaD internal research projects, and Lou Griffith for their support of this work. The thors would also like to thank David Fogel and the reviewers for their suggestions and many helpful comments.

REFERENCES

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzman Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. New York: John Wiley & Sons, 1989.
[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*. 2nd edition, Cambridge, MA: MIT Press, 1992.
[3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, 1966.
[4] D. B. Fogel, "An Evolutionary approach to the traveling salesman problem," *Biological Cybernetics*, vol. 60, No. 2, 1988.
[5] D. B. Fogel, *System Identification though Simulated Evolution: A Machine Learning Approach to Modeling*. Needham, MA: Ginn Press, 1991.
[6] D. B. Fogel, L. J. Fogel, and V. W. Porto, "Evolving neural networks," *Biological Cybernetics*, vol. 62, pp. 487–493, 1990.
[7] R. J. Williams, "Adaptive State Representation and Estimation using Recurrent Connectionist Networks," in *Neural Networks for Control*, W. T. Moller, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA: MIT Press, 1990.
[8] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 37, No. 8, 1989.
[9] R. D. Jones, Y. C. Lee, S. Qian, C. W. Barnes, K. R. Bisset, G. M. Bruce, G. W. Flake, K. Lee, L. A. Lee, W. C. Mead, M. K. O'Rouke, I. J. Poli and L. E. Thode, "Nonlinear adaptive networks: a little theory, a few applications," Technical Report LA-UR 91-273, Los Alamos National Laboratory, Los Alamos, NM, 1991.
[10] W. C. Mead, R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Glake, L. A. Lee and M. K. O'Rourke, "Prediction of chaotic time series using CNLS-NET—example: The Mackey-Glass equation," Technical Report LA-UR-91-720, Los Alamos National Laboratory, Los Alamos, NM, 1991.
[11] R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, P. S. Lewis and S. Qian, "Function approximation and time series prediction with neural networks," Technical Report LA-UR 90-21, Los Alamos National Laboratory, Los Alamos, NM, 1990.

[12] A. S. Weigned, D. E. Rumelhart, and B. A. Huberman, "Predicting the future: A connectionist approach," Technical Report Stanford-PDP-90-01 or PARC-SSL-90-20, 1990.
[13] C. Svarer, L. K. Hansen, and J. Larsen, "On design and evaluation of tapped-delay neural network architectures," *IEEE Int. Conf. on Neural Networks*, San Francisco, 1992.
[14] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight sharing," *Neural Computation*, vol. 4, No. 4, 1992.
[15] S. S. Rao and V. Ramamurti, "A hybrid technique to enhance the performance of recurrent neural networks for time series prediction," *IEEE Int. Conf. on Neural Networks*, San Francisco, 1993.
[16] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," Technical Report CMU-CS-90-100, Carnegie-Mellon University, Pittsburgh, PA, 1990.
[17] N. Saravanan, "Evolving neural networks: applications to a prediction problem," *Second Annual Conf. on Evolutionary Programming*, La Jolla, CA, 1993, pp. 72–78.
[18] J. R. McDonnell and D. Waagen, "Neural network structure design by evolutionary programming," *Second Annual Conf. on Evolutionary Programming*, La Jolla, CA, 1993, pp. 79–89.
[19] P. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. on Neural Networks*, 1994, this issue.
[20] D. B. Fogel, "Using evolutionary programming for modeling: an ocean acoustic example," *IEEE Journal on Oceanic Eng.*, vol. 17, No. 4, pp. 333–340, 1992.
[21] A. Gelb, *Applied Optimal Estimation*. Cambridge, MA: MIT Press, 1974.
[22] L. Li and S. Haykin, "A cascaded recurrent neural network for real-time nonlinear adaptive filtering," *IEEE Int. Conf. on Neural Networks*, San Francisco, 1993.
[23] X. Yao, "A review of evolutionary artificial neural networks," *Int. Journal of Intelligent Systems*, to appear.
[24] D. Waagen, P. Diercks, and J. R. McDonnell, "The stochastic direction set algorithm: A hybrid technique for finding function extrema," *First Annual Conf. on Evolutionary Programming*, La Jolla, CA, 1992, pp. 35–42.
[25] S. S. Rao, *Optimization Theory and Applications*. New York: John Wiley and Sons, 1978.
[26] D. C. Karnopp, "Random search techniques for optimization problems," *Automatica*, vol. 1, pp. 111–121, 1963.
[27] D. A. Pierre, *Optimization Theory with Applications*. Dover, 1986.
[28] Z. Michalwicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.
[29] J. Matyas, "Random optimization," *Automation and Remote Control*, vol. 26, pp. 244–251, 1965.
[30] F. J. Solis and J. B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–50, 1981.
[31] N. Baba, "A new approach for finding the global minimum of error function of neural networks," *Neural Networks*, vol. 2, pp. 19–30, 1989.
[32] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, "Generalized simulated anealing for function optimization," *Technometrics*, vol. 28, no. 3, pp. 209–218, 1986.
[33] T. Back, G. Rudolph, and H.P. Schwefel, "Evolutionary programming evolution strategies: similarities and differences," *Second Annual Conf. on Evolutionary Programming*, San Diego, 1993.
[34] D. B. Fogel, "Evolving Artificial Intelligence," Ph.D. Dissertation, University of California, San Diego, 1992.
[35] S. H. Brooks, "A discussion of random methods for seeking maxima," *Operations Research*, vol. 6, pp. 244–251, 1958.
[36] G. B. Fogel and D. B. Fogel, "Continuous evolutionary programming: Analysis and experiments," *Cybernetics and Systems*, in review.
[37] P. P. C. Yip and Y. H. Pao, "A fast universal training algorithm for neural networks," *World Congress on Neural Networks*, Portland, OR, 1993.
[38] H. H. Rosenbrock, "An automatic method for finding the greatest of least value of a function," *The Computer Journal*, vol. 3, p. 175, 1960.
[39] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks*, vol. 1, no. 1, 1990.
[40] A. D. Back and A. C. Tsoi, "FIR and IIR synapses, a new neural network architecture for time series modeling," *Neural Computation*, vol. 3, pp. 375–385, 1991.
[41] G. Cybenko, "Approximation by superpositions of sigmoidal functions," *Mathematics of Control, Signals, and Systems*, 1989.
[42] S. Lee, "Supervised learning with Gaussian potentials," in *Neural Networks for Signal Processing*, B. Kosko, Ed. Prentice-Hall, 1992.
[43] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. on Automatic Control*, vol. 19, no. 6, Dec., 1974.
[44] M. B. Priestley, *Non-Linear and Non-Stationary Time Series Analysis*. Academic Press, 1988.
[45] D. Haesloop and B. R. Holt, "Neural networks for process identification," *Int. Joint Conf. on Neural Networks*, San Diego, 1992.
[46] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, 1989.
[47] H. Tong and K. S. Lim, "Threshold autoregression, limit cycles and cyclical data," *Journal Royal Statistical Society B*, vol. 42, 1980.
[48] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems 2*. San Mateo, CA: Morgan Kaufman, 1990, pp. 598–605.
[49] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, 197, 1977.
[50] A. M. Logar, E. M. Corwin, and W. J. B. Ol[illegible] comparison of recurrent neural network learning algorithms [illegible]*nt. Conf. on Neural Networks*, San Francisco, 1993.
[51] J. Rissanen, "Stochastic complexity and modeling, *The Annals of Statistics*, vol. 14, no. 3, pp. 1080–1100, 1986.
[52] R. L. Kashyap, "Inconsistency of the AIC rule for estimating the order of autoregressive models," *IEEE Trans. on Automatic Control*, vol. 25, no. 5, pp. 996–998, 1980.

John R. McDonnell (A'87–S'88–M'89) received the B.S. degree from Texas A&M in 1985 and is currently completing the requirements for the M.S. degree in Systems Science from the University of California, San Diego. Mr. McDonnell is a principal engineer at the Naval Command, Control, and Ocean Surveillance Center (NCCOSC), RDT&E Div., where he is involved with teleoperated and autonomous robotic systems, neural network signal classification, and the design of neural network architectures by evolutionary search. He is an adjunct professor in the Mechanical Engineering Department at San Diego State University, where he teaches graduate-level courses in control systems and simulation. Mr. McDonnell is a member of the Evolutionary Programming Society and serves as a committee member for the Annual Conferences in Evolutionary Programming.

D. Waagen received the B.S. in mathematics from the University of Utah in 1984, and the M.S. degree in statistics from San Diego State University in 1993. From 1985 to 1993, he has worked as a scientist for NCCOSC, RDT&E Div., San Diego, CA. Mr. Waagen is a member of the American Statistical Association and the Evolutionary Programming Society. His current interests are in the areas of nonparametric estimation, machine learning, and stochastic optimization.