Dick G, Whigham P. Spatially-structured sharing technique for multimodal problems. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 23(1): 64–76 Jan. 2008

Spatially-Structured Sharing Technique for Multimodal Problems

Grant Dick and Peter Whigham

Department of Information Science, University of Otago, Dunedin, New Zealand

E-mail: gdick@infoscience.otago.ac.nz

Revised November 27, 2007.

Abstract Spatially-structured populations are one approach to increasing genetic diversity in an evolutionary algorithm (EA). However, they are susceptible to convergence to a single peak in a multimodal fitness landscape. Niching methods, such as fitness sharing, allow an EA to maintain multiple solutions in a single population, however they have rarely been used in conjunction with spatially-structured populations. This paper introduces *local sharing*, a method that applies sharing to the overlapping demes of a spatially-structured population. The combination of these two methods succeeds in maintaining multiple solutions in problems that have previously proved difficult for sharing alone (and vice-versa).

Keywords evolutionary algorithm, multimodal problem domain, sharing, spatially-structured population

1 Introduction

Traditional evolutionary algorithms (EAs) have difficulty in discovering and maintaining multiple solutions to multimodal fitness landscapes. Through a combination of selection and genetic drift, an EA tends to converge its population onto a single point in a fitness landscape. While this property of EAs is desirable when one only wishes to identify a single solution to a given problem, this behaviour is undesirable when one wishes to use an EA to simultaneously search for multiple solutions. Previous work has been looked to nature to provide possible solutions for delaying population convergence[1]. One method, *fitness sharing*[2], has received particular attention. This method uses the concept of finite resources to encourage elements of the population to explore different regions of the fitness landscape. Sharing has been shown to generally perform well, although it is known to have difficulty on at least one class of multimodal problem[3].

Another method which claims to encourage population diversity is to impose a structure on the population and restrict selection and mating to geographically close individuals[4]. Previous researchers have noted that, while these *spatially-structured evolutionary algorithms* (SSEAs) do not prevent population convergence to a diversity loss.

Previous work has explored the concept of merging fitness sharing with SSEAs[5]. The author noted that a sharing-like operator, combined with a spatially-structured population, increased an EA's ability to maintain multiple peaks within a single population. However, the work was peculiar in that sharing was applied globally, while selection and mating was confined to demes. A method in which sharing acted locally was proposed, however it has never materialised in subsequent work.

This paper introduces the concept of *local sharing*. The proposed method applies sharing to each deme prior to selection and offspring replacement. This method reduces the overall complexity of sharing, and is applicable to problems that sharing has difficulty handling.

The remainder of this paper is structured as follows: Section 2 describes the concepts of sharing and SSEAs as used in this paper; the proposed local sharing method is described in Section 3 and a comparison between this method and existing SSEAs is given in Section 4 and Section 5. Finally, a brief discussion of the findings of this paper and a suggested path for future work is presented in Section 6.

2 Sharing Methods and Spatially-Structured EAs

The concept of sharing finite resources in an evolutionary algorithm was first proposed by Holland[6], but *fitness sharing*[2] was the first successful attempt at modelling resource contention within a simple EA. With sharing, each optima in the fitness landscape is allocated a finite number of resources in relation to its size. The fitness of a given optimum in space must

Regular Paper

then be shared by the number of individuals representing this solution. This introduces the following two benefits over a simple EA.

1) Rather than attempt to crowd around a single peak, individuals within the population will actively seek out less populated optima. This ensures the constant presence of a selection pressure and counteracts the effects of genetic drift.

2) The effective value of a peak is reduced as the number of individuals exploiting it increases. This in turn makes the lesser-valued optima in the fitness landscape more attractive, increases their relative fitness and drives selection toward these regions. This reduces the likelihood of losing the lesser peaks from the population.

In order for sharing to work, some assumptions need to be made; it is assumed that the optima in the problem occupy similar-sized areas and that they are evenly distributed throughout the fitness landscape[7]. This is due to the mechanism sharing uses to determine which resources are being consumed by each individual. Sharing uses a radius measure, called the *sharing radius*, to determine the amount of similarity between individuals and hence the *niche count* of an individual, given by:

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

where σ_{share} is the sharing radius and α is a value that alters the shape of the function (typically, this is set to 1). In the above function, d represents the distance between two individuals in either a genotypic or phenotypic space. As sharing is applied to reduce the fitness of groups of similarly behaving individuals (i.e., individuals with similar phenotypes), the distance between individuals is ideally measured in the phenotype space[7].

Implementing fitness sharing is a simple extension to standard EAs. Its sole purpose is to dynamically alter the fitness of individuals after evaluation; in terms of implementation, it plays no direct part in the fitness function. The other operators in an EA, such as selection and recombination, are equally unaffected.

Criticism of the sharing method has often focused on the need for a fixed sharing radius and the *a priori* knowledge of the ideal sharing radius[8–10]. Suitable calibration of the sharing radius is required to provide the sharing method with *perfect discrimination* of niches[11]. However, both analytical[12] and empirical[7,13] research into sharing methods suggests that they be capable of supporting multiple niches even in cases of small-to-moderate niche overlap. Derivatives of fitness sharing, such as species identification[14–17], are more sensitive to poor calibration of the sharing radius, although recent research into these methods suggests that adaptive discovery of this parameter might be possible[18, 19].

While the most common implementation of fitness sharing is through the use of a sharing radius, it is by no means the only method of incorporating finite resources into an EA. An alternative method, implicit fitness sharing, uses discrete "objects" (for example, test cases in supervised learning) to enable individuals to compete for limited resources[8,20,21]. We do not explore implicit fitness sharing within this paper, however the techniques developed here are applicable to both explicit and implicit forms of fitness sharing.

2.1 Spatially-Structured Evolutionary Algorithms

Fitness sharing is one method in which an EA can incorporate speciation. Another method is to impose a spatial structure on the population so that there is a concept of geographic distance between individuals. Geographically close individuals form localised subpopulations, or *demes*, within the global population. Mating is confined to demes and offspring are placed within the vicinity of their parents. Evolution potentially acts on demes in different ways so that they be explore different regions of the fitness landscape. Given enough time, individuals within a deme will possess sufficiently different genotypes from those of other geographically distant demes that they could

Algorithm 1. General Sequence for a Spatially-Structured Evolutionary Algorithm
Input: A given problem
Output: A spatially-structured population of evolved candidate solutions to the problem

```
1   population ← {}
2   foreach location in space do
3   |   population [location] ← initialiseIndividual();
4   end
5   while not done do
6   |   generation ← { };
7   |   foreach location in space do
8   |   |   deme ← ConstructDeme(location);
9   |   |   parents ← Select(deme);
10  |   |   offspring ← breedOffspring(parents);
11  |   |   evaluate(offspring);
12  |   |   generation[location] ← pickSurvivor
    |   |   (offspring, deme);
13  |   end
14  |   population ← generation;
15  end
16  return population;
```

be considered a different species.

A spatially-structured evolutionary algorithm (SSEA)[4] is essentially an implementation of the parapatric speciation concept; there are no complete boundaries within the population to restrict gene flow. Instead, SSEAs use the vast distances between some individuals to slow the rate of exchange of genetic material to a sufficiently low level so as to promote local divergence of genotypes[22,23]. Though each SSEA implements a slightly different algorithm, the overall method for using a spatially-structured population within an EA is shown in Algorithm 1. Essentially, there are three basic steps to an SSEA; first, for a given location, construct a deme. After this, select the parents from this deme. Finally, the newly created offspring must be inserted into the deme.

3 Local Sharing Methods for SSEAs

Spears, as part of his simple subpopulation schemes concept, used a spatially-structured population in conjunction with a sharing method operating on tag bits[5]. The sharing component operated once per generation and considered the population as a whole. After this step, selection and reproduction continued as for a typical SSEA. Spears noted that this scheme was more effective at maintaining multiple peaks in a fitness landscape than a simple subpopulation scheme that did not impose a population structure. It is not entirely clear that the increase in performance was down to the combination of an SSEA and the sharing method, or if the improvement was purely through the adoption of spatial constraints in the population. The issue is further complicated by the fact that the sharing component of the algorithm and the local mating did not interact; the population topology played no part in determining the shared fitness of the population members. Spears himself noted this peculiarity and suggested a third subpopulation scheme which used the population topology within the sharing process[5]. However, this new method has not appeared in any subsequent work.

This paper extends Spears' idea to create a new niching method, *local sharing*. The concept is actually rather simple; at the start of a generation, each location calculates the shared fitness of its occupant by determining the individual's niche count with respect to the other members of the deme. After this, selection and reproduction are performed as per a normal SSEA except that individuals are selected via their shared fitness. Upon completion of mating, the shared fitness of offspring are calculated. Offspring are then inserted into the population if their shared fitness exceeds that of the current occupant. The local sharing method, as used in this paper, is described in Algorithms 2 and 3.

One benefit of the local sharing method is a reduced complexity compared to that of panmictic fitness sharing. Typically, fitness sharing is of $O(N^2)$ complexity, as all pairs of individuals must be considered. With local sharing, only individuals that share a common deme are compared, therefore the complexity of local

Algorithm 2. Local Sharing Method
input: A spatially-structured population of evaluated individuals
output: The same population with correctly calculated shared fitness of individuals

```
1   foreach location in space bf do
2       current ← population[location];
3       deme ← constructDeme(location);
4       m ← 0;
5       foreach member ∈ deme do
6           d ← dist(current, member);
7           m ← m + sh(d);
8       end
9       current.fitness ← current.objective/m;
10  end
11  return population;
```

Algorithm 3. General Sequence for a Spatially-Structured Evolutionary Algorithm with Local Sharing
input: A given problem
output: A spatially-structured population of evolved candidate solutions to the problem

```
1   population ← { };
2   foreach (location in space) do
3       population[location] ← initialIndividual();
4   end
5   while not done do
6       localSharing(population, σ_sh);
7       generation ← { };
8       foreach location in space do
9           deme ← constructDeme(location);
10          parents ← select(deme);
11          generation[location] ← breedOffspring(parents);
12          evaluate(offspring);
13      end
14      localSharing(generation, σ_sh);
15      foreach location in space do
16          generation[location] ← pickSurvivor(population[location], generation[location]);
17      end
18      population ← generation;
19  end
20  return population;
```

sharing given a deme size of d is $O(2Nd)$. As $d \ll N$ holds for most SSEAs, this represents a considerable saving in the number of distance comparisons required. Another benefit of local sharing is the number of sharing instances that are performed per generation. Local sharing performs multiple, smaller instances of sharing in parallel per generation. The isolation of demes allows each sharing instance to concentrate on different parts of the fitness landscape, increasing the chance of discovering multiple optima. Finally, local sharing permits the use of elitism, which helps to preserve optima once they are discovered.

4 Empirical Analysis — Peak Maintenance

The local sharing SSEA was compared with a traditional SSEA and a panmictic EA incorporating fitness sharing. Each algorithm was tested against two problems. The first problem, referred to as $M6$, is the Shekel's foxhole problem as used in De Jong's dissertation on genetic algorithms[24]. This problem is frequently used in comparisons of niching methods[25] as it has 25 local optima in the fitness landscape. The peaks vary in value, so a typical SSEA will have difficulty maintaining all but the global optimum. $M6$ is defined over a bitstring of length 34, with 17 bits for each variable in the phenotype. The second problem, $M7$, is a massively multimodal, deceptive problem[3]. It is defined as the concatenation of five unitation functions, each unit operating on 6-bit strings. This problem has over five million optima in the fitness landscape, of which only 32 are global and of interest. Fitness sharing struggles with this problem as the deceptive optima lie within the sharing radius of the global peaks and hence compete for the same resources. Together, $M6$ and $M7$ form a good test suite for comparing the hybrid local sharing SSEA. The fitness landscapes of $M6$ and $M7$ are shown in Fig.1.

Population size plays an important role in the performance of EAs. The population size for each of the test problems was taken from previous work. The population size for $M6$ was determined via a population sizing model developed for fitness sharing[26] and was 289 (a 17×17 torus for the SSEAs). The population size for $M7$ was taken from empirical studies of niching and was 676 (a 26×26 torus). This particular population size is considered insufficient for panmictic fitness sharing[3]. Other niching methods, however, are able to solve $M7$ with a population size in this vicinity[15,25].

Selection in all EAs was via fitness proportional selection. Selection in the sharing EA was via stochastic universal sampling[27] to remain consistent with previous work[25]. Roulette-wheel selection was used for the SSEAs, again for consistency with previous work[28]. Two parents were selected (without replacement) to create offspring. A single offspring was created via one-point crossover for the SSEAs, while the fitness sharing EA created two offspring for each reproduction instance.

The parameters for each test were as follows: one-point crossover with probability 1.0; mutation was via bit-flipping applied with probability 0.002 per locus; the sharing radius σ_{share} was 8 for $M6$ and 6 for $M7$; each SSEA used a torus population structure and a Von-Neumann neighbourhood was used to determine demes; elitist replacement of offspring was used in both SSEAs (via shared fitness for local sharing); each run of an EA lasted for 500 generations. All results presented in this paper are plotted with 99% confidence intervals. In the following section, a peak was considered "discovered" if it was populated by an individual with a fitness greater than or equal to 99% of the given peak's value.

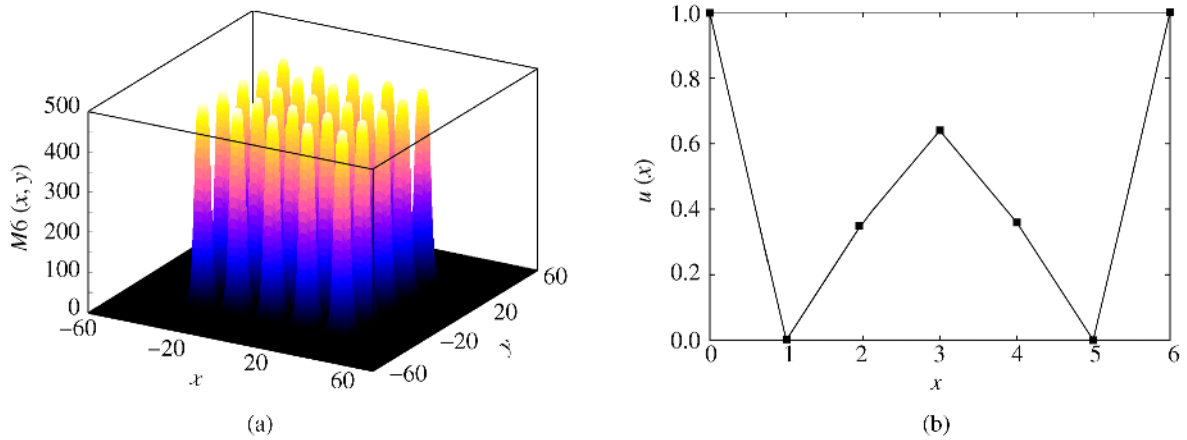

Fig.1. Fitness landscapes of the test problems used in this paper. (a) $M6$. (b) $M7$ (1 unit).

4.1 Results

The performance of each EA on the test problems was measured through two statistics used in previous niching studies[7]. The first measurement records the number of peaks discovered and maintained by the population over 500 generations. Higher values for the number of maintained niches indicates better algorithm performance. The second measure is the Chi-Square-Like performance which measures the deviation of the population from that of an "ideal" population in which all individuals reside on a peak in distributions relative to a peak's value. Lower values for the Chi-Square-Like measure indicate better performance (a value of zero being ideal). These measurements were averaged over 500 runs and are reported below.

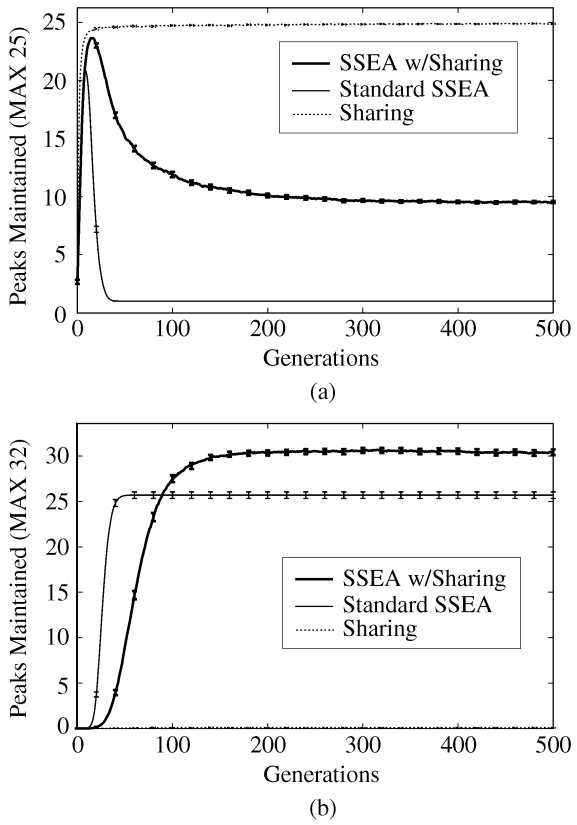

Fig.2. Number of peaks maintained by each algorithm on $M6$ and $M7$. (Error bars are 99% confidence intervals.) (a) $M6$. (b) $M7$.

The number of peaks maintained over time is shown in Fig.2. On $M6$, local sharing maintains around 40% of all optima present in the fitness landscape after 500 generations. However, local sharing on $M7$ is far superior to that of either panmictic sharing of the traditional SSEA, with nearly all 32 optima present in the population after 500 generations. The Chi-Square-Like performance of the three EAs, as shown in Fig.3, tells a similar story to the number of peaks performance; local sharing distributes individuals among peaks more efficiently than the traditional SSEA on both $M6$ and $M7$ and is superior to panmictic sharing on the $M7$ problem.

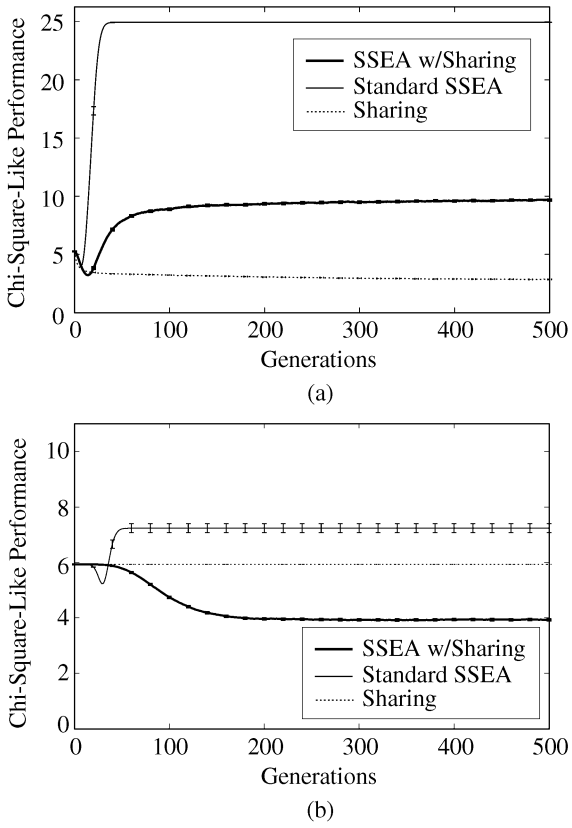

Fig.3. Chi-Square-Like performance of the tested EAs on $M6$ and $M7$. (Error bars are 99% confidence intervals.) (a) $M6$. (b) $M7$.

The results presented here indicate that local sharing is able to discover and maintain more optima than a traditional SSEA. Local sharing is also able to support multiple optima on $M7$, a problem that has traditionally been difficult to handle using sharing methods. However, the performance of local sharing on $M6$ does not match that of panmictic sharing. This may be in part due to the elitist replacement of local sharing; too much emphasis is placed on the global optimum by elitism at the expense of other niches. Elitism plays an important role in local sharing's performance on $M7$, so elitist replacement cannot be simply removed from the local sharing method. Instead, we propose a change in elitism from a strict, "always better than"

approach to a probabilistic method. In the proposed strategy, offspring enter a tournament with the current occupant to determine who is passed into the next generation. The probability of an offspring o winning the tournament over the current occupant i is:

$$p(o) = \frac{f_{\text{sh}}(o)}{f_{\text{sh}}(o) + f_{\text{sh}}(i)}$$

where f_{sh} is the shared fitness of an individual. Experiments on $M6$ were repeated using this alternative elitism method and the results are shown in Fig.4. The probabilistic elitism method allows local sharing to support more optima within the population, and the distribution of individuals among those peaks is much closer to that of panmictic sharing (as indicated by the Chi-Square-Like performance). However, this proposed elitism method is detrimental to the performance of local sharing on $M7$, as shown in Fig.5.

Rather than adopting a probabilistic elitism strategy, an alternative method to increasing local sharing's performance on $M6$ could be to increase the deme size. This would promote a greater effect of sharing within the population and help to maintain the presence of the lesser valued optima within the population. The local sharing and SSEA were reapplied to $M6$, this

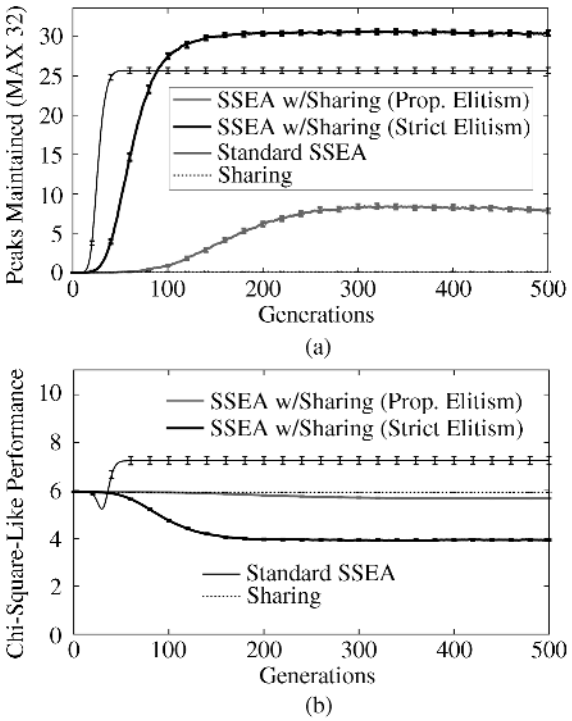

Fig.5. Comparison of elitism strategies for the local sharing method on $M7$. (Error bars are 99% confidence intervals.) (a) Peaks maintained. (b) Chi-Square-Like performance.

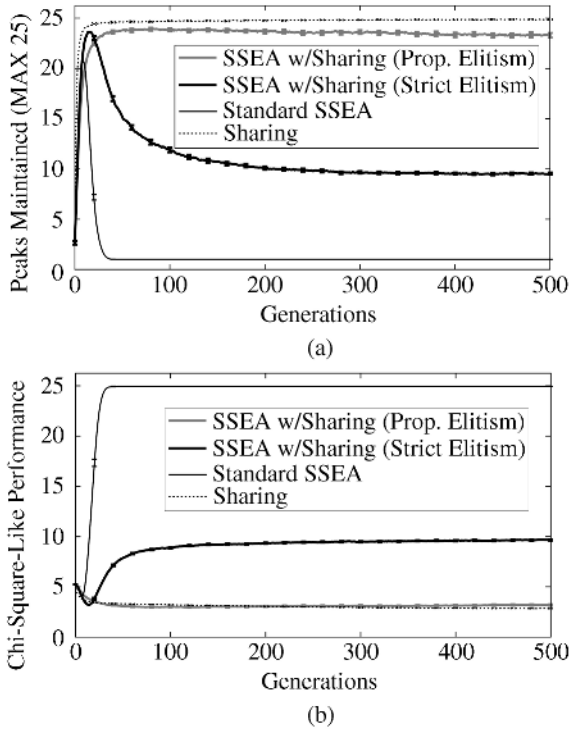

Fig.4. Comparison of elitism strategies for the local sharing method on $M6$. (Error bars are 99% confidence intervals.) (a) Peaks maintained. (b) Chi-Square-Like performance.

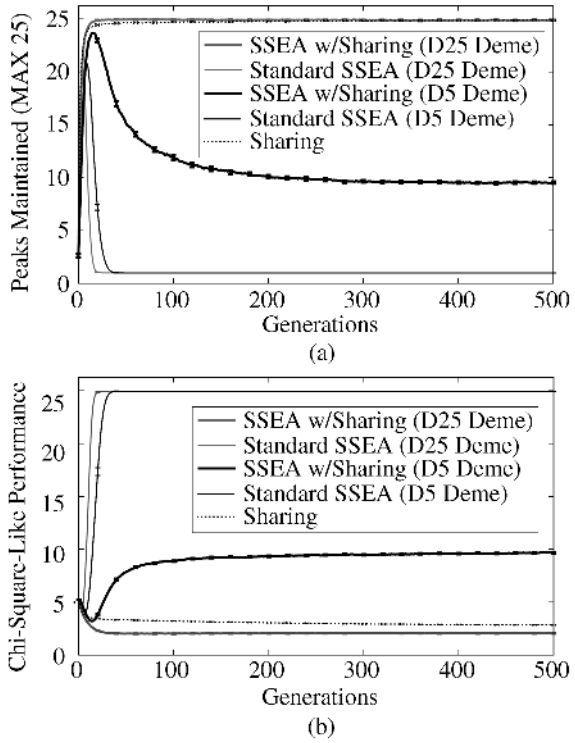

Fig.6. Comparison of deme sizes for the local sharing method on $M6$. (Error bars are 99% confidence intervals.) (a) Peaks maintained. (b) Chi-Square-Like performance.

time using a larger deme size of 25 individuals in a diamond configuration, commonly referred to as a "D25" deme[28]. As shown in Figs. 6 and 7, increasing the deme size for local sharing improves its peak maintenance performance. However, as increasing the deme size shifts the behaviour of local sharing to that of panmictic sharing, the performance of local sharing on $M7$ drops. This drop is not as significant as the change in elitism strategy, as shown in Fig.7. It is likely that using a larger population size would rectify local sharing's behaviour on $M7$. Therefore, using a D25 deme may provide the best trade-off between the spatially-structured and sharing characteristics of local sharing.

5 Empirical Analysis — Evaluation Efficiency

The previous section demonstrated the ability of local sharing to maintain peaks in two multimodal functions. However, the ability to locate multiple optima is only one desirable trait of a good niching algorithm. Additionally, a good niching EA must locate multiple optima in an efficient manner, i.e., with as few function evaluations as possible.

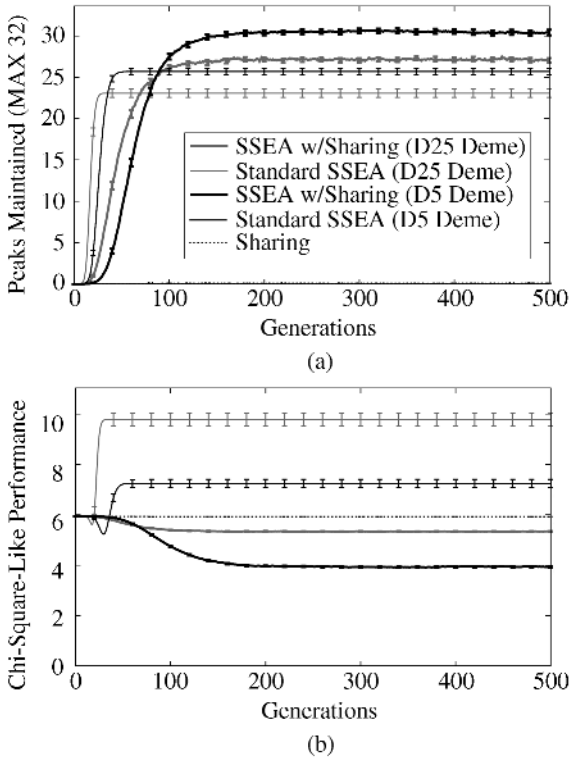

Fig.7. Comparison of deme size for the local sharing method on $M7$. (Error bars are 99% confidence intervals.) (a) Peaks maintained. (b) Chi-Square-Like performance.

Mahfoud presented a framework that compared the function evaluation performance of several niching EAs over a range of problems of varying difficulty. In this section, we will use Mahfoud's framework, with minor modification, to compare the function evaluation requirements of local sharing to that of panmictic sharing and a typical SSEA. A local search method, parallel hillclimbing, will also be included for comparison (and post-processing of the final population evolved by an EA).

The test framework uses eleven problems. We have included nine of those problems here ($M1{\sim}M9$). Two of these problems were used in the previous section ($M6$ and $M7$). $M1{\sim}M4$ are simple five-peaked sinusoidal problems defined as:

$$M1(x) = \sin^6(5\pi x),$$

$$M2(x) = 2^{-2(\frac{x-0.1}{0.8})^2} \sin^6(5\pi x),$$

$$M3(x) = \sin^6(5\pi[x^{0.75} - 0.05]),$$

$$M4(x) = 2^{-2(\frac{x-0.08}{0.854})^2} \sin^6(5\pi[x^{0.75} - 0.05]).$$

$M5$ is the modified Himmelblau function, a two-dimensional function with four global optima:

$$M5(x, y) = \frac{2186.0 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2}{2186.0}.$$

$M6$ is the Shekel's foxholes problem as used in the previous section:

$$M6(x, y) = 500 - \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{1 + i + (x - a(i))^6 + (y - b(i))^6}}$$

where $a(i) = 16[(i \bmod 5) - 2]$ and $b(i) = 16[(i/5) - 2]$. $M7$ is the massively multimodal, deceptive problem as used in the previous section. $M8$ is an exponentially scaled transformation of $M7$ that is more amenable to searching via sharing methods[3]. Finally, $M9$ is the minimum-distance function, with 2197 local optima, of which 27 are global[29].

Mahfoud's framework also includes two classification problems, *MUX*-6 and *PAR*-8. However, as shown in Mahfoud's study, the number of required function evaluations to solve these problems was actually much larger than the total search space. Essentially, an exhaustive, iterative search of the problem domain would be more efficient than searching via any niching EA. Therefore, for this study we replace these problems with the following examples. The first problem, *ABS*

is similar to $M7$ in that it is a unitation function applied to a bitstring. The fitness function of a unit is defined as:

$$ABS(u) = \frac{|u-3|}{3}$$

where u is the value of unitation. ABS is defined over a 30-bit string and has five units. As with $M7$, the function presents two global optima ({000000} and {111111}). Therefore, ABS has the same number of global optima (32) as $M7$, however there are no other optima in the search space and no deception is present. The second problem is the two-dimensional Shubert (SHU) problem used in other studies of niching EAs[16]. As sharing methods require the range of possible fitness values to have the same sign (all positive or all negative), we have transformed the function into a maximisation problem with minimum fitness of zero. SHU is therefore defined as:

$$SHU(x,y) = 211 - \left(\begin{array}{c} \sum_{i=1}^{5} i\cos[(i+1)x+i] \\ . \\ \sum_{i=1}^{5} i\cos[(i+1)y+i] \end{array} \right).$$

SHU has 760 local optima of which 18 are global.

The point at which to stop running an EA is typically difficult to determine. The adopted framework uses a similar halting mechanism to that proposed by Beasley *et al.* for use with their sequential niching method[30]. A halting window of five generations is used. If the average fitness of the current population is not more than some increment *inc* greater than the average fitness of the population five generations earlier, the EA is halted. The value of *inc* used typically varies with problem; in Mahfoud's study a value of 0.001 was used for all problems except for $M6$ which used a value of 0.1. We adopt the same values for this study.

Mahfoud argues that EAs are global optimisers, and as such should be augmented with local search methods to accurately locate optima. Therefore, after an EA has converged, the population is passed to a hillclimber. For each individual, a local search is performed so that the true local optimum is found. After post-processing via the hillclimber, the population should consist entirely of locally optimal individuals. For this paper, we used the same parallel hillclimber as described in Mahfoud's study[25].

Population size plays an important role in the efficiency of an EA. Obviously, some algorithms will require larger population sizes than others in order to properly search a given fitness landscape. The framework used here is designed to determine the smallest population size an algorithm requires in order to solve a given problem. It starts by using a population that is both a power of two and large enough to find all optima in the fitness landscape. If, after halting and post-processing with a hillclimber, the EA was unable to discover all optima, the population size is doubled and the algorithm is restarted. This process repeats until the algorithm finds all desired optima, or exceeds a predefined number of function evaluations. In this study, this limit is set to 1.5 million evaluations from the EA, or two million evaluations from the combined EA + hillclimber.

The sharing radius used for each problem was the same as that used in Mahfoud's study. $M1{\sim}M4$ used a niche radius of 0.1; $M5$, 4.24; $M6$, 8.0; $M7$ and $M8$, 6; and $M9$, 3. The two other problems, ABS and SHU used radii of 6 and 0.8 respectively. $M7{\sim}M9$ and ABS used genotypic (hamming distance) distance measures, while the remaining problems all utilised a distance measure in the phenotypic space.

We used the same parameters for each EA as Mahfoud, namely one-point crossover applied with 100% probability, no mutation and stochastic universal sampling as the selection method for sharing. Local sharing and the SSEA both used roulette-wheel selection. Both local sharing and the SSEA used a torus topology. The SSEA used the same Von-Neumann neighbourhood as in Section 4, while the local sharing method used the larger D25 deme. These particular deme sizes were selected as they demonstrated the best overall peak discovery performance for each algorithm (as demonstrated in Section 4).

Each algorithm was tested against each problem with the same seed for the random number generator. This means that each algorithm attempted each problem with the same initial population. The random number was reseeded whenever an algorithm was restarted with a larger population. To establish meaningful averages, each algorithm was applied to each test problem 500 times, using a different random number seed for each sample.

5.1 Results

The problems have been grouped in a similar manner to that suggested by Mahfoud; $M1{\sim}M5$ and ABS are grouped together as "easy" problems, $M6$ and SHU are considered to be of "moderate" complexity and $M7{\sim}M9$ are classified as "hard". The population sizes and average function evaluations for each algorithm for each of these problem groups are presented in Tables 1∼3.

It is obvious from Table 1 that the easy problems

Table 1. Performance of Hillclimbing (HC), Sharing (SH), SSEA and Local Sharing (LS) on the Six Easiest Test Functions

	Method	$\overline{n}$	GA: μ	Total: μ	Total: σ
$M1$	HC	17	17	**540**	290
	SH	32	280	1 216	595
	SSEA	214	4 091	*9 337*	4 517
	LS	40	543	1 706	960
M_2	HC	16		**515**	254
	SH	39	367	1 537	762
	SSEA		>1 500 000	*>2 000 000*	
	LS	75	944	3 113	2 275
M_3	HC	18		**556**	340
	SH	30	287	1 190	675
	SSEA	221	4 096	*9 524*	4 866
	LS	37	496	1 559	1 006
M_4	HC	17		**540**	304
	SH	36	350	1 404	733
	SSEA		>1 500 000	*>2 000 000*	
	LS	50	609	2 059	1 266
M_5	HC	12		**1 193**	666
	SH	12	105	1 303	691
	SSEA	122	2 164	*12 828*	7 349
	LS	19	165	2 027	768
ABS	HC	189		**11 540**	4 594
	SH	268	6 943	23 008	7 788
	SSEA	1 614	70 354	*118 789*	42 275
	LS	285	10 438	27 315	9 449

Note: measurements were taken over 500 runs. The average population size is reported in the $\overline{n}$ column. The mean number of function evaluations required for halting are in the GA μ column. The average number and standard deviation of total function evaluations are in columns Total: μ and Total: σ, respectively.

Table 2. Performance of Hillclimbing (HC), Sharing (SH), SSEA and Local Sharing (LS) on the Two Test Functions of Moderate Difficulty

	Method	$\overline{n}$	GA: μ	Total: μ	Total: σ
M_6	HC	259		19 840	9 302
	SH	128	1 553	**10 208**	2 792
	SSEA		>1 500 000	*>2 000 000*	
	LS	156	3 437	13 715	5 880
SHU	HC	2 394		156 964	62 594
	SH	2 214	19 503	162 288	62 090
	SSEA	3 547	240 621	*386 665*	158 837
	LS	658	29 833	**69 907**	39 231

are best solved by hillclimbing alone, which outperforms all other methods on all six problems. Last place on these problems is consistently assigned to the SSEA, which failed to find and maintain all the required optima in $M2$ and $M4$. Local sharing was consistently slower than both panmictic sharing and hillclimbing on all six problems. However, aside from the poor performance of SSEA, these results suggest little about the relative performance of the tested niching methods. In at least four of the six easy problems, the required population size for local sharing is barely larger than the deme size. In the case of $M5$, the required population size is actually smaller than the deme size used. This suggests that the performance of all niching EAs on the easy problems is dominated by the post-processing element of the framework. The hillclimbing effect is masking the performance of the niching EA, and that these problems are too simple for any meaningful comparison to be made.

Table 3. Performance of Hillclimbing (HC), Sharing (SH), SSEA and Local Sharing (LS) on the Three Hardest Test Functions.

	Method	$\overline{n}$	GA: μ	Total: μ	Total: σ
M_7	HC			*>2 000 000*	
	SH		>1 500 000	>2 000 000	
	SSEA	2 250	125 988	**193 491**	74 996
	LS	3 542	223 451	399 097	147 215
M_8	HC			*>2 000 000*	
	SH	731	15 001	**45 756**	19 677
	SSEA	2 850	126 724	212 215	86 084
	LS	824	22 954	47 661	22 157
M_9	HC			*>2 000 000*	
	SH		>1 500 000	>2 000 000	
	SSEA	2 868	185 080	253 914	108 425
	LS	1 118	51 266	**95 780**	43 089

The moderate problems, (Table 2) start to explain more about the behaviour of the various algorithms. Both local sharing and panmictic sharing significantly outperformed hillclimbing on $M6$. Local sharing requires more evaluations on $M6$ than panmictic sharing. This is because it requires a larger population size in order to create a reasonable isolation effect from the spatially-structured population. The larger population will naturally take longer to converge and require more effort from the hillclimber, resulting in an overall increase in the required number of function evaluations.

The local sharing method was considerably faster than all the other algorithms on the Shubert function. Panmictic sharing was slightly slower than hillclimbing alone, although this difference was not significant. Panmictic sharing also required a similar population size, which suggests that its performance on SHU was dictated by the post-processing done by the hillclimber, rather than its own niching abilities. Conversely, local sharing was able to locate the global

optima of *SHU* with a much smaller population, suggesting that it is less dependent on the local search mechanism.

The behaviour of the niching methods becomes clearer when they are applied to the hard problems. The results for these problems are shown in Table 3. As reported in Mahfoud's study, neither panmictic sharing or hillclimbing can solve $M7$ within the allotted number of function evaluations. Unlike its panmictic counterpart, local sharing is able to solve $M7$, however it requires more than twice as many function evaluations as a standard SSEA. Local sharing is slightly slower than panmictic sharing on $M8$, although the difference is not significant. However, on $M9$ local sharing demonstrates a clear advantage, requiring less than 40% of the function evaluations of the next best algorithm. Once again, the results for hillclimbing and panmictic sharing are consistent with earlier results; neither algorithm is able to effectively search the fitness landscape of $M9$.

The performance of local sharing on $M7$ somewhat contradicts the behaviour observed in Subsection 4.1. Here, local sharing is comprehensively outperformed by a standard SSEA. There are two possible explanations for this. One reason may be that, as demonstrated in Subsection 4.1, increasing the deme size of local sharing induces a greater sharing effect on the population, hence local sharing behaves closer to panmictic sharing. As panmictic sharing has great difficulty with this problem, we should expect local sharing with a larger deme size to face similar problems.

Another reason for the poor performance of local sharing on $M7$ may be due to the halting rule used for the experiments. It is possible that local sharing discovers additional optima after the population's mean fitness has converged. If so, the halting rule may be forcing local sharing to stop before all optima can be discovered. As a result, local sharing must rely on larger population, so that the average fitness of the population will converge more slowly. To test this, we ran local sharing and a standard SSEA on $M6$ and $M7$ with typical population sizes (289 and 1024 respectively). The algorithms were run with the normal halting rule. Upon finding the halting point, the algo-

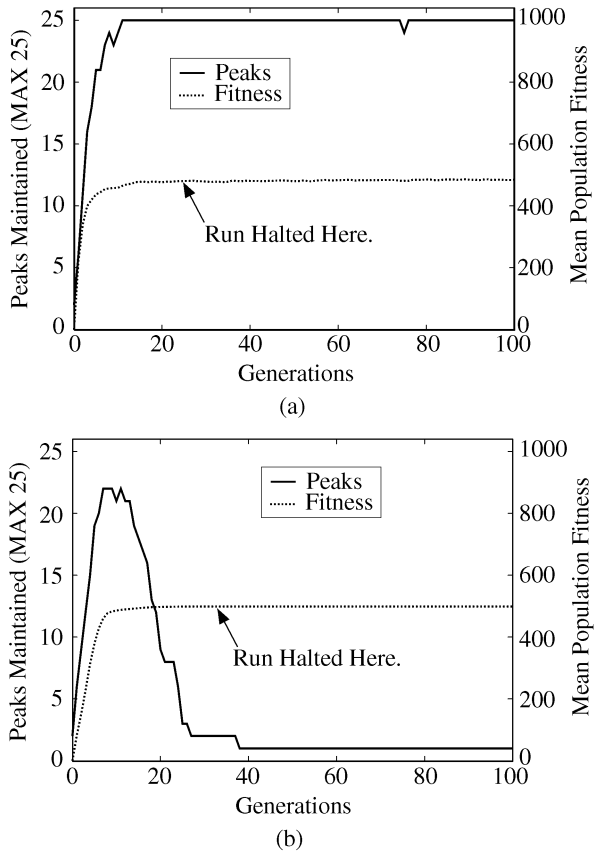

Fig.8. Halting points for a simple SSEA and local sharing on the $M6$ problem. (a) SSEA. (b) Local sharing.

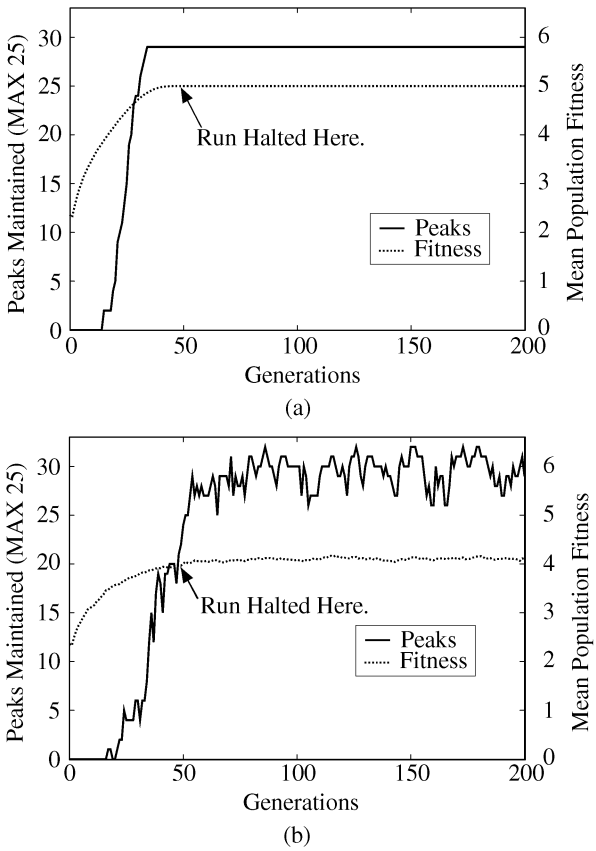

Fig.9. Halting points for a simple SSEA and local sharing on the $M7$ problem. (a) SSEA. (b) Local sharing.

rithms were restarted at the point where they had finished. The number of optima discovered, the population fitness and halting generation for typical runs of these configurations are shown in Figs. 8 and 9. As can be seen, on $M7$, local sharing would have halted long before it had stabilised on the number of optima it would find if more evaluations were performed. The SSEA does not exhibit this behaviour. Additionally, local sharing does not demonstrate this behaviour when applied to $M6$.

6 Conclusions and Future Work

Space often plays an important role in speciation of populations. Traditionally, EAs have used individual-based comparisons to perform niching in the absence of spatial population structure. This paper presents a new method for niching, local sharing, which takes the traditionally panmictic niching method of fitness sharing and applies it within the demes of a spatially-structured population. Initial results suggest that local sharing is able to effectively locate multiple solutions to problems that have proved difficult to solve by either sharing or SSEAs alone.

One of the greatest differences between local and panmictic sharing is elitist replacement of offspring; local sharing provides a simple mechanism with which offspring can compete with the individuals of the previous generation. It is this property that allows local sharing to successfully search fitness landscapes that have previously proved difficult for panmictic sharing. However, the elitist properties of local sharing can cause difficulties in peak maintenance for fitness landscapes where the fitness of the desired optima are unequal.

Local sharing was the only EA that was able to successfully search the 11 test problems within the defined two million function evaluation limit. In most cases, local sharing was not the fastest algorithm in terms of number of function evaluations, however it equaled or outperformed all other tested methods on three out of the five problems of medium to hard complexity. As the only method to solve all five moderate to hard problems, local sharing appears to scale to harder problems more effectively than a panmictic sharing approach.

6.1 Future Work

This paper has presented an initial investigation into the local sharing concept and has highlighted many aspects of its behaviour. Nevertheless, there are several interesting directions that future research into local sharing could take to provide more insights into the nature of spatially-structured niching hybrids.

Fitness sharing forms the basis on which local sharing performs its niching. However, many other niching methods have been discussed in the evolutionary computation literature. One interesting area to investigate would be to test the suitability of other niching methods, for example clearing[15], within the confines of a spatially-structured EA. Initial work in this area is underway and the results are promising[31].

Like panmictic sharing, local sharing requires the use of a niching radius in order to determine species similarity. The necessary calibration of this parameter to a given problem is often mentioned as a criticism of the sharing concept. Unlike panmictic sharing, local sharing performs several independent instances of sharing within a single generation. In essence, each deme acts in isolation from other distant demes. This may allow local sharing to use a "divide and conquer" approach to peak discovery; some demes will focus on a certain region of the fitness landscape, while other demes will work on others. Therefore, a given deme need only distinguish between a few niches, and is thus less reliant on an accurate setting for the niching radius. This is a potentially important and desirable property of local sharing that future work should address, particularly is the concept is applied to sharing derivatives, such as species identification.

Future work should investigate the elitism strategies of local sharing in greater detail. One possible direction might be to implement elitism as a Boltzmann tournament with annealing[32] so that the higher valued optima do not dominate elitism in earlier generations of a run.

As identified in Subsection 5.1, local sharing will continue to discover new optima long after the population's mean fitness has converged. This means that, at least on some problems, local sharing will not work efficiently with the use of a halting rule based on mean population fitness. Future work should investigate alternative halting rules for EAs that are based upon other population statistics. Such work would have potential benefits not only to local sharing, but to any other EAs that concatenate existing niches to discover new optima.

Local sharing is of lower typical complexity than panmictic sharing; when the deme size is fixed it is of linear time complexity with respect to population size ($O(2N)$). However, the implementation of local sharing as used in this paper is not optimised and future work should investigate possible implementations that are of even lower complexity.

References

[1] Mahfoud S W. Niching methods for genetic algorithms [Dissertation]. University of Illinois at Urbana-Champaign, Urbana, IL, USA, IlliGAL Report 95001, May 1995.

[2] Goldberg D E, Richardson J. Genetic algorithms with sharing for multi-modal function optimisation. In *Proc. the 2nd Int. Conf. Genetic Algorithms and Their Applications*, Massachusetts, USA, 1987, pp.41–49.

[3] Goldberg D E, Deb K, Horn J. Massive Multimodality, Deception, and Genetic Algorithms. Parallel Problem Solving from Nature, 2, Männer R, Manderick B (eds.), Amsterdam: Elsevier Science Publishers, B. V., 1992, pp.37–46.

[4] Tomassini M. Spatially Structured Evolutionary Algorithms. Springer, 2005.

[5] Spears W M. Simple subpopulation schemes. In *Proc. the Third Annual Conf. Evolutionary Programming*, Sebald A V, Fogel L J (eds.), Singapore, World Scientific Press, 1994, pp.296–307.

[6] Holland J H. Adaptation in Natural and Artificial Systems. Cambridge, Massachusetts: The MIT Press, 2ed., 1992.

[7] Deb K, Goldberg D E. An investigation of niche and species formation in genetic function optimization, In *Proc. the Third Int. Conf. Genetic Algorithms*, Schaffer J D (ed.), San Mateo, CA, Morgan Kaufmann, 1989, pp.42–50.

[8] Darwen P, Yao Y. Every niching method has its niche: Fitness sharing and implicit sharing compared. In *Proc. Parallel Problem Solving from Nature — PPSN IV*,Voigt H M, Ebeling W, Rechenberg I, Schwefel H P (eds.), Berlin, Springer, 1996, pp.398–407.

[9] Sareni B, Krähenbuhl L. Fitness sharing and niching methods revisited. *IEEE Trans. Evolutionary Computation*, 1998, 2(3): 97–106.

[10] Cioppa A D, Stefano C D, Marcelli A. On the role of population size and niche radius in fitness sharing. *IEEE Trans. Evolutionary Computation*, 2004, 8(6): 580–592.

[11] Mahfoud S W. Niching methods for genetic algorithms [Dissertation]. University of Illinois at Urbana-Champaign, Urbana, IL, USA, IlliGAL Report 95001, May 1995.

[12] Horn J. The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations [Dissertation]. University of Illinois at Urbana Champaign, Urbana, Illinois, 1997.

[13] Watson J P. A performance assessment of modern niching methods for parameter optimization problems. In *Proc. the Genetic and Evolutionary Computation Conference*, Banzhaf W, Daida J, Eiben A E *et al.* (eds.), Orlando, Florida, USA, vol. 1, Morgan Kaufmann, July 13–17, 1999, pp.702–709.

[14] Miller B L, Shaw M J. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In *Proc. International Conference on Evolutionary Computation*, Nayoya University, Japan, 1996, pp.786–791.

[15] Pétrowski A. A clearing procedure as a niching method for genetic algorithms. In *Proc. the 1996 IEEE Int. Conf. Evolutionary Computation*, Nayoya University, Japan, 1996, pp.798–803.

[16] Li Jian-Ping, Balazs M E, Parks G T, Clarkson P J. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 2002, 10(3): 207–234.

[17] Parrott D, Li Xiao-Dong. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Trans. Evolutionary Computation*, 2006, 10(4): 440–458.

[18] Bird S, Li Xiaodong. Adaptively choosing niching parameters in a PSO. In *Proc. the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006*, Keijzer M, Cattolico M, Arnold D *et al.* (eds.), Seattle, Washington, USA, vol 1, ACM Press, July 8–12, 2006, pp.3–10.

[19] Bird S, Li Xiaodong. Enhancing the robustness of a speciation-based PSO. In *Proc. the 2006 IEEE Congress on Evolutionary Computation*, Yen G G, Lucas S M, Fogel G *et al.* (eds.), Vancouver, BC, Canada, IEEE Press, July 16–21, 2006, pp.843–850.

[20] Smith R E, Forrest S, Perelson A S. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1993, 1(2): 127–149.

[21] Forrest S, Smith R E, Javornik B, Perelson A S. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1993, 1(3): 191–211.

[22] Wright S. Isolation by distance. *Genetics*, 1943, 28(2): 114–138.

[23] Mayr E. Populations, Species and Evolution; An Abridgment of Animal Species and Evolution. Harvard University Press, 1970.

[24] De Jong K A. An analysis of the behavior of a class of genetic adaptive systems [Dissertation]. University of Michigan, Ann Arbor, MI, 1975, *Dissertation Abstracts International*, University Microfilms Number 76-9381, 36(10): 5140B.

[25] Mahfoud S W. A comparison of parallel and sequential niching methods. In *Proc. the Sixth International Conference on Genetic Algorithms*, Eshelman L (ed.), San Francisco, CA, Morgan Kaufmann, 1995, pp.136–143.

[26] Mahfoud S W. Population size and genetic drift in fitness sharing. In *Proc. Foundations of Genetic Algorithms 3*, Whitley L D, Vose M D (eds.), San Francisco, Morgan Kaufmann, 1995, pp.185–224.

[27] Baker J E. Reducing bias and inefficiency in the selection algorithm. In *Proc. Genetic Algorithms and Their Applications* (*ICGA'87*), Grefenstette J J (ed.), Hillsdale, New Jersey, Lawrence Erlbaum Associates, 1987, pp.14–21.

[28] Sarma J. An analysis of decentralized and spatially distributed genetic algorithms [Dissertation]. George Mason University, Fairfax VA, USA, 1998.

[29] Horn J, Goldberg D E. Genetic algorithm difficulty and the modality of fitness landscapes. In *Proc. Foundations of Genetic Algorithms 3*, Whitley L D, Vose M D (eds.), Morgan Kaufmann, San Francisco, CA, 1995, pp.243–269.

[30] Beasley D, Bull D R, Martin R R. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1993, 1(2): 101–125.

[31] Dick G. A comparison of localised and global niching methods. In *Proc. The 17th Annual Colloquium of the Spatial Information Research Centre*, Dunedin, New Zealand, 2005, pp.91–101.

[32] Goldberg D E. A note on Boltzmann tournament selection for genetic algorithms and population–oriented simulated annealing. *Complex Systems*, 1990, 4(4): 445–460.

Grant Dick received his B.Sc. degree (Hons) in information science from University of Otago in 2001 and is currently working towards a Ph.D. degree Since 2006, Grant has been working as a lecturer in the Department of Information Science within the University of Otago, Dunedin, New Zealand. His research interests include adaptive business intelligence, computational intelligence methods (particularly evolutionary computation) and evolving systems (particularly the role that population structure plays within models of speciation).

Peter Whigham received his B.Sc. degree (Hons) in computer science from the Australian National University in 1983, and his Ph.D. degree in computer science from the University of New South Wales in 1996. He was a scientist at the Commonwealth Scientific and Industrial Research Organisation (CSIRO) Division of Water and Land from 1986 until 1998, where his research included spatial expert systems and environmental decision support systems. Since 1999 he has been a senior lecturer in the Information Science Department of the University of Otago, New Zealand. He is currently director of the Spatial Information Research Centre and has active research and teaching interests in spatial and network processes, theoretical population genetics, evolutionary computation and ecological modelling. He is on the editorial board of Genetic Programming and Evolvable Machines and Ecological Informatics.

Call for Papers for the 9th International Conferencefor Young Computer Scientists (ICYCS2008)

Sponsor: China Computer Federation
Organizer: School of Info. Sci. & Eng., Central South University
In Co-Operation with IEEE Computer Society and National Natural Science Foundation of China
Zhang Jia Jie, China, November 18-21, 2008
http://www.csu.edu.cn/ICYCS2008 or http://www.ccf.org.cn/ICYCS2008

Introduction

The theme of the 9th International Conference for Young Computer Scientists (ICYCS 2008) is Computer and Communications Frontiers (CCF). The combination of computing technologies and communications services will benefit modern people with more convenient and better lives. Following the tradition and success of previous ICYCS conferences, ICYCS 2008 will provide an international forum for scientists and engineers in academia and industry to exchange and discuss their experiences, new ideas, research results, and applications on all aspects of Computer Science and Technology and its related disciplines. It will feature keynote speeches, technical presentations, panel discussions and workshops.

Scope and Interests

Technical papers on all aspects of Computer Science and Technology and its related disciplines reporting original and unpublished research work are solicited for presentation and publication in ICYCS 2008. Topics of interest include, but are not limited to:

Submission and Publication Information

The conference and workshop proceedings will be published by IEEE Computer Society (Pending, EI source). Manuscripts should be written in English conforming to the IEEE standard conference format ($8.5'' \times 11''$, Two-Column). Manuscripts should be submitted through the conference website http://www.csu.edu.cn/ICYCS2008/. The first author of each submitted paper should be under the age of 45. Each accepted paper is limited to 6 pages (or 8 pages with over length charge). Distinguished papers, after further revisions, will be published in a special issue of *Journal of Computer Science and Technology* (SCI&EI source) and a supplement issue of *Journal of Software* (EI source). The program committee will select one winner for the Best Paper Awards for the main conference and each workshop.

Important Dates

Paper submissions due: May 1, 2008
Notification of acceptances: July 1, 2008
Camera-ready papers due: August 1, 2008

Conference Contact: Guojun Wang, Ming Liu, Zhigang Chen

Tel: 0731-8877711, 8876677, 8830797

Email: csgjwang@mail.csu.edu.cn; x-info@mail.csu.edu.cn; czg@mail.csu.edu.cn