Efficient and Scalable Pareto Optimization by Evolutionary Local Selection Algorithms

Filippo Menczer
Management Sciences Department
University of Iowa
Iowa City, IA 52242, USA
filippo-menczer@uiowa.edu

Melania Degeratu
Mathematics Department
University of Iowa
Iowa City, IA 52242, USA
mdegerat@math.uiowa.edu

W. Nick Street
Management Sciences Department
University of Iowa
Iowa City, IA 52242, USA
nick-street@uiowa.edu

February 15, 1999

Abstract

Local Selection (LS) is a simple selection scheme in evolutionary algorithms. Individual fitnesses are accumulated over time and compared to a fixed threshold, rather than to each other, to decide who gets to reproduce. LS, coupled with fitness functions stemming from the consumption of finite shared environmental resources, maintains diversity in a way similar to fitness sharing. However it is more efficient than fitness sharing, and lends itself to parallel implementations for distributed tasks. While LS is not prone to premature convergence, it applies minimal selection pressure upon the population. LS is therefore particularly suitable in problem classes involving cover and Pareto optimization. This paper presents experiments in which ELSA, an evolutionary algorithm employing local selection, is applied to two multi-criteria optimization problems. Because of its efficiency, ELSA significantly outperforms other well-known evolutionary computation approaches to Pareto optimization (VEGA and NPGA). These experiments confirm earlier results in which LS algorithms were applied to a general class of graph search problems. The paper also discusses the issue of scalability and potential distributed applications of the algorithm.

1 Introduction

When there are multiple solutions to a problem, each of which meets some requirements of the problem better than others, but no solution meets all of the requirements, we have a *multi-criteria* or *Pareto* optimization problem. If we know some good way to combine the performance of a solution at the various criteria, then we can optimize a single combined fitness function, and there are many techniques (including evolutionary algorithms) that apply. But if we do not know how the different criteria should be combined, then we need to find the set of solutions that represent the best compromises between the conflicting criteria. This set is called the *Pareto front*. By quickly locating the Pareto front of a multi-criteria optimization problem, we can devote further efforts to more expensive evaluations of the various alternatives and their trade-offs.

Evolutionary algorithms (EAs) are based on the idea that a population searches the space in a parallel fashion, and therefore they seem amenable to Pareto optimization. The population of solutions at a given time may be used to represent the current consensus on the set of solutions deserving further attention. Ideally, the population would converge not to a single point, but to the entire Pareto front.

Many characteristics of evolutionary algorithms have been considered in the search for ways to devise efficient and effective Pareto optimization algorithms. Selection schemes have emerged as the aspect of evolutionary computation that most directly addresses the issue of multi-criteria optimization. In fact, selective pressure determines how fast the population converges to a solution. Even in standard (single-criterion) optimization, 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.

This paper discusses the *locality* of a selection scheme and its effects on an evolutionary algorithm's behavior with respect to convergence, performance, efficiency, and parallelism. We can loosely define *local selection* (LS) as a selection scheme that minimizes interactions among individuals. The issue of locality in selection schemes is certainly not new in the evolutionary computation community [9, 11, 5, 23].

Parallel EAs often impose geographic constraints on evolutionary search to assist in the formation of diverse subpopulations [12, 3]. 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 [26], 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 EAs are more amenable to cover optimization than to standard convergence criteria, due to the limited communication inherent in most parallel implementations. For example, applying a parallel EA to optimize a bimodal fitness function, the traditional selection schemes — truncation, linear rank, and proportional selection — cause the population to rapidly converge to one mode of the fitness function or the other, while localized selection strategies generate two separate populations that

have converged, each to a separate peak [26].

The problem of ill-convergence exhibited by global selection schemes for multi-criteria fitness functions is related to the issue of *niching*. According to Goldberg [7], we may view a niche intuitively as an organism's job or role in an environment, and we can think of a species as a class of organisms with common characteristics:

> As reproduction, crossover, and mutation proceed, the population climbs the [fitness] hills, ultimately distributing most of the strings near the top of one hill [...] This ultimate convergence on one peak or another without differential advantage is caused by genetic drift — stochastic errors in sampling caused by small population sizes. Somehow we would like to reduce the effect of these errors and enable stable subpopulations to form around each peak [...] We also might like to modify the performance of simple evolutionary algorithms in multi-criteria problems where the peaks are not all of the same magnitude [...] Perhaps we would even like to allocate subpopulations to peaks in *proportion* to their magnitude [...]

Although there is a well-developed biological literature in both niching and speciation [14], its transfer to the artificial evolutionary search has been limited [7]. Standard EAs are ineffective for niching, or multi-criteria function optimization, due to high selective pressure and premature convergence [6]. 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 [34]. 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 [8].

The most notable selection variations explicitly aimed at niching are *crowding* [4, 20] and *fitness sharing* [10, 15, 21]. 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 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 super-linearly with H (with a large constant) [22]. The role of selection for multi-criteria and parallel optimization remains an active area of research in the evolutionary computation community [13, 23, 16].

This paper describes ELSA, an EA based on a biologically-inspired local selection scheme, which lends itself naturally to multi-criteria and cover optimization applications. The algorithm is illustrated in the next section, and its main distinctions from other evolutionary algorithms are outlined. In the following section we report on experiments in which ELSA is compared with other EAs in three broad classes of computational tasks. Finally, we discuss the results of these experiments and try to address the advantages, limitations, and possible applications of local selection.

```
initialize population of agents, each with energy θ/2
while there are alive agents
   for each agent a
      a' ← mutate(clone(a))
      for each energy source k
         ΔE ← min(Fitness(a', k), E^k_envt)
         E^k_envt ← E^k_envt − ΔE
         E_a ← E_a + ΔE
      endfor
      E_a ← E_a − E_cost
      if (E_a > θ)
         insert a' into population
         E_a' ← E_a/2
         E_a ← E_a − E_a'
      else if (E_a < 0)
         remove a from population
      endif
   endfor
   replenish E_envt
endwhile
```

Figure 1: ELSA pseudo-code.

2 Local Selection Algorithms

2.1 ELSA

The original motivation for considering local selection stemmed from an interest in ecological modeling [30, 29]. 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 finite shared resources along with other agents. We can best characterize local selection and succinctly describe its differences from global schemes by casting the EA into the same ecological framework. The resulting algorithm, which we call ELSA (Evolutionary Local Selection Algorithm), is illustrated at a high level of abstraction in Figure 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 similar to itself (in its mutation neighborhood). The agent is taxed with E_{cost} for this action and collects ΔE from the environment. The net energy intake of an agent corresponds to its fitness. This is a function of how well the agent performs with respect to the (possibly many) criteria being optimized. But the energy also depends on the state of the environment. For example, in the experiments illustrated in this paper, the environment corresponds to the set of possible values for each of the criteria being optimized.[1] 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

$$Fitness(a,k) = \frac{F_k(a)}{P_k(F_k(a))}. \tag{1}$$

This is equivalent to a sort of crowding. However, 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.

[1] If a criterion takes continuous values, these can be discretized.

```
E_replenish ← E_bin · B
for each  bin v (v_max..v_min)
   for each  energy source k
      δE ← min(E_replenish, E_bin − E^k_envt(v))
      E_replenish ← E_replenish − δE
      E^k_envt(v) ← E^k_envt(v) + δE
   endfor
endfor
```

Figure 2: Pseudo-code for energy replenishment in ELSA .

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. In a non-distributed or single-criterion task, the environment may reduce to a single global process with the trivial role of ensuring balanced allocation of computational resources to the individuals.

In the experiments illustrated in this paper, E_{cost} for any action is a constant unless otherwise stated. Figure 2 shows how the environment is replenished at each time step. The quantity E_{bin} is also typically a constant:

$$E_{bin} \propto E_{cost}. \tag{2}$$

The idea is to fill each bin to E_{bin}, starting with the bins with highest criteria

values and until we run out of replenishment energy. B is the number of values taken by any criterion, or bins into which the values of a continuous criterion are discretized. Thus the total amount of replenishment energy typically depends on the size of the problem.

On the other hand, if we want to maintain the population average around some fixed value p_0 irrespective of problem size, we can let

$$E_{bin} = p_0 \cdot E_{cost}/B. \tag{3}$$

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):

$$\begin{aligned} \langle p E_{cost} \rangle &= B E_{bin} \\ \langle p \rangle E_{cost} &= p_0 E_{cost} \\ \langle p \rangle &= p_0 \end{aligned}$$

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

In an implementation based on the pseudo-code of Figure 1, some other details would need to be filled in. For example, if crossover is to be used, a candidate solution 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. The same applies for any other genetic operators or problem-specific local search steps warranted by the task at hand. Recombination is not used with ELSA in any of the experiments in this paper. The only genetic operator used is mutation, and the details are

illustrated for each problem.

2.2 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, selected, 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. 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* optimization — all good solutions being represented in the population — than to standard convergence criteria. The bias is to exploit all resources in the environment, rather than to locate the single best resource. This is particularly appropriate in multi-criteria optimization applications.

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. This means that the larger the problem search space, the larger B and the more energy is injected into the environment during replenishment, yielding larger carrying capacity and average population size. The population size need not be adjusted depending on the problem size.

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 func-

Feature	Global Selection	Local Selection
reproduction threshold	$\theta = f(E_1, \ldots, E_{pop})$	$\theta = const$
conserved quantity	selective pressure	entropy
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

Table 1: Schematic comparison between local and global selection schemes.

tion 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 EA, an individual is selected for reproduction based on how its 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 any direct comparisons with other individuals. Table 1 illustrates schematically the main features that differentiate the two classes of selection schemes.

3 Experiments

In this section we will compare the performance of ELSA with other evolutionary algorithms on problems requiring cover or multi-criteria optimization.

3.1 Previous work: Graph search

One application of evolutionary algorithms that achieve cover optimization is the problem of retrieving relevant information in a hypertext environment. Distributed information retrieval is an lively area of research due to the popularity of the Web and the scalability limitations of search engine technology [33, 32]. In previous work, 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. Here we summarize the results of those experiments [31, 28].

3.1.1 Problem description

The problem can be broadly described as searching large graphs in sublinear time. Imagine a very large graph, where each node is associated with some

payoff. The population of agents visits the graph as agents traverse its edges. The problem is well studied in the case of populations of random-walkers. The idea is to maximize the collective payoff of visited nodes, given that there is only time to visit a fraction of the nodes in the graph. Since the modeled search graph is typically distributed across remote servers, agents are charged costs for using its resources, e.g., traversing edges and evaluating nodes' payoff. Nodes model hypertext documents, edges model hyperlinks, and payoff models some measure of relevance.

The graph search task is general enough that it can also be applied to model several other interesting problems. Another example would use the graph as a model of a 2-dimensional environment in which agents have to sense their position and move to reach some goal. This would be a typical task for a situated robot. A third example would be to reduce the graph to model any of the environments used in reinforcement learning problems.

In our instances of the graph search task, each link l is associated with a feature vector with components $f_1^l, \ldots, f_{N_f}^l \in [0,1]$. These features are environmental cues, such as word frequencies, that can guide a browsing agent toward relevant nodes. Actual payoffs are assigned to nodes from an *ad-hoc* probability distribution in the unit interval. This construction process guarantees the existence of a set of weights that, if used as coefficients in a linear combination of a link's feature vector, yield an "accurate" prediction of the payoff of the node pointed by the link. The prediction accuracy of such optimal weight vector is a user-defined graph construction parameter.

3.1.2 Algorithmic details

In the ELSA implementation for this problem, each agent's genotype comprises a single-layer neural net or perceptron, i.e., a weight vector with components $w_1, \ldots, w_{N_f+1} \in R$. The agent's representation also specifies where (on which node) the agent is currently situated. Therefore an action consists, first of all, of evaluating each outlink from the current node. To this end, the output of the neural net is computed for each of these links:

$$o(l) = \frac{1}{1 + e^{-\left(w_{N_f+1} + \sum_{i=1}^{N_f} w_i f_i^l\right)}}$$

and represents the agent's prediction of the payoff $p(l) \in [0, 1]$ of the node that l points to.

The agent follows a link that is picked by a stochastic selector among the links from the current node, with probability distribution:

$$\Pr[l] = \frac{e^{\beta o(l)}}{\sum_{l' \in node} e^{\beta o(l')}}$$

where the β parameter is another component of the genotype. Both β and some fraction of the weights are mutated by additive uniform noise (with the constraint $\beta \geq 0$).

The energetic benefit of an action is the payoff of the newly visited node, provided it had not been previously visited by any agent. Nodes are therefore "marked" to keep track of used resources. A constant energy cost is charged for any new node visited. A smaller cost is also charged for previously visited nodes, to prevent endless paths through visited nodes.

The goal is to evolve agents with optimal genotypes, enabling them to follow the best links and thus achieve maximum payoff intake. This task can be seen as single-criterion in the sense that payoff is the only source of energy, but it can also be seen as multi-criteria in the sense that payoff can be scattered across separate clusters (niches) of nodes. We aim at cover optimization because we want the agents to locate as many relevant nodes as possible. To this end, the environment is not replenished for this task ($E_{bin} = 0$); a node yields energy only once.

3.1.3 Results

To gauge the performance of ELSA in the graph search problem, we decided to compare local selection with a traditional EA employing global selection. Binary tournament selection was chosen as a representative of global selection schemes mainly because it does not require global operations such as averaging, and thus it fits naturally within the steady-state framework of ELSA. Basically the same algorithm outlined above is used for tournament selection, with the difference that the energy level of a randomly chosen member of the population is used in place of both θ for reproduction and 0 for death (cf. Figure 1).

We ran a set of experiments on random graphs generated according to three distinct parameterizations. The parameter G stands for the fraction of nodes whose payoff is above some threshold; we call these "good" nodes. Small G values flag "needle in a haystack" problems. The parameter H is the number of clusters into which the good nodes are grouped. For $H > 1$, we consider the task to be multi-criteria. By "grouped" we mean that a node has

G	0.025	0.05	0.1	0.2
R	0.2	0.4	0.6	0.8
H	1	2	4	8

Table 2: Parameterizations of the graph search problem.

a higher probability to be linked to another node in the same cluster than to nodes in other clusters or to "bad" nodes. The conditional probability that a node points to another node in the same cluster is the third parameter, R.[2] Table 2 shows the parameter values used for the experiments.

In these experiments all the graphs had the same number of nodes (1000), average fanout (5), and number of link features (16). The feature vectors were constructed in such a way that the optimal neural net predicted payoff within an accuracy of 0.99. The algorithm was stopped when 50% of the nodes had been visited, and *recall* (the fraction of good nodes found by the population) up to that point was recorded. Across all graph 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.

In these experiments the performance improvement of ELSA over tournament selection did not vary with G, the density of good nodes. Two trends were instead observed relative to the other graph parameters. Decreasing

[2] G is the background probability and a lower bound for R; if $R = G$, clusters are meaningless. We showed elsewhere that for applications such as information retrieval on the Web, it is realistic to assume that $R > G$ [27].

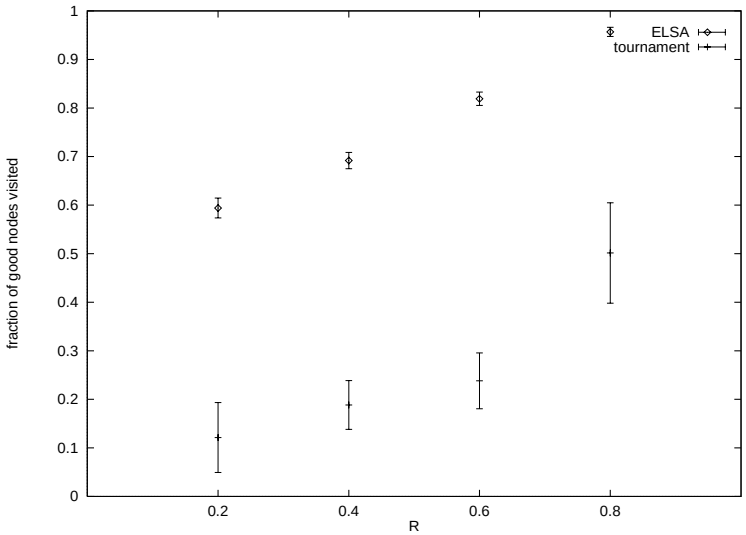

Figure 3: Performance of ELSA and tournament selection on graph search problems with $H = 1$, $G = 0.1$, and various values of R.In this and the following plots, error bars indicate standard errors across multiple runs of the same algorithm with the same parameters but different initial random conditions.

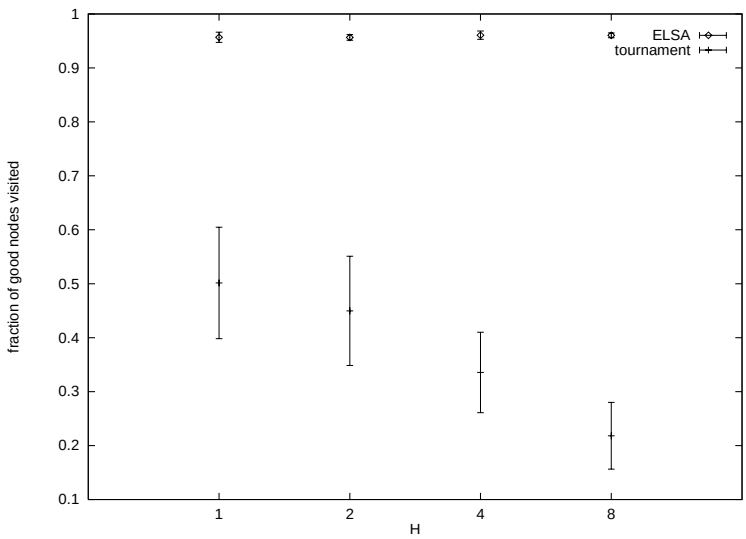

Figure 4: Performance of ELSA and tournament selection on graph search problems with $R = 0.8$, $G = 0.1$, and various values of H.

R, the correlation among good nodes, was equivalent to decreasing the importance of locality; where an agent was situated had smaller consequence in determining how well it would do in the future. We therefore expected ELSA's performance to degrade accordingly. Figure 3 shows the case of unimodal graphs ($H = 1$) and intermediate G. The performance of tournament selection also decreased with R, yielding a consistent advantage in favor of ELSA.

The second trend was observed varying H. Increasing H makes the problem multi-criteria and therefore we expected tournament selection to degrade in performance due to premature convergence. Figure 4 illustrates this trend in the case of high R and intermediate G. The advantage in favor of ELSA increased with H as predicted.

3.2 A simple test problem: Ones and pairs

The success of ELSA in the graph search problem led us to test the approach on other, better known problems, for which we could obtain a more fair comparison with other evolutionary algorithms developed specifically for Pareto optimization. We wanted to pick both a problem and a set of alternative approaches well described in the literature. We first focused on a simple test problem called "unitation versus pairs" [17] and on two state-of-the-art evolutionary algorithms for multi-criteria optimization, VEGA and NPGA [16]. Here we report on new experiments comparing ELSA with VEGA and NPGA on the unitation versus pairs task.

3.2.1 Problem description

The problem is very simple. Consider the combinatorial search space of size N binary strings $\{s = (s_0, \ldots, s_{N-1}) | s_i \in \{0, 1\}\}$. Two criteria are defined:

$$\begin{aligned} Ones(s) &= \sum_{i=0}^{N-1} s_i \\ Pairs(s) &= \sum_{i=0}^{N-1} (s_i \oplus s_{(i+1) \bmod N}). \end{aligned}$$

Clearly the two criteria cannot be both completely satisfied, as maximizing the number of ones in a string will minimize the number of $\{01, 10\}$ pairs, and maximizing the number of pairs will halve the number of ones. Therefore this is an example of a Pareto optimization problem.

We can now give a more formal definition of the Pareto front. The criteria define a partial order among solutions. One solution s_1 is said to *dominate*

another solution s_2 if $\forall k : F_k(s_1) \geq F_k(s_2)$ and $\exists k : F_k(s_1) > F_k(s_2)$, where F_k is the k-th criterion. Neither solution dominates the other if $\exists k, l : F_k(s_1) > F_k(s_2), F_l(s_2) > F_l(s_1)$. The Pareto front is defined as the set of undominated solutions, i.e., no solution dominates any of the solutions in the Pareto front.

The main advantage of simple problems such as unitation versus pairs is that the Pareto front is known. Therefore algorithms can be tested conveniently, and one can even assess what fraction, if any, of the Pareto front is covered by a population.

3.2.2 Algorithmic details

The application of ELSA to this problem is a straightforward implementation of the algorithm in Figure 1 and Equation 1, with the functions $Ones(\cdot)$ and $Pairs(\cdot)$ as the criteria. Bins are created in correspondence of each of the possible values of the criteria. Replenishment takes place as shown in Figure 2, with E_{bin} determined according to Equation 3 in order to maintain an average population size equal to that used in the other EAs described below. The values of the various algorithm parameters are shown in Table 3. Note that a mutation in ELSA corresponds to flipping a single bit.

The first multi-criteria EA alternative that we consider is the Vector Evaluated Genetic Algorithm (VEGA) [37, 38]. In VEGA, the population is divided into subpopulations, one associated with each of the criteria. In each subpopulation, the corresponding criterion is used as the fitness function. VEGA has been used with some success in Pareto optimization and therefore has become one of the best known multi-criteria EAs, although it has been reported that VEGA tends to favor extreme points of the Pareto

Parameter	Value
E_{cost}	4.0
θ	2.0
p_0	300
B	$2(N+1)$
$\Pr(mutation)$	1

Table 3: Parameter values used with ELSA in the unitation versus pairs problem.

front over intermediate, trade-off points [7]. VEGA can be used in conjunction with any selection scheme. Here we use binary tournament selection in conjunction with fitness sharing, since this combination has been shown to improve VEGA's coverage of the Pareto front [17].

The second multi-criteria EA alternative that we consider is the Niched Pareto Genetic Algorithm (NPGA) [17, 16]. 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 dominated 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 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 here for VEGA and NPGA are taken in part from Horn *at al.* [17] and verified by reproducing their results in the

Parameter	Value
σ_{share}	4.0
t_{dom}	16
p	300
$\Pr(crossover)$	0.9
$\Pr(mutation)$	0.01

Table 4: Parameter values used with NPGA and VEGA in the unitation versus pairs problem.

unitation versus pairs problem. They are shown in Table 4.

3.2.3 Results

The concept of a generation is meaningless in ELSA because the population fluctuates. Even if we define a generation as a unit of time in which each candidate solution is evaluated once (the loop over the agents in Figure 1), the time complexities of the basic steps of the three algorithms are very different from one another. To insure a fair comparison, we count basic "operations" uniformly across the three algorithms. A basic operation is any comparison that has to do with the execution of the algorithm. These include both fitness evaluations and algorithmic steps, so that the operation count is incremented inside every loop that is required by the algorithm. The operation count is therefore a measure of time complexity.

The first experiment simply compares the performance of the three algorithms by measuring the cover of the Pareto front achieved over time. The size of the problem in this experiment is $N = 32$ bits. Figure 5 shows the results. ELSA is the clear winner. NPGA also achieves almost complete

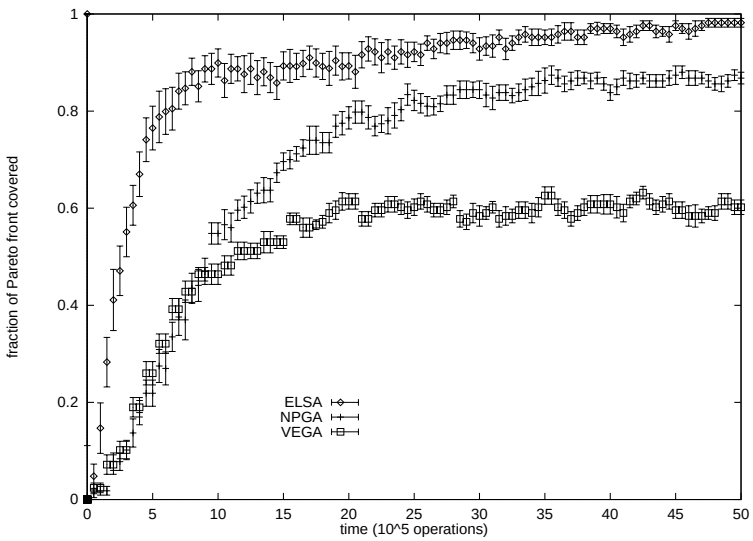

Figure 5: Coverage of the Pareto front for the unitation and pairs problem, plotted for the three evolutionary algorithms versus time complexity.

coverage, but it takes longer because the algorithm has a higher time complexity. Every time a candidate solution is evaluated, many operations must be carried out to compute Pareto domination and fitness sharing. VEGA appears to converge on a subset of the Pareto front.

The second experiment aims at evaluating the scalability of the two more competitive algorithms with respect to problem size. To this end we drop VEGA and test ELSA and two versions of NPGA on problems of different sizes between $N = 16$ and 128 bits. The reason for the two version of NPGA is in the fact that like the majority of EAs, NPGA has a fixed population size. How does one determine the right population size for a given problem? Such problem-dependence is a problem with EAs in general. For large problems, the Pareto front is large and therefore a large population is necessary. Conversely, a smaller population should be sufficient for small problems. The

problem-dependence of population size is of consequence for the scalability of the algorithm. If we don't increase population size with problem size, Pareto cover performance will suffer. So in one NPGA version we keep the population size constant (100), and in another we make it proportional to the problem size ($4N$).[3]

By using the replenishment scheme of Figure 2 in conjunction with Equation 2 ($E_{bin} = 4E_{cost}$ in these runs), ELSA does not have a problem-dependence issue. In fact, the average population size is determined by the carrying capacity of the environment, which is automatically proportional to problem size. The user does not need to make a decision based on knowledge of the problem. Apart from these differences in population size, both ELSA and NPGA use the parameters values shown in Figures 3 and 4, respectively.

For each problem size, Figure 6 plots the coverage of the Pareto front achieved by the different algorithms after a fixed time (one million operations). Unsurprisingly, the NPGA version with proportional population does better than the version with fixed population on large problems (where the fixed population is too small) and worse on small problems (where the fixed population is larger). Once again ELSA is the winner. Not only does its population adjust automatically to problem size, but it also outperforms both versions of NPGA where the differences are significant. As N grows, the differences becomes less significant and ultimately all of the algorithms fail to find any point in the Pareto front in the given time.

[3]We use the knowledge that the size of the Pareto front is also proportional to N for this problem.

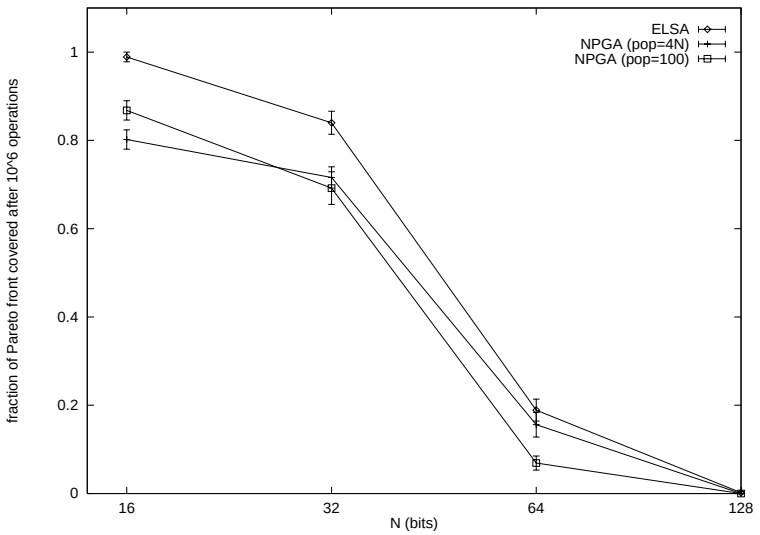

Figure 6: Coverage of the Pareto front for the unitation and pairs problem achieved after a fixed number of operations, plotted for the two evolutionary algorithms versus problem size.

3.3 A real problem: Feature selection in inductive learning

Unitation versus pairs has allowed us to quantitatively compare the performance of ELSA and other multi-criteria EAs on a well defined problem, with known Pareto front. We now turn to a more realistic problem, where the Pareto front is unknown.

3.3.1 Problem description

In these experiments we consider the problem of feature subset selection in inductive or similarity-based machine learning. Given two disjoint sets $\mathcal{A}_1$ and $\mathcal{A}_2$ of feature vectors in some k-dimensional space, the problem is to construct a separating surface that allows future examples to be correctly

classified as being members of either $\mathcal{A}_1$ or $\mathcal{A}_2$. Popular techniques for this task include decision trees, artificial neural networks, and nearest neighbor methods. In this work the learn separation model is a simple k-dimensional hyperplane, comparable to a perceptron.

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 (number of features); it is difficult to determine, for a given problem, the relative importance of these two competing objectives. The combinatorial feature selection problem has been studied extensively in the machine learning literature, with approximate solutions being found using both filter and wrapper models [18, 19], and exact solutions found using integer programming [36]. Further, iterative selection of feature subsets is fundamental to traditional predictive techniques such as regression and decision tree construction.

3.3.2 Pattern separation via linear programming

The problem of optimally (in a least-norm error sense) separating two point sets can be solved using linear programming. We describe here the Robust Linear Program (RLP) [1, 24] used in this application. The available training points from $\mathcal{A}_1$ and $\mathcal{A}_2$ can be represented as matrices A_1 and A_2, with each row of A_1 (respectively A_2) containing the feature values for one training

example from set $\mathcal{A}_1$ ($\mathcal{A}_2$). The goal is to construct a separating plane $x^T w = \gamma$ in the feature space of these examples such that all the points of A_1 lie on one side of the plane (say, $A_1 w > e\gamma$, where e is a vector of ones of appropriate dimension) and all the points of A_2 lie on the other ($A_2 w < e\gamma$). This will only be possible if A_1 and A_2 are linearly separable, which in general is not the case. We therefore choose to minimize the average distance between the plane and the misclassified points. This is achieved with the following normalized minimization problem:

$$\underset{w,\gamma}{\text{minimize}} \qquad \frac{1}{n_1} \|(-A_1 w + e(\gamma+1))_+\|_1 + \frac{1}{n_2} \|(A_2 w - e(\gamma-1))_+\|_1$$

where $\|z\|_1$ denotes the 1-norm of z and $(z)_+$ denotes $((z)_+)_i = max\{z_i, 0\},\ i = 1, \ldots, m$, for $z \in R^m$.

This objective can be shown to be equivalent to the following linear program:

$$\begin{array}{ll} \underset{w,\gamma,y,z}{\text{minimize}} & \dfrac{ey}{n_1} + \dfrac{ez}{n_2} \\ & A_1 w - e\gamma + y \geq e \\ \text{subject to} & -A_2 w + e\gamma + z \geq e \\ & y, z \geq 0. \end{array} \tag{4}$$

This linear program has a number of favorable properties, including:

- If A_1 and A_2 are linearly separable, a separating plane is found.
- The null solution $w = 0, \gamma = 0$ is never unique, and is only obtained when the centroids of the two point sets are equal.

- Use of the 1-norm error, rather than the more traditional 2-norm error, makes Program 4 more resistant to the effects of outliers.

Mathematical programming models have also been used to solve the feature selection problem by incorporating feature minimization directly into the objective. In [2], the RLP formulation was augmented with a term that penalizes the number of non-zero coefficient values. The resulting concave optimization problem was solved efficiently using a series of linear programs, resulting in significant generalization improvements on high-dimensional problems. One of the problem domains explored in that paper is also used in our experiments: the prediction of breast cancer recurrence based on cytological features [25, 39]. A version of this data set is available at the UCI Machine Learning Repository [35].

3.3.3 Algorithmic details

In this application the EA individuals are bit strings with length equal to the dimensionality of the feature space. Each bit is set to 1 if the feature is to be used in training, and 0 otherwise. We thus measure the complexity of the classifier as simply the number of features being used. Accuracy is measured using the training correctness of the resulting classifier. While this is not a reliable estimate of generalization ability, it does allow an exploration of the accuracy vs. complexity trade-off. Our criteria to be maximized are therefore

$$\begin{aligned} F_{complexity}(s) &= 1 - \frac{Ones(s)}{N} \\ F_{accuracy}(s) &= \text{prediction training accuracy using feature vector } s. \end{aligned}$$

Parameter	Value
E_{cost}	0.6
θ	0.3
p_0	300
B	$(N+1)+100$
$\Pr(mutation)$	1

Table 5: Parameter values used with ELSA in the feature selection problem.

The application of ELSA to this problem is a straightforward implementation of the algorithm in Figure 1 and Equation 1, with the functions $F_{accuracy}(\cdot)$ and $F_{complexity}(\cdot)$ as the criteria. Bins are created in correspondence of 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 100 bins for the accuracy criterion. The values of the various algorithm parameters are shown in Table 5. Some of the parameters are preliminarly tuned. Replenishment takes place as shown in Figure 2, with E_{bin} determined according to Equation 3 in order to maintain an average population size equal to that used in NPGA.[4]

For this problem we again compare ELSA with NPGA, since VEGA has proven a less competitive algorithm. The various parameters used for NPGA are shown in Table 6. Note that crossover is not used in NPGA because for this problem we don't expect any correlation across features. The higher mutation rate, as well as the other parameter values used with NPGA, are the result of preliminary parameter tuning.

[4]In this experiment only about one third of the available energy is used by the ELSA population, so that the actual population size oscillates around $\langle p \rangle = 100$.

Parameter	Value
σ_{share}	14.0
t_{dom}	16
p	100
$\Pr(crossover)$	0.0
$\Pr(mutation)$	0.5

Table 6: Parameter values used with NPGA in the feature selection problem.

The data set used in these experiments has $N = 33$ features, so that the candidate solutions in ELSA and NPGA have 33 bits.

3.3.4 Results

In the feature subset selection problem, the evaluations of the criteria appear as black boxes to the evolutionary algorithms. In fact, the accuracy computation is very expensive and completely dominates the time complexities of the algorithms.[5] Given this observation, it was decided for fairness that in the feature subset selection experiments, the only operations that should contribute to the measured time complexities of the algorithms are the accuracy criterion computations. Therefore time is measured in number of $F_{accuracy}$ evaluations and all other algorithmic costs are assumed to be negligible. This kind of assumption is realistic for actual multi-criteria applications in general.

We ran the two EAs for 100,000 function evaluations. Each run took approximately 3 hours on a dedicated 400 MHz P2 Linux workstation. Figure 7 pictures the population dynamics of the algorithms in Pareto phase-space,

[5] The complexity computation only takes time $O(N)$.

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 general conclusion that we can draw from Figure 7 is that ELSA tends to cover a wider range of trade-off solutions, while NPGA focuses on a smaller range but gets closer to the Pareto front in that range thanks to its stronger selection pressure. These behaviors are not entirely surprising given the analysis of Section 2.2. If closeness to the actual Pareto front is more important than coverage, then NPGA remains a very strong competitor.

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

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

where $P_X(t)$ is the population of algorithm X at time t.[6] Figure 8 plots the areas S_{ELSA} and S_{NPGA} versus time. These measures highlight the superior coverage achieved by ELSA. Consequently, by looking at the extremities of the population front, we can use ELSA to easily answer questions like, What

[6]Note that the dummy variable c in the summation does in fact take discrete values corresponding to $F_{complexity}$.

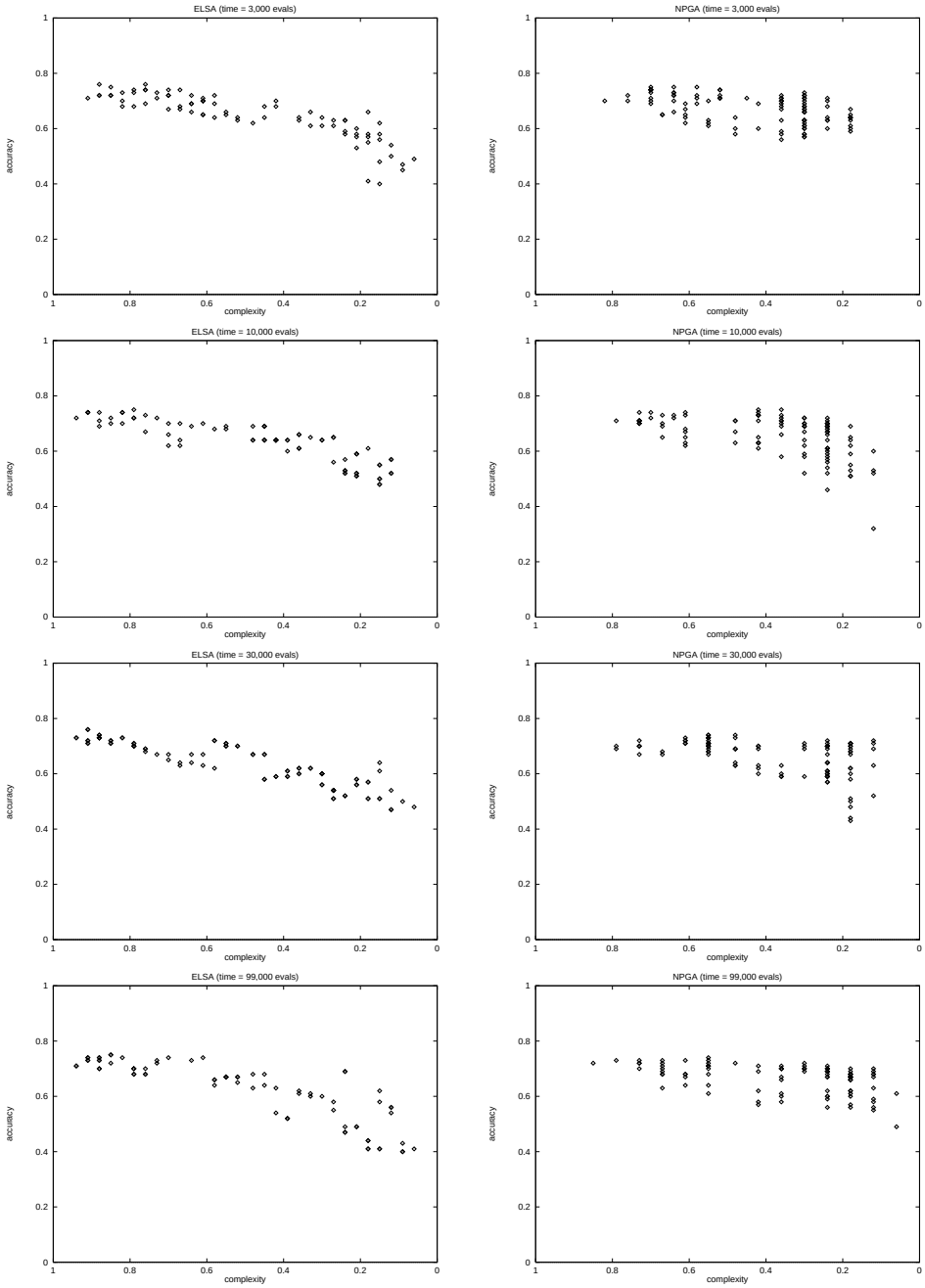

Figure 7: Snapshots of the ELSA and NPGA populations in Pareto phase-space after 3, 10, 30, and 99 thousand function evaluations.

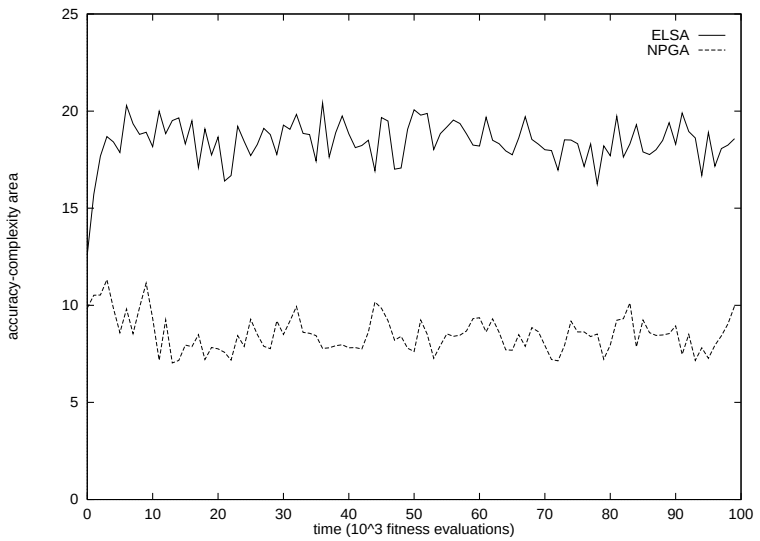

Figure 8: Plot of the area covered in Pareto phase-space by the ELSA and NPGA populations over time.

features are the best and worst single predictors? Or, What is the maximum complexity of the feature subsets that we should consider, beyond which no increase in accuracy is expected? To quickly achieve such coverage of the Pareto space is an important property for a multi-criteria optimization algorithm.

4 Discussion

We now discuss the advantages and limitations of the local selection algorithm that we have observed based on the results described in the previous section.

4.1 Performance

Addressing the performance of selection strategies is equivalent to discussing the tension between exploration and exploitation. A balance between these two opposite forces is clearly necessary even in single-criterion EAs, so that the population can make progress toward good solutions without prematurely converging to suboptimal ones. All multi-criteria optimization EAs enforce diversity by constraining exploitation, to keep the population from converging at all. The different techniques to achieve this goal result in different algorithmic behaviors, which are naturally problem-dependent. This paper is intended to discuss local selection rather than to compare and critique different evolutionary approaches to Pareto optimization (see, e.g., [16] for such a review).

LS is a weak selection scheme, so it ensures better performance in tasks where the maintenance of diversity within the population is more important than a speedy convergence to the optimum. We have shown that ELSA is well suited to multi-criteria optimization and sublinear graph search applications. In fact the population quickly distributes itself across the whole ranges taken by the criteria. And for easy problems such as unitation versus pairs, ELSA outperforms other effective multi-criteria EAs such as NPGA at locating the entire Pareto front.

On the other hand, the exploitation of information is also necessary to cut the search space and make progress. For problems requiring effective selection pressure, local selection may be too weak, as we have shown in the case of the feature subset selection problem. We have also applied ELSA to NP-complete combinatorial optimization problems such as SAT and TSP, with

little success [31]. The only selection 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 may be in the fact that the appropriate mapping of a problem onto an environmental model may be hard to determine.

The observations that NPGA is better at multi-criteria optimization in a more focused area of Pareto space, while ELSA is more efficient in covering the whole Pareto space, suggest that the two approaches could be combined. Local selection could be used at the beginning of a run, and once a wide range of trade-off solutions has been located, then further resources can be allocated to domination selection.

4.2 Efficiency

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 problem, visited nodes are marked so that the same node does not yield payoff multiple times.[7] If marking is allowed and performed in constant time, the time complexity of LS is also $O(1)$ per individual. This is a big win over fitness sharing and Pareto domination tournaments, the best alternative schemes for distributed or multi-criteria optimization. The efficiency advantage of ELSA contributes to its success over NPGA and VEGA in the unitation versus pairs problem. Further, in a distributed implementation there is little communication overhead and thus the parallel speedup can be significant.

4.3 Scalability

In the graph search problem, we have shown that ELSA responds robustly to increases in the the number of clusters. As the search space becomes more niched, subpopulation naturally form to cover the various niches.

The unitation versus pairs experiment with increasing problem size also provides us with evidence of the scalability properties of local selection. ELSA suffers less than NPGA from the severe (doubly exponential) growth of the search space.

Another aspect of ELSA's scalability is the robustness of its population size in the face of problems of increasing complexity. Unlike other evolutionary algorithms, ELSA does not require a problem-dependent decision by the

[7] In a distributed information retrieval application, this issue would entail a discussion on distributed caching.

user regarding on appropriate population size. These scalability properties seem crucial for evolutionary algorithms in general.

4.4 Applications

Since one of the major strengths of ELSA is its minimal centralized control and consequent parallelization potential, distributed and parallel computation tasks seem amenable to the local selection approach. Examples of such problems that we might attack with ELSA include distributed scheduling, distributed resource allocation, and constrained search.

Local selection has already been applied to agent-based distributed information retrieval, e.g. for autonomous on-line Web browsing. The InfoSpiders project [28, 32] shows promise in taking advantage of several properties of evolutionary local selection algorithms: (i) coverage, for quickly locating many relevant documents; (ii) distributedness, to conserve bandwidth by enabling agents to run on the remote servers where documents reside; and (iii) localization, so that each agent may face an easier learning problem by focusing on a limited set of features — words — that are locally correlated with relevance.

The application to inductive learning can be extended to perform wrapper-model feature subset selection. Local selection can be applied as in the experiments described in this paper 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 also that the measure of complexity can easily be adapted to other predictive models such as artificial neural networks or decision trees.

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 "orthoganal" 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.

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.

4.5 Future directions

The applications outlined above are attractive directions for future work aimed at further analysis and testing of evolutionary local selection algorithms. The two multi-criteria problems considered in this paper are insufficient to fully evaluate ELSA. We hope to identify applications that will allow us to better assess the performance and efficiency of local selection, and to better characterize the problem domains in which ELSA may be feasible and successful.

One aspect of scalability that we have not addressed is the algorithmic

behavior with respect to increases in the number of criteria. We have followed the common practice of testing algorithms on "toy" problems and real-world problems with only two criteria. It is imperative to explore the performance of ELSA in real Pareto optimization problems with large numbers of criteria.

The interactions of local selection with recombination operators, in particular with local rather than panmictic mating, have not been explored and deserve attention in the future.

We also intend to study the effect of reinforcement learning within an agent's lifetime and its interaction with local and global selection schemes. Some initial experiments suggest a strong duality between local selection and reinforcement learning, where the two adaptive mechanisms can be viewed as attempts to internalize environmental cues at multiple spatial and temporal scales [28].

Acknowledgments

The authors are grateful to Richard K. Belew and David Fogel for helpful comments. This work was supported in part by University of Iowa CIFRE grant 50254180 and NSF grant IIS 99-96044 (formerly IIS 97-01992).

References

[1] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

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

[3] 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.

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

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

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

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

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

[9] 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.

[10] DE Goldberg and J Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proc. 2nd ICGA*, 1987.

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

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

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

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

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

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

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

[18] 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.

[19] 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.

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

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

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

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

[24] O. L. Mangasarian. Mathematical programming in neural networks. *ORSA Journal on Computing*, 5:349–360, 1993.

[25] 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.

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

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

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

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

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

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

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

[33] 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. Forthcoming.

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

[35] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html], 1996. University of California, Irvine, Department of Information and Computer Sciences.

[36] 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.

[37] JD Schaffer. *Some experiments in machine learning using vector evaluated genetic algorithms.* PhD thesis, Vanderbilt University, 1984.

[38] JD Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proc. 1st ICGA*, 1985.

[39] 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 Publishers.