Feature Selection for Structure−Activity Correlation Using Binary Particle Swarms

Dimitris K. Agrafiotis* and Walter Cedeño

3-Dimensional Pharmaceuticals, Inc., 665 Stockton Drive, Exton, Pennsylvania 19341

Received October 10, 2001

We present a new feature selection algorithm for structure−activity and structure−property correlation based on particle swarms. Particle swarms explore the search space through a population of individuals that adapt by returning stochastically toward previously successful regions, influenced by the success of their neighbors. This method, which was originally intended for searching multidimensional continuous spaces, is adapted to the problem of feature selection by viewing the location vectors of the particles as probabilities and employing roulette wheel selection to construct candidate subsets. The algorithm is applied in the construction of parsimonious quantitative structure−activity relationship (QSAR) models based on feed-forward neural networks and is tested on three classical data sets from the QSAR literature. It is shown that the method compares favorably with simulated annealing and is able to identify a better and more diverse set of solutions given the same amount of simulation time.

I. Introduction

In recent years, there has been an increasing need for novel data-mining methodologies that can analyze and interpret large volumes of data. Artificial intelligence methods, such as artificial neural networks, classification and regression trees, and *k* nearest-neighbor classifiers, have been used extensively for this purpose.[1−3] Most of these methods work by correlating some experimentally determined measure of biological activity with a set of physicochemical, structural, and/or electronic parameters (descriptors) of the compounds under investigation. Since it is not possible to know a priori which molecular properties are most relevant to the problem at hand, a comprehensive set of descriptors is usually employed, chosen based on experience, software availability, and computational cost.

However, it is well-known, both in the chemical and statistical fields, that the accuracy of classification and regression techniques is not monotonic with respect to the number of features employed by the model. Depending on the nature of the regression technique, the presence of irrelevant or redundant features can cause the system to focus attention on the idiosyncrasies of the individual samples and lose sight of the broad picture that is essential for generalization beyond the training set.[4−6] This problem is compounded when the number of observations is also relatively small. If the number of variables is comparable to the number of training patterns, the parameters of the model may become unstable and unlikely to replicate if the study were to be repeated.

Feature selection attempts to remedy this situation by identifying a small subset of relevant features and using only them to construct the actual model. Feature selection algorithms can be divided into three main categories:[7] (1) those where the selection is embedded within the basic regression algorithm, (2) those that use feature selection as a filter prior to regression, and (3) those that use feature selection as a wrapper around the regression. The last has a long history in statistics and pattern recognition and is the method of choice for quantitative structure−activity relationship (QSAR) and quantitative structure−property relationship (QSPR).

From an algorithmic perspective, feature selection can best be viewed as an heuristic search, where each state in the search space represents a particular subset of the available features. In all but the simplest cases, an exhaustive search of the state space is impractical, since it involves 2^n possible combinations, where n is the total number of available features. Several search algorithms have been devised, ranging from simple greedy approaches such as forward selection or backward elimination[8] to more elaborate methodologies such as simulated annealing,[9] evolutionary programming,[10] genetic algorithms,[11−14] and artificial ants.[15,16]

In this paper, we present a novel approach inspired from the study of human sociality, known as particle swarms.[17−19] Particle swarms explore the search space through a population of individuals, which adapt by returning toward previously successful regions. Their movement is stochastic and is influenced by the individuals' own memories as well as the memories of their peers. This method, which was originally intended for searching multidimensional continuous spaces, is adapted to the problem of feature selection by viewing the location vectors of the particles as probability thresholds and employing roulette wheel selection to construct candidate subsets. In the remaining sections, we provide the key algorithmic details of the binary particle swarm optimizer and discuss its advantages compared to simulated annealing using three classical data sets from the QSAR literature. While the present study uses artificial neural networks to construct the models, the method is general and can be used with any machine-learning technique, including multilinear regression,

* To whom correspondence should be addressed. Phone: (610) 458-6045. Fax: (610) 458-8249. E-mail: dimitris@3dp.com.

10.1021/jm0104668 CCC: $22.00
Published on Web 02/01/2002

case-based reasoning (nearest-neighbor regression), fuzzy logic, and many others.

II. Methods

Neural Networks. Our analysis was based on three-layer, fully connected multilayer perceptrons, trained with the standard error back-propagation algorithm.[20] The logistic transfer function $f(x) = 1/(1 + e^{-x})$ was used for both hidden and output layers. During feature selection, each network was trained for 200 epochs, using a linearly decreasing learning rate from 1.0 to 0.01 and a momentum of 0.8. During each epoch, the training patterns were presented to the network in a randomized order. To minimize the risk of back-propagation getting trapped in local minima in synaptic weight space, each model was trained three times and the model with the lowest training error was retained. The best models identified by the particle swarm and simulated annealing algorithms were retrained and cross-validated 50 times using leave-one-out cross-validation to establish their true learning and generalization capabilities. These were quantified by the Pearson correlation coefficient:

$$R = \frac{N\sum_{i=1}^{N} y_i\tilde{y}_i - \sum_{i=1}^{N} y_i \sum_{i=1}^{N} \tilde{y}_i}{\sqrt{[N\sum_{i=1}^{N} y_i^2 - (\sum_{i=1}^{N} y_i)^2][N\sum_{i=1}^{N} \tilde{y}_i^2 - (\sum_{i=1}^{N} \tilde{y}_i)^2]}} \quad (1)$$

where N is the number of training patterns and y_i and $\tilde{y}_i$ are the measured and predicted activities of the ith compound, respectively.

Particle Swarms. Particle swarms (PS) is a relatively new optimization paradigm introduced by Kennedy and Eberhart.[17,18] The method is based on the observation that social interaction, which is believed to play a crucial role in human cognition, can serve as a valuable heuristic in identifying optimal solutions to difficult optimization problems. Particle swarms explore the search space using a population of individuals, each with an individual, initially random location and velocity vector. The particles then "fly" over the state space, remembering the best solution encountered. Fitness is determined by an application-specific objective function $f(\mathbf{x})$. During each iteration, the velocity of each particle is adjusted on the basis of its momentum and the influence of the best (most fit) solutions encountered by itself and its neighbors. The particle then moves to a new position, and the process is repeated for a prescribed number of iterations. In the original PS implementation,[21] the trajectory of each particle is governed by the equations

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + \eta_1 \cdot r \cdot (\mathbf{p}_i - \mathbf{x}_i(t)) + \eta_2 \cdot r \cdot (\mathbf{p}_{b(i)} - \mathbf{x}_i(t)) \quad (2)$$

and

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (3)$$

where $\mathbf{x}_i$ and $\mathbf{v}_i$ are the current position and the velocity of the ith particle, $\mathbf{p}_i$ is the position of the best state visited by the ith particle, $b(i)$ is the particle with the best fitness in the neighborhood of i, and t is the iteration number. The parameters η_1 and η_2 are called the *cognitive* and *social learning rates* and determine the relative influence of the memory of the individual versus that of its neighborhood. In the psychological metaphor, the cognitive term represents the tendency of organisms to repeat past behaviors that have been proven to be successful or have been reinforced by their environment, whereas the social term represents the tendency to emulate the successes of others, which is fundamental to human sociality. In effect, these terms introduce a tendency to sample regions of space that have demonstrated promise. r is a random number, whose upper limit is a constant parameter of the system, and is used to introduce a stochastic element in the search process.

Kennedy defined four models of PS:[21] the *full model*, which places equal influence for the cognitive and social influence, the *social-only model*, which involves no cognitive learning, the *cognitive-only model*, which has no social component, and the *selfless model*, which is a social-only model in which the individual is excluded from consideration in determining its neighborhood's best. The neighborhood represents a subset of the population surrounding a particular particle. The neighborhood size defines the extent of social interaction and can range from the entire population to a small number of neighbors on either side of the particle (i.e., for the ith particle, a neighborhood size of 3 would represent particles $i - 1$, i, and $i + 1$).

The present work employs Shi and Eberhart's[22,23] variant of the PS algorithm, which makes use of an inertia weight, ω, to dampen the velocities during the course of the simulation and allows the swarm to converge with greater precision:

$$\mathbf{v}_i(t+1) = \omega \cdot \mathbf{v}_i(t) + \eta_1 \cdot r \cdot (\mathbf{p}_i - \mathbf{x}_i(t)) + \eta_2 \cdot r \cdot (\mathbf{p}_{b(i)} - \mathbf{x}_i(t)) \quad (4)$$

Larger values of ω induce larger transitions and thus enable global exploration, whereas lower values facilitate local exploration and fine-tuning of the current search area.

Binary Particle Swarms. The particle swarm algorithm, which was originally intended for searching multidimensional continuous spaces, can be adapted to the discrete problem of feature selection by viewing the location vectors of the particles as probabilities and employing roulette wheel selection to construct candidate subsets. In this scheme, the elements of the location vectors x_{ij} and p_{ij} can only take the values 0 and 1 and indicate whether the jth feature is selected in the ith particle (subset). A discretization step is introduced following the application of eq 3, which converts the fractional coordinates, x_{ij}, to binary values using probabilistic selection. During this step, the fractional values of x_{ij} are treated as probability thresholds to determine subset membership. Two possibilities can be envisioned to prevent overfitting: (1) to select each feature on the basis of its own probability and employ an objective function that penalizes solutions containing a large number of features, such as Rao's lack-of-fit;[24] (2) to select a predefined number of features on the basis of the ratio of the number of training patterns to the number of freely adjustable parameters in the model. In the latter case, which is the one employed in this work, the features comprising the model are determined by roulette wheel selection. In this method, each feature is assigned a slice of a roulette wheel whose size is equal to the probability assigned to that feature. The subset is assembled by spinning the wheel and selecting the features under the wheel's marker. This process is repeated k times, where k is the number of desired features in the model (duplicates are excluded).

The actual probabilities, p_{ij}, are computed by eq 5:

$$p_{ij} = \frac{x_{ij}^{\,a}}{\sum_{j=1}^{n} x_{ij}^{\,a}} \quad (5)$$

where x_{ij} is the fractional coordinates obtained by applying eq 3 (confined in the interval [0, 1]) and α is a scaling factor referred to as selection pressure. If α is greater than 1, the selection tends to emphasize highly fit individuals, whereas if it is less than 1, the differences between the individuals are attenuated and less fit individuals have an increased chance of being selected. To eliminate redundant computation, the program caches all visited states and their fitness into a lookup table, implemented as a sorted vector of pointers with ($O(n \log n)$ insertion)/(retrieval time).

Simulated Annealing. Simulated annealing (SA) is a global, multivariate optimization technique based on the

Table 1. Data Set Size and Neural Network Topology Used

data set	N^a	M^b	K^c	H^d	ref
AMA	31	53	3	3	8
BZ	57	42	6	2	29
PYR	74	27	6	2	31

[a] Number of samples. [b] Number of features in the original data set. [c] Number of features used in the models. [d] Number of hidden neurons.

Metropolis Monte Carlo search algorithm.[25] The method starts from an initial random state and walks through the state space associated with the problem of interest by a series of small stochastic steps. In the problem at hand, a state represents a particular subset of k features and a step is a small change in the composition of that subset (i.e., replacement of one of the features comprising the subset). As with particle swarms, an objective function, $f(\mathbf{x})$, maps each state to a real value that represents its energy or fitness. While downhill transitions are always accepted, uphill transitions are accepted with a probability that is inversely proportional to the energy difference between the two states. This probability is computed using Metropolis' acceptance criterion $p = e^{-\Delta E/(KT)}$, where K is a constant used for scaling purposes and T is an artificial temperature factor that controls the ability of the system to overcome energy barriers. The temperature is systematically adjusted during the course of the simulation in a manner that gradually reduces the probability of high-energy transitions (in this case, using a Gaussian cooling schedule with a half-width of 5 deviation units). This protocol results in two optimization phases: one in which the system explores the state space relatively freely and one in which it equilibrates around a low-energy minimum.

To circumvent the problem of selecting an appropriate value for K and to ensure that the transition probability is properly controlled, we use an adaptive approach in which K is not a true constant, but rather it is continuously adjusted during the course of the simulation on the basis of a running estimate of the mean transition energy.[26,27] In particular, at the end of each transition, the mean transition energy is updated and the value of K is adjusted so that the acceptance probability for a mean uphill transition at the final temperature is some predefined small number (here, 0.1%).

Data Sets. The methods were tested on three well-known data sets: antifilarial activity of antimycin analogues (AMA),[8,10−12,28] binding affinities of ligands to benzodiazepine/$GABA_A$ receptors (BZ),[29,30] and inhibition of dihydrofolate reductase by pyrimidines (PYR[31]). These data sets have been the subject of extensive QSAR studies and have served as a test bed for many feature selection algorithms. To allow comparison with previous neural-network-based approaches, the number of input features and hidden neurons was taken from the literature. These details are summarized in Table 1. In all three cases, the descriptor data were normalized to [0, 1] prior to network modeling.

Implementation. All programs were implemented in the C++ programming language and are part of the DirectedDiversity[32] software suite. They are based on 3-Dimensional Pharmaceuticals' Mt++ class library[33] and are designed to run on all Posix-compliant Unix and Windows platforms. All calculations were carried out on a Dell Inspiron 8000 laptop computer equipped with a 1 GHz Pentium IV Intel processor running Windows 2000 Professional.

III. Results and Discussion

Model Assessment. Our study was designed to accomplish two main goals. The first was to establish a reasonable set of parameters for the particle swarm optimizer, and the second was to determine whether the method offers any advantages over other commonly used search algorithms that employ neural networks for activity prediction. The choice of simulated annealing was based on its programmatic simplicity, robustness, and track record in structure−activity correlation and multimodal optimization in general. For comparison purposes, the best models obtained from another insect-based optimization algorithm[16] using a similar protocol are listed in Table 2.

Following common practice, two measures were used to define the quality of the resulting models. The first is the training correlation coefficient R (eq 1), and the second is the cross-validated correlation coefficient R_{CV} resulting from leave-one-out cross-validation. The latter is obtained by systematically removing one of the patterns from the training set, building a model with the remaining cases, and predicting the activity of the removed case using the optimized weights. This is done for each pattern in the training set, and the resulting predictions are compared to the measured activities to determine their degree of correlation. Since back-propagation is susceptible to local minima, each network model was trained three times using a different set of randomly chosen initial weights and the lowest training error was used to score the corresponding subset.

Given the fact that both particle swarms and simulated annealing are stochastic in nature, we carried out 50 independent runs for each set of optimization parameters, each time starting with a different random seed. The same procedure was followed for all three data sets under investigation, and the results are summarized in Tables 3, 5, and 7 for AMA, BZ, and PYR, respectively. To ensure a fair comparison, the PS, SA, and random sampling methods were given the same amount of simulation time, measured in terms of function evaluations. Past experience with these data sets suggested that 1000 trials represented a good tradeoff between accuracy and speed and would be a reasonable limit for comparing the three methods.

To gain additional confidence, the best subsets discovered from all 50 runs of the reference PS and SA configurations (for PS, 100 particles, 10 iterations; for SA, 30 temperature cycles, 33 sampling steps) were retrained 50 times using 1000 training epochs and tested using leave-one-out cross-validation. The mean and standard deviation of the training and cross-validated correlation coefficients R and R_{CV} are listed in Tables 4, 6, and 8 for the AMA, BZ, and PYR data sets, respectively, along with the number of times that each subset was discovered by each optimizer. The entries are listed in descending order of R.

In a comparison of optimization techniques, it is important to recognize that the relative quality of the various algorithms can only be assessed on the basis of the functions that they were designed to optimize. Although cross-validation is an essential component of QSAR, feature selection is guided exclusively by the training error, which is often poorly correlated with the generalization ability of the resulting model. For example, the correlation coefficients between the average training and cross-validated correlation coefficients of the models listed in Tables 4, 6, and 8 were only 0.448, 0.347, and 0.425, respectively. This suggests that even when the mapping device has a limited number of adjustable parameters to prevent overfitting, the training R, which is in effect the function optimized by the search algorithm, is a rather poor indicator of the model's generalization ability. Thus, for the purposes

Table 2. Top Models Selected by the Artificial Ant Algorithm (See Ref 16)

data set	selected variables (names)	selected variables (indices)[a]	$\mu(R_{CV})$[b]	$\sigma(R_{CV})$[c]
AMA	NSDL8, MOFI_Y, LOGP	31, 37, 49	0.838	0.010
	NSDL8, MOFI_Z, LOGP	31, 38, 49	0.826	0.007
	MOFI_Y, LOGP, SUM_F	37, 49, 51	0.807	0.023
BZ	μ_7, π_7, σ_{m7}, MR_1, R_1, μ_2	0, 1, 5, 9, 11, 14	0.890	0.009
	μ_7, π_7, F_7, MR_1, σ_{p1}, σ_{m2}	0, 1, 3, 9, 13, 19	0.888	0.019
	μ_7, π_7, F_7, MR_1, σ_{m2}, π_6	0, 1, 3, 9, 19, 22	0.885	0.011
PYR	SZ_3, FL_3, Hd_3, πA_3, SZ_5, HA_5	1, 2, 3, 5, 19, 22	0.796	0.005
	Hd_3, πA_3, FL_4, PO_4, SZ_5, HA_5	3, 5, 11, 16, 19, 22	0.786	0.035
	SZ_3, FL_3, πA_3, SZ_5, Hd_5, HA_5	1, 2, 5, 19, 21, 22	0.785	0.037

[a] Zero-based indices of selected variables. [b] Average leave-one-out cross-validated R over 10 cross-validation runs. [c] Standard deviation of leave-one-out cross-validated R over 10 cross-validation runs.

Table 3. Quality of Models Selected by PS, SA, and Random Selection for the AMA Data Set, with Each Row Representing 50 Independent Optimization Runs

opt[a]	size[b]	steps[c]	η_1[d]	η_2[e]	ω[f]	α[g]	trials[h]	$\mu(R)$[i]	$\sigma(R)$[j]	min (R)[k]	max (R)[l]
PS	100	10	1	1	0.9	2	3	0.905	0.021	0.842	0.931
PS	50	20	1	1	0.9	2	3	0.901	0.019	0.829	0.932
PS	25	40	1	1	0.9	2	3	0.895	0.021	0.839	0.932
PS	10	100	1	1	0.9	2	3	0.862	0.039	0.742	0.915
PS	100	10	1	0.5	0.9	2	3	0.899	0.021	0.851	0.935
PS	100	10	1	0	0.9	2	3	0.882	0.021	0.826	0.928
PS	100	10	0.5	1	0.9	2	3	0.897	0.018	0.859	0.931
PS	100	10	0	1	0.9	2	3	0.899	0.017	0.845	0.927
PS	100	10	1	1	0.5	2	3	0.902	0.022	0.834	0.930
PS	100	10	1	1	0.1	2	3	0.894	0.020	0.851	0.937
PS	100	20	1	1	0.9	2	3	0.905	0.019	0.851	0.931
SA	30	33					3	0.890	0.002	0.884	0.894
SA	30	66					3	0.891	0.002	0.888	0.900
SA	30	133					3	0.892	0.002	0.889	0.898
R	1000						3	0.872	0.023	0.809	0.916

[a] Optimizarion algorithm (PS, particle swarm; SA, simulated annealing; R, random selection). [b] Number of particles in the swarm. [c] Number of iterations. [d] Cognitive learning rate. [e] Social learning rate. [f] Inertia. [g] Selection pressure. [h] Number of back-propagation refinements. [i] Mean training R. [j] Standard deviation of training R. [k] Minimum training R. [l] Maximum training R.

Table 4. Best Models Selected by PS[a] and SA[b] for the AMA Data Set (50 Runs, 1000 Training Epochs)

rank	features[c]	freq (PS)[d]	freq (SA)[e]	$\mu(R)$[f]	$\sigma(R)$[g]	$\mu(R_{CV})$[h]	$\sigma(R_{CV})$[i]
1	3, 4, 49	23	0	0.945	0.010	0.818	0.013
2	31, 34, 49	9	2	0.919	0.004	0.825	0.012
3	31, 37, 49	0	8	0.918	0.007	0.837	0.008
4	6, 49, 50	0	21	0.913	0.012	0.796	0.016
5	31, 35, 49	4	0	0.912	0.006	0.831	0.009
6	16, 49, 50	0	2	0.910	0.009	0.790	0.012
7	31, 38, 49	0	1	0.910	0.007	0.826	0.007
8	2, 3, 49	1	0	0.909	0.020	0.687	0.037
9	37, 49, 51	2	0	0.909	0.006	0.799	0.014
10	3, 6, 49	0	16	0.908	0.019	0.755	0.013
11	35, 49, 51	7	0	0.907	0.022	0.799	0.014
12	34, 49, 51	3	0	0.905	0.021	0.764	0.023
13	4, 12, 49	1	0	0.903	0.023	0.619	0.029

[a] Swarm parameters: 100 particles, 10 iterations, $\eta_1 = 1$, $\eta_2 = 1$, $\omega = 0.9$, $\alpha = 2$. [b] Annealing parameters: 30 temperature cycles, 33 sampling steps. [c] Zero-based indices of features comprising the best models identified by 50 independent runs of the PS and SA algrotihms. [d] Number of times that the PS algorithm converged to this model. [e] Number of times that the SA algorithm converged to this model. [f] Mean training R. [g] Standard deviation of training R. [h] Mean leave-one-out cross-validated R. [i] Standard deviation of leave-one-out cross-validated R.

of this study, it is the model with the highest R rather than R_{CV} that is considered to have the best quality.

Effects of Swarm Parameters. The performance of the particle swarm optimizer is controlled by seven parameters: (1) the population size, (2) the number of iterations, (3) the neighborhood size, (4) the cognitive learning rate η_1, (5) the social learning rate η_2, (6) the inertia ω, and (7) the selection pressure α. In this work, we examined the influence of all these parameters except the neighborhood size and selection pressure, which were set to 5 and 2, respectively. Our study was carried out in two stages. During the first stage, we studied four representative combinations of population size and number of iterations, constraining the total number of function evaluations to 1000 and setting η_1, η_2, and ω to some "reasonable" default values (1, 1, and 0.9, respectively). We found that the results were generally similar as long as the number of particles was sufficiently large. When the population size is very small, the trajectories of the particles are synchronized prematurely, and the swarm converges to the nearest local minimum, which is often suboptimal, losing much of its exploratory ability.

This is nicely illustrated by plotting the number of unique states visited during the search for each pair of simulation parameters (Figure 1). In all three cases, this number decreases sharply with decreasing population size and drops below 20% for populations of size 10. On the other extreme, random search (labeled "1000 0" in Figure 1) is virtually impervious to this problem. The differences among the three data sets reflect the underlying combinatorics, which for the n-choose-k problem are given by the binomial coefficient:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{6}$$

For the AMA, BZ, and PYR feature selection tasks (see Table 1), the number of possible combinations is 23 426, 5 245 786, and 296 010, respectively, which implies that the likelihood of producing unique states decreases in the order BZ > PYR > AMA, as observed in Figure 1.

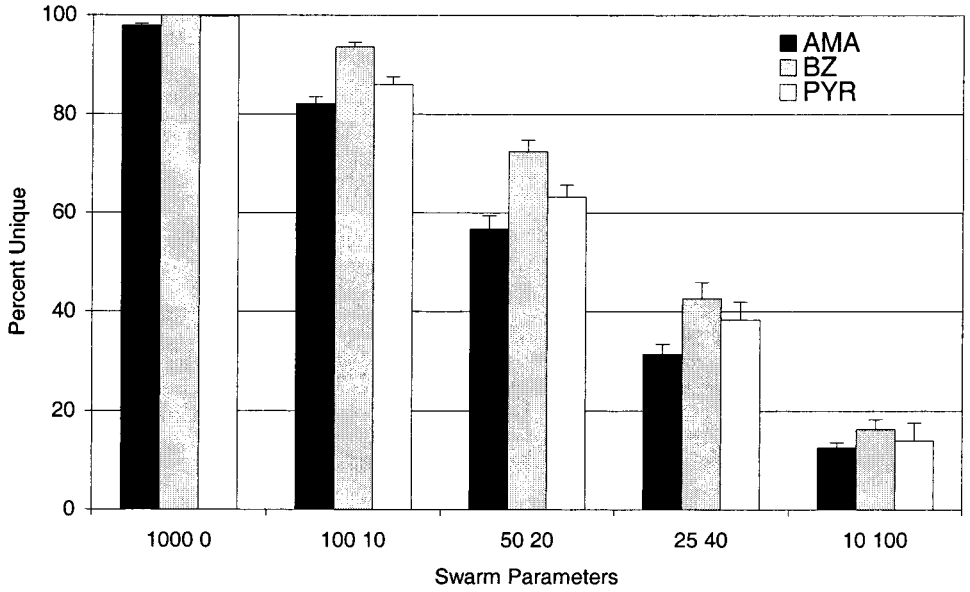

Figure 1. Mean and standard deviation of the percentage of unique states generated in 50 independent runs of several configurations of the particle swarm algorithm, involving a total of 1000 sampling steps.

Table 5. Quality of Models Selected by PS, SA, and Random Selection for the BZ Data Set, with Each Row Representing 50 Independent Optimization Runs

opt[a]	size[b]	steps[c]	η_1[d]	η_2[e]	ω[f]	α[g]	trials[h]	$\mu(R)$[i]	$\sigma(R)$[j]	min(R)[k]	max(R)[l]
PS	100	10	1	1	0.9	2	3	0.946	0.011	0.902	0.960
PS	50	20	1	1	0.9	2	3	0.946	0.013	0.888	0.960
PS	25	40	1	1	0.9	2	3	0.949	0.011	0.909	0.960
PS	10	100	1	1	0.9	2	3	0.944	0.011	0.914	0.958
PS	100	10	1	0.5	0.9	2	3	0.941	0.024	0.813	0.959
PS	100	10	1	0	0.9	2	3	0.938	0.012	0.894	0.958
PS	100	10	0.5	1	0.9	2	3	0.943	0.012	0.900	0.959
PS	100	10	0	1	0.9	2	3	0.945	0.011	0.902	0.961
PS	100	10	1	1	0.5	2	3	0.947	0.011	0.914	0.960
PS	100	10	1	1	0.1	2	3	0.944	0.013	0.900	0.958
PS	100	20	1	1	0.9	2	3	0.949	0.010	0.920	0.960
SA	30	33					3	0.954	0.003	0.946	0.959
SA	30	66					3	0.957	0.002	0.950	0.960
SA	30	133					3	0.958	0.001	0.955	0.960
R	1000						3	0.932	0.018	0.843	0.952

[a] Optimizarion algorithm (PS, particle swarm; SA, simulated annealing; R, random selection). [b] Number of particles in the swarm. [c] Number of iterations. [d] Cognitive learning rate. [e] Social learning rate. [f] Inertia. [g] Selection pressure. [h] Number of back-propagation refinements. [i] Mean training R. [j] Standard deviation of training R. [k] Minimum training R. [l] Maximum training R.

This general trend reflects the classical dilemma between *exploration* and *exploitation* as two opposing strategies in optimization. A good optimizer must be able to simultaneously *explore* the state space to gather information about the environment and *exploit* existing knowledge to increase the reward and ensure that the optimum is identified within a finite number of sampling steps. These two goals are mutually conflicting, and finding the optimum balance between them is the key to constructing a successful optimizer. Moving away from this optimum in either direction is detrimental; in the case at hand, if the swarm is too small, it ceases to explore and becomes trapped in local minima, and if it is too large, it degrades to random search.

The next set of experiments is aimed at investigating the relative impact of the cognitive and social learning rates η_1 and η_2 and the momentum ω. These experiments were based on swarms consisting of 100 particles allowed to evolve for 10 iterations. With regard to the learning rates, we found that in all three cases, strong social learning was essential for good performance and the best results were obtained with equal contributions from the cognitive and social components (see Tables 3, 5, and 7). Although the differences are relatively small, it is clear that particle swarms draw their strength from their cooperative nature and are most effective when nostalgia and envy coexist in a balanced mix. Similar effects were observed with the momentum ω, which controls the ability of the swarm to change direction on the basis of dynamic feedback from its environment. If the momentum is too small, the swarm loses its ability to drift away in search of new experiences and becomes rapidly dominated by its recent past (see Tables 3, 5, and 7). It is a well-known fact that while learning from past experiences is essential for social progress, conformism most often leads to social stagnation and demise.

Comparison with Simulated Annealing. The results in Tables 3, 5, and 7 reveal an interesting difference between particle swarms and simulated annealing. While the average fitness of the resulting models does not lead to any definitive conclusions (particle swarms are better than annealing for AMA, worse for BZ, and comparable for PYR), there is a

Table 6. Best Models Selected by PS[a] and SA[b] for the BZ Data Set (50 Runs, 1000 Training Epochs)

rank	features[c]	freq (PS)[d]	freq (SA)[e]	$\mu(R)$[f]	$\sigma(R)$[g]	$\mu(R_{CV})$[h]	$\sigma(R_{CV})$[i]	rank	features[c]	freq (PS)[d]	freq (SA)[e]	$\mu(R)$[f]	$\sigma(R)$[g]	$\mu(R_{CV})$[h]	$\sigma(R_{CV})$[i]
1	1, 4, 5, 9, 15, 23	1	2	0.963	0.001	0.874	0.019	42	1, 3, 6, 9, 18, 19	0	1	0.951	0.008	0.857	0.008
2	1, 3, 4, 9, 15, 23	0	3	0.963	0.000	0.876	0.007	43	0, 1, 5, 9, 19, 20	1	0	0.951	0.009	0.874	0.008
3	1, 3, 6, 9, 15, 23	0	1	0.962	0.001	0.881	0.010	44	0, 1, 3, 9, 19, 31	0	1	0.950	0.007	0.886	0.014
4	1, 4, 5, 9, 20, 22	1	0	0.962	0.002	0.869	0.018	45	1, 4, 6, 9, 16, 18	0	1	0.950	0.010	0.897	0.011
5	1, 3, 4, 9, 15, 22	0	2	0.961	0.002	0.845	0.033	46	0, 1, 5, 9, 19, 31	1	0	0.949	0.009	0.883	0.005
6	1, 4, 5, 9, 15, 27	0	1	0.961	0.000	0.896	0.003	47	1, 5, 9, 13, 19, 36	1	0	0.949	0.004	0.871	0.007
7	1, 4, 6, 9, 15, 27	1	2	0.961	0.001	0.870	0.019	48	0, 1, 5, 9, 14, 16	1	0	0.949	0.009	0.883	0.023
8	1, 4, 5, 9, 15, 38	0	1	0.960	0.000	0.882	0.018	49	1, 2, 4, 6, 9, 19	1	0	0.949	0.012	0.853	0.021
9	0, 1, 3, 9, 15, 16	0	1	0.960	0.000	0.871	0.009	50	0, 1, 3, 9, 14, 18	0	1	0.949	0.010	0.862	0.008
10	1, 3, 4, 7, 9, 15	1	0	0.960	0.000	0.857	0.016	51	0, 1, 3, 9, 13, 19	0	1	0.948	0.006	0.880	0.010
11	1, 4, 6, 9, 15, 22	2	0	0.960	0.002	0.894	0.014	52	0, 1, 5, 9, 11, 14	0	1	0.946	0.008	0.864	0.016
12	1, 3, 4, 9, 10, 15	0	1	0.960	0.001	0.870	0.003	53	1, 3, 9, 11, 14, 34	0	1	0.946	0.003	0.887	0.007
13	0, 1, 4, 5, 9, 15	1	0	0.959	0.001	0.868	0.014	54	1, 3, 4, 9, 18, 19	2	1	0.946	0.005	0.855	0.011
14	1, 3, 6, 9, 15, 27	3	0	0.959	0.010	0.860	0.016	55	1, 3, 6, 9, 17, 18	0	1	0.946	0.006	0.839	0.005
15	0, 1, 3, 9, 16, 19	0	1	0.959	0.006	0.850	0.016	56	1, 3, 9, 17, 18, 22	0	1	0.945	0.001	0.857	0.023
16	1, 3, 4, 9, 15, 38	1	1	0.959	0.007	0.885	0.007	57	1, 4, 5, 9, 12, 19	1	0	0.945	0.002	0.880	0.013
17	1, 2, 3, 4, 9, 20	1	0	0.959	0.005	0.864	0.004	58	1, 3, 9, 11, 14, 23	0	1	0.945	0.005	0.888	0.003
18	1, 3, 4, 8, 9, 15	1	0	0.959	0.000	0.876	0.022	59	0, 1, 3, 9, 18, 19	1	2	0.945	0.007	0.871	0.019
19	0, 1, 5, 9, 19, 41	1	0	0.958	0.000	0.886	0.011	60	1, 2, 3, 9, 17, 31	0	1	0.945	0.008	0.862	0.014
20	1, 4, 6, 8, 9, 15	0	1	0.958	0.001	0.881	0.007	61	1, 2, 5, 9, 17, 18	0	1	0.943	0.008	0.826	0.015
21	1, 4, 5, 9, 15, 20	1	1	0.958	0.002	0.899	0.017	62	1, 3, 9, 15, 19, 33	1	0	0.943	0.002	0.866	0.019
22	1, 3, 6, 8, 9, 15	1	0	0.957	0.001	0.872	0.023	63	0, 1, 3, 9, 11, 14	0	1	0.942	0.005	0.879	0.013
23	1, 4, 6, 9, 15, 38	1	0	0.957	0.004	0.899	0.009	64	0, 1, 3, 9, 14, 20	0	1	0.942	0.007	0.868	0.014
24	1, 4, 5, 9, 15, 24	1	0	0.957	0.002	0.868	0.044	65	0, 1, 4, 9, 15, 19	1	0	0.942	0.003	0.861	0.014
25	1, 5, 6, 8, 9, 15	1	0	0.957	0.001	0.871	0.019	66	1, 5, 9, 16, 20, 26	1	0	0.941	0.008	0.883	0.006
26	0, 1, 5, 9, 15, 16	0	1	0.956	0.004	0.880	0.005	67	1, 2, 3, 5, 9, 19	1	0	0.940	0.001	0.866	0.016
27	1, 3, 4, 9, 17, 20	1	0	0.954	0.007	0.858	0.008	68	1, 3, 4, 9, 15, 29	0	1	0.940	0.013	0.870	0.010
28	1, 3, 4, 9, 15, 27	0	2	0.953	0.020	0.875	0.005	69	0, 1, 5, 9, 19, 23	1	0	0.939	0.003	0.890	0.010
29	1, 4, 5, 9, 15, 28	0	1	0.953	0.009	0.866	0.030	70	1, 3, 4, 9, 16, 20	0	1	0.938	0.010	0.862	0.009
30	1, 4, 5, 8, 9, 15	1	0	0.953	0.007	0.873	0.020	71	0, 1, 5, 9, 19, 27	0	1	0.937	0.001	0.888	0.007
31	1, 3, 5, 9, 15, 23	1	0	0.953	0.008	0.852	0.016	72	0, 2, 5, 9, 15, 23	0	1	0.935	0.018	0.872	0.013
32	1, 5, 6, 9, 10, 15	1	0	0.953	0.006	0.856	0.007	73	0, 1, 5, 6, 9, 20	1	0	0.934	0.029	0.845	0.010
33	1, 4, 6, 9, 20, 21	1	0	0.953	0.005	0.900	0.019	74	1, 5, 6, 9, 12, 15	1	0	0.933	0.031	0.856	0.009
34	1, 2, 4, 5, 9, 20	1	0	0.952	0.007	0.862	0.006	75	1, 3, 9, 15, 22, 23	0	1	0.932	0.016	0.867	0.009
35	1, 4, 5, 9, 20, 41	1	0	0.952	0.005	0.884	0.017	76	1, 4, 6, 9, 15, 23	0	2	0.930	0.038	0.886	0.018
36	1, 3, 4, 9, 15, 17	1	0	0.952	0.008	0.834	0.024	77	0, 1, 3, 7, 9, 15	1	0	0.928	0.032	0.864	0.002
37	1, 3, 4, 9, 15, 28	1	0	0.952	0.020	0.874	0.008	78	1, 4, 5, 9, 11, 15	1	0	0.920	0.019	0.859	0.020
38	1, 5, 6, 9, 19, 20	1	0	0.951	0.007	0.846	0.015	79	1, 5, 6, 9, 15, 25	1	0	0.919	0.028	0.835	0.014
39	0, 1, 3, 9, 15, 20	0	1	0.951	0.007	0.867	0.005	80	1, 4, 5, 9, 20, 23	0	1	0.917	0.020	0.901	0.005
40	1, 4, 5, 9, 17, 20	1	0	0.951	0.008	0.867	0.035	81	1, 3, 6, 9, 15, 26	1	0	0.916	0.031	0.872	0.010
41	1, 3, 8, 9, 15, 23	0	1	0.951	0.003	0.877	0.019	82	0, 1, 2, 5, 9, 14	0	1	0.915	0.131	0.519	0.424

[a] Swarm parameters: 100 particles, 10 iterations, $\eta_1 = 1$, $\eta_2 = 1$, $\omega = 0.9$, $\alpha = 2$. [b] Annealing parameters: 30 temperature cycles, 33 sampling steps. [c] Zero-based indices of features comprising the best models identified by 50 independent runs of the PS and SA algorithms. [d] Number of times that the PS algorithm converged to this model. [e] Number of times that the SA algorithm converged to this model. [f] Mean training R. [g] Standard deviation of training R. [h] Mean leave-one-out cross-validated R. [i] Standard deviation of leave-one-out cross-validated R.

marked difference in the standard deviations, which are almost 4−10 times larger in PS compared to those in SA. However, the fitness of the best models obtained by each method shows a clear advantage for particle swarms, which discovered better solutions in two of the three data sets. Thus, although annealing does converge with greater precision, it converges to suboptimal models that are perhaps more easily accessible in the fitness landscape. Interestingly enough, increasing the simulation time did not significantly affect either one of the algorithms, with improvements limited to less than 0.003 R units in all three cases. In the case of particle swarms, we found that prolonging the simulation time does not induce any additional sampling and simply leads to repeated evaluation of the same subsets.

To get a better appreciation of the relative distribution of scores, the best models discovered in 50 reference PS and SA runs were retrained using 1000 training epochs and were cross-validated by the leave-one-out procedure described in Methods. The models and their statistics are listed in the order of decreasing R in Tables 4, 6, and 8. The most intriguing results are those for the AMA data set (Table 4), where the best model comprising features 3, 4, and 49 was discovered in nearly half of the PS runs but not even once by SA. This model is substantially more fit than any other model discovered by the two algorithms and lies 0.026 R units above the second best (features 31, 34, and 49), which was found nine times by the particle swarm and only twice by annealing. In fact, this second model is the only one that was discovered by both algorithms, suggesting that each method has a clear and distinct preference for certain types of local minima.

The BZ results present a different situation (Table 6). Here, there are a much larger number of good-quality models and none of the algorithms show any notable preference for any particular solution. Indeed, neither PS nor SA converged more than 2−3 times to the same answer, producing no less than 46 and 42 distinct solutions, respectively, with only 6 of them in common. Although simulated annealing produced slightly better models, the differences in fitness were very small and probably statistically insignificant (the top 25 models were within only 0.006 units from each other on the R scale).

Table 7. Quality of Models Selected by PS, SA, and Random Selection for the PYR Data Set, with Each Row Representing 50 Independent Optimization Runs

opt[a]	size[b]	steps[c]	η_1[d]	η_2[e]	ω[f]	α[g]	trials[h]	$\mu(R)$[i]	$\sigma(R)$[j]	min(R)[k]	max(R)[l]
PS	100	10	1	1	0.9	2	3	0.918	0.019	0.861	0.940
PS	50	20	1	1	0.9	2	3	0.920	0.022	0.859	0.946
PS	25	40	1	1	0.9	2	3	0.916	0.021	0.865	0.944
PS	10	100	1	1	0.9	2	3	0.898	0.034	0.751	0.942
PS	100	10	1	0.5	0.9	2	3	0.904	0.039	0.702	0.941
PS	100	10	1	0	0.9	2	3	0.908	0.021	0.834	0.934
PS	100	10	0.5	1	0.9	2	3	0.913	0.025	0.817	0.938
PS	100	10	0	1	0.9	2	3	0.912	0.031	0.763	0.941
PS	100	10	1	1	0.5	2	3	0.911	0.030	0.807	0.942
PS	100	10	1	1	0.1	2	3	0.909	0.036	0.743	0.935
PS	100	20	1	1	0.9	2	3	0.922	0.018	0.877	0.943
SA	30	33					3	0.920	0.005	0.904	0.931
SA	30	66					3	0.922	0.004	0.909	0.931
SA	30	133					3	0.924	0.003	0.914	0.931
R	1000						3	0.898	0.055	0.561	0.938

[a] Optimizarion algorithm (PS, particle swarm; SA, simulated annealing; R, random selection). [b] Number of particles in the swarm. [c] Number of iterations. [d] Cognitive learning rate. [e] Social learning rate. [f] Inertia. [g] Selection pressure. [h] Number of back-propagation refinements. [i] Mean training R. [j] Standard deviation of training R. [k] Minimum training R. [l] Maximum training R.

The PYR data set falls somewhere in between. As with AMA and BZ, the number of models discovered by both algorithms is disappointingly small (4), with PS producing by far the best solutions (22 of the top 25 models, spanning 0.014R units). In addition, PS generated 43 unique solutions, proving once again that it is capable of unveiling the diversity of the solution space when the models are of comparable fitness. On the other hand, annealing produced only 16 unique models, the most frequent of which ranked 35th (13 times), 46th (8 times), 48th (6 times), 51st (6 times), and 26th (5 times), respectively.

These results suggest that particle swarms are a very competitive optimization technique that offers two important advantages over simulated annealing. The first is the ability to discover better solutions, particularly when there is a significant variation in fitness, and the second is the ability to produce a more diverse set of solutions of comparable quality. We believe that this is achieved by casting a wider net over the state space and capitalizing on the parallel nature of the search. Some researchers may find this uncomforting. After all, an ideal optimizer should be able to discover the global minimum both quickly and reliably. Unfortunately, none of the methods examined here offers that luxury, and as the PYR results demonstrate, consistency may often come at the expense of quality.

A note on cross-validation: The results in Tables 4, 6, and 8 show the futility of comparing optimization algorithms on the basis of cross-validated errors, unless these are the functions that are being explicitly optimized during the search. It is indicative that the models with the best R_{CV} rank 3rd, 80th, and 41st in terms of training R for the AMA, BZ, and PYR data sets, respectively, whereas the models with the highest training R rank 5th, 31st, and 19th in terms of R_{CV}. It is unclear whether the optimal solutions in a cross-validated sense are even listed in Tables 4, 6, and 8. One thing is certain: the two algorithms explore the state space in fundamentally different ways and are greatly affected by the nature of the fitness landscape. If the aim is to produce models that perform well on cross-validation, then the cross-validation error should be taken directly into account *during* feature selection. Unfortunately, the instability of neural networks (i.e., their susceptibility to the training parameters) makes this a very difficult and computationally challenging proposition.

Finally, we should point out that although several other optimization algorithms have been tested on these data sets, a direct comparison is not possible because of the instability problem of neural networks mentioned above. Our results are based on the statistical distribution of R scores collected over a large number of runs and are not as sensitive to initialization. A more systematic comparison of several optimization heuristics in the context of feature selection is currently underway and will be presented in due course.

Swarm Trajectories. Valuable insight into the behavior of particle swarms can be gleaned by projecting the trajectories of the particles onto a two-dimensional plane using nonlinear mapping.[34] This method attempts to visualize a set of objects described by a (dis)similarity or distance matrix onto a low-dimensional display plane in a way that preserves the proximities of the objects as much as possible. More specifically, given a set of k objects, a symmetric matrix, d_{ij}, of dissimilarities between these objects, and a set of images on an n-dimensional display plane $\{\mathbf{y}_i, i = 1, 2, ..., k; \mathbf{y}_i \in \mathscr{R}^n\}$, nonlinear mapping attempts to place $\mathbf{y}_i$ onto the plane in such a way that their Euclidean distances $\delta_{ij} = ||\mathbf{y}_i - \mathbf{y}_j||$ approximate as closely as possible the corresponding values d_{ij}.

To visualize the movement of the swarm population, the location vectors $\mathbf{x}_i(t)$ of all the particles are monitored throughout the simulation and recorded on an $m \times n$ matrix, where m is the number of particles times the number of iterations and n is the total number of features in the input data. At the end of the simulation, the location vectors are mapped from $\mathscr{R}^n$ to $\mathscr{R}^2$ using the Euclidean distance as a measure of similarity. In this work, the actual projection was carried out using a fast nonlinear mapping algorithm developed by our group.[35,36] The result for a typical PS run on the AMA data set involving 100 particles and 50 iterations is illustrated in Figure 2. The three snapshots show the location of the swarm at the beginning (Figure 2a), middle (Figure 2b), and end (Figure 2c) of the simulation. The map has a Kruskal stress of 0.262 and exhibits an interesting structure comprising four partial concentric rings. This structure stems from the

Table 8. Best Models Selected by PS[a] and SA[b] for the PYR Data Set (50 Runs, 1000 Training Epochs)

rank	features[c]	freq (PS)[d]	freq (SA)[e]	$\mu(R)$[f]	$\sigma(R)$[g]	$\mu(R_{CV})$[h]	$\sigma(R_{CV})$[i]
1	0, 5, 10, 11, 19, 22	1	0	0.951	0.002	0.777	0.020
2	0, 2, 5, 10, 19, 22	2	0	0.951	0.007	0.786	0.023
3	5, 8, 11, 16, 19, 22	2	0	0.949	0.012	0.799	0.013
4	0, 4, 5, 10, 19, 22	1	0	0.948	0.006	0.771	0.026
5	5, 8, 10, 11, 19, 22	1	0	0.947	0.009	0.773	0.026
6	0, 1, 5, 10, 19, 22	1	0	0.946	0.004	0.790	0.017
7	0, 5, 10, 16, 19, 22	1	0	0.945	0.018	0.808	0.014
8	4, 5, 8, 10, 19, 22	1	1	0.944	0.006	0.781	0.024
9	0, 5, 10, 19, 21, 22	0	1	0.944	0.003	0.726	0.026
10	0, 5, 8, 10, 19, 22	2	0	0.944	0.004	0.804	0.015
11	0, 5, 10, 14, 19, 22	1	0	0.943	0.004	0.775	0.013
12	2, 5, 6, 16, 19, 22	1	0	0.941	0.005	0.793	0.022
13	5, 8, 10, 19, 20, 22	1	0	0.941	0.004	0.749	0.024
14	0, 5, 10, 19, 22, 25	1	0	0.941	0.005	0.726	0.010
15	0, 5, 10, 12, 19, 22	1	0	0.941	0.012	0.772	0.013
16	0, 5, 16, 19, 21, 22	1	1	0.940	0.011	0.742	0.025
17	2, 5, 6, 10, 19, 22	1	0	0.940	0.006	0.750	0.031
18	5, 8, 10, 18, 19, 22	1	0	0.940	0.011	0.754	0.026
19	0, 3, 5, 10, 19, 22	1	0	0.940	0.015	0.777	0.036
20	5, 8, 10, 13, 19, 22	1	0	0.939	0.010	0.771	0.026
21	0, 5, 6, 10, 19, 22	1	0	0.939	0.007	0.779	0.018
22	0, 4, 6, 10, 19, 22	1	0	0.939	0.016	0.748	0.013
23	5, 7, 10, 11, 19, 22	1	0	0.938	0.007	0.784	0.010
24	1, 4, 5, 10, 22, 25	1	0	0.938	0.016	0.618	0.018
25	5, 6, 10, 18, 19, 22	1	0	0.937	0.009	0.731	0.029
26	3, 5, 11, 16, 19, 22	0	5	0.937	0.009	0.792	0.034
27	2, 5, 8, 9, 19, 22	1	0	0.936	0.008	0.730	0.034
28	5, 8, 11, 19, 22, 25	1	0	0.936	0.006	0.632	0.029
29	3, 5, 6, 10, 19, 22	0	1	0.936	0.009	0.774	0.054
30	5, 11, 16, 19, 21, 22	1	1	0.936	0.009	0.732	0.026
31	5, 6, 16, 19, 21, 22	3	0	0.935	0.010	0.729	0.021
32	5, 6, 10, 19, 21, 22	3	0	0.935	0.014	0.702	0.038
33	0, 3, 5, 16, 19, 22	0	2	0.934	0.012	0.787	0.021
34	5, 8, 16, 19, 21, 22	1	0	0.934	0.021	0.748	0.031
35	3, 5, 6, 16, 19, 22	1	13	0.934	0.009	0.781	0.034
36	1, 5, 10, 22, 25, 26	1	0	0.933	0.013	0.626	0.012
37	0, 3, 10, 19, 22, 23	1	0	0.932	0.019	0.682	0.025
38	5, 10, 11, 19, 22, 25	1	0	0.932	0.007	0.686	0.028
39	1, 4, 10, 22, 23, 25	1	0	0.932	0.018	0.628	0.008
40	5, 10, 11, 19, 21, 22	1	0	0.930	0.016	0.717	0.020
41	1, 2, 5, 8, 19, 22	1	0	0.930	0.002	0.808	0.026
42	5, 6, 12, 16, 19, 22	1	0	0.930	0.017	0.771	0.030
43	1, 2, 4, 5, 19, 21	0	1	0.930	0.007	0.778	0.040
44	5, 8, 9, 19, 22, 25	1	0	0.929	0.003	0.659	0.022
45	0, 4, 16, 19, 22, 25	1	0	0.928	0.026	0.739	0.011
46	2, 3, 5, 6, 19, 22	0	8	0.927	0.012	0.787	0.023
47	4, 5, 10, 11, 19, 22	0	1	0.927	0.010	0.710	0.016
48	1, 2, 3, 5, 19, 22	0	6	0.926	0.004	0.795	0.015
49	1, 10, 19, 22, 25, 26	1	0	0.925	0.005	0.551	0.008
50	4, 5, 10, 19, 20, 22	0	1	0.925	0.007	0.728	0.022
51	2, 3, 5, 16, 19, 22	0	6	0.924	0.005	0.781	0.032
52	0, 1, 2, 16, 19, 21	0	1	0.920	0.015	0.785	0.034
53	3, 5, 8, 10, 21, 25	1	0	0.916	0.019	0.639	0.023

[a] Swarm parameters: 100 particles, 10 iterations, $\eta_1 = 1$, $\eta_2 = 1$, $\omega = 0.9$, $\alpha = 2$. [b] Annealing parameters: 30 temperature cycles, 33 sampling steps. [c] Zero-based indices of features comprising the best models identified by 50 independent runs of the PS and SA algorithms. [d] Number of times that the PS algorithm converged to this model. [e] Number of times that the SA algorithm converged to this model. [f] Mean training R. [g] Standard deviation of training R. [h] Mean leave-one-out cross-validated R. [i] Standard deviation of leave-one-out cross-validated R.

quantized nature of the distance function, which for three-feature subsets can only assume the values 0, $2^{1/2}$, $4^{1/2}$, and $6^{1/2}$ (subsets can only differ by 0, 2, 4, and 6 features, respectively). In the beginning, the particles are spread out across the state space, but as the simulation progresses, they begin to converge and eventually coalesce into a small number of local minima. This result seems to suggest that particle swarms are capable of niching and speciation; at some point in the simulation, the swarm can split into subgroups, which gradually drift away from each other, following their own independent trajectories. This is typical of many population-based algorithms[37,38] and could be exploited to introduce diversity in the resulting models.

A more quantitative measure of the degree of localization can be provided by information theory and the concept of Shannon entropy. Two types of entropies can be defined. The first is the *particle entropy* S_i which is based on the factional coordinates of the location vectors prior to roulette wheel selection:

$$S_i = -\sum_{j=1}^{n} p_{ij} \ln p_{ij} \tag{7}$$

where p_{ij} values are the probabilities computed by eq 5. The second is the *swarm entropy* S, which measures the distribution of features across the entire swarm:

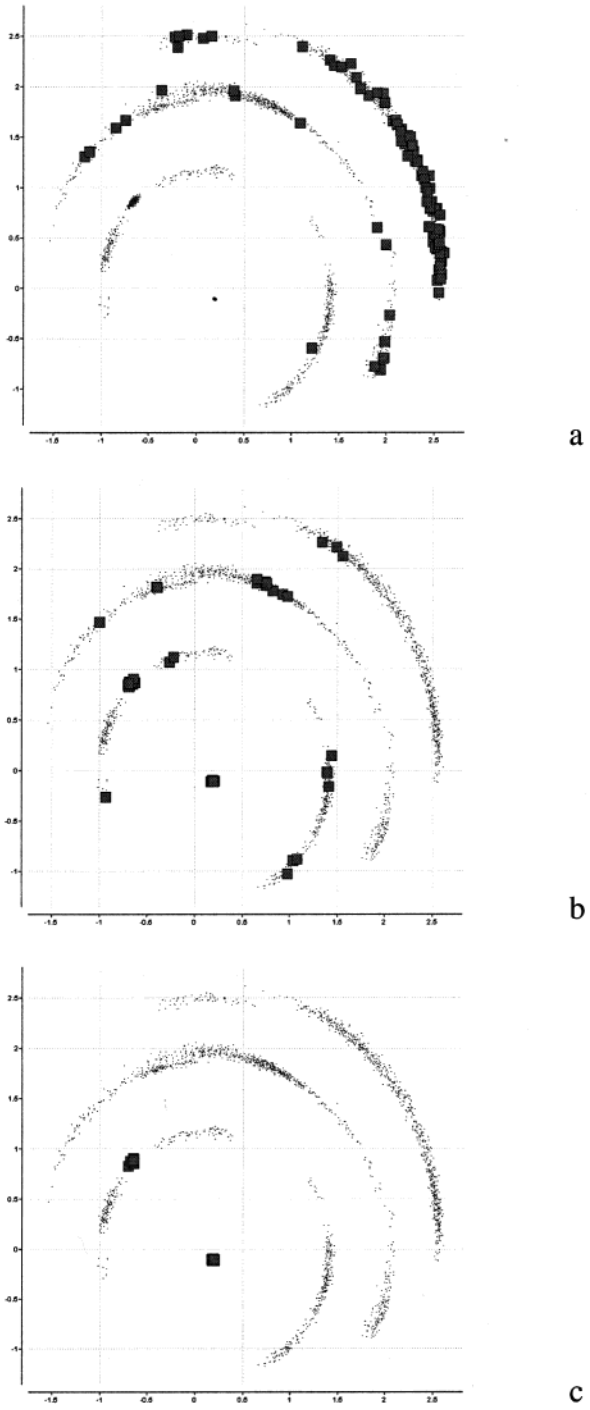

Figure 2. Two-dimensional nonlinear map of the location vectors of the particles generated in a typical run of the particle swarm algorithm for the AMA data set (100 particles, 50 iterations, $\eta_1 = \eta_2 = 1$, $\omega = 0.9$, $\alpha = 2$). Dots represent all visited states, and squares represent the position of the swarm at the (a) beginning, (b) middle (iteration 25), and (c) end of the simulation.

$$S = -\sum_{i=1}^{n} q_i \ln q_i \tag{8}$$

where q_i is the fraction of occurrences of the ith feature in the entire population. Thus, S_i represents the diversity of the environment of the ith particle, whereas S reflects the fixation of features and overall convergence of the binary swarm. These two entropies are closely related but not identical.

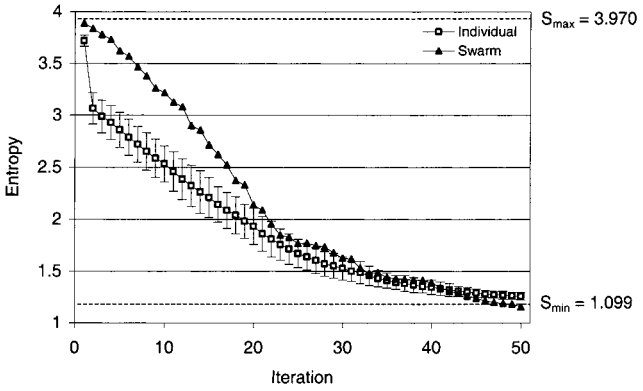

Figure 3. Swarm entropy and mean particle entropy as a function of time for the AMA run described in Figure 2.

A plot of these entropies as a function of time for the same 100-particle, 50-step simulation described above is shown in Figure 3. Both entropy functions are bounded by the two theoretical extrema, $S_{max} = 3.970$ and $S_{min} = 1.099$, which represent maximum diversity (all features are sampled equally) and maximum degeneracy (only three features are sampled), respectively. Throughout the majority of the simulation, the swarm entropy is substantially higher than the average individual entropy, which reflects the collective diversity of the population compared to local neighborhoods. However, as the simulation progresses and the swarm begins to converge, these differences are narrowed and eventually a crossover occurs around the 41st cycle, which manifests the strong selection pressure at the final stages of the optimization.

IV. Conclusions

A promising new algorithm for selecting relevant features for QSAR and QSPR has been described. The method is based on a binary adaptation of particle swarms, an intrinsically parallel optimization paradigm inspired from the study of human sociality. In two out of three data sets that were tested, the method was able to identify a better and more diverse set of solutions than simulated annealing given the same amount of simulation time. The method appears capable of niching and speciation, but unlike simulated annealing, it does not converge as reliably to the same minimum. Nonlinear mapping was shown to be very effective in visualizing the trajectories of the swarms, as were Shannon entropies in quantifying their convergence. Our study involved the construction of nearly 10 000 000 neural network models to ensure that the results were statistically meaningful. While such an effort is not required for routine applications on novel QSAR problems, it is important for validation purposes, where the objective is to assert the superiority of one method over another, and when the underlying predictor is inherently unstable, as is the case with neural networks, classification and regression trees, etc. Finally, the reader is most certainly aware that, as in most QSAR studies of this type, the conclusions may not necessarily extend to other chemical and biological systems. True validation can only come from extensive experimentation in real-world applications.

Acknowledgment. The authors thank Dr. Victor S. Lobanov, Dmitrii N. Rassokhin, and Sergei Izrailev of

3-Dimensionsl Pharmaceuticals, Inc. for many useful discussions and Dr. Raymond F. Salemme for his insightful comments and support of this work.

References

(1) Devillers, J., Ed. *Neural Networks in QSAR and Drug Design*; Academic Press: New York, 1996.
(2) van de Waterbeemd, H., Ed. Chemometric methods in molecular design. In *Methods and Principles in Medicinal Chemistry*; VCH: Weinheim, Germany, 1995; Vol. 2.
(3) Hansch, L.; Leo, C. *Exploring QSAR. Fundamentals and Applications in Chemistry and Biology*; American Chemical Society: Washington, DC, 1996.
(4) Andrea, T. A.; Kalayeh, H. Application of neural networks in quantitative structure−activity relationships of dihydrofolate reductase inhibitors. *J. Med. Chem.* **1991**, *34*, 2824−2836.
(5) So, S.-S.; Richards, W. G. Applications of neural networks: quantitative structure−activity relationhips of the derivatives of 2,4-diamino-(5-substituted-benzyl)pyrimidines as DHFR inhibitors. *J. Med. Chem.* **1992**, *35*, 3201−3207.
(6) Manallak, D. T.; Ellis, D. D.; Livingston, D. J. Analysis of linear and nonlinear QSAR data using neural networks. *J. Med. Chem.* **1994**, *37*, 3758−3767.
(7) John, G.; Kohavi, R.; Pfleger, J. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*; Cohen, W. W., Hirsh, H., Eds.; Rutgers University: New Brunswick, NJ, 1994; pp 121−129.
(8) Selwood, D. L.; Livingstone, D. J.; Comley, J. C. W.; O'Dowd, A. B.; Hudson, A. T.; Jackson, P.; Jandu, K. S.; Rose, V. S.; Stables, J. N. *J. Med. Chem.* **1990**, *33*, 136−142.
(9) Sutter, J. M.; Dixon, S. L.; Jurs, P. C. Automated descriptor selection for quantitative structure−activity relationships using generalized simulated annealing. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 77−84.
(10) Luke, B. T. Evolutionary programming applied to the development of quantitative structure−activity relationships and quantitative structure−property relationships. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1279−1287.
(11) Rogers, D. R.; Hopfinger, A. J. Application of genetic function approximation to quantitative structure−activity relationships and quantitative structure−property relationships. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 854−866.
(12) So, S.; Karplus, M. Evolutionary optimization in quantitative structure−activity relationship: an application of genetic neural networks. *J. Med. Chem.* **1996**, *39*, 1521−1530.
(13) Yasri, A.; Hartsough, D. Toward an optimal procedure for variable selection and QSAR model building. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1218−1227.
(14) Hasegawa, K.; Miyashita, Y.; Funatsu, K. GA strategy for variable selection in QSAR studies: GA-based PLS analysis of calcium channel antagonists. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 306−310.
(15) Izrailev, S.; Agrafiotis, D. K. A new method for building regression tree models for QSAR based on artificial ant colony systems. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 176−180.
(16) Izrailev, S.; Agrafiotis, D. K. Variable selection for QSAR by artificial ant colony systems. *SAR QSAR Environ. Res.*, in press.
(17) Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995; IEEE Service Center: Piscataway, NJ, 1995; pp 39−43.
(18) Kennedy, J.; Eberhart, R. C. Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, 1995; IEEE Service Center: Piscataway, NJ, 1995; Vol. IV, pp 1942−1948.
(19) Eberhart, R.; Shi, Y. Comparison between genetic algorithms and particle swarm optimization. Presented at the 7th Annual Conference on Evolutionary Programming, San Diego, CA, 1998.
(20) Haykin, S. *Neural Networks*; Macmillan: New York, 1994.
(21) Kennedy, J. The particle swarm: social adaptation of knowledge. In *IEEE International Conference on Evolutionary Computation*, Indianapolis, IN, 1997; IEEE Service Center: Piscataway, NJ, 1997; pp 303−308.
(22) Shi, Y. H.; Eberhart, R. C. A modified particle swarm optimizer. Presented at the IEEE International Conference on Evolutionary Computation, Anchorage, AL, 1998.
(23) Shi, Y. H.; Eberhart, R. C. Parameter selection in particle swarm optimization. Presented at the 7th Annual Conference on Evolutionary Programming, San Diego, CA, 1998.
(24) Rao, C. R. Some problems involving linear hypotheses in multivariate analysis. *Biometrika* **1959**, *46*, 49−58.
(25) Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by simulated annealing. *Science* **1983**, *220* (459), 671−680.
(26) Agrafiotis, D. K. Stochastic algorithms for maximizing molecular diversity. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 841−851.
(27) Rassokhin, D. N.; Agrafiotis, D. K. Kolmogorov−Smirnov statistic and its applications in library design. *J. Mol. Graphics Modell.* **2000**, *18*, 370−384.
(28) Wikel, J. H.; Dow, E. R. The use of neural networks for variable selection in QSAR. *Bioorg. Med. Chem. Lett.* **1993**, *3*, 645−651.
(29) Maddalena, D. J.; Johnson, G. A. R. Prediction of receptor properties and binding affinity of ligands to benzodiazepine/$GABA_A$ receptors using artificial neural networks. *J. Med. Chem.* **1995**, *38*, 715−724.
(30) So, S. S.; Karplus, M. Genetic neural networks for quantitative structure−activity relationships: improvements and application to benzodiazepine affinity for benzodiazepine/$GABA_A$ receptors. *J. Med. Chem.* **1996**, *39*, 5246−5256.
(31) Hirst, J. D.; King, R. D.; Sternberg, M. J. E. *J. Comput.-Aided Mol. Des.* **1994**, *8*, 405−420.
(32) Agrafiotis, D. K.; Bone, R. F.; Salemme, F. R.; Soll, R. M. U.S. Patents 5,463,564 (1995), 5,574,656 (1996), 5,684,711 (1997), and 5,901,069 (1999).
(33) Copyright by 3-Dimensional Pharmaceuticals, Inc., 1994−2000.
(34) Sammon, J. W. *IEEE Trans. Comput.* **1969**, *C18*, 401−409.
(35) Agrafiotis, D. K.; Lobanov, V. S. Nonlinear mapping networks. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 1356−1362.
(36) Rassokhin, D. N.; Lobanov, V. S.; Agrafiotis, D. K. Nonlinear mapping of massive data sets by fuzzy clustering and neural networks. *J. Comput. Chem.* **2001**, *22*, 373−386.
(37) Cedeño, W.; Vemuri, V. Analysis of Speciation and Niching in the Multi-Niche Crowding GA. *Theor. Comput. Sci.* **1999**, *229*, 177−197.
(38) Vemuri, V.; Cedeño, W. Multi-niche crowding for multimodal search. In *Practical Handbook of Genetic Algorithms and Applications*; Chambers, L., Ed.; CRC Press: Boca Raton, FL, 1995; Vol. II.

JM0104668