An Explicit Spatial Model for Niching in Genetic Algorithms

Grant Dick

[1]Department of Information Science
University of Otago. Dunedin, New Zealand
Phone: +64 3 479-7391 Fax: +64 3 479-8311
Email: gdick@infoscience.otago.ac.nz

Presented at SIRC 2003 - The 15th Annual Colloquium of the Spatial Information Research Centre
University of Otago, Dunedin, New Zealand
December 1st-3rd 2003

ABSTRACT

A niching technique is an important component of the genetic algorithm when attempting to solve problems that have multiple optimal solutions. Traditional niching techniques use an explicit concept of similarity to perform the actual niche formation. Often, the definition of this similarity function is difficult or requires *a priori* knowledge of the problem domain. This paper investigates the use of an explicit spatial structure to perform niching. This technique differs from other niching techniques in that it does not require a definition of similarity between individuals in order to form niches. Early results indicate that using this technique can allow a GA to maintain multiple peaks in some multi-modal functions.

1 Introduction

A vast number of problems exist in the real world for which there is no single global solution. When traditional genetic algorithms are applied to these problems they have difficulty in discovering all optima. This has lead to the development of niching techniques for use in GAs which can assist in the simultaneous discovery of multiple optima.

The sdGA is a new technique which disperses a population onto a two dimensional surface. This paper investigates the sdGA's ability to provide niching properties to genetic algorithms. The applicability of the sdGA to niching problems is interesting because it does not require a definition of similarity between individuals to be supplied to the system and instead relies on spatial distribution to form stable subpopulations.

This paper is broken into the following sections: Section 2 presents a brief discussion of some niching methods and the implications of their use, Section 3 introduces the sdGA, Section 4 presents and discusses the results of experiments using the sdGA and Section 5 concludes the paper with a summary and suggestions for future work in this field.

2 Niching Techniques

There are a number of techniques that permit niching to occur in genetic algorithms. A comprehensive study of these was performed by Mahfoud (Mahfoud 1995). This section briefly describes two common niching techniques that will be used in later in this paper.

2.1 Sharing

Fitness sharing was first implemented by Goldberg and Richardson for use in multi-modal functions (Goldberg & Richardson 1987). This technique uses the concept of similarity to determine the degree of sharing between individuals in the population. An individual's fitness is then scaled by the degree of similarity that it has with other population members.

There are two important components of fitness sharing. The first is the distance function. This is used to determine the amount of overlap between individuals. The distance function can work on either the genotype or the phenotype of an individual. The second component is the sharing function. A sharing function will return a value of 1 for identical individuals and values closer to zero for increasingly dissimilar individuals. The sharing function will return zero if the two individuals cross a “threshold of similarity”. A typical sharing function is

$$sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha}, & \text{if } d < \sigma_{share}; \\ 0\,, & \text{otherwise,} \end{cases} \tag{1}$$

where α is a constant that regulates the shape of the sharing function, d is the value returned by the distance function and σ_{share} is the threshold of similarity.

Once similarity between individuals is established, the shared fitness of an individual can be determined. To set the shared fitness for an individual, the following function is employed:

$$f_s(i) = \frac{f(i)}{\sum_{j=1}^{n} sh(d(i,j)),} \tag{2}$$

where $f(i)$ is the regular fitness of the individual, n is the population size and $d(i, j)$ is the distance function.

2.2 Crowding

Crowding techniques can be used in genetic algorithms with overlapping populations (De Jong 1995). Offspring that are created replace existing individuals based on their similarity. A newly created individual is compared to a randomly chosen subset of the population. The size of this subset is determined by the crowding factor (CF). The individual from this subset which most closely matches the new individual is selected for replacement.

Crowding techniques are inspired by natural phenomenon that occur when similar members in natural populations compete for limited resources. More accurately, individuals in a natural population that can each fill the same ecological niche must compete in order to utilise the resources that niche provides. Likewise, dissimilar individuals tend to occupy different niches and are unlikely to compete for the same resource. This property of crowding techniques has shown to be of some use in multi-modal functions where typical crowding factors $CF = 2$ and $CF = 3$ are used.

Like the fitness sharing technique, crowding requires the use of a similarity function. Early work by DeJong used Hamming distance functions on an individual’s genotype, while subsequent research has investigated the use of phenotypic similarity measures.

2.3 Limitations of Niching Techniques

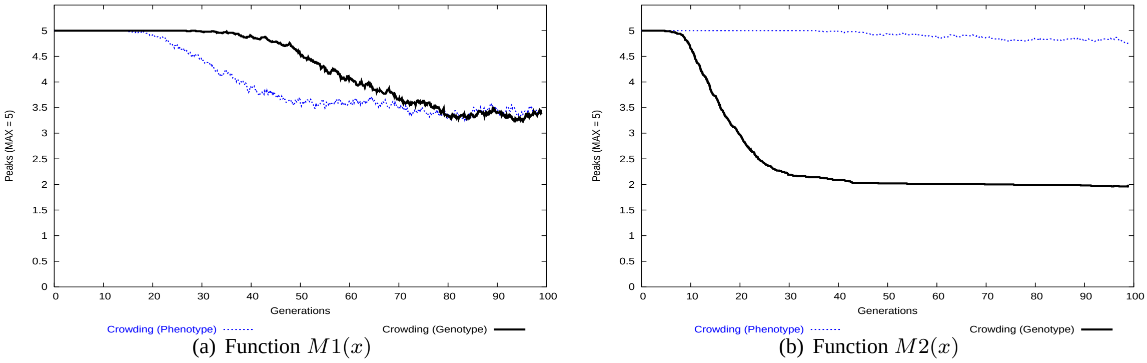
While traditional niching techniques have been successfully applied to a number of different problems, they are not completely without their own drawbacks. One limitation of these traditional methods is the need to calibrate the functions that control the niching process. This often implies *a priori* knowledge of the domain that is being worked on (Goldberg 1989).

While the need for calibration could be considered a minor issue, a more significant concern is the problem-dependent nature of traditional niching techniques (Goldberg & Richardson 1987). In order for these methods to operate, a definition of similarity between individuals must be supplied. Frequently, the best methods for this work in the phenotypic space. If a given problem is not well understood it will be difficult to implement a suitable distance function. In this case, a distance function working on genotypes needs to be employed, which may not perform as efficiently.

To demonstrate the importance of accurate distance measure, the following example is given. The crowding tests for this paper all used a distance function that worked in phenotypic space. A distance function working on genotypes (hamming distance) was also implemented and tested. Figures 1(a) and 1(b) show the results of these tests against trials using the phenotype distance function. As can be seen, the genotype distance function performs better than the phenotype function on $M1(x)$, but is significantly outperformed phenotypic distance measures on $M2(X)$.

3 The Spatially Dispersed GA (sdGA)

The spatially-dispersed genetic algorithm (sdGA) is an alternative method of incorporating population genetics models into genetic algorithms (Dick & Whigham 2002, Whigham & Dick 2002, Dick 2003). The spatial model used to hold the members of the population is a Euclidean space in two dimensions. The space is infinite and continuous. The model places individuals on a two dimensional plane and allows them to breed only with individuals that are “visible” to them.



(a) Function $M1(x)$ (b) Function $M2(x)$

Figure 1: Comparison of crowding distance functions.

3.1 Population arrangement

An individual in a sdGA has a pair of values that represent its location on a two dimensional surface. The individuals in the initial population have their coordinates randomly assigned so that they fit into a bounded area of size x^2 where x is a user specified parameter. For subsequent iterations of the system, this boundary is removed and offspring can be placed in locations that fall outside this area. Figure 2(a) shows a typical distribution of individuals on the plane.

3.2 Individual selection

In a sdGA, the first parent is selected from the entire population via whichever selection method the system is currently using. A second parent is then selected from the deme that is visible to the initial parent. This concept of visibility essentially creates subpopulations based on the location of individuals on the surface. Figure 2(b) shows the selected individual and highlights its visibility radius. Figure 2(c) shows two individuals that could possibly be selected for breeding.

It is important to note that sdGAs do not alter the actual mechanism for selecting an individual (for example, tournament, rank-based or roulette-wheel). The only difference that sdGAs impose is that the second parent is selected from a sub-population which is determined by the first parent.

3.3 Offspring placement

When new individuals are created in the system, they are placed in a location that maintains the relationship with their parents' coordinates. To introduce new offspring into the population, one parent is chosen randomly and the new individual is placed randomly at coordinates which fall within that parent's visibility radius.

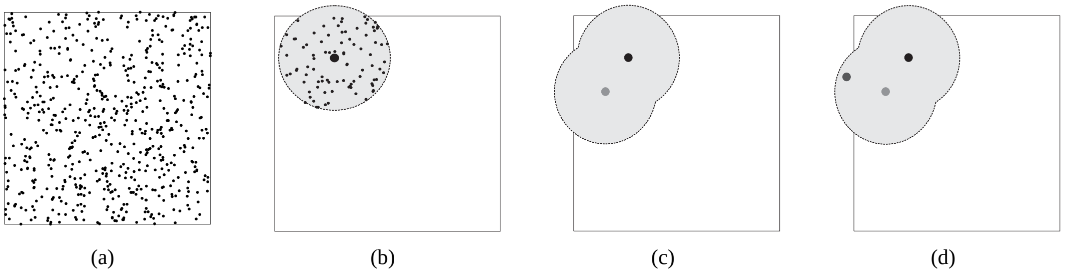


Figure 2: Constraining selection with space: (a) shows the placement of individuals on a plane; (b) shows the visibility radius of the initially selected individual (in bold) and the deme from which the second parent is selected; (c) shows the selected parents and the area available to place offspring; (d) shows the possible location for the parent's offspring.

3.4 Niching with the sdGA

The sdGA has been applied to various different problems with varying levels of success. One particular problem that the sdGA appears to work well on is the Travelling Salesman Problem (TSP). One point that distinguishes the TSP over other problems tested on the sdGA is that the TSP has more than one global optima. The success of the sdGA on the Travelling Salesman problem is the motivation for exploring the use of the sdGA on other problems with several different niches. Hopefully, the spatial properties of the sdGA will somewhat isloate individuals that occupy different niches. This should reduce competition between the niches and help to form stable subpopulations around different peaks.

4 Experimental Results

The sdGA was applied to two sine functions picked from the experimental framework used by Mahfoud. The performance of the sdGA on these functions was compared with the equivalent tests using fitness sharing, crowding and a canonical genetic with panmitic mating. For each function, the number of peaks maintained during the course of a run was collected and averaged over 100 runs.

The tests were initially run with a population size of 200. In addition, the crowding, canonical GA and sdGA tests were repeated with a larger population size of 2000 individuals. All tests were run using a multi-point crossover where the number of crossover points was randomly selected from a poison distribution ($\lambda = 2.816$). The probability of crossover was set to 1.0 and no mutation was used for any tests. All experiments were run with stochastic remainder selection. Each test was allowed to run for 100 generations. The sdGA tests placed the initial population within the bounds [-25,-25] and [25,25]. A crowding factor, CF = 2, was used for all the crowding tests. The distance function for both the crowding and fitness sharing tests worked in the phenotypic space and was defined as $|x_i - x_j|$ where x_n is the decoded bit-string of individual n. The bit-string length for all tests was set to 30 bits.

4.1 Function M1(x)

The first function, $M1(x)$, was defined as

$$\begin{aligned} M1(x) &= \sin^6(5\pi x) \; ; \\ 0 <= x &<= 1.0. \end{aligned} \tag{3}$$

This function has five equidistant maxima all of equal value. A plot of this function is shown in Figure 3.

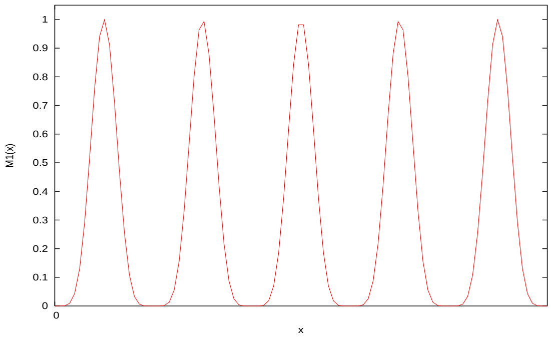


Figure 3: Plot of function $M1(x)$

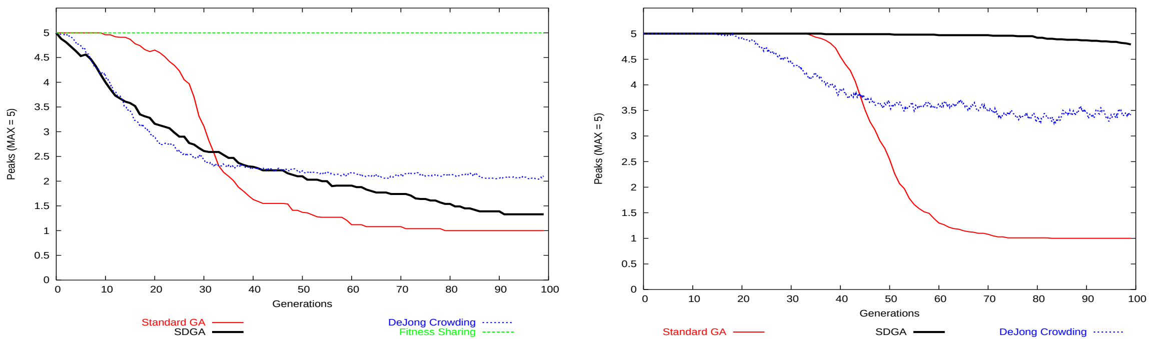
Figure 4 shows the number of peaks maintained by each system on function $M1(x)$ with populations sizes of 200 and 2000 respectively. With a small population size, crowding and the sdGA both perform poorly, being outperformed by the canonical GA in the early generations. The fitness sharing technique is the only algorithm capable of consistently maintaining all 5 peaks with the small population size. The sdGA improves significantly when a larger population size is introduced. The crowding algorithm also improves with a larger population, although not to the same level as the sdGA.

4.2 Function M2(x)

The second test function, $M2(x)$, was defined as

$$\begin{aligned} M2(x) &= e^{-2(\ln\;2)(\frac{x\,-\,0.1}{0.8})^2} \sin^6(5\pi x) \; ; \\ 0 <= x &<= 1.0 \end{aligned} \tag{4}$$

This function has five equidistant maxima that decrease in magnitude with increasing x value. A plot of this function is shown in Figure 5.



(a) Population Size 200

(b) Population Size 2000

Figure 4: Number of peaks maintained, function M1(x)

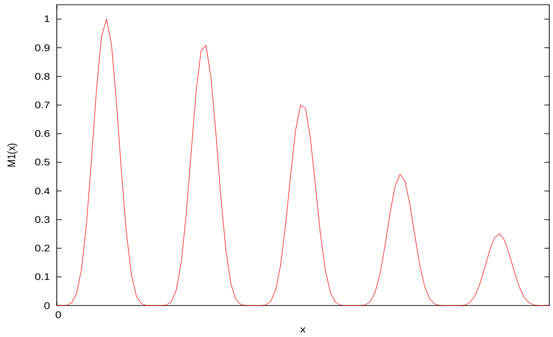
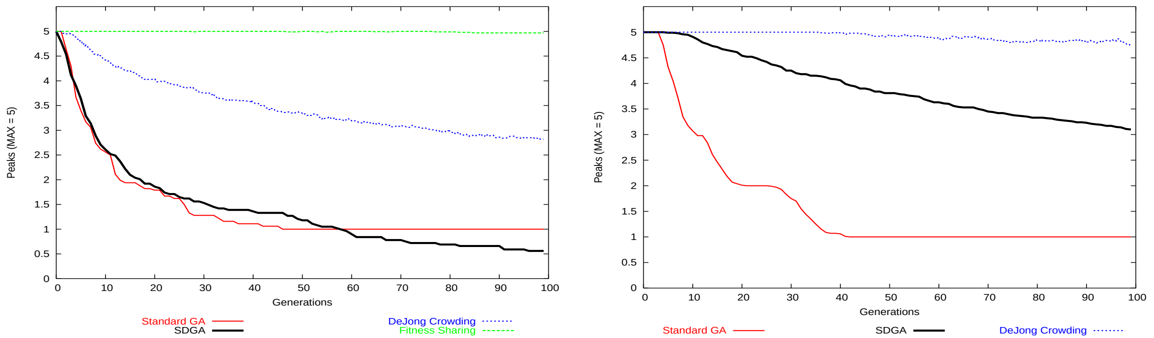


Figure 5: Plot of function $M2(x)$

Figure 6 shows the number of peaks maintained by each system on function $M2(x)$ with population sizes of 200 and 2000 respectively. As observed with function $M1(x)$, the fitness sharing technique has little trouble maintaining all maxima throughout the duration of a run. Using a small population, the crowding technique is able to maintain half the peaks of the function. The sdGA performs very poorly, being outperformed by the canonical GA. The sdGA frequently loses all five peaks when using the small population size.

The sdGA and crowding both improve significantly when a larger population size is introduced. The crowding technique is able to maintain all five peaks on most runs. The sdGA is also able to maintain significantly more peaks when allowed to use a larger population.



(a) Population Size 200

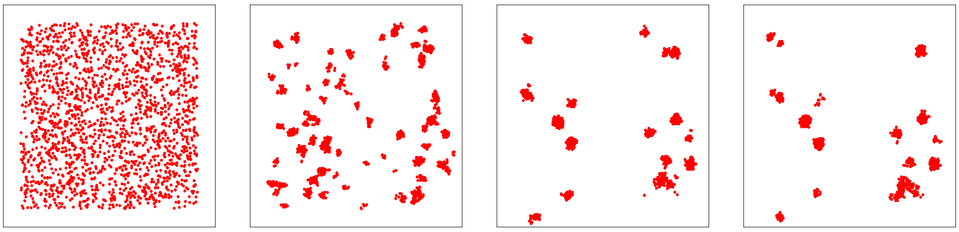
(b) Population Size 2000

Figure 6: Number of peaks maintained, function M2(x)

4.3 Discussion of Results

The crowding and sdGA techniques exhibit different properties depending on the population size used. Both are much more successful at maintaining multiple peaks when they are given larger populations. This could be partly due to the genetic drift present in the smaller population. Genetic drift should be less of a concern in the larger population, which would give individuals on different peaks a better chance of being selected.

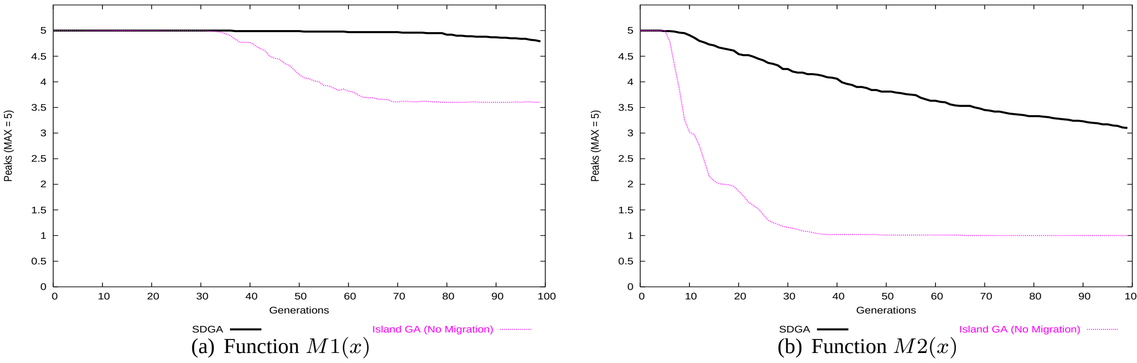
The sdGA appears to be able to provide some niching abilities to genetic algorithms. This effect emerges purely by the inclusion of space–there is no concept of similarity between individuals implemented in the sdGA. Niching properties are provided by distributing the individuals onto a surface and allowing subpopulations to form around isolated fit individuals. Figure 7 shows the formation of small, isolated populations forming on the surface during a typical run of the sdGA. By 25 generations, small populations are forming, which continue to increase in density in future generations. In the final generation, a small number of tightly clustered subpopulations scattered over the landscape have emerged.



(a) Generation 0 (b) Generation 25 (c) Generation 75 (d) Generation 100

Figure 7: Population arrangement over time for typical sdGA run.

Since the sdGA produces small, isolated, non-communicating subpopulations on a surface, it would be reasonable to assume that an island GA with no migration could produce similar results. Figures 8(a) and 8(b) show a comparison between the sdGA and a 10-island GA without migration on both test functions. Tests using an island GA with migration were also run. They are not included because they actually performed worse than the tests that did not use migration. Although the island GA is able to maintain multiple peaks on function $M1(x)$, it cannot prevent the loss of the smaller peaks present in function $M2(x)$. The sdGA is able to significantly outperform the island model on both $M1(x)$ and $M2(x)$. This would indicate that, in addition to the inclusion of space, there are other, more subtle properties of the sdGA that increase the likelihood of niches forming.



(a) Function $M1(x)$ (b) Function $M2(x)$

Figure 8: Comparison of sdGA and an island GA with no migration.

5 Conclusion

Niching techniques play an important role in genetic algorithms. The use of niching methods to promote the formation of stable subpopulations on different peaks can help in the solving and understanding of many problems. Traditional niching implementations require the use of a similarity function, which needs to be defined by the user. This paper has explored the use of the sdGA for its niching properties. Unlike other niching techniques, the sdGA does not require a function that determines similarity between individuals. Instead, the sdGA uses space to isolate individuals in different niches, reducing the competition between them. This property of the sdGA could be useful if a method for determining similarity between individuals was difficult to define.

5.1 Future Work

This is the first piece of work utilising the niching properties of sdGA and has raised some interesting questions that future work could focus on.

- *Discovery of optima in multi-modal problems:-* The results of this paper focused purely on maintaining peaks in multi-modal functions. The applicability of the sdGA o problems where the multiple peaks need to be discovered first before they are maintained is something that now needs exploration.
- *Hybrid implementation:-* The sdGA technique could be potentially used in conjunction with other niching methods. This could potentially lead to even more effective niching properties of both systems.

References

De Jong, K. A. (1995). *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis University of Michigan Ann Arbor. Dissertation Abstracts International 36(10), 5140B; UMI 76-9381.

Dick, G. (2003). “The Spatially-Dispersed Genetic Algorithm” In E. Cantú-Paz (ed.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 2003*. Springer Verlag.

Dick, G. & Whigham, P. (2002). “Spatially constrained selection in evolutionary computation” *Australia-Japan Joint Workshop on Intelligent and Evolving Systems*..

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goldberg, D. E. & Richardson, J. (1987). “Genetic Algorithms with Sharing for Multimodal Function optimization” In J. J. Grefenstette (ed.), *Genetic algorithms and their applications: Proc. of the Second Int. Conf. on Genetic Algorithms*. Lawrence Erlbaum Assoc. Hillsdale, NJ pp. 41–49.

Mahfoud, S. W. (1995). *Niching methods for genetic algorithms*. PhD thesis University of Illinois at Urbana-Champaign Urbana, IL, USA. IlliGAL Report 95001.
*ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/95001.ps.Z

Whigham, P. & Dick, G. (2002). “A study of spatial distribution and evolution” *The 14th Annual Colloquium of the Spatial Information Research Centre, Wellington, New Zealand*.. pp. 157–166.