A Dilemma for Fitness Sharing with a Scaling Function

Paul Darwen and Xin Yao
Department of Computer Science
University College, The University of New South Wales
Australian Defence Force Academy, Canberra AUSTRALIA
Email: darwen@cs.adfa.oz.au, xin@cs.adfa.oz.au

ABSTRACT

Fitness sharing has been used widely in genetic algorithms for multi-objective function optimization and machine learning. It is often implemented with a scaling function, which adjusts an individual's raw fitness to improve the performance of the genetic algorithm. However, choosing a scaling function is an *ad hoc* affair that lacks sufficient theoretical foundation. Although this is already known, an explanation of why scaling works is lacking. This paper explains why a scaling function is often needed for fitness sharing. We investigate fitness sharing's performance at multi-objective optimization, demonstrate the need for a scaling function of some kind, and discuss what form of scaling function would be best.

We provide both theoretical and empirical evidence that fitness sharing with a scaling function suffers a dilemma which can easily be mistaken for deception. Our theoretical analyses and empirical studies explain why a larger-than-necessary population is needed for fitness sharing with a scaling function to work, and give an explanation for common fixes such as further processing with a hill-climbing algorithm. Our explanation predicts that annealing the scaling power during a run will improve results, and we verify that it does.

1. Introduction

Fitness sharing is a speciation method for genetic algorithms, allowing a GA population to find multiple optima in a multimodal search space. Fitness sharing has been used in machine learning [1] and multimodal function optimization [6]. Fitness sharing often uses a scaling function. However, choosing a scaling function is a procedure lacking theoretical conviction [5, page 124] [9]. Common scaling functions are known to suffer certain drawbacks, the most important of which is the need for post-processing with a hill-climbing algorithm. Why this need exists has been an open question. This paper provides a theoretical and empirical explanation for some well-known traits of fitness sharing that have long been known but not explained.

We investigate fitness sharing's performance at multiobjective optimization, demonstrate the need for a scaling function, and review the optimal form of a scaling function. A simple non-deceptive search space is used in order to concentrate on fitness sharing. We explain why fitness sharing so often needs post-processing with a hill-climber, or alternatively fails to find all optima, unless one uses a very large population or a local search algorithm.

Using the insight gained by explaining this behaviour, we review techniques commonly used with fitness sharing, and explain why they were needed and why they worked. Our explanation predicts an improvement to fitness sharing by increasing the scaling power during a run. We test this prediction, and as expected it gives results better than with any fixed scaling power. This simple improvement, which seems obvious in the light of our explanation, is an example of the practical innovations made possible by this paper's contribution to a deeper understanding of genetic algorithms.

1.1. Fitness Sharing and Multi-objective Optimization

Fitness sharing modifies a search landscape by reducing payoff in densely-populated regions, encouraging search in unexplored regions and causing subpopulations to form. Consider an individual i with fitness f_i. Its niche count m_i [5, page 191] [10] measures how many other individuals with which i shares fitness. The shared fitness is $f_i^s = \frac{f_i}{m_i}$. The niche count m_i is calculated with a distance metric d_{ij} that describes the difference between individuals i and j, usually their Hamming distance, $m_i = \sum_{j=1}^{\text{population size}} \text{sh}(d_{ij})$ where the *sharing function* $\text{sh}(d_{ij})$ makes distant individuals share less:

$$\text{sh}(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_s}\right)^\alpha & \text{for } 0 \leq d_{ij} < \sigma_s \\ 0 & \text{for } d_{ij} \geq \sigma_s \end{cases} \qquad (1)$$

Parameter α changes the sharing function's shape: $\alpha = 1$ gives a linear sharing function. More important is σ_s, the *sharing radius* or cutoff distance: two strings σ_s or further apart do not share fitness.

One way to find multiple optima is to make several runs of an ordinary GA: for each run, genetic drift will cause the population to converge to one optimum, but on the next run it will probably converge to a different optimum. Thus, multiple optima are found from multiple runs. Beasley *et al* [2] take this a step further: each time the ordinary GA converges to an optimum, the payoff is reduced around that optimum to prevent the GA from converging there again, to prevent wasteful repetition. Fitness sharing similarly reduces payoff, but at heavily-populated regions of the search space, and it does it dynamically during a single run, which reduces the repeated partial exploration suffered by sequential schemes like that of Beasley *et al* [2].

Nonetheless, fitness sharing has its own limitations, including the following:

1. σ_s is the same for all individuals, so all optima in the search space must be equidistant or nearly so [12].
2. To set σ_s, you need to know *a priori* how far apart optima are, and their (unshared) fitness. Until you search the space, this information is unknown [10].

These limitations can cause fitness sharing to fail to find all optima if they are not equidistant and equal-valued, or if the estimated distance between optima is incorrect. To get around these problems, some schemes (in effect) do not use a fixed sharing radius [8, 12].

1.2. Organization of this Paper

In the rest of this paper, we will demonstrate that even without the above limitations, in a space where the peaks are equidistant and equal-valued, fitness sharing still finds fewer peaks than expected. We explain why, and shed light on why genetic algorithms so often require augmentation with a local search algorithm, and why a GA with fitness sharing often needs a suspiciously large population to find all optima.

The reason is that a scaling function is usually required to improve the results of fitness sharing. We briefly review why scaling is needed: it is known that for our and other similar problems, the best scaling function ϕ is a power function, $\phi(f) = f^{\beta}$ for some β [9]. High values of β cause premature convergence and missed optima, low values cause subpopulations to form around optima but not at them, similar to the effect of deception. In short, a power function is the best scaling function, but no constant scaling power can get around the above dilemma. This explains many of the known problems with scaled fitness sharing, and lets us understand the known fixes to those problems. More importantly, it helps us devise new and more effective solutions.

In Section 2, we describe a simple multimodal search space, show why a scaling function is needed, and review what is known about optimal scaling functions. We give a theoretical explanation and empirical examples of the dilemma facing scaling functions in fitness sharing. With this deeper understanding, in Section 3 we re-examine several common techniques used in fitness sharing, and review alternative GA extensions that promise better performance. In particular, our explanation predicts that the simple innovation of a varying scaling power should give better search results, and we demonstrate that it does. Section 4 concludes with a brief summary of this paper.

2. Fitness Sharing and Scaling Functions

In this section, we demonstrate that fitness sharing requires the use of a scaling function and that we should be able to use a small population. We explain both theoretically and empirically why a large population or similar fix is required in previous studies.

2.1. An Artificial Search Space

The search space consists of all binary strings of length 30 bits, the same size used by Goldberg *et al* [6]. Of the $2^{30} \approx 10^9$ strings, only 10 are global optima. Unshared payoff is 1.0 for an exact match of an optimum, and declines linearly to zero for a $\frac{2}{3}$ correct match, or 10 incorrect bits out of 30. The global optima are selected to be at a Hamming distance of 10 bits or more from each other. If an individual partially matches more than one optimum, it receives the payoff from the closest match only, i.e., no addition. Incidentally, allowing addition of partial matches causes "generalizers" to evolve, with few species partially matching many optima, an interesting problem similar to the immune system [12].

The correct population size has two needs. First, the random first generation must sample individuals in each payoff region. Since the space contains a mere $2^{30} \approx 10^9$ strings, enumerative search is feasible and finds that each optima is surrounded by 2×10^7 paying strings. For a space of 2^{30} strings, that leaves 98.13% of the space outside a particular peak. The chance of a particular peak being missed completely by a random population of size n is therefore 0.9813^n. The average number of peaks represented in the random initial population is 10 peaks $\times(1-0.9813^n)$. A population of 200 samples the regions around almost all 10 optima:

$$\begin{aligned} &\text{Population size} = 200 \\ &\Rightarrow \text{Avg peaks sampled} = 10\left(1-\left(1-\tfrac{2\times10^7}{2^{30}}\right)^{200}\right) \\ &\approx 9.77 \text{ peaks} \end{aligned} \tag{2}$$

This is actually a conservative estimate, as crossover during the first few near-random generations create even more random individuals.

Secondly, the population must be large enough to maintain a subpopulation around each optimum. Like Deb and Goldberg [4, page 49], we found that assortative crossover (crossover between similar parents) worked better than random crossover, and the experiments below use assortative crossover. Assortative mating produces few inter-species hybrids, so we may neglect crossover when sizing the population. The minimum population size n for maintaining c species of equal fitness for G generations with probability γ is then [11, equation 13]:

$$n = \frac{\ln(\frac{1-\gamma^{\frac{1}{G}}}{c})}{\ln(\frac{c-1}{c})} \tag{3}$$

For $c = 10$ optima, maintained for $G = 100$ generations with probability $\gamma = 0.999$, the required population size is $n = 131$ individuals, so a population of 200 is enough to maintain subpopulations. In our experiments, elitism is used. The mutation rate is 0.006, which was the best result from a modest amount of searching. To discourage "super-individuals" from dominating, linear ranked selection is used, with the best individual expecting 2 offspring and the worst, no offspring.

2.2. Fitness Scaling Revisited

For a typical run, Figure 1 shows the population distribution by how many zeros in a string. One optimum in the search space consists of all 30 bits set to zero, and the closest optima have only 20 zeros. Instead of converging to the optimum with 30 zeroes, we see a subpopulation as a bulge at 26 zeros instead of 30, as if there were deceptive optima. But the space is not deceptive. Why is this subpopulation error-prone?

Shared payoff depends on nearby individuals. Recall that unshared payoff is 1 for a perfect match of an optimum, decreasing linearly to zero payoff for 10 incorrect bits. The sharing radius σ_s is also 10 bits. Thus, an individual at the optimum shares with everyone in the surrounding payoff region, but an individual on the region's edge shares with far fewer peers, raising its shared payoff.

We demonstrate this by calculating the shared payoff around an isolated peak for a typical population distribution. A realistic distribution will be

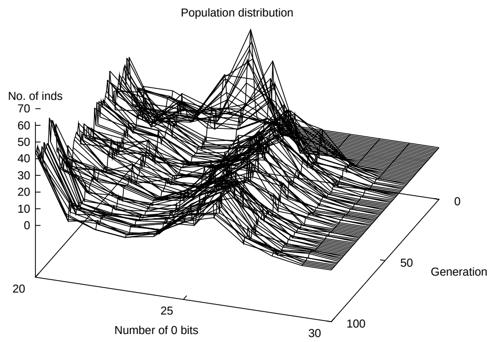

Fig. 1: Without scaling, this subpopulation stays away from the all-zero peak.

zero outside the payoff region. Inside, we assume a uniform distribution, i.e., all individuals have an equal probability of existing. Any other distribution with more individuals at the edge of the payoff region would encourage individuals in the center, and vica-versa, so a uniform distribution is reasonable for this demonstration.

The calculation of m_k is available elsewhere [3]. Figure 2 shows the shared payoff for no scaling (power = 1) for this population distribution. We see the highest shared payoff is not at the optimum, but a short Hamming distance away. Figure 2 shows why, without scaling, a subpopulation stays *near* an optimum rather than at it in Figure 1: an individual at the optimum must share with more individuals.

With this explanation in mind, the reason why fitness scaling works becomes obvious. A scaling function is usually a power function [5, page 124] [6] [9, page 12]. If we raise the unshared payoff to a power β before sharing, we get a shared payoff that makes the optimum more attractive than the surrounding region:

$$f_i^s = \frac{(f_i)^\beta}{m_i} \tag{4}$$

Figure 2 shows shared payoff for different scaling function powers, at the same population distribution described above.

2.3. Optimal Scaling Function

Why a power function? Choosing a scaling function is an *ad hoc* procedure that lacks a solid theoretical foundation [5, page 124]. Kreinovich *et al* [9] give the form of the optimal scaling function, given the criteria of certain properties of the search problem and assuming the scaling function's dimension. Space precludes a fuller discussion, but for problems of the form of our problem, which is typical of many but not all search problems, Kreinovich *et al* [9, page 18] have shown that:

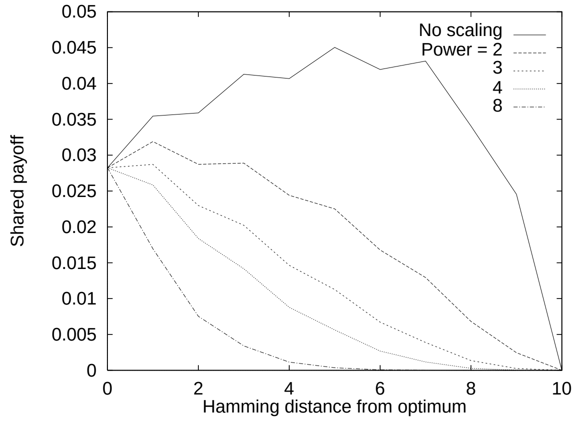

Fig. 2: Shared payoff at uniform population distribution for different scaling powers. Low fitness scaling means the highest shared payoff iis not at the optimum.

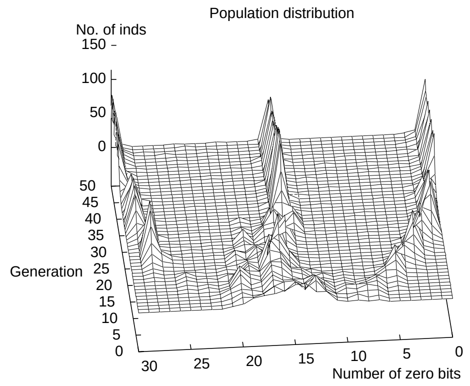

Fig. 3: Scaling power of 10 finds fewer peaks.

1. If we assume the scaling function is scalar, then the optimal function is $\phi(f) = f^{\beta}$ for some β.
2. If we assume the scaling function has a finite basis, $\phi(f) = \sum_{i=1}^{\eta} C_i \phi_i(f)$ for some η, then each subfunction is of the following form, for some β, γ, and θ:

$$\phi_i(f) = (\ln(f))^i f^{\beta} \sin(\gamma(\ln(f)) + \theta) \qquad (5)$$

There is however no way of knowing if a particular problem works better with a scalar, finite, or infinite basis. If one assumes a finite basis, optimizing the parameters $\eta, \beta, \gamma, \theta$ and C_i is itself a time-consuming problem, requiring many runs of the original GA search for different parameter values. For practicality's sake, we will consider a single-dimensional scaling function of the form $\phi(f) = f^{\beta}$.

It is worth noting that Kreinovich *et al* [9] cannot tell us what power β works best for a particular problem. In the next section, we will show that no power β finds all optima even in our simple artificial problem, and thus that fitness sharing with power scaling has room for improvement.

2.4. No Power Function Works Perfectly

Figure 3 shows a typical run with a scaling power of $\beta = 10$. Each subpopulation converges to its optimum, solving the problem of near-optimal subpopulations in Figure 1, but only a few out of the 10 optima are actually found.

Why? The random initial population contains individuals in nearly every payoff region, sampling most regions as predicted by Equation 2. However, scaling means that the lucky few who are closer to their optima have their higher payoff raised to the high scaling power, making them comparatively much fitter. These "super individuals" multiply and drive to extinction other individuals which are further away from an optimum. Those regions are not explored, and are lost to the population. This happens despite linear ranked selection, with the best individual receiving only 2 offspring. With proportional selection, the "super-individuals" have well above twice the population's average payoff, and so get more than 2 offspring, making the results even worse.

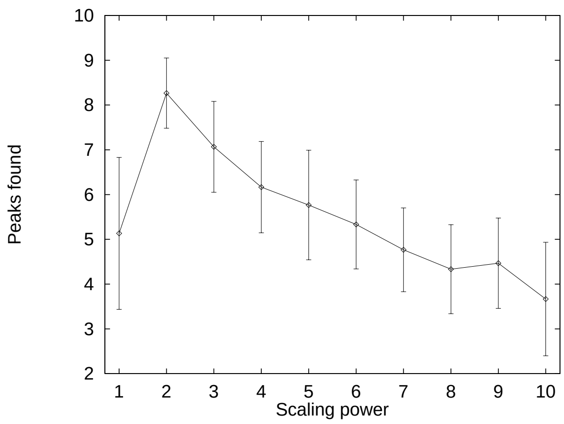

Fig. 4: Peaks found (out of 10) for different scaling powers. We did 30 runs for each scaling power.

To control these super-individuals, we reduce the scaling power. The number of peaks found for different powers is shown in Figure 4, and we find a dilemma:

- A high power means the individuals closest to an optimum receive much higher payoff than individuals not quite as close to other optima, who die out, despite the use of linear ranked selection. Fewer optima are found.
- A low power means that individuals stay a few bits away from the peak, because any individual at the peak must share with more and thus have a lower shared payoff, as shown in Figure 2.

What a dilemma! Equation 2 says that with a population size of 200, the GA should find nearly all peaks. However, we cannot adjust the scaling power to achieve this without one of the above problems, and the best we can do is about 8 peaks found instead of 9.77 peaks. And since Kreinovich *et al* [9] show that the best single-dimensional scaling function for our problem is a power function, and no power meets our expectations, then no suitable single-dimensional scaling function exists. If we make assume a multi-dimensional scaling function would work better (an assumption unsupported by theory), this requires optimizing all the parameters in Equation 5 — a difficult problem in itself. This knowledge explains many long-known features of fitness sharing.

3. Discussion

3.1. Old Solutions in a New Light

Armed with our explanation of why scaled fitness sharing misses optima, we re-examine some techniques commonly used with fitness sharing. Like many GA techniques, these were developed in a trial-and-error fashion without a theoretical insight into *why* they worked.

We have explained how low scaling powers make an optimum less attractive than the surrounding region, as shown in Figure 2. In a *deceptive* search space, numerous local optima around each global optimum would therefore be even more attractive to a GA with fitness sharing, which would make the effect of deception appear worse that it really is. A high scaling power can compensate for a deceptive search space, as well as compensating for the "stay away from the optimum" effect of fitness sharing at low scaling powers. For a deceptive search space, users often justify high scaling powers only to "increase the capacity of the niches centered on the global optima, relative to the niches centered on deceptive attractors" [6], unaware of the additional effect of fitness sharing that this paper explains. Deception is only *partly* to blame.

To compensate for high scaling powers, a larger population can achieve satisfactory results, by causing a rare "super-individual" to occur in every single payoff region. For example, Goldberg *et al* [6] used fitness sharing on a deceptive space, and found that the algorithm worked well using a large population (of 5000 individuals) and a very high scaling power (of 15). Their qualitative reason was that a scaling power was needed to make global optima much more attractive than deceptive non-global optima. But our non-deceptive problem behaved similarly, and required a similar fix, suggesting that deception was only *partly* to blame. Also, they made no justification — "We achieved stable subpopulations at each global optima by using [a scaling power of] $\beta = 15$ and a population size of 5000" [6], indicating a trial-and-error increase of population and scaling power until global optima were found. This paper fully explains why high scaling powers with large populations were needed. It should be kept in mind that large populations usually require long computation time.

Our result also sheds light on another aspect of the speciated genetic algorithm. In nature, as in genetic search, evolution is good at finding near-optimal solutions but not optimal solutions, and so it is common to augment a GA with another method [7, page 50], such as hill-climbing. Exactly why has been a mystery, until now. As shown in Figure 2, fitness sharing with no scaling or with low power scaling causes subpopulations to form *around* an optimum, instead of at it. Hill-climbing is a rather *ad hoc* way to cover that gap.

Our deeper insight allows us to invent more sensible innovations — even our simple suggestion of annealing the scaling power, as described below, gives improved results.

3.2. An Obvious Solution: Annealing the Scaling Power

Genetic algorithms derive their power by balancing the priorities of exploration and exploitation. With a fixed scaling power, we unbalance these priorities. So we vary the scaling power with time. A low power early in the run would enhance the exploration of new regions, a high power later in the run would drive the sub-populations towards their optima, removing the need for a hill-climbing step. This simple idea, even when implemented in a simple manner, produces improved results. Figure 5 shows the average number of peaks found for fixed scaling powers, and for a varying scaling power. The varying scaling power β starts at $\beta = 1$ until generation 50, and rises linearly to $\beta = 10$ at generation 200.

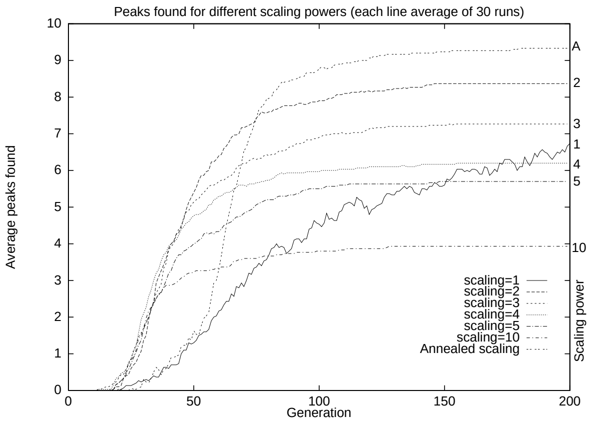

Fig. 5: Annealing the scaling power gives better results than any fixed scaling power.

We do not insist that annealing the scaling power is the best possible technique. In fact, this annealing schedule is relatively inflexible as it does not take into account the state of the population, and something more general could be devised. It merely shows that even this simple application of our explanation can give better results. The fact that annealing the scaling power gives better results, as predicted by our explanation, demonstrates the practical benefits of a deeper theoretical understanding of genetic algorithms.

4. Conclusion

Fitness sharing with a scaling function suffers a serious dilemma. For our and many other problems, the best scaling function is a power function, and:

- Too high a power means an individual slightly closer to an optimum receives so much payoff it becomes a "super-individual" (even with ranked selection to prevent it) and drives to extinction other individuals not as close to their optimum. Thus, fewer optima are found.
- Too low a power means that species stay around the optimum because any individual at the optimum must share with more individuals, and thus have a lower shared payoff.

Knowing that this dilemma exists can let us understand why some extensions are commonly made to GAs with fitness sharing. Using a low scaling power gives results resembling deception, i.e., nearby sub-optimal regions are more attractive, due to the effect on shared payoff as shown in Figure 2. High powers are a fix for deception, and we have seen they also fix the problems of low scaling powers, which is why this effect has been overlooked until now. To compensate for the new problems of high scaling powers, a larger population can achieve satisfactory results by causing a rare "super-individual" to occur at each payoff region, but this wastes computational power. Another common way to compensate for low scaling powers is to use a local search algorithm to bridge the gap from near-optimal species to the nearby optimum.

With the deeper insight that our explanation provides, the reason for using these *ad hoc* techniques becomes clear, and provides the background to invent more effective extensions. Our simple suggestion of annealing the scaling power works better than any fixed scaling power, as predicted by our understanding of the dilemma facing fixed scaling powers. That such a simple extension can explain why hill-climbing is necessary, and suggest other ways to meet that necessity, demonstrates the power of a deeper theoretical understanding of genetic algorithms.

Acknowledgements

We would like to thank H. Foote, P. Triantafillopoulos, and R. Clifford for administrative support, and A. Brown for literature searches.

References

[1] H. C. A. Andersen, *A Constructive Algorithm for a Multilayer Perceptron based on Co-Operative Population Concepts in Genetic Algorithms.* Master's thesis, University of Queensland, 1993.

[2] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, 1993.

[3] P. Darwen and X. Yao, "How good is fitness sharing with a scaling function," Tech. Rep. CS 8/95, University College, The University of New South Wales, Canberra, Apr. 1995.

[4] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 42–50, Morgan Kaufmann, June 1989.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[6] D. E. Goldberg, K. Deb, and J. Horn, "Massive multimodality, deception, and genetic algorithms," in *Parallel Problem Solving from Nature 2*, pp. 37–46, North-Holland, Sep. 1992.

[7] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, pp. 44–50, July 1992.

[8] J. Horn, D. E. Goldberg, and K. Deb, "Implicit niching in a learning classifier system: Nature's way," *Evolutionary Computation*, vol. 2, no. 1, pp. 37–66, 1994.

[9] V. Kreinovich, C. Quintana, and O. Fuentes, "Genetic algorithms: What fitness scaling is optimal?," *Cybernetics and Systems*, vol. 24, no. 1, pp. 9–26, 1993.

[10] S. W. Mahfoud, "Genetic drift in sharing methods," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 67–72, IEEE Press, June 1994.

[11] S. W. Mahfoud, "Population sizing for sharing methods," Tech. Rep. 94005, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Aug. 1994. To appear in the Third Workshop on the Foundations of Genetic Algorithms.

[12] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evolutionary Computation*, vol. 1, no. 2, pp. 127–149, 1992.