Local Selection

Filippo Menczer and Richard K. Belew

Computer Science and Engineering Department
University of California, San Diego
La Jolla, CA 92093–0114, USA
{fil,rik}@cs.ucsd.edu

Abstract. Local selection (LS) is a very simple selection scheme in evolutionary algorithms. Individual fitnesses are 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 shared environmental resources, maintains diversity in a way similar to fitness sharing; however it is generally 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 more appropriate than other, stronger selection schemes only on certain problem classes. This papers characterizes one broad class of problems in which LS consistently outperforms tournament selection.

1 Introduction

The study of selection schemes is central to evolutionary computation and closely related to the issue of convergence. In particular, the *locality* of a selection scheme affects both the parallelization potential and the convergence behavior of evolutionary algorithms (EAs), with efficiency and performance implications [3]. We call *local selection* (LS) a selection scheme that minimizes interactions among individuals.

Parallel EAs often impose geographic constraints on evolutionary search to assist in the formation of diverse subpopulations [1]. This is done to minimize the communication overhead of selection; different processors are allocated to subpopulations to minimize interprocess dependencies and thus improve efficiency. Parallel EAs are more amenable to *cover* optimization — all good solutions being represented in the population — than to standard convergence criteria, due to the limited communication in most parallel implementations.

The problem of ill-convergence exhibited by global selection schemes for multimodal fitness functions is a general issue related to aspects of natural adaptation — niching and speciation — that have seldom been included in formal treatments of EAs in spite of a well-developed biological literature [9]. Standard EAs are ineffective for niching, or multimodal function optimization, due to high selective pressure and premature convergence [5]. Several methods have been devised to deal with this problem by maintaining diversity: from variations of proportional selection that tune the selective pressure adaptively [16] to different selection methods, such as tournament selection, that converge slowly and have niching effects [6].

The most notable selection variations explicitly aimed at niching are *crowding* [2] and *fitness sharing* [7]. In both of these methods, selection is 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, the population size required to maintain a cover across niches grows very rapidly with the number of niches [11]. The role of selection for multimodal and parallel optimization remains an active area of research [8, 12].

In Section 2 we describe a simple mechanism for local selection. In Section 3 we show the results of experiments in which LS is compared with a global selection scheme in two broad classes of computational tasks. In Section 4 we discuss these results and try to identify some of the advantages and limitations of local selection.

2 Local Selection

Our motivation for local selection stems from an ecological modeling perspective. Consider a population of organisms, evolving to efficiently forage in a simulated physical environment [15, 14]. In this model, fitness is the result of individual interactions with the environment and its finite shared resources. By casting the EA into the same framework, we can best characterize local selection and succinctly describe its differences from global schemes.

The agents making up the population use and collect energy from local interactions with the environment. The algorithm is illustrated in Figure 1. Note (step 3) that 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. Energetic costs are incurred by any action. Costs and replenishment determine the *carrying capacity* of the environment that in turn determines population size. An agent reproduces or dies based on the comparison between its current energy level and the α, ω thresholds (step 4). Energy intake is the currency by which we measure the success or failure of behaviors with respect to the environment. The only form of communication among agents is their shared use of the finite energy resources.

In a standard EA implementation based on the pseudocode of Figure 1, the input and output steps correspond to the evaluation of a candidate solution. They may also include a local search phase, if warranted by the problem/algorithm. The net energy intake is the individual's fitness. The reproduction step of course comprises cloning, random mutations, and optionally other genetic operators such as crossover and/or problem-specific local search. In a non-distributed task/algorithm, the environment may reduce to a single global

```
initialize population of agents, each with energy E_0
while there are alive agents
  for each agent i
    1. input: sense environment
    2. output: compute action a
    3. update energy:
       E_envt <- E_envt - benefit(a)
       E_i <- E_i + benefit(a) - cost(a)
    4. selection:
       if (E_i > α)
         reproduce
         E_offspring <- E_i/2, E_i <- E_i/2
       else if (E_i <= ω)
         die
       end
  end
  E_envt <- E_envt + E_replenish
end
```

Fig. 1. EA pseudocode. For further details on the algorithm, see [15, 14].

process with the trivial role of ensuring balanced allocation of computational resources to the individuals.

In a multimodal or distributed task, the environment models the problem space and the resources that are locally available to individual solutions. In such cases 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 choice of thresholds $\alpha = \omega = \langle E \rangle$, where $\langle \cdot \rangle$ indicates population average; the reproduction rate would then be proportional to how rapidly an agent accumulates energy, with respect to the rest of the population. Likewise, binary tournament selection can be modeled by $\alpha = \omega = E_r$ where the subscript r indicates a randomly picked individual. We will define LS as a scheme in which α and ω are independent of the rest of the population. In the rest of the paper we will use $\alpha = 2E_0 = const > 0$ and $\omega = 0$. This way, interference among individuals is reduced to the sharing of local resources — the amount of interaction allowed by the distributedness or multimodality of the task at hand. Energy is consumed by action costs, created by environmental replenishment, and conserved at reproduction (parents share energy with offspring) and death (agents die as soon as they run out of energy).

The removal of selection's centralized bottleneck allows for parallel EA implementations. It is also evident that LS is an implicitly niched scheme and therefore it naturally enforces the maintenance of population diversity. These factors in our opinion make LS a central issue in today's EA community [3, 12]. It is then essential to characterize the problem domains in which LS is applicable and/or advantageous, and estimate the costs involved.

3 Experiments

In this section we will compare the performance of two variants of the evolutionary algorithm in Figure 1, one using LS and one using deterministic binary tournament selection. We chose the latter 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 the algorithm.

3.1 Graph Search

The first class of problems 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 goal 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.

The problem is interesting because typically the graph is distributed, so that agents are charged costs for using its resources, e.g., traversing edges and evaluating nodes' payoff. The issue of distributed algorithms is therefore central for this problem class. Furthermore, the graph search task is general enough that it can be reduced to several interesting special cases. If we use nodes to model hypertext documents, edges for hyperlinks, and payoff for some measure of relevance, then the problem is that of distributed information retrieval, a popular topic due to the current popularity of the World Wide Web. As another example, the graph could be used to model a 2-dimensional environment in which agent have to sense their position and move to reach some goal. This would be a typical task for a situated robot.

In our instances of the graph search task, each node is assigned a payoff p from a uniform probability distribution in the unit interval. Furthermore, each link l is annotated with a "feature vector" with N_f real components $f_1^l, \ldots, f_{N_f}^l \in [0,1]$. The idea is that these features, if properly interpreted, can guide agents by allowing them to predict the payoff of a node based on the features of a link pointing to that node.

To make this possible, each agent's genotype comprises a single-layer neural net or perceptron, i.e., a vector of weights $w_1, \ldots, w_{N_f+1} \in \Re$. An agent receives in input, for each outgoing link from the node where it is currently situated, the link's feature vector (step 1 of the algorithm). It then uses its neural net to compute $o(l) = 1/[1 + \exp(-w_{N_f+1} - \sum_{i=1}^{N_f} w_i f_i^l)]$, i.e., its prediction of the payoff $p(l)$ of the node that l points to. Finally (step 2) the agent follows a link that is picked by a stochastic selector among the links from the current node, with probability distribution $\Pr[l] = \exp[\beta o(l)] / \sum_{l' \in node} \exp[\beta o(l')]$ where the β parameter is another component of the genotype. There exists by construction an optimal weight vector such that the corresponding neural net is a perfect predictor of payoff. Agents with such a genotype can follow the best links and thus achieve optimal fitness (maximum payoff intake). The energetic benefit of an action is the payoff of the newly visited node, provided it had not been previously visited by any agent (step 3; nodes are "marked" to keep track of used

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

Table 1. Parameterizations of the graph search problem.

resources). The environment is not replenished ($E_{replenish} = 0$); a node yields energy only once. 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. At reproduction (step 4), an agent's genotype is cloned and mutated to obtain the offspring genotype. Both β and some of the weights are mutated by additive uniform noise (with the constraint $\beta \geq 0$). To study the effect of local selection in isolation from other factors, we did not use crossover in the graph search task.

We have run a set of preliminary experiments comparing LS and tournament selection on the graph search problem. Random graphs were generated according to three distinct parameterizations. First, 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. Second, the parameter H is the number of niches into which the good nodes are clustered. For any $H > 1$, the task is multimodal. By "clustered" we mean that a node has a higher probability to be linked to another node in the same niche than to nodes in other niches or to "bad" nodes. The conditional probability that a node points to another node in the same niche is the third parameter, R.[1] Table 1 shows the parameter values used for the experiments.

In these experiments all the graphs have 1000 nodes, an average fan-out of 5, and $N_f = 16$. The feature vectors are constructed in such a way that the optimal neural net predicts payoff within an accuracy of 0.01. The algorithm is stopped when 50% of the nodes have been visited, and the fraction of good nodes found by the population up to that point is recorded.

Across all graph parameterizations, LS significantly and consistently outperforms tournament selection. The improvement varies depending on the graph parameters, but is generally between two- and ten-fold. Typically, LS populations continue to discover a constant rate of good nodes, while tournament populations converge prematurely.

Varying G, the density of good nodes, does not result in any noticeable trend across all other experimental conditions. Decreasing R, the correlation among good nodes, is equivalent to decreasing the importance of locality; where an agent is situated has smaller consequence in determining how well it will do in the future. We therefore expect LS's performance to degrade accordingly. Figure 3.1 shows the case of unimodal graphs and intermediate G. We observe that the

[1] G is the background probability and a lower bound for R; if $R = G$, niches are meaningless. We have shown elsewhere [13] that for applications such as information discovery on the Web, it is realistic to assume that $R > G$.

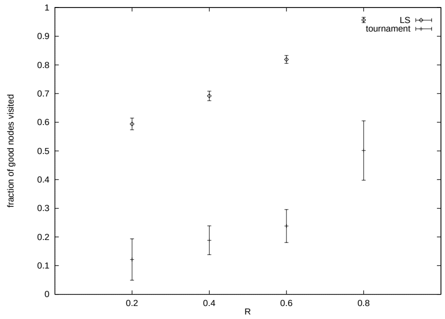

Fig. 2. Performance in graphs with $H = 1$, $G = 0.1$, and various R values.

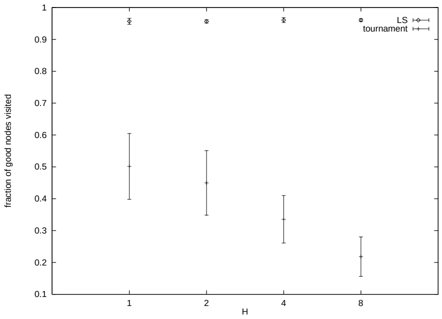

Fig. 3. Performance in graphs with $R = 0.8$, $G = 0.1$, and varying H.

performance of tournament selection also decreases with R, yielding a consistent advantage in favor of LS.[2]

Finally, increasing H makes the problem multimodal and therefore we expect tournament selection to degrade in performance due to premature convergence. Figure 3.1 illustrates this trend in the case of high R and intermediate G. The advantage in favor of LS increases with H as predicted.

3.2 Combinatorial Optimization

The second problem domain in which we explore local selection is combinatorial optimization. We consider two NP-hard optimization problems, TSP and SAT. In each case steps 1 and 2 of the algorithm reduce to evaluating a new candidate solution. In order to apply LS, we need to implement step 3 through some model of environmental resources and energy benefits and costs.

For the TSP, we generate a problem instance by distributing points uniformly in the unit square and using Euclidean distances. An agent's genotype represents a tour, i.e., a permutation of the order in which points are to be visited. While no crossover is used, two ad-hoc mutation operators are applied: (i) swapping two random points, (ii) reversing the subtour between two random edges.[3]

For LS, edges between points represent the shared resources. Every time an agent tests a tour, a usage count associated with each traversed edge is incremented. The agent is then charged an energy cost based on the accumulated usage counts, and receives an energy benefit based on how good (short) the tour is. At replenishment, usage counts are redistributed uniformly across edges and decreased by an amount that determines carrying capacity.[4] For tournament selection, only the tour length is used to compute fitness.

[2] In this and the following plots, error bars indicate standard errors across multiple runs of the same algorithm with the same graph parameters.

[3] This operation, called *2-Opt*, is a well-known local search strategy for the TSP [10].

[4] This model resembles the Ant Colony algorithms [4], in which agents deposit pheromone on used edges; edges are chosen based on both length and amount of pheromone accumulated by the passage of previous agents.

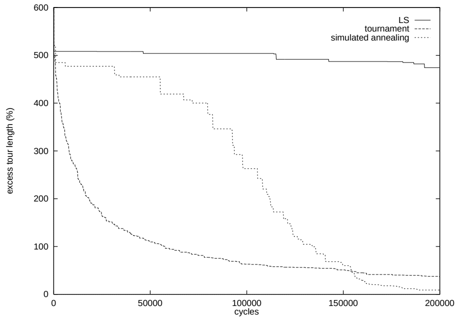

Fig. 4. Single-run performance on a 100-point Euclidean TSP with random initial tours.

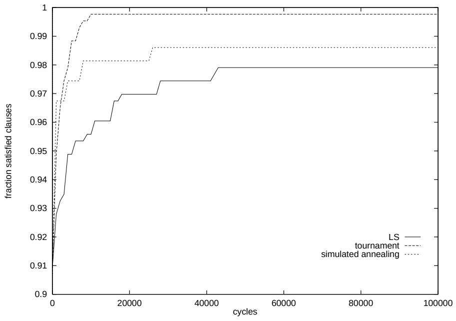

Fig. 5. Single-run performance on a 100-variable, 430-clause MAX-3-SAT problem.

Figure 3.2 plots the excess tour length over the Held-Karp lower bound for Euclidean tours in the unit square.[5] It shows that the comparison between LS and tournament selection performance on the TSP is in clear favor of the global scheme. LS does not seem to apply sufficient selective pressure. In this experiment, agent are initialized with random tours. In another experiment, in which agents are initialized with good tours,[6] LS outperforms tournament selection; the latter consistently converges prematurely to less optimal tours.

For the second combinatorial optimization problem, MAX-3-SAT, we generate random CNF formulas in which each clause contains 3 literals, and seek truth assignments that maximize the number of satisfied clauses. The number of clauses in the formulas is chosen so as to make the problem difficult [18]. Individual genotypes contain truth assignments that are randomly initialized. Point crossover is applied in this problem, with panmictic mating. A local search strategy is applied at each step; a random literal is selected from one of the unsatisfied clauses and negated. This is an optimal strategy for SAT [17], and we also use it as a mutation operator.

For LS, the energy cost charged for each evaluation is a constant. The energy benefit, however, depends on the fraction of clauses that are satisfied by the agent's truth assignment, and for which there are available resources. Environmental resources are associated with each clause, and decremented every time an agent satisfies that clause. At replenishment, all clause resources are incremented by a constant that determines carrying capacity. For tournament selection, fitness is simply the fraction of satisfied clauses.

Figure 3.2 shows that for SAT, as for the TSP, tournament selection performs better than LS. Again, LS does not seem to exercise a sufficient selective pressure.

[5] In this and the following plot the performance of a simple simulated annealing implementation, with geometric annealing schedule, is also shown for sake of comparison.

[6] The nearest-neighbor heuristic is used to produce Euclidean tours whose average length is 29.3% above the Held-Karp lower bound [10].

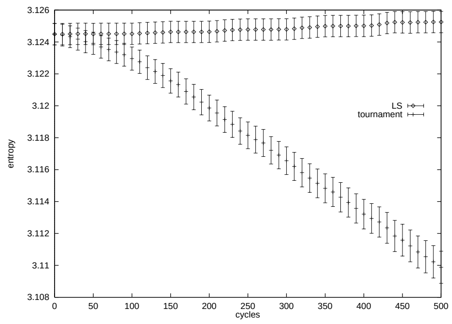

Fig. 6. Entropy for 1000-point Euclidean TSP instances with greedy initial tours.

4 Discussion

In this section we discuss the advantages and limitations of LS observed in the preliminary experiments outlined in the previous section. Although our observations are subdivided between performance and efficiency, the two issues are certainly not independent of each other.

4.1 Performance

Discussing the performance of selection strategies is related to discussing the tension between exploration and exploitation, a classic issue for EA practitioners. The interplay between these two opposite forces determines the EA's capability to make progress toward good solutions without prematurely converging to suboptimal ones. The appropriate balance, of course, is problem dependent.

LS is a very 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. This is the case for multimodal optimization and sublinear graph search, as we have shown. LS does in fact maintain population diversity at a much higher level than tournament selection. This is illustrated in Figure 6, where population entropy is plotted over time for the two selection schemes in TSP problems with good initial tours. Entropy is computed by $S = -\sum_l f_l \log(f_l)$ where f_l is the frequency with which edge l is traversed by agents in the current population. Since all of the tours making up the population are quite good in these runs, tournament selection exploits the information too quickly, excluding good solutions and eventually converging prematurely to suboptimal tours.

On the other hand, exploiting information is also necessary to cut the search space and guarantee progress. For problems requiring effective selection pressure, LS may just be too weak. This may explain its failure at the combinatorial optimization problems that we have considered in our preliminary experiments. The only selection pressure that LS can apply comes from the sharing of resources. Therefore the way in which environmental resources are coupled with the problem space in a particular implementation of LS is crucial to its success. We have explored a few alternatives for TSP and SAT, but LS has been consistently outperformed by tournament 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 resources associated with fitness. 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. 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, the obvious alternative for distributed or multimodal optimization (cf. Section 1). Further, in a distributed implementation there is no communication overhead and thus the parallel speedup can be as large as the number of agent processes.

Note, however, that all problem spaces do not lend themselves to being used as data structures. For example, a task may have a fitness function that is static and expensive to evaluate. And for problems such as continuous function optimization, marking the environment would imply a discretization of the search space; arbitrary precision would make LS at least as expensive as fitness sharing.

5 Future Directions

More experiments are needed to better assess the performance and efficiency of LS, and to better characterize the problem domains in which LS is feasible and successful. In particular, for combinatorial optimization problems where LS has failed to prove advantageous, we need to study alternative environmental resource models in order to determine whether the poor performance is to be attributed to inadequate resource models or to weak selection pressure. For example, the way of associating resources in the TSP described here is opposite to the one behind the Ant Colony algorithm [4], in which agents are encouraged to agree upon, rather than diversify, their use of edges. Such alternatives should be explored in the LS framework.

The graph search problem class, in which LS has proven successful, can be extended to provide for better models of real applications such as robotics and Web information retrieval. It can also be constrained in such a way as to create deceptive problem instances, where the performance comparison between LS and global selection schemes would be of even greater relevance.

The role of crossover has not been explored in this paper but deserves attention in the future. In particular, for parallel or distributed problems or implementations, the issue of local vs. panmictic mating becomes crucial to limit additional sources of communication and interference between distant agents.

A direction currently under study is the effect of reinforcement learning within an agent's lifetime and its interaction with local selection. LS may be viewed as an integration of reinforcement learning over longer time scales. Environments rich with features that correlate with fitness at different time scales should prove useful problem domains to study these interactions.

Acknowledgments We thank the Cognitive Computer Science Research Group at UCSD, David Fogel, and two anonymous reviewers for their helpful comments.

References

1. Y. Davidor. A naturally occurring niche and species phenomenon: The model and first results. In R. Belew and L. Booker, editors, *4th ICGA*, 1991.
2. K. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.
3. K. De Jong and J. Sarma. On decentralizing selection algorithms. In *6th ICGA*, 1995.
4. M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. on Evolutionary Computation*, 1(1):53–66, 1997.
5. L. Eshelman and J. Schaffer. Crossover's niche. In *5th ICGA*, 1993.
6. D. Goldberg. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Sustems*, 4:445–460, 1990.
7. D. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *2nd ICGA*, 1987.
8. G. Harik. Finding multimodal solutions using restricted tournament selection. In *6th ICGA*, 1995.
9. D. Hartl and A. Clarke. *Principles of Population Genetics.* Sinauer Associates, 1989.
10. D. Johnson. Local optimization and the traveling salesman problem. In *17th Colloquium on Automata, Languages and Programming*, 1990.
11. S. Mahfoud. Population sizing for sharing methods. In *FOGA 3*, 1994.
12. S. Mahfoud. A comparison of parallel and sequential niching methods. In *6th ICGA*, 1995.
13. F. Menczer. Arachnid: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. In *14th ICML*, 1997.
14. F. Menczer and R. Belew. From complex environments to complex behaviors. *Adaptive Behavior*, 4:317–363, 1996.
15. F. Menczer and R. Belew. Latent energy environments. In *Adaptive Individuals in Evolving Populations: Models and Algorithms.* Addison Wesley, 1996.
16. F. Menczer and D. Parisi. Recombination and unsupervised learning: effects of crossover in the genetic optimization of neural networks. *Network*, 3:423–442, 1992.
17. C. Papadimitriou. On selecting a satisfying truth assignment. In *32nd FOCS*, 1991.
18. B. Selman, D. Mitchell, and H. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81:17–29, 1996.