

An Empirical Investigation into Correlation Functions in a Spatially-Dispersed Evolutionary Algorithm

Grant Dick¹

¹Spatial Information Research Centre
University of Otago, Dunedin, New Zealand
Email: gdick@infoscience.otago.ac.nz

Presented at SIRC 2004 - The 16th Annual Colloquium of the Spatial Information Research Centre
University of Otago, Dunedin, New Zealand
November 29th-30th 2004

ABSTRACT

Spatially-structured populations play an important role in controlling selection pressure in evolutionary algorithms. The imposing space on a evolving population has traditionally been biased toward the underlying architecture that the population is executed on. The spatially dispersed evolutionary algorithm (sdEA) is an attempt to model population structures incorporating more probabilistic measures into the construction of demes. One important component of the sdEA in determining demes is the correlation function. This paper introduces three new correlation functions into the realm of the sdEA and compares their resultant behaviours on four differing test cases. Initial results indicate that the design of a correlation function should bias deme construction to small areas in the population surface.

Keywords and phrases: evolutionary algorithms, spatial structures, deme construction, location correlation

1 Introduction

Spatially-structured populations play an important role in the realm of evolutionary computation. Evolutionary algorithms (EAs) incorporating a spatially organised collection of individuals have been used to successfully on many difficult problem domains. Early experiments using space within an EA did so with more than just an impact of selection pressure in mind. Another benefit of spatially-structured populations is that they often provide an easy mechanism for partitioning the individuals for distributed processing. Indeed, it could be argued that it was the distributed processing properties of a space, not the behaviour of selection, that were the motivating factor in the adoption of spatially-structured populations for EAs.

The design of existing spatially-structured populations was driven, in part at least, by the computational platform chosen for their implementation. The Island model, for example, are often referred to as a coarse-grained implementation, which relates to the processing architecture that it is most readily suited to. Likewise, a diffusion model EA is a fine-grained implementation, as it is highly suited to implementation on multi-processor machines where the cost of inter-process communication is low.

While the existing spatial-structures are powerful tools for adjusting the behaviour of an EA, their historical use on parallel machines means that they are rigid in the size and shape of the space and therefore the connectedness of the resultant locations. Because of this, it is unknown if they are in fact optimal methods of controlling selection pressure in an EA. Also, because of their biases towards certain computational architectures, it is usually difficult to incorporate features from one spatial structure into another, limiting the number of interesting hybrids that may result¹.

1.1 The Spatially Dispersed Evolutionary Algorithm (sdEA)

Distributed processing is an important aspect of evolutionary computational methods. However, the role that selection plays in such systems is also vital and the creation of a spatially-structured population that is not biased

¹There has been work in hybrid parallel EAs in the past. The models produced in the previous work are more realistically viewed as two separate spatial models, with the locations of one spatially-structured population in fact containing not individuals, but complete spatially-structured populations themselves.

in any way by an underlying architecture would be a useful tool for studying the properties of space in populations. One such implementation of is the Spatially-Dispersed Evolutionary Algorithm (sdEA) which uses explicit, coordinate-based spatial models to maintain populations (Dick 2003b, Dick 2003a, Dick & Whigham 2002). In the sdEA used in this paper, individuals of the global population are placed at randomly generated x - y coordinates within a unit space. The space is a torus, so the the maximum distance between any two points is $\sqrt{\frac{1}{2}}$. Localised subpopulations, or demes, are created by selecting individuals from locations in space probabilistically. The probability that a location l will be included in a deme originating from a location o is determined by a *correlation function*.

1.1.1 Correlation Functions

The correlation between two locations in space is the probability that both locations will share at least one common deme. The correlation value is determined by determining the distance between the two locations and passing the result into a correlation function. The correlation function can be relatively simple. Indeed, all previous implementations of the sdEA have used a “cutoff” function, which assigns a correlation of 1 to all locations that fall within a given radius, and 0 to all those that fall outside it. This type of correlation function closely resembles the neighbourhood construction techniques of cellular EAs in that a location is either included in or excluded from a deme with 100% certainty.

Given that the sdEA is an attempt to move away from the rigid structures seen in traditional spatially-structured EAs, it makes sense that other correlation functions should be investigated. For example. it should be possible to devise correlation functions that introduce a temporal element into deme construction.

This paper introduces several new correlation functions that aim to eliminate rigid deme construction and introduce a more probabilistic nature to the formation of localised subpopulations.

As stated previously, existing implementations of the sdEA have used a cutoff function, defined as:

$$C(d) = (d < x) ? 1 : 0$$

where x is a value in the range [0..1]. Normally x is set relatively low in order to promote small localised subpopulations forming. This paper includes this function so that proper comparisons with the new correlation functions can be made. For this paper, x is set to 0.1, which results in an average deme size of 9 individuals.

A simple correlation function to implement is:

$$C(d) = 1 - d$$

which simply introduces a linearly decreasing correlation as the distance between two locations increases. This function introduces much larger deme sizes into the sdEA.

The above correlation function applies an even bias on locations based on distance. It may be desirable in some situations to apply more bias to locations with less distance between them. One example correlation function that does this is:

$$C(d) = -\log(d)^2 / -\log(d_{min})^2$$

where d_{min} is a number set very close to zero in order to produce a value in the range [0..1]. This function produces a curve that assigns a high correlation to very close locations, which decreases relatively rapidly as distance increases.

An alternative to the above correlation function is:

$$C(d) = e^{-\frac{1}{2}\left(\frac{d-\mu}{\sigma}\right)^2}$$

which produces a bell-shaped curve of height 1 when $d - \mu = 0$. This function is used twice in this paper, once with $\mu = 0$ and $\sigma = 0.07$ and a second time with $\mu = 0.05$ and $\sigma = 0.04$. While the first configuration assigns the highest correlation to individuals that share the same location, the second configuration introduces a condition whereby the highest correlation between two locations is established when there is a small distance between them. This could potentially help to reduce inbreeding, which may be of some benefit to some problems.

A visual comparison of each of these correlation functions is shown in Figure 1. With the exception of the linear correlation function, each of these functions produces the same deme size on average. The real difference between them is the concentration of the deme, that is how highly clustered it is to the originating location.

In addition to the above correlation functions, this paper includes two other correlation functions that serve as additional control functions. One is simply:

$$C(d) = 1$$

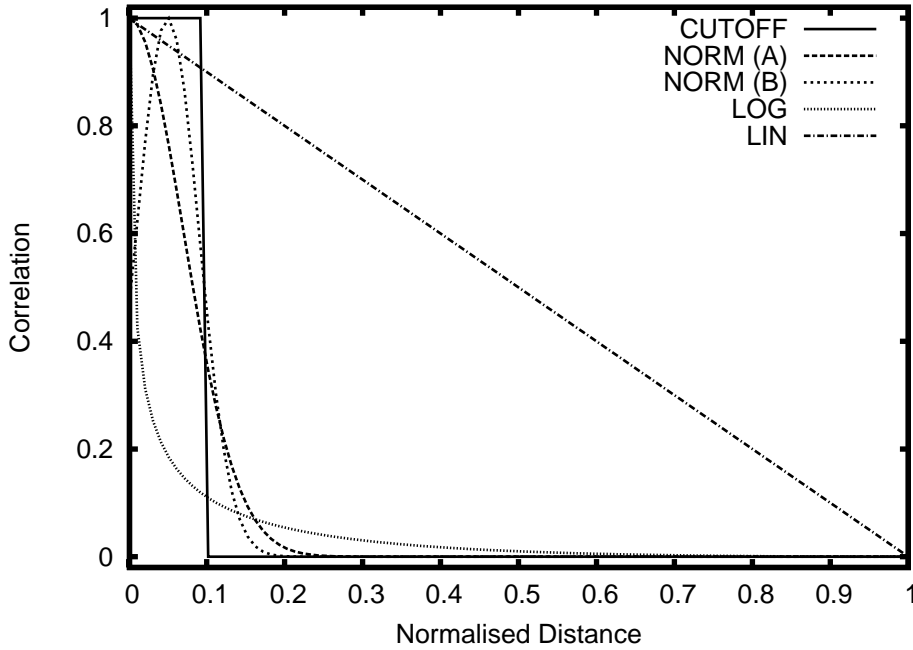


Figure 1: A comparison of the correlation functions used in this paper

which effectively turns the sdEA into a panmitic system. The other function is:

$$C(d) = (rnd() < x) ? 1 : 0$$

which is similar to the above panmitic system except that the average deme size will be equal to Nx , where N is the global population size. For this paper, x is set to produce an average deme size of 9.

2 Experimental Setup and Test Problems

A number of experiments were planned to test the various correlation functions. A population size of 512 individuals was used for each experiment. These individuals were initially placed at randomly generated locations within a unit torus. A steady-state implementation was used, meaning that for every individual that was created, one was destroyed. The procedure for a mating event was as follows:

1. Pick an individual i_1 from a location in space chosen at random
2. Determine the distance between i_1 and every other individual in the space.
3. Pass these distances into the correlation function to determine the correlation between locations in space
4. Use the determined correlation values to construct D , the deme originating from the location containing i_1
5. Select an individual i_2 from D via Roulette-wheel selection.
6. Cross and mutate i_1 and i_2 to produce the offspring individual o^2 .
7. If o is fitter than its parent, insert it into the population in place of i_1 (elitist replacement)³

Each experiment ran for a total of 100000 or 250000 mating events, depending on the difficulty of the problem. At each mating event the change in best fitness, mean fitness and standard deviation were recorded. For every problem in the test suite, a total of 100 runs were performed and the averages of all recorded values were taken.

The sdEA as used in this paper differs slightly from its implementation as used in previous work. The sdEA can, with a given probability, move an offspring from its original location by a small amount. In order to see the effect of correlation functions in isolation, this operation was not performed in this paper.

²In the Plus-One-Recall-Store problem, two offspring were created

³In the One-Max problem, the offspring replaced its parent regardless of fitness

2.1 Test Problems

The test problems selected for this paper are a subset of functions that used previously in studying spatially-structured EAs (Bryden, Ashlock & Corns 2003, Mühlenbein 1991). These problems are known to possess several qualities that make them suitable candidates for inclusion in a test suite (Whitley, Rana, Dzuberka & Mathias 1996).

2.1.1 One-Max Problem

The One-Max problem is a common benchmark for testing evolutionary algorithms. It is simply attempting to individuals represented as n -length bit-strings. The fitness of an individual is given by the function:

$$f(x_i |_{i=1,N}) = \sum_{i=1}^N x_i, x_i \in [1, 0], N = 20$$

in other words, the fitness is determined by the number of activated (1) bits in the string.

The setup for the One-Max problem as used in this paper was as follows: Two-point crossover was used and was applied with 100% probability at each mating event. Mutation was also used with a probability of performing it of 0.01%. A mutation event simply flipped a bit in the string from 1 to 0 and vice-versa.

2.1.2 Griewangk's Function

Griewangk's function is a sum of quadratic bowls with cosine terms added to the, translated to be positive function. It has been examined in (Whitley et al. 1996) and used in (Bryden et al. 2003, Mühlenbein 1991) and is considered a good function for inclusion in any test suite. The cosine term for each parameter creates numerous local optima within the function, although these optima decrease in size as the number of parameters increase, since the scale of the product term in the function decreases with increasing dimensionality. This paper uses this function in three dimensions. The fitness function for this problem is:

$$f(x_i |_{i=1,N}) = 1 + \sum_{i=1}^N \frac{x_i^2}{4000} - \prod_{i=1}^N \left(\cos \left(\frac{x_i^2}{\sqrt{i}} \right) \right), x_i \in [-512, 511], N = 3$$

Individuals in the Griewangk test are represented as bit-strings, with 10 bits per parameter. Therefore, each individual in the population was stored and manipulated as a 30-bit string.

2.1.3 Self-Avoiding Walk Problem

The Self-Avoiding Walk (SAW) problem, like the Griewangk and One Max problems, is a string based test. The goal of the problem is, when given a set of movement operations, to evolve a string that successfully visits every cell of an $n \times x$ grid. The four operations that are available to the system are shown in Table 1.

Name	Description	Symbol
Up	Move up one space. If already at top, ignore move	U
Down	Move down one space. If already at bottom, ignore move	D
Left	Move left one space. If already at furthest left, ignore move	L
Right	Move right one space. If already at furthest right, ignore move	R

Table 1: The operators and terminals available to the PORS problem.

Individuals in the SAW problem are represented by $((n \times x) - 1)$ -length strings. The sequence of operations produced by the string is called a *walk*. Any operation in the walk that would cause the walk to move off the grid is ignored. In the SAW problem, there is no limit to the number of times a cell in the grid may be visited during a walk. The fitness of the walk is the number of cells visited in the grid. The fittest walk then is one that visits every cell just once, which by definition is a self-avoiding walk.

2.1.4 Plus-One-Recall-Store (PORS) Problem

The Plus-One-Recall-Store problem was first described in (Ashlock & Lathrop 1998) and used in the test suite for (Bryden et al. 2003). It is a maximisation problem from the genetic programming domain. Given the operators and terminals in Table 2, the PORS attempts to evolve trees of at most n nodes that produce the highest integer result when executed. An example optimal tree for the 16-node PORS problem is shown in Figure 2.

Name	Description	Type	Symbol
Plus	Adds two subtrees and returns the result	Binary Operator	+
One	The constant value 1	Terminal	1
Recall	Returns the value current stored in memory	Unary Operator	Rcl
Store	Stores the result of its subtree in memory and returns that value	Terminal	Sto

Table 2: The operators and terminals available to the PORS problem.

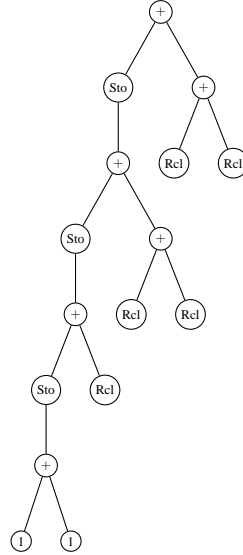


Figure 2: An Optimal Tree in the PORS Problem

3 Experimental Results

Each of the experiments was run 100 times and averages were taken. The statistics of interest were the rate of change of fitness of the best individual in the population and the standard deviation of the mean fitness of the population. The first statistic, best individual fitness, gives some indication into how suited the algorithm is to solving the given problem. The standard deviation of mean fitness is a simple metric for discussing the diversity of the population. If two algorithms produce the same fitness curve but one of them maintains higher standard deviation levels, it is considered the better method.

For each problem, the panmitic, random and cutoff correlation functions were used as baseline statistics from which comparisons with the remaining correlation functions were made.

3.1 OneMax Problem

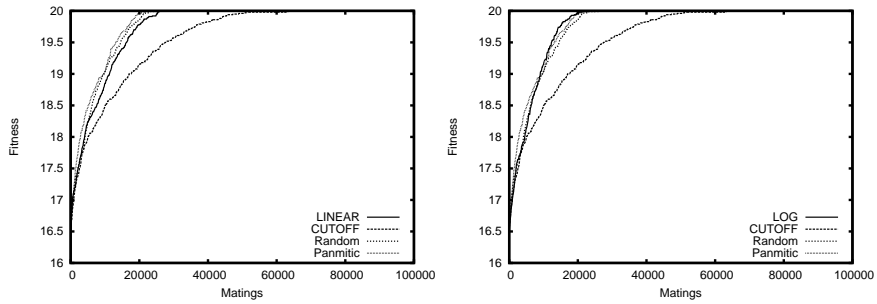
Plots for the rate of change in fitness for the OneMax problem are shown in Figure 3. Plots for the change in standard deviation of the average fitness are shown in Figure 4. The linear and \log^2 correlation functions have an effect on the sdEA that bring its behaviour closer to that of a panmitic system. The bell-shaped correlation functions, conversely, have behaviour closer to that of the cutoff function, with similar fitness curves and standard deviation rates.

3.2 Griewangk's Function

Plots for the rate of change in fitness for the 3-dimension Griewangk function are shown in Figure 5. Plots for the change in standard deviation of the average fitness are shown in Figure 6. As with the OneMax problem, there is little difference here between the \log^2 correlation function and a panmitic system. The linear function behaves closer to the cutoff function. The behaviour of the bell-shaped function on this test case appears to be effected more by parameter configuration for this problem than it was on the OneMax problem.

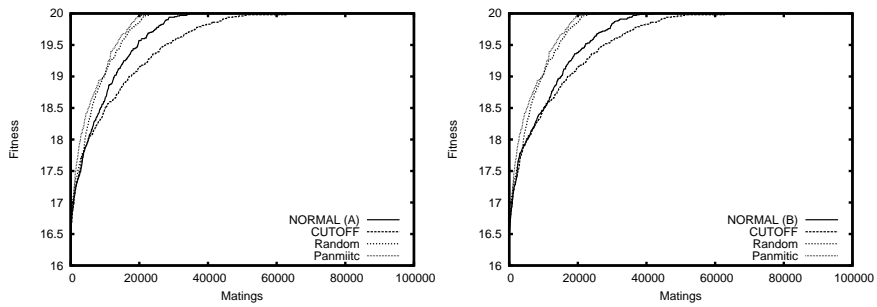
3.3 SAW Problem

Plots for the rate of change in fitness for the 4×4 SAW problem are shown in Figure 7. Plots for the change in standard deviation of the average fitness are shown in Figure 8. The results for this problem closely resemble



(a) Linear Probability

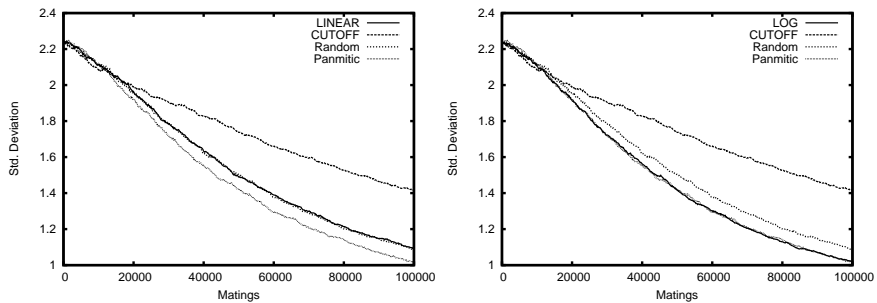
(b) Log² Probability



(c) Normal Probability $\mu = 0, \sigma = 0.07$

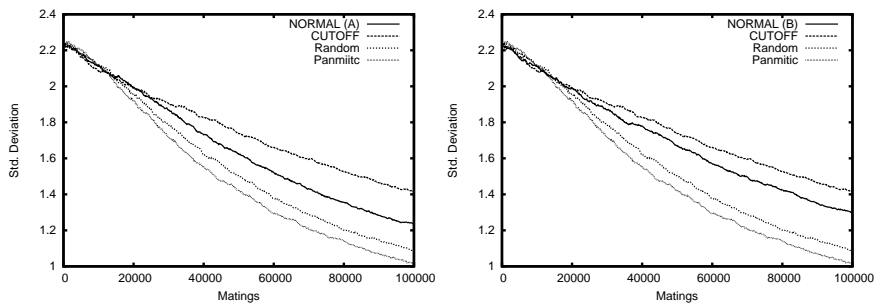
(d) Normal Probability $\mu = 0.05, \sigma = 0.04$

Figure 3: Experimental results for best individual fitness over time in the OneMax problem.



(a) Linear Probability

(b) Log² Probability



(c) Normal Probability $\mu = 0, \sigma = 0.07$

(d) Normal Probability $\mu = 0.05, \sigma = 0.04$

Figure 4: Standard deviation of mean fitness over time in the OneMax problem.

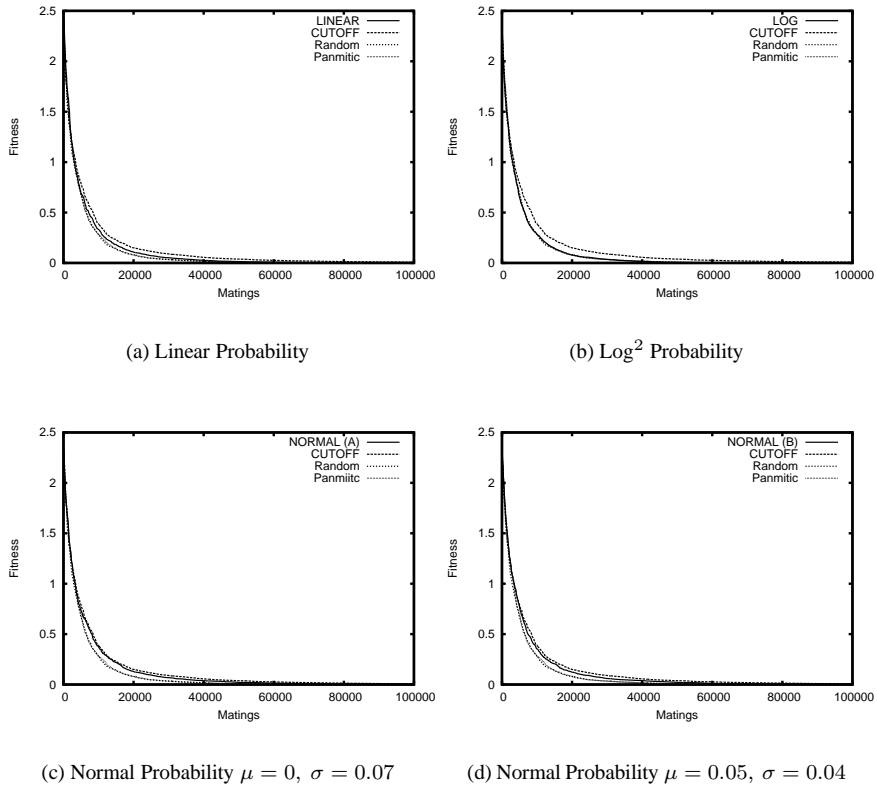


Figure 5: Experimental results for best individual fitness over time in the Griewangk problem.

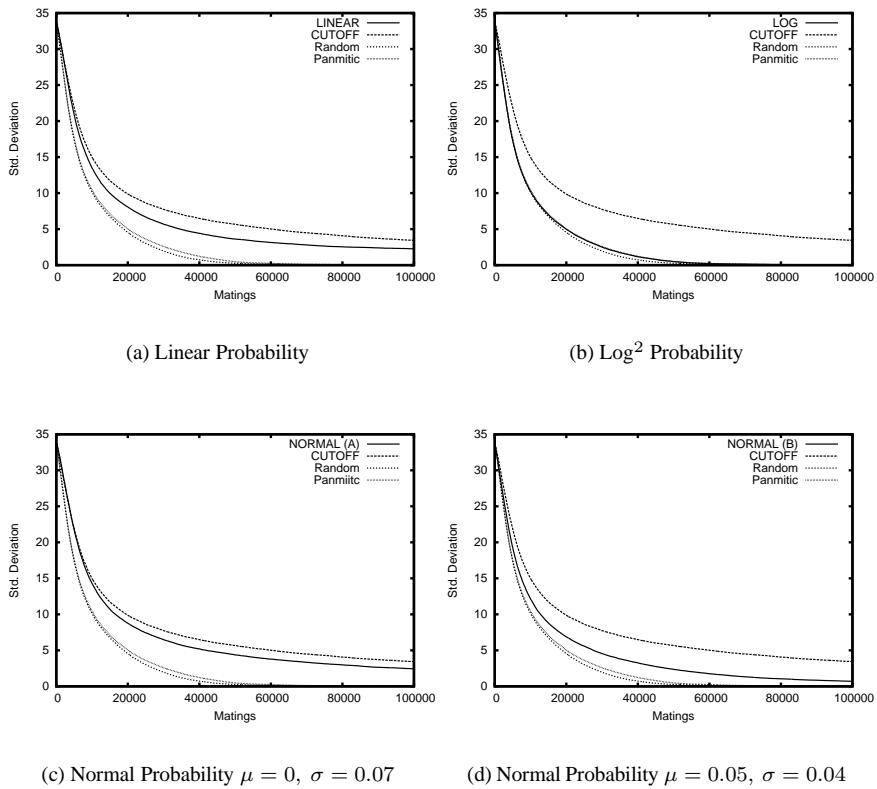


Figure 6: Standard deviation of mean fitness over time in the Griewangk problem.

that of the OneMax problem,

Next to OneMax, this is perhaps the easiest problem in the test suite, at least in its configuration for this paper (Bryden et al. 2003). This may help to explain why the differences in behaviour of the various correlation functions on this problem do not differ as greatly as on the other test cases in this paper.

3.4 PORS Problem

Plots for the rate of change in fitness for the 15-PORS problem are shown in Figure 9. Plots for the change in standard deviation of the average fitness are shown in Figure 10. As with Griewangk's function, the behaviour of the sdEA is markedly affected by the implemented correlation function. Again it is shown that the linear and logarithmic functions emit a behaviour from the sdEA that closely resembles that of a panmitic system, while the bell-shaped curves continue to produce similar results to that of the cutoff function.

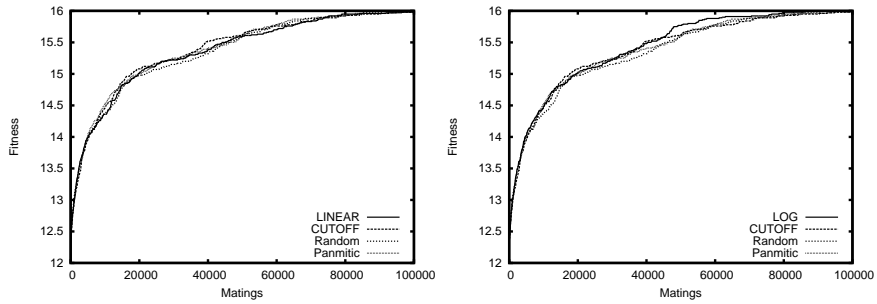
Perhaps the most peculiar observation to be made of this problem is the rapid increase in diversity that is seen. This phenomenon has been observed previously by other researchers working on similar problems (Folino, Pizzuti, Spezzano, Vanneschi & Tomassini 2003). The PORS problem is unique in the suite of test problems here in that it uses a variable length encoding to represent individuals, namely trees of operations. The actual mechanics behind the initial increase in diversity are unknown, although it could possibly be related to crossover of trees. Crossing two trees on average produces larger offspring trees. Since there are more possible permutations of a larger tree, it could be the crossover operator that produces an increase in diversity.

4 Conclusion

Space is an important component in controlling selection pressure in evolutionary algorithms. Most traditional spatially-structured EAs have focused on rigid spatial topologies with which to construct and arrange populations. This paper extends work done on the sdEA, which is an attempt to move away from these rigid spatial constructs into more irregular and temporal structures. In particular, this paper has focused of the behaviour of the correlation function within the sdEA. Three new correlation functions were introduced, each with different characteristics that were intended to alter the grouping of individuals in space into demes for selection, with each having a marked effect on the behaviour of the sdEA.

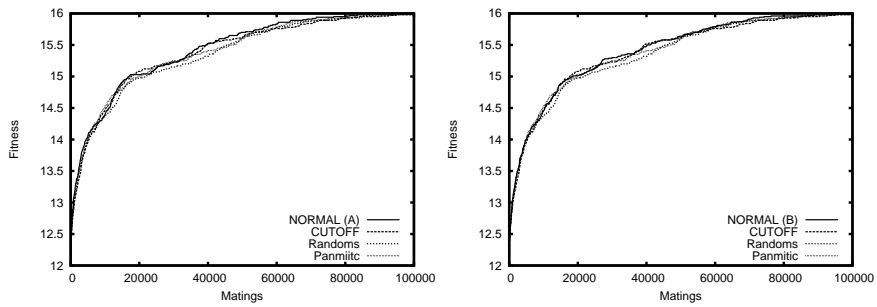
Apart from the linear correlation function, each new correlation function was configured to return on average the same number of individuals for inclusion in a deme. Despite this fact, there were markedly different behaviours in each of the correlation functions. For example, The \log^2 and linear correlation functions both caused the sdEA to exhibit panmitic-like behaviour on all most of the test problems in this paper. Conversely, the bell-shaped functions – Normal (A) and Normal (B) – behaved in a similar fashion to the cutoff correlation function. The marked difference in behaviour between these functions may be explained somewhat by their shape. Figure 1 compares the four correlation functions against the traditional cutoff function. As can be seen, the bell-shaped functions produce a correlation value of effectively zero for any distances greater than 0.25. The linear and \log^2 functions, however, continue to produce higher values of correlation for much greater distances. This means that while the final deme size may be similar, the actual area it was constructed from is significantly larger, indicating that small, strongly localised demes are preferred for diversity maintenance.

This paper is an initial attempt at understanding the role that the correlation function has on the behaviour of the sdEA. Further work, including testing on more problem domains, testing additional correlation functions and different sdEA topologies is needed before more robust conclusions can be made.



(a) Linear Probability

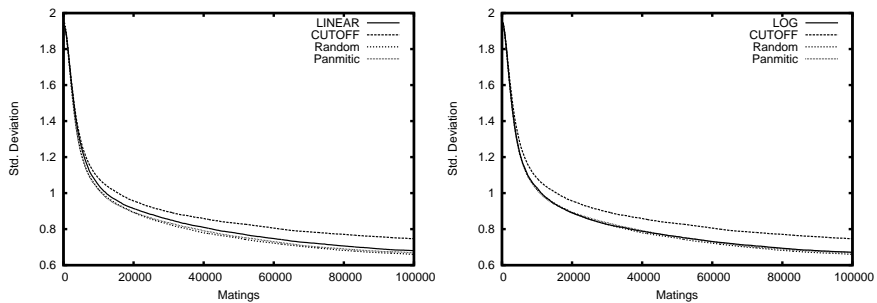
(b) Log² Probability



(c) Normal Probability $\mu = 0, \sigma = 0.07$

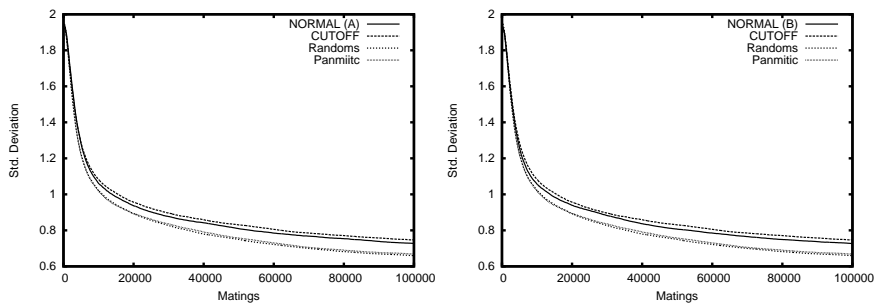
(d) Normal Probability $\mu = 0.05, \sigma = 0.04$

Figure 7: Experimental results for best individual fitness over time in the SAW problem.



(a) Linear Probability

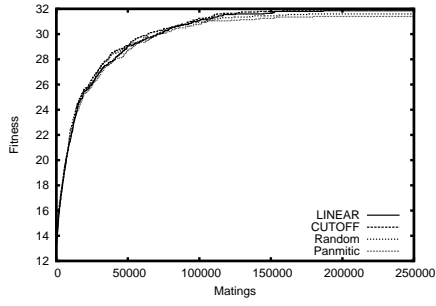
(b) Log² Probability



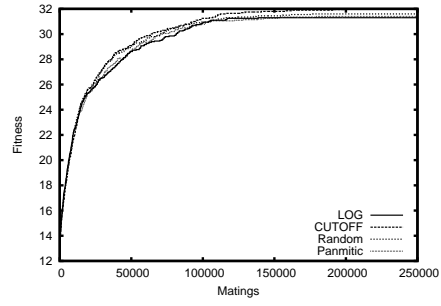
(c) Normal Probability $\mu = 0, \sigma = 0.07$

(d) Normal Probability $\mu = 0.05, \sigma = 0.04$

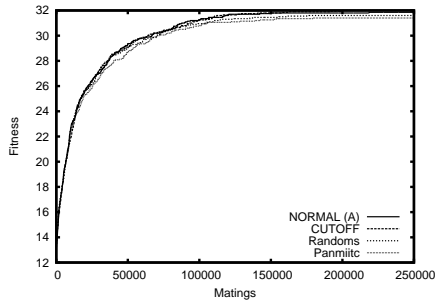
Figure 8: Standard deviation of mean fitness over time in the SAW problem.



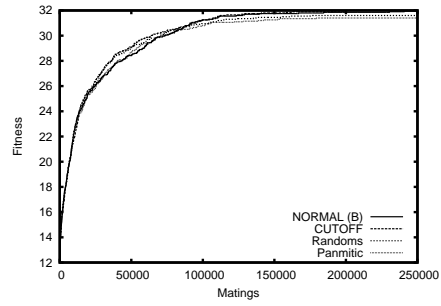
(a) Linear Probability



(b) Log^2 Probability

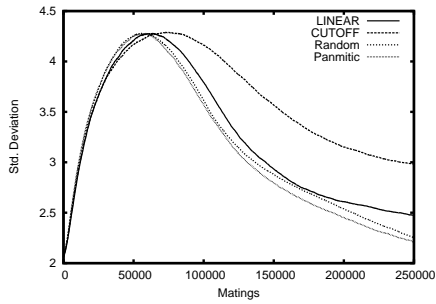


(c) Normal Probability $\mu = 0, \sigma = 0.07$

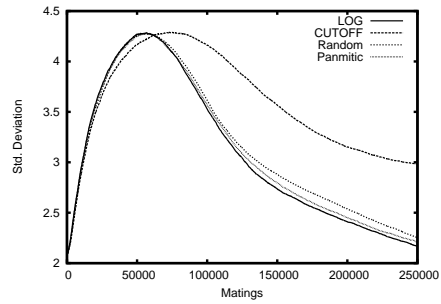


(d) Normal Probability $\mu = 0.05, \sigma = 0.04$

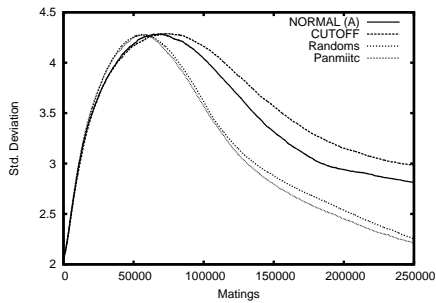
Figure 9: Experimental results for best individual fitness over time in the PORS problem.



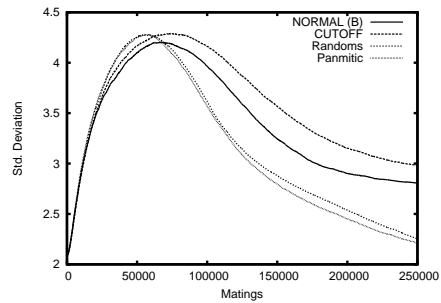
(a) Linear Probability



(b) Log^2 Probability



(c) Normal Probability $\mu = 0, \sigma = 0.07$



(d) Normal Probability $\mu = 0.05, \sigma = 0.04$

Figure 10: Standard deviation of mean fitness over time in the PORS problem.

References

- Ashlock, D. & Lathrop, J. I. (1998). "A Fully Characterized Test Suite for Genetic Programming" In V. W. Porto, N. Saravanan, D. Waagen & A. E. Eiben (eds), *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming*. Vol. 1447 of LNCS Springer-Verlag Mission Valley Marriott, San Diego, California, USA pp. 537–546.
- Bryden, K. M., Ashlock, D. A. & Corns, S. M. (2003). "Graph Based Evolutionary Algorithms" *Technical report*. Department of Mechanical Engineering, Iowa State University. In revision for submission to IEEE Transactions on Evolutionary Computation.
- Dick, G. (2003a). "The Spatially-Dispersed Genetic Algorithm" In E. Cantú-Paz (ed.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 2003*. Springer Verlag.
- Dick, G. (2003b). "The Spatially Dispersed Genetic Algorithm: An Explicit Spatial Model for GAs" In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam & T. Gedeon (eds), *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*. IEEE Press Canberra pp. 2445–2461.
- Dick, G. & Whigham, P. (2002). "Spatially constrained selection in evolutionary computation" *Australia-Japan Joint Workshop on Intelligent and Evolving Systems..*
- Folino, G., Pizzuti, C., Spezzano, G., Vanneschi, L. & Tomassini, M. (2003). "Diversity analysis in cellular and multipopulation genetic programming" In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam & T. Gedeon (eds), *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*. IEEE Press Canberra pp. 305–311.
- Mühlenbein, H. (1991). "Evolution in Time and Space – The Parallel Genetic Algorithm" In G. J. Rawlins (ed.), *Foundations of Genetic Algorithms*. Morgan Kaufmann Publishers. pp. 316–337.
- Whitley, D., Rana, S., Dzubera, J. & Mathias, K. E. (1996). "Evaluating evolutionary algorithms" *Artif. Intell.*, **85**(1-2): 245–276.