1 Evolving Heterogeneous Neural Agents by Local Selection

Filippo Menczer, W. Nick Street, and Melania Degeratu

Evolutionary algorithms have been applied to the synthesis of neural architectures, but they normally lead to uniform populations. Homogeneous solutions, however, are inadequate for certain applications and models. For these cases, *local selection* may produce the desired heterogeneity in the evolving neural networks. This chapter describes algorithms based on local selection, and discusses the main differences distinguishing them from standard evolutionary algorithms. The use of local selection to evolve neural networks is illustrated by surveying previous work in three domains (simulations of adaptive behavior, realistic ecological models, and browsing information agents), as well as reporting on new results in feature selection for classification.

1.1 Introduction

The synthesis of neural architectures has been among the earliest applications of evolutionary computation [60, 1, 50, 13]. Evolutionary algorithms have been used to adjust the weights of neural networks without supervision [51, 46], to design neural architectures [49, 28, 20, 48], and to find learning rules [5].

Evolutionary algorithms, however, typically lead to uniform populations. This was appropriate in the above applications, since some optimal solution was assumed to exist. However, homogeneous solutions — neural or otherwise — are inadequate for certain applications and models, such as those requiring *cover* [14, 36] or *Pareto* [58, 24] optimization. Typical examples stem from expensive or multi-criteria fitness functions; in these cases, an evolutionary algorithm can be used to quickly find a set of alternative solutions using a simplified fitness function. Some other method is then charged with comparing these solutions.

Selection schemes have emerged as the aspect of evolutionary computation that most directly affects heterogeneity in evolutionary algorithms. In fact, selective pressure determines how fast the population converges to a uniform solution. The *exploration-exploitation* dilemma is commonly invoked to explain the delicate tension between an algorithm's efficiency and its tendency to prematurely converge to a suboptimal solution.

Parallel evolutionary algorithms often impose geographic constraints on evolutionary search to assist in the formation of diverse subpopulations [19, 8]. The motivation is in avoiding the communication overhead imposed by standard selection schemes; different processors are allocated to subpopulations to minimize inter-process dependencies and thus improve efficiency. The poor match between parallel implementations and the standard notion of optimization by convergence is noted for example by McInerney [36], who distinguishes between *convergence* — all individuals converging on the best solution — and *cover* — all good solutions being represented in the population — as measures of successful termination. Parallel evolutionary algorithms are more amenable to cover optimization than to standard convergence criteria, due to the limited communication inherent in most parallel implementations.

The problem of ill-convergence exhibited by traditional selection schemes is related to the issue of *niching*. In a niched evolutionary algorithm, according to Goldberg [14], stable subpopulations would ultimately form around each of the local fitness optima. Individual solutions in such subpopulations would be allocated in proportion to the magnitude of the fitness peaks.

Although there is a well-developed biological literature in niching [22], its transfer to artificial evolutionary search has been limited [14]. Standard evolutionary algorithms are ineffective for niching, due to high selective pressure and premature convergence [12]. Several methods have been devised to deal with this problem by maintaining diversity. One example for proportional selection is to tune the selective pressure adaptively, by a nonlinear scaling of the fitness function [47]. Different selection methods of course impose varying degrees of selection pressure. For example, tournament selection is known to converge slowly and to have niching effects [15].

The most notable selection variations explicitly aimed at niching are *crowding* [9, 31] and *fitness sharing* [17, 23, 32]. In both of these methods, selection is somehow altered to take into account some measure of similarity among individuals. Shortcomings of both methods are problem-dependency and inefficiency; if p is the population size, selection with sharing or crowding has time complexity $O(p)$ rather than $O(1)$ *per individual*. The slowdown can be important for practical cases with large populations, and computing similarity imposes a large communication overhead for parallel implementations. Moreover, even assuming that the number of niches H is known a priori, it is estimated that the population size required to maintain the population across niches grows rapidly with H [33]. The role of selection for multi-criteria and parallel optimization remains an active area of research in the evolutionary computation community [21, 34, 24].

This chapter discusses the *locality* of a selection scheme and its effects on an evolutionary algorithm's behavior with respect to convergence. We can loosely define *local selection* (LS) as a selection scheme that minimizes interactions among individuals. Locality in selection schemes has been a persistent theme in the evolutionary computation community [16, 18, 10, 34].

The chapter is organized as follows. Section 1.2 describes ELSA, an evolutionary algorithm based on a local selection scheme, which lends itself naturally to maintaining heterogeneous populations for cover optimization and multi-criteria applications. We illustrate the algorithm and outline its main distinctions from other evolutionary algorithms. Sections 1.3 and 1.4 demonstrate the problems and advantages of local selection by discussing its application in four neural domains, for cover and Pareto optimization, respectively. We show that although local selection is not a panacea, it may produce heterogeneous populations of neural networks when such diverse solutions are called for. Finally, in section 1.5 we summarize our conclusions and consider directions for further applications of local selection.

1.2 Evolution by Local Selection

Our original motivation for considering local selection in evolutionary algorithms stemmed from an interest in ecological modeling [41, 40]. Local selection is a more realistic reproduction scheme in an evolutionary model of real populations of organisms. In such a model, an agent's fitness must result from individual interactions with the environment, which contains shared resources along with other agents, rather than from global interactions across the population.

The Algorithm

We can best characterize local selection and succinctly describe its differences from global schemes by casting the evolutionary algorithm into an ecological framework. The resulting algorithm, which we call ELSA (Evolutionary Local Selection Algorithm), is illustrated at a high level of abstraction in figure 1.1.

Each agent (candidate solution) in the population is first initialized with some random solution and an initial reservoir of *energy*. If the algorithm is implemented sequentially, parallel execution of agents can be simulated with randomization of call order.

In each iteration of the algorithm, an agent explores a candidate solution (possibly including an action) similar to itself. The agent is taxed with E_{cost} for this action and collects ΔE from the environment. In the applications illus-

```
initialize population of $p_0$ agents, each with energy $\theta/2$
while there are alive agents
    for each agent $a$
        $a' \leftarrow perturb(a)$
        $\Delta E \leftarrow e(Fitness(a'), E_{envt})$
        $E_{envt} \leftarrow E_{envt} - \Delta E$
        $E_a \leftarrow E_a + \Delta E$
        $E_a \leftarrow E_a - E_{cost}$
        if ($E_a > \theta$)
            $new(a')$
            $E_{a'} \leftarrow E_a/2$
            $E_a \leftarrow E_a - E_{a'}$
        else if ($E_a < 0$)
            $die(a)$
        end
    end
    $E_{envt} \leftarrow E_{envt} + E_{replenish}$
end
```

Figure 1.1
ELSA pseudo-code.

trated in this chapter, E_{cost} for any action is a constant unless otherwise stated. The net energy intake of an agent, expressed by the function $e()$, depends both on its fitness and on the state of the environment, i.e., on the number of other agents considering similar solutions (either in search or in criteria space, depending on the application). This is equivalent to a sort of environment-mediated crowding. Actions result in energetic benefits only inasmuch as the environment has sufficient energetic resources; if these are depleted, no benefits are available until the environmental resources are replenished.

In the selection part of the algorithm, an agent compares its current energy level with a threshold θ. If its energy is higher than θ, the agent reproduces. The mutated clone that was just evaluated becomes part of the population, with half of its parent's energy. When an agent runs out of energy, it is killed.

The environment acts as a data structure that keeps track of the net effects of the rest of the population. This way, direct communications between individuals (such as comparisons, ranking, or averaging of fitness values) become unnecessary, and the only interactions consist in the indirect competition for the finite environmental resources.

If we want to maintain the population average around some fixed value p_0 irrespective of problem size, we can let

$$E_{replenish} = p_0 \cdot E_{cost}. \tag{1.1}$$

In fact, since energy is conserved, the average amount of energy that leaves the system per unit time (through costs) has to be equal to the amount of energy that enters the system per unit time (through replenishment):

$$\begin{aligned}\langle pE_{cost}\rangle &= E_{replenish}\\ \langle p\rangle E_{cost} &= p_0 E_{cost}\\ \langle p\rangle &= p_0\end{aligned}$$

where $\langle\cdot\rangle$ indicates time average.

In an implementation based on the pseudo-code of figure 1.1, some other details must be filled in. In particular, for neural agents, a solution can be represented by the weight vector of the neural net. If crossover is to be used, a candidate network can be recombined with another member of the population before being evaluated or before being inserted into the population, in case the parent is selected for reproduction. There has been ample discussion in the literature about the feasibility of recombination in the evolution of neural networks (see, e.g., [47, 46]), but such discussion is outside the scope of this chapter. Mutation is the only genetic operator used in the domains discussed in this chapter. The mutation operator provides evolving neural nets with a local search step, and the details of its dynamics are discussed for each task-specific application.

Local versus Global Selection

Selection is the central point where this algorithm differs from most other evolutionary algorithms. Here an agent may die, reproduce, or neither (corresponding to the solution being eliminated from the pool, duplicated, or maintained). Energy is always conserved. The selection threshold θ is a constant independent of the rest of the population — hence selection is *local*. This fact reduces communication among agent processes to a minimum and has several positive consequences.

First, two agents compete for shared resources only if they are situated in the same portion of the environment (i.e., of the search or criteria space, depending on the application). It is the environment that drives this competition and the consequent selective pressure. No centralized decision must be made about how long an agent should live, how frequently it should reproduce, or when it should die. The search is biased directly by the environment.

Second, LS is an implicitly niched scheme and therefore it naturally enforces the maintenance of population diversity. This makes the search algorithm more amenable to cover and multi-modal optimization than to standard convergence criteria. The bias is to exploit all resources in the environment, rather than to locate the single best resource.

Third, the size of the population, rather than being determined a priori, emerges from the *carrying capacity* of the environment. This is determined by (i) the costs incurred by any action, and (ii) the replenishment of resources. Both of these factors are independent of the population.

Finally, the removal of selection's centralized bottleneck makes the algorithm parallelizable and therefore amenable to distributed implementations. ELSA is therefore an ideal candidate to study the potential speedup achievable by running agents on multiple remote hosts.

Local selection of course has disadvantages and limitations as well. Imagine a population of agents who can execute code on remote servers in a distributed environment, but have to look up data on a central machine for every action they perform. A typical example of such a situation would be a distributed information retrieval task in which agents share a centralized page cache. Because of communication overhead and synchronization issues, the parallel speedup achievable in this case would be seriously hindered. As this scenario indicates, the feasibility of distributed implementations of evolutionary algorithms based on local selection requires that the environment can be used as a data structure. Like natural organisms, agents must be able to "mark" the environment so that local interactions can take advantage of previous experience.

Local selection algorithms cannot immediately be applied to any arbitrary problem. First, a problem space may not lend itself to being used as a data structure. For example, marking the environment in continuous function optimization with arbitrary precision might hinder discretization and thus compromise the feasibility of local data structures. Second, it may be difficult to devise an isomorphism of the problem such that the environmental resource model could be applied successfully. For example, associating environmental resources to partial solutions of a combinatorial optimization problem may require a decomposition property that the problem is not known to possess.

In a multi-criteria or distributed task, the environment models the problem space and the resources that are locally available to individual solutions. It is in such cases that the distinction between local and global interactions among individuals becomes important; the selection mechanism and environmental resource model capture the nature of such interactions. In a standard evolutionary algorithm, an individual is selected for reproduction based on how its

Table 1.1
Schematic comparison between local and global selection schemes. r-selection and K-selection refer to population models commonly used in ecology, which assume infinite and bounded resources, respectively.

Feature	Global selection	Local selection
reproduction threshold	$\theta = f(E_1, \ldots, E_{pop})$	$\theta = const$
search bias	exploitation	exploration
adaptive landscape	single-criterion	multi-criteria
convergence goal	single-point	cover
solution quality	best (fragile)	good (robust)
biological equivalent	r-selection	K-selection

fitness compares with the rest of the population. For example, proportional selection can be modeled by a selection threshold $\langle E \rangle$, where $\langle \cdot \rangle$ indicates population average, for both reproduction (in place of θ) and death (in place of 0). Likewise, binary tournament selection can be modeled by a selection threshold E_r where the subscript r indicates a randomly picked individual. In local selection schemes, θ is independent of the rest of the population and the computations that determine whether an individual should die or reproduce can be carried out without need of direct comparisons with other individuals. Table 1.1 illustrates schematically the main features that differentiate the two classes of selection schemes.

1.3 Heterogeneous Neural Agents for Cover Optimization

We refer to neural agents here to mean evolving agents whose genetic representation is, or is associated with, a neural network. Many applications have been tackled with neural systems, and when examples are not available to supervise the training of the neural network, evolutionary algorithms represent one way to train the network in an unsupervised fashion — provided an evaluation function is available to compute fitness.

In this section we summarize previous work done in three different domains. In each case, a local selection algorithm similar to ELSA was used to evolve a population of neural agents for a distinct task. Each of the following subsections describe one such domain, motivates the use of the evolutionary local selection algorithm, and draws some observations based on the main results.

Coevolving Sensors and Behaviors in Toy Environments

The first domain we consider is an artificial life model simulating environments constructed *ad-hoc* to study the evolution of sensory systems.

Sensors represent a crucial link between the evolutionary forces shaping a species' relationship with its environment, and the individual's cognitive abilities to behave and learn. We used *latent energy environments* (LEE) models to define environments of carefully controlled complexity, which allowed us to state bounds for random and optimal behaviors, independent of strategies for achieving the behaviors. A description of the LEE framework is outside the scope of this chapter, and the interested reader is referred to [41, 40].

LEE provided us with an analytic basis for constructing the environments, a population of neural networks to represent individual behaviors, and an evolutionary local selection algorithm to model an adaptive process shaping the neural nets, in particular their input (sensory) system and their connection weights [39]. The experiments considered different types of sensors, and different sets of environmental resources that such sensors would detect. The idea was to study the conditions under which *efficient* (informative) sensory systems could be found by evolutionary synthesis.

The optimal sensory system was known by design, given the task faced by the LEE agents. Each agent had sensors providing perceptual information about its external (local) environment and sensors providing memory information about its internal ("gut") environment. The task involved navigating in a grid world and eating appropriate combinations of resources: if the gut contained a *dark* resource element, then the agent would gain energy by eating a *light* element, and vice versa. Combining two elements of the same resource would result in a loss of energy. The situation is illustrated in figure 1.2.

Evolving the weights of neural agents with optimal sensors resulted in behaviors corresponding to a carrying capacity approximately 3.5 times that of populations of random walkers (agents with "blind" sensors). However, evolving both neural network weights and sensory systems together yielded significant subpopulations with suboptimal sensors, and high variability of emerging behaviors caused by genetic drift. Optimal sensor configurations were not robust in the face of mutations, because they required finely tuned neural net weights. Conversely, many suboptimal sensory systems were robust because they could share weights, and the corresponding behaviors would not be necessarily disrupted by sensor mutations.

The genetic drift effect was overcome by allowing the neural nets to learn

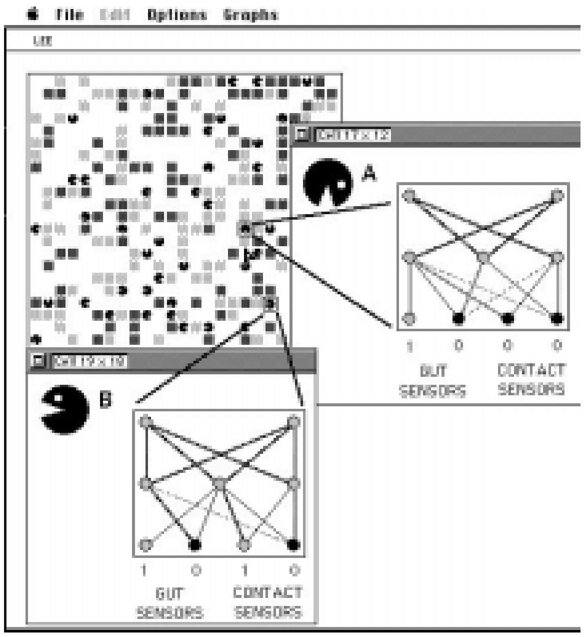

Figure 1.2
Two agents in similar circumstances but different sensory systems. Both agents have a light element in their gut and one in front of them. The right move would be to veer away. Agent A has inefficient sensors, because both of its contact sensors only detect dark elements in the cell facing the agent. So the light element is not detected, and the agent might lose energy by going forward and eating it. Agent B has an efficient sensory system. Its contact sensors detect the light element in front of it, and the agent will be able to perform the right action.

via reinforcement learning during an agent's life. The weight changes due to learning were not inherited by offspring, however, so that learning was non-Lamarckian. This indirect interaction between learning and evolution was an example of the so-called *Baldwin effect* [2]. This phenomenon provided the neural agents with significantly more efficient sensory systems, as illustrated in figure 1.3.

An important observation can be drawn from these experiments regarding the use of evolutionary local selection algorithms with neural agents. Heterogeneity is not always a good thing: for this task, the selective pressure was insufficient to evolve both informative sensory systems and connection weights that could exploit the sensory information. Evolutionary local selection algorithms are appropriate when we want to maintain a diverse population of local,

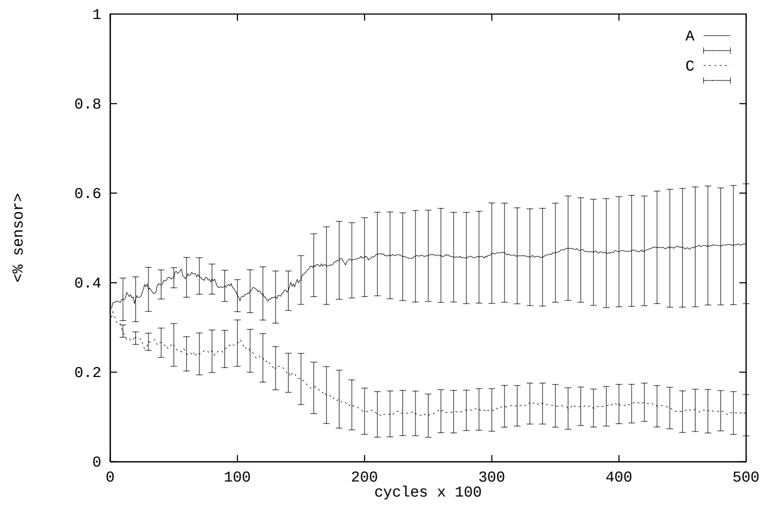

Figure 1.3
Percentages of efficient sensors (A) versus inefficient sensors (C) evolved in simulations with non-Lamarckian reinforcement learning during life. (Reprinted from [39] with permission.)

possibly suboptimal solutions, not when we are trying to converge to some optimal solution.

Foraging Efficiently in Marine Environments

Our second domain is a somewhat more realistic model of ecological environment, in which we study the evolution of behaviors and compare the predictions of the model with actual field data.

Tropical oceanic fishes, such as tunas, billfishes, and sharks, live together in the same environment — the pelagic open ocean. *In situ* experiments have been developed to observe horizontal and vertical movements of individuals of different pelagic predator species, but there is a need for models that explain the evolutionary origin of such movement behaviors, and their relationships with these fishes' shared biotic environment or with other constraints, e.g., physiological ones.

We proposed a minimal model the explore the hypothesis that movement behaviors are driven entirely by prey distributions and dynamics, and tested the predictive power of this model with simulations based on the LEE framework

[6]. In these experiments, the behaviors of an evolving population of artificial fishes adapted in a three-dimensional environment. Spatial distributions, energetic values, and temporal dynamics (depth versus time of day) of two types of prey were based on data from observations of the open ocean in French Polynesia.

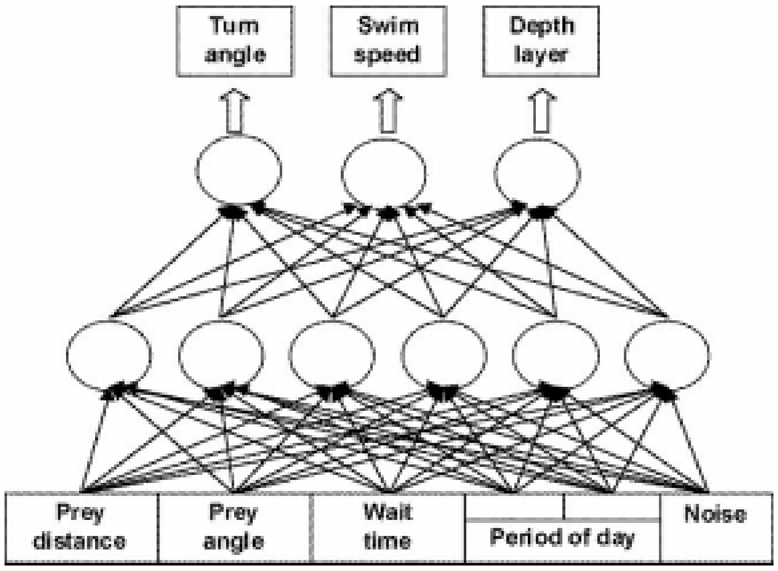

Figure 1.4
Architecture of the neural networks used to model the behaviors of tropical oceanic pelagic predator fishes.

Individuals were represented by simple neural agents as illustrated in figure 1.4. Agents could sense prey from the closest patch, without discerning prey types. They also had sensors indicating the time elapsed since the last eaten prey and the time of day. The energy spent swimming was a function of the swimming speed.

The evolutionary local selection algorithm was used to evolve a heterogeneous set of agents. At reproduction, mutations would add random noise (drawn from a uniform distribution between -1 and $+1$) to approximately 2% of the weights, randomly selected. All weights were bounded to the interval $[-5, +5]$.

MIT Press Math6X9/1999/09/30:19:43 Page 12

Table 1.2
Correspondence between the most frequent vertical swimming patterns as predicted by the evolutionary model based on local selection and as observed in the most common tropical oceanic predator species. (Data from [6].)

Model frequency rank	Nighttime layer	Daytime layer	Species
1	surface	intermediate	Albacore tuna
2	surface	surface	Skipjack tuna, Striped marlin, Pacific blue marlin
2	surface	intermediate, surface	Yellowfin tuna
4	surface	intermediate, deep	Blue shark
5	surface	deep	Bigeye tuna, Swordfish

The movement behaviors of evolved neural agents across ten simulations were analyzed and classified into patterns, then compared with those observed in real fishes (four species of tunas, three species of billfish, and one species of shark), mostly by acoustic telemetry. Beyond our expectations, most of the artificial individuals evolved vertical patterns virtually identical to those exhibited by fishes in the wild. The agreement between this simple computational model and ethological data, illustrated in table 1.2, validated the use of minimal assumption models for the study of behaviors in multi-species ecosystems.

From the viewpoint of the evolutionary synthesis of these neural agents, the main observation we draw from this work is that local selection allowed for the concurrent evolution of many heterogeneous behaviors to model this co-adapted multi-species system. Had a standard evolutionary algorithm been applied to this model, the population would have likely converged to one of the observed patterns. Given that the purpose of the experiment was not to search for one “optimal” behavior, but rather to identify a set of behaviors that could be compared with those of actual species, local selection was key in achieving this goal.

Browsing Adaptively in Information Environments

The third domain in which we have applied evolutionary local selection algorithms is the search for relevant information in documents across the Internet. Distributed information retrieval is an lively area of research due to the popularity of the Web and the scalability limitations of search engine technology [37, 30]. We tested the feasibility of local selection algorithms for distributed information retrieval problems by building artificial graphs to model different aspects of Web-like search environments [42, 38].

In each experiment we constructed a large graph, where each node was associated with some payoff. Nodes, edges, and payoffs modeled hypertext documents, hyperlinks, and relevance, respectively. The population of agents visited the graph as agents traversed its edges. The idea was to maximize the collective payoff of visited nodes, given that there would be only time to visit a fraction of the nodes in the graph. Since the modeled search graph is typically distributed across remote servers, agents were charged costs for traversing edges and evaluating nodes' payoff. Each link l was also associated with a feature vector with components $f_1^l, \ldots, f_{N_f}^l \in [0, 1]$. These features modeled environmental cues, such as word frequencies, that could guide a browsing neural agent toward relevant nodes.

Each agent's genotype comprised a single-layer neural net or perceptron, i.e., a weight vector with components $w_1, \ldots, w_{N_f+1} \in R$. Link feature vectors were used as inputs by these neural agents. Node payoffs and link feature vectors were constructed in such a way as to guarantee the existence of a set of weights that, if used by a neural agent, would yield an "accurate" prediction of the payoff of the node pointed to by the input link. The prediction accuracy of such optimal weight vector was a user-defined parameter.

The agent's representation also specified the node on which the agent was currently situated. Therefore an action consisted, first of all, of evaluating each outlink from the current node. The outputs of the neural net would represent the agent's predictions of the payoffs of the nodes reachable from the current node. The agent then followed a link picked by a stochastic selector. As for the other applications, some fraction of the weights were mutated by additive uniform noise at reproduction.

The energetic benefit of an action was the payoff of the newly visited node, provided it had not been previously visited by any agent. Nodes were therefore "marked" to keep track of used resources (no replenishment). A constant energy cost was charged for any node visited. The goal was to evolve agents with optimal genotypes, enabling them to follow the best links and thus achieve maximum payoff intake.

To gauge the performance of ELSA in the graph search problem, we compared local selection with a traditional evolutionary algorithm. To this end we replaced local selection by binary tournament selection because the latter scheme was "traditional" (i.e., global) and yet it did not require operations such as averaging, and thus it fit naturally within the steady-state framework of ELSA. Basically the same algorithm of figure 1.1 was used, with the difference that the energy level of a randomly chosen member of the population was used in place of both θ for reproduction and 0 for death.

We ran a set of experiments on random graphs with 1000 nodes, an average fanout of 5 links, and 16 features per link. Nodes with payoff above some threshold were considered "good." The algorithm was stopped when 50% of all the nodes had been visited. Then the fraction of good nodes found by the population up to that point (*recall*) was recorded. The random graphs were generated according to several distinct parameterizations. Across all such parameterizations, ELSA significantly and consistently outperformed tournament selection. Local selection populations continued to discover a constant rate of good nodes, while tournament populations tended to converge prematurely. The improvement depended on the graph parameters, but was generally between two- and ten-fold.

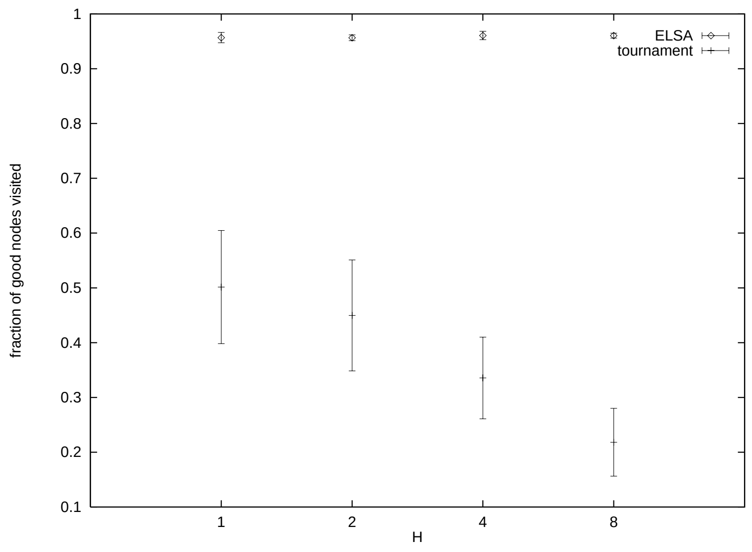

Figure 1.5
Performance of ELSA and tournament selection on graph search problems with various values of H. (Data from [42].)

An example of this performance is shown in figure 1.5. Here, the algorithms were tested on graphs with various values of H, the number of "good clusters." Good nodes were clustered in the sense that they had a higher probability to be linked to other nodes in the same cluster than to nodes in other clusters or to "bad" nodes. Increasing H made the problem multi-modal, and therefore tournament selection degraded in performance due to premature convergence.

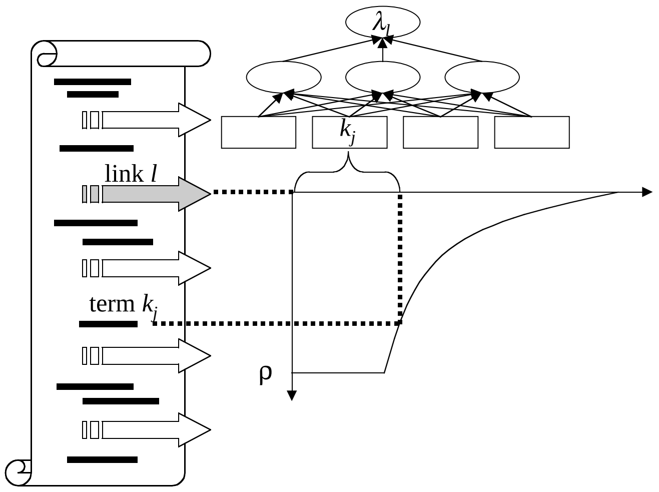

Figure 1.6
How an *InfoSpider* estimates each link from the current document. An agent genotype comprises both a neural net and a vector of terms, one per input. For each link in the document, each input of the neural net is computed by counting the document words matching the keyword corresponding to that input, with weights that decay with distance from the link. (Reprinted from [43] with permission.)

A more sophisticated representation of neural agents, called *InfoSpiders*, is shown in figure 1.6. These agents adapt to user preferences and to local context through both reinforcement learning and an evolutionary local selection algorithm. Evolution is used also to select word features, which determine the inputs of the neural net. Reinforcement learning is used to adjust the connection weights and improve link estimation during an agent's life. *InfoSpiders* have been used to search through actual Web data with very promising preliminary results [45, 43]. For example, figure 1.7 shows the result of a comparison in performance between *InfoSpiders* and best-first-search, on a hypertext collection from Encyclopaedia Britannica Online [38]. *InfoSpiders* had a significant advantage when the relevant set was not too many links away from the starting page. These results point to the need for hybrid systems that use search engines to provide good starting points and *InfoSpiders* to search for recent information.

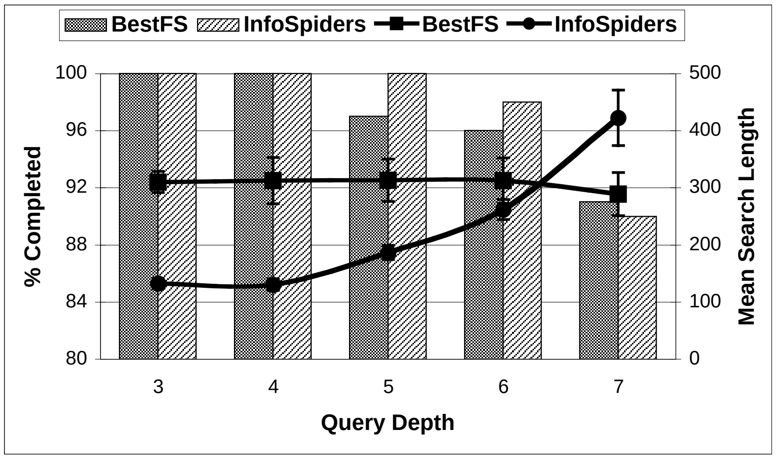

Figure 1.7
Performance of *InfoSpiders* versus best-first-search. Completed queries (bars) were those for which the search algorithms could locate 10% of the relevant documents before running out of time. For these queries we recorded *search length* (lines), the number of non relevant pages visited, averaged over queries whose relevant sets were at the same distance (*depth*) from the starting page. (Data from [38].)

The same algorithmic observation made for the ecological modeling domain applies to applications in the information search domain. Here, cover optimization is necessary because we want the agents to locate as many relevant documents as possible, rather than one "most relevant" document. Local selection algorithms make this objective evolutionarily achievable.

1.4 Heterogeneous Neural Agents for Pareto Optimization

In these experiments we consider the problem of feature subset selection in inductive or similarity-based machine learning. Given two disjoint sets A_1 and A_2 of feature vectors in some n-dimensional space, the problem is to construct a separating surface that allows future examples to be correctly classified as being members of either A_1 or A_2. Here we use neural networks as the classification method, and limit ourselves to two-class problems.

In order to construct classifiers that generalize well to unseen points, it is important to control the complexity of the model. In many domains, biasing the learning system toward simpler models results in better accuracy, as well as more interpretable models. One way to control complexity is through the selection of an appropriate subset of the predictive features for model building. There is a trade-off between training accuracy and model complexity; it is difficult to determine, for a given problem, the relative importance of these two competing objectives.

The feature selection problem is an example of multi-criteria or Pareto optimization, in which more than one objective is optimized at the same time [55, 58]. In the general case, when faced with a multi-criteria problem, the decision maker does not know how the various objectives should be combined. Therefore, the goal of the solver is to find the set of solutions that represents the best compromises between conflicting criteria. This set is called the *Pareto front*. The ELSA framework is particularly well-suited for Pareto optimization because it maintains populations of heterogeneous solutions. By mapping different criteria to different environmental resources, we can create the conditions for agents exploiting different regions of the Pareto space to coexist [44]. Competition occurs along the shared resources, i.e., orthogonally to the Pareto front.

Feature Selection

From a data mining perspective, the problem of determining which predictive features contribute the most to the accuracy of the model is often an important goal in its own right. This is particularly true in medical domains, where a particular feature may be the result of a diagnostic test that can be time-consuming, costly, or dangerous. We examine one such domain in our experiments.

The combinatorial feature selection problem has been studied extensively in the machine learning and statistics literature. Heuristic search methods such as forward selection and backward elimination have long been used in contexts such as regression [11]. Exact solutions may be found using integer programming [53] if one assumes that the error metric is monotonic, which is not the case when estimating out-of-sample error. John [26] makes the distinction between *wrapper* models, which choose feature subsets in the context of the classification algorithm, and *filter* models, such as Relief [27], which choose a subset before applying the classification method. Bradley *et al.* [4] build feature minimization directly into the classification objective, and solve using parametric optimization.

The method described here uses the evolutionary algorithm to search through the space of possible subsets, estimating generalization accuracy for each examined subset. A similar approach, along with a thorough overview of the field, can be found in [62]. Feature selection in neural networks can be seen as a more specific instance of the architecture selection problem; see for instance [54]. Other recent surveys of the feature selection problem may be found in [29] and [7].

Algorithmic Details

In order to isolate the effect of feature selection, the architecture of each neural network is set in such a way to keep the complexity of the networks approximately constant. We use the heuristic that the number of weights in the network should be no more than some small fraction of the number of training cases [52]. For example, for a training set with 500 instances, a network with 5 input features would require 3 hidden units, while a network with 10 input features would have only 2 hidden units.

Training of the individual neural networks is performed using standard backpropagation [56]. Note that this is in contrast to the applications of section 1.3, in which evolution was used to adjust the weights and biases in the networks. Here, the evolutionary algorithm is used solely to search through the space of possible feature subsets.

```
$\Delta E \leftarrow e(Fitness(a'), E_{envt})$
$E_{envt} \leftarrow E_{envt} - \Delta E$
$E_a \leftarrow E_a + \Delta E$
```

```
for each energy source $k$
    $\Delta E \leftarrow \min(Fitness(a', k), E^k_{envt})$
    $E^k_{envt} \leftarrow E^k_{envt} - \Delta E$
    $E_a \leftarrow E_a + \Delta E$
end
```

Figure 1.8
Original ELSA pseudo-code (left, cf. figure 1.1) and the version adapted for multi-criteria optimization (right).

The ELSA algorithm in figure 1.1 was slightly modified to account for the multiple environmental resources corresponding to the fitness criteria [44]. The change is illustrated in figure 1.8. The environment corresponds to the set of possible values (or intervals) for each of the criteria being optimized. We imagine an energy source for each criterion, divided into bins corresponding to its values. So, for criterion F_k and value v, the environment keeps track of the energy $E^k_{envt}(v)$ corresponding to the value $F_k = v$. Further, the environment keeps a count of the number of agents $P_k(v)$ having $F_k = v$. The energy corresponding to an action (alternative solution) a for criterion F_k is given by

$$Fitness(a, k) = \frac{F_k(a)}{P_k(F_k(a))}. \tag{1.2}$$

```
$E_{replenish} \leftarrow E_{bin} \cdot B$
for each energy source $k$
    for each bin $v$
        $\delta E \leftarrow \min(E_{replenish}, E_{bin} - E^k_{envt}(v))$
        $E_{replenish} \leftarrow E_{replenish} - \delta E$
        $E^k_{envt}(v) \leftarrow E^k_{envt}(v) + \delta E$
    end
end
```

Figure 1.9
ELSA energy replenishment pseudo-code for multi-criteria optimization.

Figure 1.9 shows how the environment is replenished at each time step. The quantity E_{bin} is typically a constant (cf. $E_{replenish}$ in equation 1.1). The idea is to fill each bin to E_{bin}. B is the total number of values taken by all criteria, or bins into which the values of continuous criteria are discretized. Thus the total amount of replenishment energy could depend on the size of the problem. However, in order to maintain the population average around some fixed value irrespective of the problem size, we can set

$$E_{bin} = \frac{E_{replenish}}{B} = \frac{p_0 \cdot E_{cost}}{B}. \tag{1.3}$$

In this application the genotype of a neural agent is a bit string s with length equal to the dimensionality of the feature space, N. Each bit is set to 1 if the feature is to be used, and 0 otherwise. We thus measure the complexity of the classifier as simply the fraction of features being used. Each time an agent is evaluated, 2/3 of the examples in the data set are randomly chosen as a training set for a newly constructed neural net (using the features selected by the genotype as inputs). Online backpropagation is applied until the training error converges. Then the remaining 1/3 of the examples is used as a test set

Table 1.3
Parameter values used with ELSA in the feature selection problem. Note that mutation consists of flipping one random bit.

Parameter	Value
E_{cost}	0.3
θ	0.2
p_0	100
B	$(N+1)+20$
$\Pr(mutation)$	1

and the generalization error is used to compute the prediction accuracy of the trained neural agent. Our criteria to be maximized are therefore

$$F_{complexity}(s) = \frac{\text{number of zeros in } s}{N} \tag{1.4}$$

$$F_{accuracy}(s) = \text{generalization accuracy using feature vector } s. \tag{1.5}$$

The application of ELSA to this problem is a straightforward implementation of the algorithm in figure 1.1, with the change of figure 1.8 and equation 1.2, and the two criteria of equations 1.4 and 1.5. Bins are created in correspondence to each of the possible values of the criteria. While $F_{complexity}$ has $N+1$ discrete values (between 0 and 1), $F_{accuracy}$ takes continues values and thus must be discretized into bins. We use 20 bins for the accuracy criterion. The values of the various algorithm parameters are shown in table 1.3. Some of the parameters are preliminarily tuned. Replenishment takes place as shown in figure 1.9, with E_{bin} determined according to Equation 1.3.[1]

For these experiments we compare the performance of ELSA with a well-known multi-criteria evolutionary algorithm, the Niched Pareto Genetic Algorithm (NPGA) [25, 24]. In NPGA, Pareto domination tournament selection is used in conjunction with fitness sharing. Pareto domination tournaments are binary tournaments in which the domination of each candidate is assessed with respect to a randomly chosen sample of the population. If a candidate dominates the whole sample and the other candidate does not, then the dominant candidate wins the tournament. If both or neither candidates dominate the whole sample, then the tournament is won by the candidate with the lower niche count. The latter is, roughly, a weighted count of individuals within dis-

1 In this experiment only about half of the available energy is used by the ELSA population, so that the actual population size oscillates around $p_0/2$.

Table 1.4
Parameter values used with NPGA in the feature selection problem. As for ELSA, mutation consists of flipping one random bit.

Parameter	Value
σ_{share}	14.0
t_{dom}	16
p	100
$\Pr(mutation)$	1

tance σ_{share} in criteria space. The size of the sample, t_{dom}, is used to regulate the selective pressure. NPGA has proven very successful in Pareto optimization over a range of problems. The various parameters used for NPGA are shown in table 1.4. Several of the parameter values are the result of preliminary parameter tuning. For example, the performance of NPGA was deteriorated by the use of crossover in this setting, and therefore the recombination operator was eliminated.

Data Sets

Experiments were performed on two data sets, both of which are available at the UC-Irvine Machine Learning Repository [3].

Wisconsin Prognostic Breast Cancer (WPBC) data: The predictive problem in the WPBC [35] data is to determine whether a particular breast cancer patient can be expected to have a recurrence of the disease within 5 years after surgery. The examples in the each contain $N = 33$ predictive features. They include morphometry information on the individual cell nuclei taken from a cytology slide, along with traditional prognostic factors such as tumor size and metastatic status of local lymph nodes. There are 62 positive examples and 166 negative examples.

Ionosphere data: The task in the Ionosphere data [57] is to discriminate “good” (structured) vs. “bad” (unstructured) antenna returns from free electrons in the ionosphere. The examples consist of $N = 34$ continuous attributes. There are 351 examples, divided between 225 good and 126 bad returns.

Experimental Results

In the feature subset selection problem, the evaluation of the criteria appears as a black box to the evolutionary algorithms. The accuracy computation is expensive and completely dominates the time complexities of the algorithms. Therefore the only operations that contribute to the measured time complexity of the algorithms are the accuracy criterion computations, and time is measured in number of $F_{accuracy}$ evaluations.

We ran the two algorithms for 100,000 function evaluations. Each run, on a dedicated 400 MHz P2 Linux workstation, took approximately 5 minutes for the WPBC data set and 47 minutes for the Ionosphere data set.

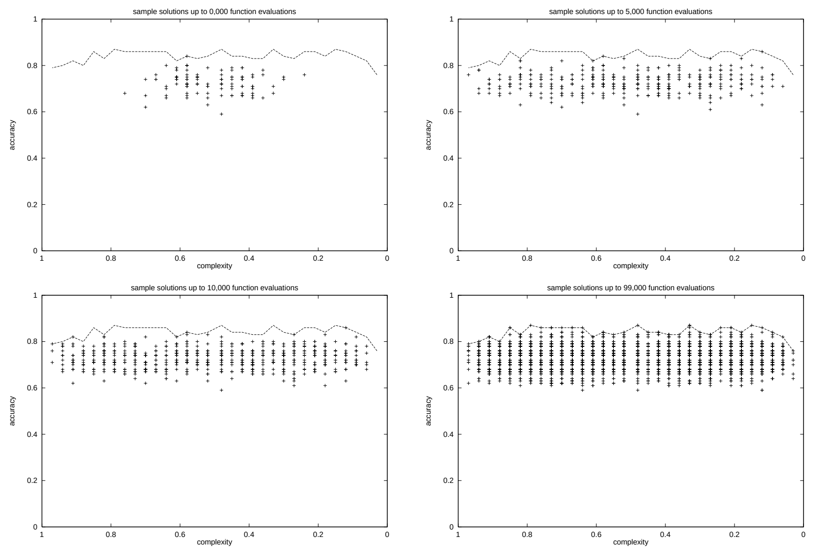

Figure 1.10
Cumulative ELSA populations in Pareto phase-space for the WPBC data set. The x-axis labels indicate the fraction of features used. The snapshots plot samples of the neural agents up to 0, 5, 10, and 99 thousand function evaluations. The convex hull of the sampled populations up to 99,000 evaluations is also shown; the actual Pareto front is unknown.

Figure 1.10 pictures the population dynamics of the ELSA algorithm for the WPBC data set in Pareto phase-space, i.e., the space where the criteria values are used as coordinates. The Pareto front is unknown, so we can only observe the populations and qualitatively assess their progress relative to one another. Since we want to maximize accuracy and minimize complexity, we know that a solution represented as a point in Pareto phase-space is dominated by solutions above it (more accurate) or to its right-hand side (less complex).

The estimated Pareto front in figure 1.10 is nearly flat, reflecting the fact that the complexity of the neural networks was held nearly constant. Still, we see that using only one or two features is insufficient to learn the concept, and using nearly all of them leads to overfitting. The apparent best agent used five input features, reflecting a mix of nuclear size (area, radius) and nuclear shape (symmetry, concavity) features. Interestingly, it did not use lymph node status, a traditional prognostic factor. Removal of these lymph nodes is an extra surgical procedure, performed purely for prognosis, that leaves the patient vulnerable to infection and lymphedema, a painful swelling of the arm. This result supports previous evidence that detailed nuclear morphometry is equal or superior to lymph status as a prognostic measure [59, 61].

ELSA is able to cover the entire range of feature vector complexities. A quantitative measure of coverage can be obtained by measuring the "area" of Pareto space covered by the population of algorithm X at time t:

$$S_X(t) = \sum_{c=0}^{1} \max_{a \in P_X(t)} (F_{accuracy}(a) | F_{complexity}(a) = c)$$

where $P_X(t)$ is the population of algorithm X at time t. Figure 1.11 plots the areas S_{ELSA} and S_{NPGA} versus time, for each data set. NPGA eventually converges to a subpopulation with inferior coverage of the Pareto front, and ELSA achieves significantly better performance.

1.5 Conclusion

This chapter discussed a particular type of evolutionary algorithms, those using local selection, and their use in the synthesis of neural architectures. We have shown that local selection can be feasible, efficient, and effective in evolving populations of heterogeneous neural agents. This is necessary when *cover* or *Pareto* optimization are more desirable than convergence to a single (global or local) optimum.

As demonstrated in section 1.3, there are certainly many cases when a strong selective pressure is necessary, and local selection is inappropriate. We have found this to be the case when we applied ELSA to NP-complete

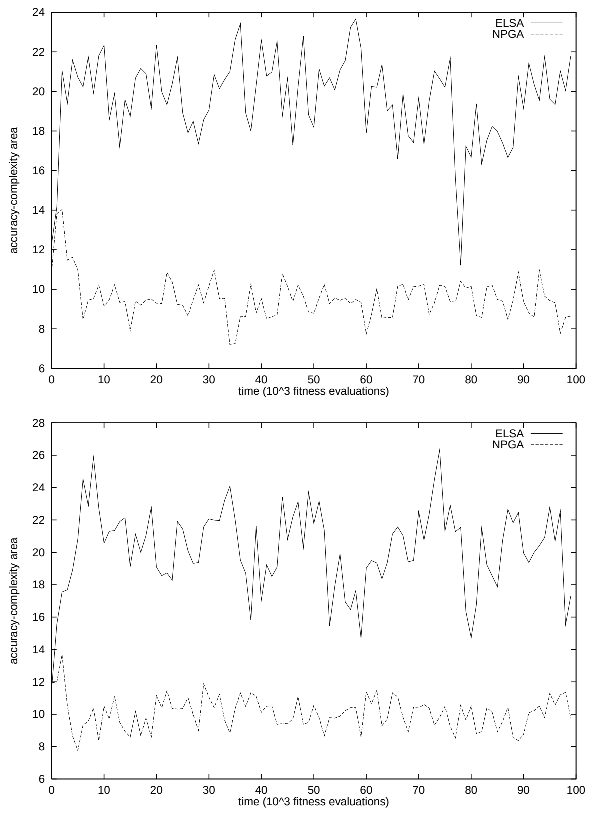

Figure 1.11
Plots of the areas covered in Pareto phase-space by the ELSA and NPGA populations over time, for the WPBC (top) and Ionosphere (bottom) data sets.

combinatorial optimization problems such as SAT and TSP [42]. The only selective pressure that ELSA can apply comes from the sharing of resources. Therefore the way in which environmental resources are coupled with the problem space in a particular application of ELSA is crucial to its success. One limitation of ELSA is in the fact that the appropriate mapping of a problem onto an environmental model may be hard to determine.

However, there are many applications where we need to maintain a diverse population of alternative solutions. Section 1.3 illustrated two such examples from the domains of ecological modeling and autonomous information agents.

Furthermore, heterogeneous neural architectures are required for multi-criteria problems in which the goal is to maintain a population that approximates the Pareto front. Section 1.4 discussed two examples of real-world classification tasks, where the evolutionary local selection algorithm was used to select important features, allowing neural networks to be trained as both parsimonious and accurate predictors.

LS algorithms can be used whenever the fitness function is evaluated by an external environment, in the sense that the environment provides appropriate data structures for maintaining the shared resources associated with fitness. Consider, for example, evaluating a robot in a physical environment: the environment itself holds information about its state. The robot prompts for some of this information through its sensors, and storing such information may be less efficient than simply prompting for the same information again as needed. It may be impossible to store *all* relevant observations about a distributed, dynamic environment. The environment therefore takes the role of a data structure, to be queried inexpensively for current environmental state.

At a minimum, in order for LS to be feasible, an environment must allow for "marking" so that resources may be shared and in finite quantities. In the graph search domain, we have seen that visited nodes are marked so that the same node does not yield payoff multiple times. If InfoSpiders were implemented with mobile agents, this would create a problem of distributed caching. If marking is allowed and performed in constant time, LS also has constant time complexity per individual. This is a big win over the selection schemes of alternative niched evolutionary algorithms, such as fitness sharing and Pareto domination tournaments, whose complexity scales linearly with the population size [24].

Evolutionary local selection algorithms can be extended and applied in many ways. One extension that we have not discussed is the use of recombination operators in ELSA. As stated in section 1.2, crossover is not used in evolving any of the neural agents described in this chapter. Actually, we ran the experiments described in section 1.4 using crossover as well. For ELSA, the mate was a randomly chosen agent. Since we did not expect any correlation across features in these classification tasks, uniform crossover was applied: each bit for whose value the parents disagreed was set to 0 or 1 with equal probability. As it turned out, the performance with crossover was inferior. These results and, more in general, the interactions between local selection and recombination, in particular with local rather than panmictic mating, deserve further analysis in the future.

The application to inductive learning discussed in this chapter can be extended to perform wrapper-model feature subset selection. Local selection can be applied as in the experiments described in section 1.4 to identify promising feature subsets of various sizes. The best of these can then be subjected to a more thorough and costly analysis such as cross-validation to obtain a more reliable estimate of generalization accuracy. This approach would be particularly attractive in an "any-time learning" context, in which little overhead would be required to maintain a record of the best individual encountered so far. Note that the measure of complexity can easily be adapted to other predictive models such as decision trees or linear classifiers [44]. We are also using ELSA for Pareto optimization in clustering problems, where the number of clusters can be used as a third fitness criterion.

Local selection can also serve as a framework for experiments with ensemble classifiers. By extending the environmental model to associate resources with features, in addition to criteria values, we can encourage individual classifiers to specialize in particular regions of the feature space. The predictions of these "orthogonal" classifiers can then be combined (say, by voting) to produce a single classification system that is more accurate than any of the individuals working alone.

Finally, distributed robotics is another application area for which evolutionary local selection algorithms may prove feasible. For example, populations of robots may be faced with unknown, heterogeneous environments in which it is important to pay attention to many sensory cues and maintain a wide range of behaviors to be deployed depending on local conditions.

Acknowledgements

The authors are most grateful to Rik Belew, Laurent Dagorn, and Alvaro Monge, who collaborated on the research reported in section 1.3. The work described in this chapter was supported in part by a travel grant from the University of Iowa, Apple Computers, Encyclopaedia Britannica, University of Iowa CIFRE grant 50254180, NSF grant IIS 99-96044, and the University of Iowa College of Business Summer Grant Program.

References

[1]RF Albrecht, CR Reeves, and NC Steele, editors. *Proc. International Conference on Artificial Neural Networks and Genetic Algorithms*. Springer-Verlag, 1993.

[2]RK Belew and M Mitchell, editors. *Adaptive Individuals in Evolving Populations: Models and Algorithms*. Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley, Reading, MA, 1996.

[3]C. L. Blake and C. J. Merz. UCI repository of machine learning databases [http://www.ics.uci.edu/˜mlearn/MLRepository.html], 1998. University of California, Irvine, Department of Information and Computer Sciences.

[4]P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998.

[5]DJ Chalmers. The evolution of learning: an experiment in genetic connectionism. In *Proc. 1990 Connectionist Models Summer School*, 1990.

[6]L Dagorn, F Menczer, P Bach, and RJ Olson. Co-evolution of movement behaviors by tropical pelagic predatory fishes in response to prey environment: A simulation model. Submitted to Ecological Modeling.

[7]M. Dash and H. Liu. Feature selectin for classification. *Intelligent Data Analysis*, 1(3):131–156, 1997.

[8]Y Davidor. A naturally occurring niche and species phenomenon: The model and first results. In RK Belew and LB Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, 1991.

[9]KA De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[10]KA De Jong and J Sarma. On decentralizing selection algorithms. In *Proc. 6th ICGA*, 1995.

[11]N. R. Draper. *Applied Regression Analysis*. John Wiley and Sons, New York, 3rd edition, 1998.

[12]LJ Eshelman and JD Schaffer. Crossover's niche. In *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993.

[13]DB Fogel. *Evolutionary Computation: The fossil record*, chapter 17, pages 481–484. IEEE Press, 1989.

[14]DE Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*, pages 185–197. Addison-Wesley, Reading, MA, 1989.

[15]DE Goldberg. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Sustems*, 4:445–460, 1990.

[16]DE Goldberg and K Deb. A comparative analysis of selection schemes used in genetic algorithms. In G Rawlings, editor, *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.

[17]DE Goldberg and J Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the 2nd International Conference on Genetic Algorithms*, 1987.

[18]VS Gordon and D Whitley. Serial and parallel genetic algorithms as function optimizers. In *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993.

[19]PB Grosso. *Computer Simulation of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. PhD thesis, University of Michigan, 1985.

[20]F Gruau. Genetic synthesis of boolean neural networks with a cell rewriting developmental process. In *COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, 1992.

[21]G Harik. Finding multimodal solutions using restricted tournament selection. In *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995.

[22]D Hartl and A Clarke. *Principles of Population Genetics*. Sinauer Associates, 1989.

[23]J Horn. Finite markov chain analysis of genetic algorithms with niching. In *Proc. 5th ICGA*, 1993.

[24]J Horn. Multicriteria decision making and evolutionary computation. In *Handbook of Evolutionary Computation*. Institute of Physics Publishing, 1997.

[25]J Horn, N Nafpliotis, and DE Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proc. 1st IEEE Conf. on Evolutionary Computation*, 1994.

[26]G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning*, San Mateo, CA, 1994. Morgan Kaufmann.

[27]K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 129–134, San Mateo, CA, 1992. Morgan Kaufmann.

[28]H Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[29]P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 1–5. AAAI Press, 1994.

[30]SR Lawrence and CL Giles. Searching the world wide web. *Science*, 280:98–100, 1998.

[31]SW Mahfoud. Crowding and preselection revisited. In *Parallel Problem Solving from Nature 2*, 1992.

[32]SW Mahfoud. Simple analytical models of genetic algorithms for multimodal function optimization. In *Proc. 5th ICGA*, 1993.

[33]SW Mahfoud. Population sizing for sharing methods. In *Foundations of Genetic Algorithms 3*, 1994.

[34]SW Mahfoud. A comparison of parallel and sequential niching methods. In *Proc. 6th ICGA*, 1995.

[35]O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-August 1995.

[36]J McInerney. *Biologically Influenced Algorithms and Parallelism in Non-Linear Optimization*. PhD thesis, University of California, San Diego, 1992.

[37]F Menczer. ARACHNID: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. In *Proc. 14th Intl. Conf. on Machine Learning*, 1997.

[38]F Menczer. *Life-like agents: Internalizing local cues for reinforcement learning and evolution*. PhD thesis, University of California, San Diego, 1998.

[39]F Menczer and RK Belew. Evolving sensors in environments of controlled complexity. In R Brooks and P Maes, editors, *Artificial Life IV*, Cambridge, MA, 1994. MIT Press.

[40]F Menczer and RK Belew. From complex environments to complex behaviors. *Adaptive Behavior*, 4:317–363, 1996.

[41]F Menczer and RK Belew. Latent energy environments. In *Adaptive Individuals in Evolving Populations: Models and Algorithms*. Addison Wesley, 1996.

[42]F Menczer and RK Belew. Local selection. In *Proc. 7th Annual Conference on Evolutionary Programming*, San Diego, CA, 1998.

[43]F Menczer and RK Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the web. *Machine Learning*, 1999. Forthcoming.

[44]F Menczer, M Degeratu, and WN Street. Efficient and scalable pareto optimization by evolutionary local selection algorithms. Submitted to Evolutionary Computation Journal.

[45]F Menczer and AE Monge. Scalable web search by adaptive online agents: An InfoSpiders case study. In M Klusch, editor, *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*. Springer, 1999.

[46]F Menczer and D Parisi. Evidence of hyperplanes in the genetic learning of neural networks. *Biological Cybernetics*, 66:283–289, 1992.

[47]F Menczer and D Parisi. Recombination and unsupervised learning: Effects of crossover in the genetic optimization of neural networks. *Network*, 3:423–442, 1992.

[48]O Miglino, S Nolfi, and D Parisi. Discontinuity in evolution: How different levels of

organization imply preadaptation. In *Adaptive Individuals in Evolving Populations: Models and Algorithms*. Addison Wesley, 1996.

[49]GF Miller, PM Todd, and SU Hedge. Designing neural networks using genetic algorithms. In *Proc. 3rd International Conf. on Genetic Algorithms*, 1989.

[50]M Mitchell. *An introduction to genetic algorithms*. MIT Press, 1996.

[51]DJ Montana and LD Davis. Training feedforward networks using genetic algorithms. In *Proc. Internationsl Joint Conf. on Artificial Intelligence*, 1989.

[52]JE Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In *Advances in Neural Information Processing Systems 4*, 1992.

[53]P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9):917–922, September 1977.

[54]D. W. Opitz and J. W. Shavlik. Connectionist theory refinement: Genetically searching the space of network topologies. *Journal of Artificial Intelligence Research*, 6(1):177–209, 1997.

[55]V. Pareto. *Manual of political economy*. Kelley, New York, 1971.

[56]D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8. MIT Press, Cambridge, MA, 1986.

[57]V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266, 1989.

[58]R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley and Sons, 1986.

[59]W. N. Street. A neural network model for prognostic prediction. In J. Shavlik, editor, *Machine Learning: Proceedings of the Fifteenth International Conference*, pages 540–546, San Francisco, CA, 1998. Morgan Kaufmann.

[60]LD Whitley and JD Schaffer, editors. *COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*. IEEE Computer Society Press, 1992.

[61]W. H. Wolberg, W. N. Street, and O. L. Mangasarian. A comparison of computer-based nuclear analysis versus lymph node status for staging breast cancer. *Clinical Cancer Research*, 1999. Forthcoming.

[62]J Yang and V Honavar. Feature subset selection using a genetic algorithm. In H Motoda and H Liu, editors, *Feature Extraction, Construction, and Subset Selection: A Data Mining Perspective*. Kluwer, New York, NY, 1998.