The Island Model Genetic Algorithm: On Separability, Population Size and Convergence

Darrell Whitley, Soraya Rana, Robert B. Heckendorn
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
(whitley,rana,heckendo)@cs.colostate.edu

November 4, 1998

Abstract

Parallel Genetic Algorithms have often been reported to yield better performance than Genetic Algorithms which use a single large panmictic population. In the case of the Island Model genetic algorithm, it has been informally argued that having multiple subpopulations helps to preserve genetic diversity, since each island can potentially follow a different search trajectory through the search space. It is also possible that since linearly separable problems are often used to test Genetic Algorithms, that Island Models may simply be particularly well suited to exploiting the separable nature of the test problems. We explore this possibility by using the infinite population models of simple genetic algorithms to study how Island Models can track multiple search trajectories. We also introduce a simple model for better understanding when Island Model genetic algorithms may have an advantage when processing some test problems. We provide empirical results for both linearly separable and nonseparable parameter optimization functions.

1 Introduction

Island Models are a popular and efficient way to implement a genetic algorithm on both serial and parallel machines[1, 15, 24, 8]. In a parallel implementation of an **Island Model** each machine executes a genetic algorithm and maintains its own subpopulation for search. The machines work in consort by periodically exchanging a portion of their populations in a process called **migration**. For example, a total population N_{total} for a serial algorithm could be spread across M machines by giving each machine a population size of $N_{island} = N_{total}/M$. The Island Model introduces the two parameters: **migration interval**, the number of generations (or evaluations) between a migration, and **migration size**, the number of individuals in the population to migrate.

Parallel Island Models have often been reported to display better search performance than serial single population models, both in terms of the quality of the solution found and effort as measured in the total number of evaluations of points sampled in the search space [12, 24]. One reason for the improved search quality is that the various "islands" maintain some degree of independence and thus explore different regions of the search space while at the same time sharing information by means of migration. This can also be seen as a means of sustaining genetic diversity [14]. Some researchers [12, 2] have gone back to the work of Fisher [4] and Wright [25] in biology to try to better understand the role of locality (e.g., maintaining distinct islands or some other form of spatial separation) in evolution.

The partially isolated nature of the island populations suggests that Island Models may be well adapted for use on problems that are loosely composed of many separate problems that can be solved independently. Many test problems that have been used to measure the performance of genetic algorithms turn out to have exactly this property in that they are **linearly separable**. That is, the problem can be decomposed into the sum of a set of subproblems each of which can be solved independently and each of which usually takes a limited subset of the entire set of function arguments.

Island Model genetic algorithms have sometimes done well against single population models on such test problems [14, 12]. We examine whether Island Model genetic algorithms, in fact, do take advantage of the linear separability of a problem to improve performance by solving each subproblem separately in island populations and assembling the partial solutions into a complete solution through the use of migration.

We begin by reviewing the evidence which suggests that having multiple islands can generally improve search performance. In particular, we show that islands can exploit separate and distinct fixed points in the space of possible populations for a single population genetic algorithm. We then propose a simplified probabilistic model of Island Model genetic algorithms to further motivate our explorations. We present the test problems and algorithms we used and discuss our results. These results suggest that our hypothesis of improved performance for the Island Model on linearly separable problems is only partly correct. We discuss possible mechanisms that might explain our observations and explore the pros and cons of the Island Model genetic algorithm.

2 Initial Conditions and Island Model GAs

Vose [16] and Whitley et al. [18] independently introduced exact models of a simple genetic algorithm using infinite population assumptions. Whitley [23] provides one derivation of these models. Vose [13] extends this model to look at finite populations using Markov models.

One of the difficulties of using the infinite population model is that one cannot take into account the sampling bias introduced by using finite populations. The finite population Markov model on the other hand, is too expensive to actually execute except for extremely small problems and extremely small populations (e.g. [10]). One way in which we can introduce finite effects into the infinite population model is by initializing the infinite population model using a distribution taken from a finite population.

In this section, we present a very simple application of the infinite population model to look at the behavior of multiple simple genetic algorithms running in parallel which also exchange information using an idealized form of migration. The multiple simple genetic algorithms have different behaviors due to the different initial conditions that result from initializing the models using different finite population samples.

In particular, we were interested to see if we could find behaviors consistent with the claims that have been made regarding parallel Island Model genetic algorithms: we wanted to see if different islands following different search trajectories could be a significant source of genetic diversity. Mutation is not used in these experiments, since we wanted to better isolate genetic diversity resulting from the interaction between islands.

2.1 The Infinite Population Model

Vose [16] uses the function $\mathcal{G}$ to represent the trajectory of the infinite population simple genetic algorithm [6]. The function $\mathcal{G}$ acts on a vector, p, where component p_i of the vector is the proportion of the population that samples string i. We denote the population at time t by p^t. Note that p^t

may be the sampling distribution of a finite or an infinite population. Using function $\mathcal{G}$ we can generate

$$p^{t+1} = \mathcal{G}(p^t)$$

where p^{t+1} is the next generation of an infinitely large population. The vector p^{t+1} also is the expected sampling distribution of a finite population. Details on the construction of $\mathcal{G}$ are given by Whitley [23]; the models have also been extended to include Hybrid Genetic Algorithms using local search as a supplemental search operator [19].

The function $\mathcal{G}$ exactly corresponds to the behavior of a simple genetic algorithm when the population is infinitely large. In the short term, the trajectory of $\mathcal{G}$ moves to a fixed point in the space of possible populations; in the long term, it may transition to other fixed points due to the effects of mutation. When working with finite populations, larger populations tend to track the behavior of $\mathcal{G}$ with greater fidelity than smaller populations [17]. Thus as the population size approaches infinity the trajectory followed by finite populations converges toward the path followed by the infinite population model, $\mathcal{G}$.

Populations can diverge from the trajectory of $\mathcal{G}$ if the actual trajectory followed by the finite population crosses over into the basin of attraction for a different fixed point. Also, due to the effects of sampling, the *initial population* p^0 of a finite population can sample only a small finite subset of the 2^L strings of length L that make up the search space. Thus, finite populations may actually start in a different basin of attraction than an infinitely large population that uniformly samples all 2^L strings in the space. While the infinite population model does not provide precise information about where a finite population will converge in general, we can study the effects of initializing the infinite population model using a finite population sample and ask how this impacts convergence.

The fact that smaller populations may converge to different fixed points in the space of possible populations suggests that there may be a potential advantage in running several smaller populations for a shorter amount of time instead of running a single larger population for a longer amount of time. The fact that small populations are less likely to follow the trajectory of the infinite population model could be an advantage because the different subpopulations potentially move into "different basins of attraction", and thus, potentially explore different parts of the search space. It should be stressed that these "different basins of attraction" exist with respect to the space of possible populations configurations and these basins exist with respect to $\mathcal{G}$, which is a single population model. An Island Model induces a different space of possible population configurations and of course can only be in one configuration at a time.

While the function $\mathcal{G}$ assumes an infinitely large population, we can use $\mathcal{G}$ to examine the impact of starting the single population genetic algorithm or the Island Models from different initial population distributions based on actual finite population samples.

We can use $\mathcal{G}$ in a simple way to model an idealized Island Model genetic algorithm. The Island Model assumes that the total population used by the genetic algorithm is broken down into a number of subpopulations referred to as islands. Each island is in effect executing a version of the genetic algorithm on each subpopulation. In the model used here, each island is executing a separate copy of $\mathcal{G}$ for the standard simple genetic algorithms using 1-point crossover. *Migration* occurs between subpopulations in a restrictive fashion based on temporal and spatial considerations. The crossover rate is 0.6.

The Island Model used here is based on a popular scheme in which migration occurs every X generations and copies of the individuals that make up the most fit 5% of the island population are allowed to migrate [14, 7]. This top 5% may be copies of the same string, or it may be some combination of strings. The island receiving these strings deletes the least fit 5% of its

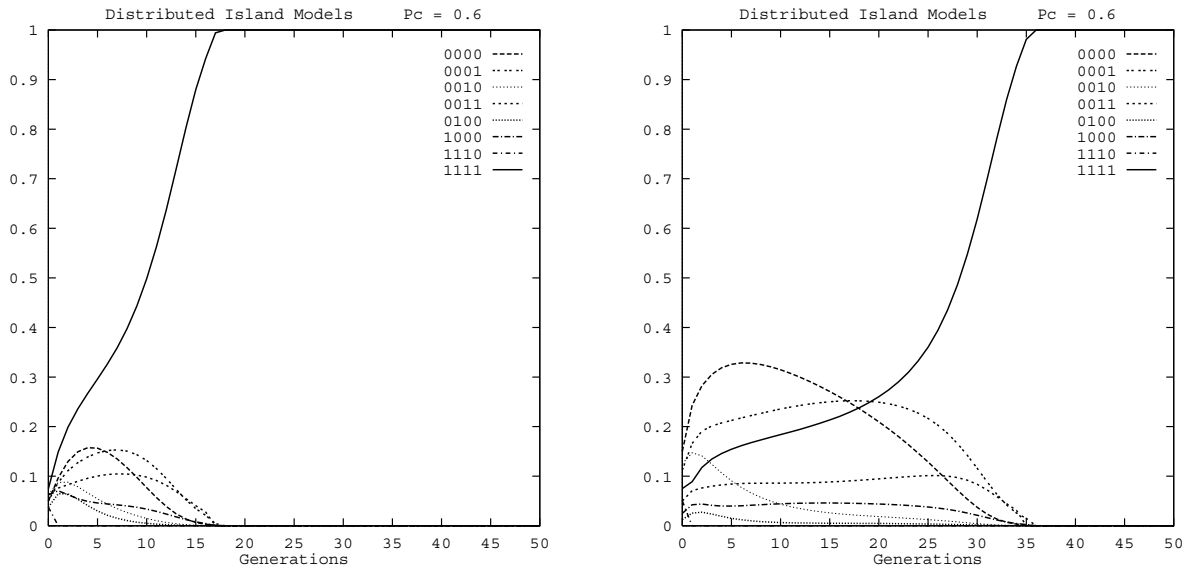

Figure 1: The Proportional Representation of Strings in 2 out of 4 Islands When Migration Occurs **Every** Generation.

own population. The migration scheme assumes that the islands are arranged in a ring. On the first migration, strings move from their current island to their immediate neighbor to the left. Migrations occur between all islands simultaneously. On the second migration, the islands send copies of the top 5% of their current population to the island which is two moves to the left in the ring. In general, the migration destination address is incremented by 1 and moves around the ring. Migrations occur every X generations until each island has sent one set of strings to every other island (not including itself); then the process is repeated.

2.2 Modeling Island Trajectories

Four subpopulations are modeled. The distribution of strings in the infinite subpopulations are initialized by generating distinct *finite* subpopulations. The initial distribution for each island was based on a sample of 80 random strings ($80 = 2^4 * 5$). However, infinite population sizes are still assumed after the initial generation. Thus, the trajectories only model the effects of having different starting points in the different subpopulations.

Results for a 4 bit fully deceptive problem [22] are shown in Figs. 1 and 2 with a global optimum of `1111` and a local optimum of `0000`. The exact function is also given in section 3.2 of this paper. The choice of *fully* deceptive functions is in no way critical to illustrate the computational behavior of the models. It is important, however, to use a function with competing solutions that are some distance removed from one another in Hamming Space and that the GA-surface induced by $\mathcal{G}$ for a single population GA have multiple fixed points. On functions with a single optimum, the GA produces predictable behavior: quick monotonic convergence to the global optimum in all subpopulations.

The graphs in Fig. 1 show the computation of two out of four islands when migration occurs every generation. In this case, the islands all show the same convergence behavior, although the islands vary in the low level details of their behavior. The other two islands (not shown) had

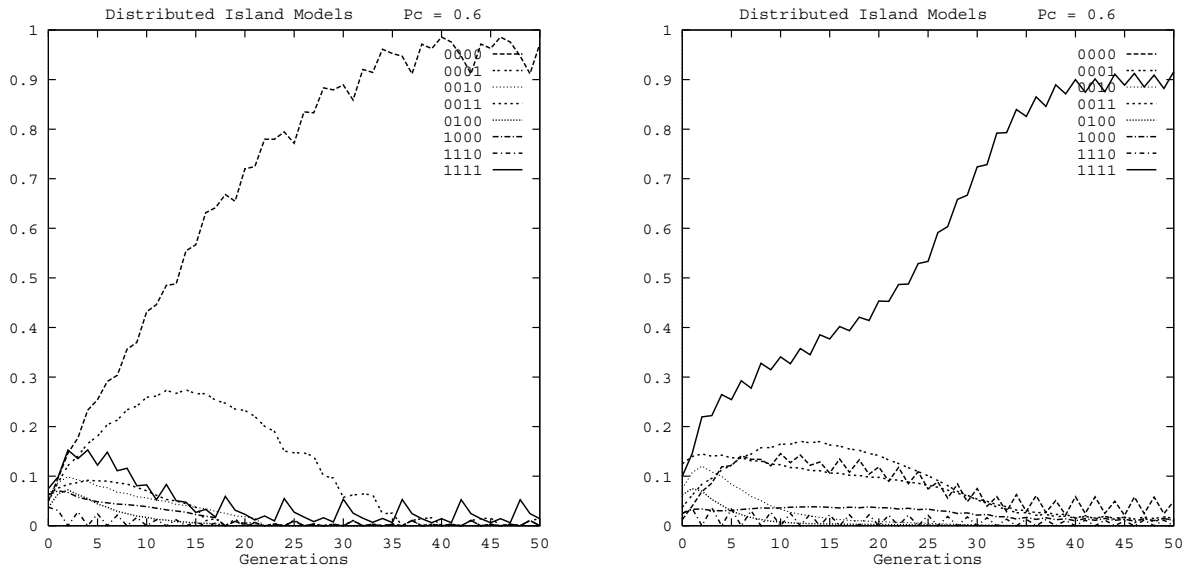

Figure 2: The Proportional Representation of Strings in 2 out of 4 Islands When Migration Occurs **Every Second** Generation.

behavior that was quite similar to one of these two. Thus, two of the islands quickly converge to the global optimum, while two of the islands initially moved toward the local optimum at `0000` and are later drawn toward `1111`, the global optimum.

It should be noted that a single population model started with a uniform representation of strings converges to `0000`. One reason that the Island Models converge to `1111` is that migration of the best strings acts as an additional selection mechanism. The best strings are not only increasing due to fitness proportionate selection; the strings that migrate also tend to double their representation, while the worst strings are deleted.

The graphs in Fig. 2 show two out of four islands when migration is occurring every *second* generation. Migration produces the spike-like jumps in representation, since local representations will rise or fall when migration occurs.

Figures 1 and 2 show that the migration of the best strings acts to increase selection due to fitness. Convergence is faster in the graphs in Fig. 1 where migration occurs every generation. Furthermore, the additional selective pressure produced by migration helps to drive the convergence to the global optimum in Fig. everygenfig. With migration every generation, all subpopulations converge to `1111`. With migration every second generation, only one of the four subpopulations converges to `1111` and convergence is more gradual. (The results for the two islands not shown are similar to the computation represented by the leftmost graph.) In additional runs, not shown here, migrating every 3 generations resulted in all subpopulations converging to the local optimum `0000` (just as in the single population case when there is no additional selection boost due to migration) and again the convergence rate is slower.

The results presented in the figures also suggest that the isolation of subpopulations afforded by the Island Model maintains diversity even with the additional selective pressure of migration. Although the Island Model equations are restricted by the assumption of infinite populations *after the first generation*, the models nevertheless show that having different starting points for the individual islands may be sufficient to cause each island to follow a unique search trajectory. And

although migration will tend to cause the set of equations representing the various islands to become more similar over time, surprisingly, the results using migration every second generation show that it is possible for different islands to converge to different solutions. When these results are extended forward in time, the pattern emerging in the graphs in Figure 2 around generation 35 appears to be stable.

These results also show how convergence to distinct populations that would be near different fixed points in the infinite single population case actually can help to preserve genetic diversity. Normally, without a mutation operator a population would converge to a single string and thus become stuck. However, with the two islands converging to different solutions and with migration to generate low level mixing; new strings are constantly generated at a relatively low rate.

3 Linearly Separable Problems

In the previous section, a general argument was presented to explain why Island Model genetic algorithms can display improved search performance compared to single population models. In this section, we specifically look at the application of Island Models to decomposable separable functions.

3.1 A Probabilistic Argument for Better Performance

A probabilistic argument can be made for why an Island Model genetic algorithm may display more rapid convergence to a given quality of answer for a linearly separable problem. Consider an evaluation function F composed of the sum of S independent nonlinear subproblems G_i:

$$F(V_1, V_2, ...V_S) = G_1(V_1) + G_2(V_2) + ... + G_S(V_S).$$

Assume we apply a genetic algorithm with population size N_x, which solves a specific subproblem G_i from F in t evaluations with probability X_t. Similarly, a genetic algorithm with population N_y solves the same subproblem in t evaluations with probability Y_t. Let $N_y = M * N_x$. In this case N_x represents an island population size for M processors giving a total population size of N_y.

For simplicity, we make several strong assumptions. First, we assume all the subfunctions G_i are identical. We assume that each subproblem G_i is solved *independently* during genetic search, regardless of whether a single population or multiple subpopulations are used. We assume that migration and recombination between subpopulations will build a solution in a post-hoc fashion from the best solution for each subproblem out of the various subpopulations. And finally, we assume the amount of time it takes to assemble to full solution is small in comparison with the time to find the individual solutions. Strictly speaking, this model is not specific to the Island Model genetic algorithm, but rather treats each island as if it were an independent subpopulation. This nevertheless provides insight into why certain problems may be easier to solve using an Island Model genetic algorithm.

A subproblem G_i is **solved** when the correct value is found to optimize G_i, or alternatively, when a solution within ϵ of the optimal solution is found. The exact conditions for a problem being solved in our empirical tests are specified for each problem in our test suite.

Given these assumptions we may reason as follows. Consider an Island Model genetic algorithm with M islands. Let $X_{t/M}$ be the probability of any one island solving a specific G_i in t/M evaluations and Y_t be the probability of the total population, in a *single* population model, solving the same G_i. Note that the probabilities $X_{t/M}$ and Y_t are the same for all G_i under our assumptions. It is clear that the M Island Model and the single population model will use t total evaluations but

the probability of solution may be different. It is this difference that may justify our expectation of a performance improvement.

Since we have M subpopulations, we have M chances of solving the subproblem G_i. Under our model, G_i is solved if it is solved in any subpopulation. The probability that G_i is **not** solved by **any** island population is given by

$$(1 - X_{t/M})^M$$

therefore the probability that it is solved is

$$1 - (1 - X_{t/M})^M$$

Since there are S subproblems with identical probabilities to solve and we assumed they are solved independently, the probability of solving all S is

$$(1 - (1 - X_{t/M})^M)^S$$

In the case of a single population, any one of the subproblems will be solved with a probability Y_t, hence it solves all S independent problems with probability $(Y_t)^S$.

Under what conditions would we expect the probability to be greater by using islands than using a single population? The reasoning above gives us an upper bound on Y_t for this condition:

$$1 - (1 - X_{t/M})^M > Y_t$$

From fixed values of $X_{t/M}$ and Y_t an estimate of the number of islands necessary for equivalent performance can be made as follows. If we assume

$$1 - (1 - X_{t/M})^{M'} = Y_t$$

we can solve for M'

$$\frac{\ln(1 - Y_t)}{\ln(1 - X_{t/M})} = M'$$

which is the number of islands, M', of population N_{total}/M running for t/M evaluations each that would be necessary to have the same likelihood of solving F as a single population of size N_{total} running for t evaluations.

For example, if we know the probabilities $X_{t/M} = 0.05$ and $Y_t = 0.1$ for $M = 10$ and populations of size $N_{island} = 200$, $N_{total} = 2000$ then the model predicts that with $M' = 2$ subpopulations of 200 each running for $t/10$ evaluations will yield approximately the same results as a single population of 2000 running for t evaluations. That is:

$$\begin{array}{ll} (1 - (1 - X_{t/M})^{M'}) & \approx Y_t \\ (1 - (1 - 0.05)^2)^S & \approx .1^S \end{array}$$

In cases where $M' \leq M$ we can conclude that increasing the number of machines from M' to M will only improve the odds of finding the solution. In this case, an Island Model genetic algorithm with 10 subpopulations of 200 each (i.e., a total population of 2000) will probably yield much better performance than just two subpopulations of 200 or a single population of 2000. In general, when the ratio $\ln(1 - Y_t)/\ln(1 - X_{t/M})$ is less than M then the Island Model may display a performance advantage over a single population.

For example, if $X_{t/M} = 0.01$ and $Y_t = 0.1$ when M = 10, the single population will have a slight advantage. (Note that for the range of small probability values we have used in these examples, the ratio between $X_{t/M}$ and Y_t directly tracks the linear scaling of the number of subpopulations.) In reality, the 10 subpopulations will probably be at a real disadvantage because the various distributed subsolutions have to be migrated and assembled, and this process is not trivial.

It should be noted that this model also generalizes to cover nonseparable problems. Assuming that all bits interact with one another, we must know the probability that the entire problem is solved as opposed to subproblems. In other words, the model also covers the special case where we have a single subproblem.

There are obviously also many effects that are ignored by this model. For example, **Hitchhiking** occurs when strings that carry bit values that solve one subproblem, say G_i, carry bit values elsewhere in the string that do not solve other subproblems G_k, $i \neq k$. As a string propagates due to selection, it also propagates other bit values that do not solve G_k. Overall however, the model makes the point that for separable problems if the ability to solve multiple subproblems does not scale with population size, then there may be an advantage to using several small populations rather than one large single population.

3.2 The Test Problems

We conducted our experiments using two linearly separable problems and two nonseparable problems. The two linearly separable problems were chosen as test problems because they represent examples from a well known set of linearly separable test problems. Fortunately they also display different behavior under our tests and lead to a better understanding of what factors contribute to Island Model performance. The two nonseparable problems were chosen to provide a baseline and to illustrate how the added interactions between parameters can affect the performance of the genetic algorithms.

The first separable problem is based on deceptive functions. The following is the **fully deceptive** order-4 function used in the examples presented in Figures 1 and 2.

```
f(1111) = 30        f(0000) = 28
f(0111) = 0         f(1011) = 2
f(1101) = 4         f(1110) = 6
f(1100) = 8         f(1010) = 10
f(1001) = 12        f(0110) = 14
f(0101) = 16        f(0011) = 18
f(1000) = 20        f(0100) = 22
f(0010) = 24        f(0001) = 26
```

In a fully deceptive function, every hyperplane competition is misleading, i.e. deceptive. For example, for a 3 bit function this implies the following hyperplane fitness relationships:

```
f(0**) > f(1**)        f(00*) > f(11*), f(01*), f(10*)
f(*0*) > f(*1*)        f(0*0) > f(1*1), f(0*1), f(1*0)
f(**0) > f(**1)        f(*00) > f(*11), f(*01), f(*10)
```

where f(H) is the average fitness of the strings in the hyperplane denoted by H. The function is fully deceptive because the global optimum is at `111` while all hyperplanes support `000` as a potential solution. Fully deceptive functions are clearly artificial constructs; yet it would also seem unlikely for a function to have no deception. What is important for the current study is that there is strong competition between two complementary points in the space, `000` and `111`, or in the case of our 4 bit fully deceptive function, `0000` and `1111`. As already seen in the experiments using

infinite population models, a genetic algorithm can converge toward 0000 in some circumstances and toward 1111 in other situations. This makes this an interesting test function for Island Models.

Goldberg, Korb and Deb [5] defined a 30 bit function constructed from 10 copies of a fully deceptive 3-bit function. The problem is made more difficult when the bits are arranged so that each of the 3 bits of the subfunctions are uniformly and maximally distributed across the encoding. Thus each 3 bit subfunction, i, has bits located at positions i, $i+10$, and $i+20$.

For our experiments we constructed a 40 bit function composed of 10 copies of our 4-bit fully deceptive function with the 4 bits of the 10 subfunctions distributed at positions i, $i+10$, $i+20$, $i+30$, for subfunctions $i=1$ to 10. We also picked this problem because parallel Island Models have previously been shown to have better results than single population approaches [14] on this problem. Our genetic algorithm is designed to minimize, so the evaluation function is $g(x) = 300 - f(x)$ where $f(x)$ is the deceptive function result for the 40-bit string x.

The second linearly separable test function is Rastrigin's Function.

$$f(x_i \mid_{i=1,N}) = 10N + \sum_{i=1}^{N} \left(x_i^2 - 10\cos(2\pi x_i) \right); \qquad x_i \in [-5.12, 5.11]$$

We used $N = 10$ and 10 bits per argument giving us a 100 bit representation. Each 10 bit argument is Gray coded and the bits are contiguous. The minimum for this function is 0. Furthermore, all bits are treated uniformly and no information about parameter boundaries was used in crossover and mutation.

The two nonseparable functions that we chose to use are Powell's Singular function [11] and Rana's function [20]. Powell's Singular Function is defined as:

$$f(x_1, x_2, x_3, x_4) \;=\; (x_1 + 10x_2)^2 \;+\; ((x_2 - 2x_3)^2)^2 \;+\; (\sqrt{5}(x_3 - x_4))^2 \;+\; (\sqrt{10}(x_1 - x_4)^2)^2$$

Powell's function is normally treated as an unbounded, continuous optimization problem [11]. For our experiments, we use a bit representation with each argument encoded as a 20-bit Gray coded string. When decoded, each input value was scaled to fall in the interval $[-512, 512)$. The minimum for this function is 0.

Rana's Function is:

$$f(x, y) = x\sin(\sqrt{|y+1-x|})\cos(\sqrt{|x+y+1|}) + (y+1)\cos(\sqrt{|y+1-x|})\sin(\sqrt{|x+y+1|})$$

Rana's function was designed to be a multi-modal scalable parameter optimization problem. It was also designed so that it is not symmetric (i.e. $f(x, y) \neq f(y, x)$ when $x \neq y$). The two dimensional primitive function can be scaled to higher dimensions by using the *weighted wrap* method for expanding functions [20]. For example, the expansion of a two dimensional function $f(x, y)$ to a four dimensional function is:

$$g(x_1, x_2, x_3, x_4) = w_1 f(x_1, x_2) + w_2 f(x_2, x_3) + w_3 f(x_3, x_4) + w_4 f(x_4, x_1)$$

The wrap method allows every parameter to occur as input to every argument of the underlying primitive function. This scaling technique is particularly effective at generating difficult functions when the the primitive function is nonsymmetric. Random weights are used for each term in the summation because an unweighted expansion can smooth the surface of the higher dimensional function. Notice that the parameter combinations used in Powell's function are the same as in a four dimensional wrap.

For our experiments, we used a 10-dimensional weighted expansion of Rana's function with random weights ranging from 0 to 1 chosen from a uniform distribution. Each input parameter

is encoded using a 10-bit Gray coded string and is scaled to the range $[-512, 511]$. The global optimum for this function occurs when all parameters are simultaneously set to -512. Additionally, we normalize our output for the scaled function so the optimum value is always $f(-512, -512) \approx -511.70$.

4 Experimental Results

The search engine in our experiments is GENITOR [21], a steady-state genetic algorithm with a biased ($bias = 1.25$) rank based selection. In the first set of experiments, each problem was run with a fixed total population size of 5000 but with varying numbers of islands as shown in Table 1. The **migration interval** in the table is the number of evaluations *per island* between migrations. The parameters in Table 1 were chosen to be the same as those used in a previous study by Starkweather, Whitley, and Mathias [14]. 30 trials were run for each parameter set. The plots represent the overall best evaluation for each of the genetic algorithm configurations averaged over the 30 runs. All of the results are scaled so that the x-axis is the total number of evaluations across all subpopulations. The genetic algorithms were run until solution was found or until a maximum number of function evaluations was reached for that trial. The genetic algorithms were run for a maximum of $200,000$ evaluations on the separable functions and $400,000$ evaluations on the nonseparable functions. The added time was necessary to allow most of the populations to converge to a solution on the nonseparable functions. No mutation was used; the crossover rate was 1.0 since this is a steady-state genetic algorithm.

Island Population	Number of Islands	Number of Emigrants	Migration Interval
50	100	2	250
100	50	2	500
500	10	5	2500
1000	5	5	5000
5000	1	N/A	N/A

Table 1: Experimental Population and Migration Sizes for a total population size 5000.

The results in Figs. 3 and 4 show that increased parallelism helps on the deceptive problem, but not for Rastrigin's function. Note that when run without migration, the islands are really just independent runs of Genitor using different population sizes. Tanese referred to these independent runs as the partitioned genetic algorithm [15].

Larger populations do not help for the deceptive problem but do help a great deal for Rastrigin's function. This would also partially explain why the Island Model is effective on the deceptive problem: it benefits from having multiple distinct populations converging to different fixed points rather than one large population converging to a single fixed point. The fixed points for the deceptive problem each contain varying numbers of optimal subsolutions. This behavior is consistent with the insights associated with our abstract model of the Island Model Genetic Algorithm; if increasing the size of the population does not increase the probability of solving a subproblem (i.e., if Y_t is not significantly better than $X_{t/M}$) then the Island Model has an advantage.

We should also point out that these results are peculiar in the following way: *the answers for the various subproblems are likely contained in the initial population of* 5000. On Rastrigin's function using a 10 bit encoding and random initialization, each population member has a 1 chance in 1024 of containing the optimal solution to a given subproblem yielding a 99.2% chance over the whole

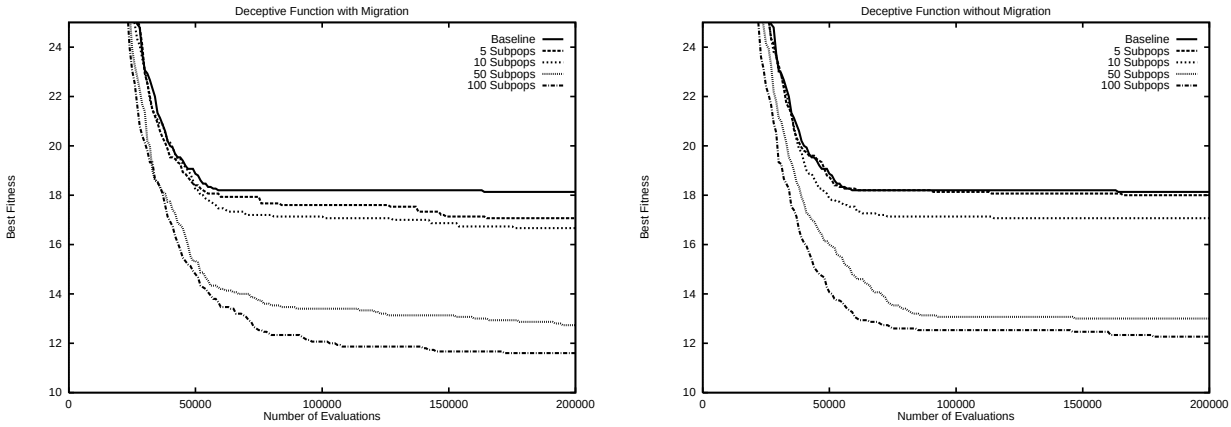

Figure 3: The Deceptive Function with and without Migration.

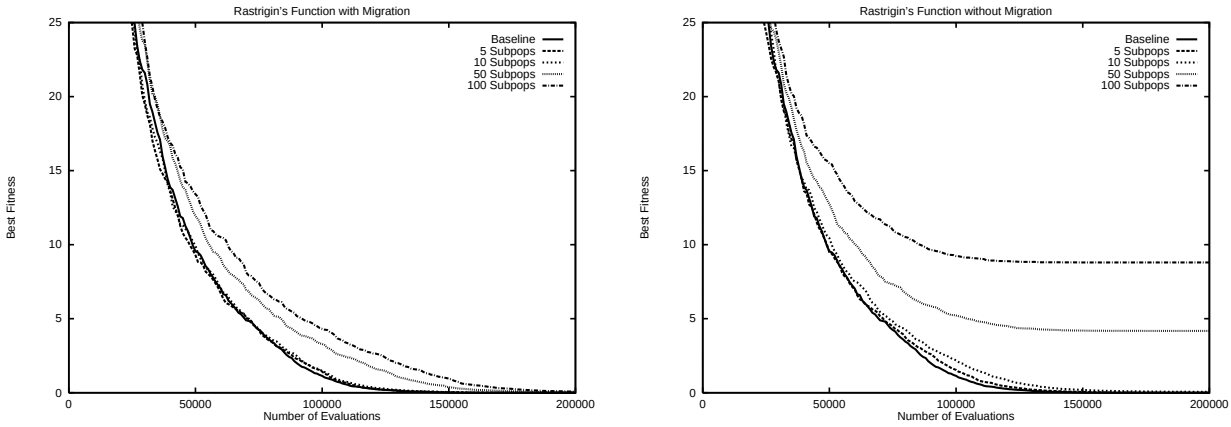

Figure 4: Rastrigin's Function with and without Migration.

population of 5000. This is also a typical situation for deceptive problems; the deception is usually defined over subfunctions constructed using a small number of bits and the subfunctions do not interact. For our deceptive function, there is a 1 in 16 chance that the optimal subsolution occurs in the initial setting for any given subproblem. Hence, the optimal subsolutions to the deceptive subfunctions are almost surely contained in the initial population for all configurations tested. By having many isolated populations that are each able to maintain solutions to the subfunctions, it is increasingly likely that each of the optimal subsolutions will occur in at least one of the populations. This gives the opportunity for these subsolutions to be assembled into the full answer. In this case, the real question is whether or not the optimal subsolutions propagate under crossover.

The last two problems in our test suite are the two nonseparable functions. None of the genetic algorithm configurations tested solved either of the nonseparable problems. The plot for Powell's function is given in Figure 5. Note that the results for Powell's function are given using a log scale. Without migration, the set of isolated small populations are clearly at a disadvantage compared to a single population. When migration is added, the single population is still better in general; however, the configuration using 50 subpopulations of size 100 appears to outperform the single population. The results of running the genetic algorithms on Rana's function are given in Figure 6. Again, without migration, the population size used dramatically affects the performance of the genetic algorithm. In this case, when migration is used, the convergence behavior of the Island

Model genetic algorithm mimics that of the single population.

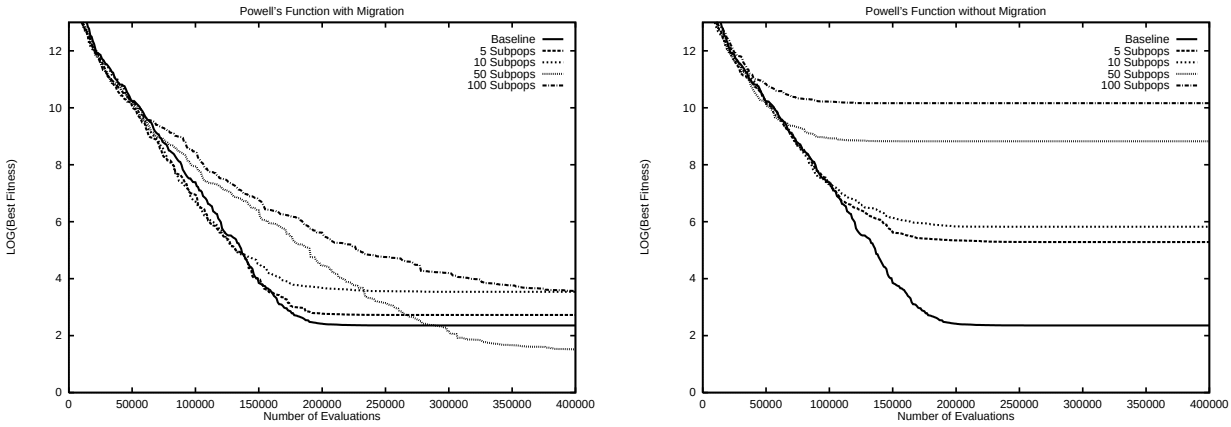

Figure 5: Powell's Function with and without Migration.

Although we cannot solve for each function argument independently for the nonseparable functions, there is still a nonzero probability that the optimal solution for a particular argument will occur in the initial population. For separable problems, we would call an optimal setting for a function argument an optimal subsolution. While we know that these individual arguments can form the optimal solution, the fitness of strings containing subsets of the optimal parameter settings may not be highly fit in nonseparable functions. When optimizing nonseparable functions, the success of the genetic algorithm is dependent on the ability of optimal subsolutions to propagate under both selection and crossover.

Since Powell's function uses a 20-bit encoding for each argument, it is unlikely (.48%) that an optimal parameter setting for a specific subfunction will occur in any of the initial subpopulations. Rana's function, however, uses a 10 bit encoding. There is a 1 in 1024 chance that an optimal parameter setting for any parameter will occur in the initial population. Thus, the initial populations of 5000 will contain optimal parameter settings even for the nonseparable functions. However, the *best* solutions found over the set of runs rarely contained instances of the optimal setting -512. This indicates that Rana's function is somewhat deceptive. The quality of the solutions found was dependent on population size for the partitioned genetic algorithms. The Island Model genetic algorithms performed similarly to a single population since migration was used. Even for nonsepa-

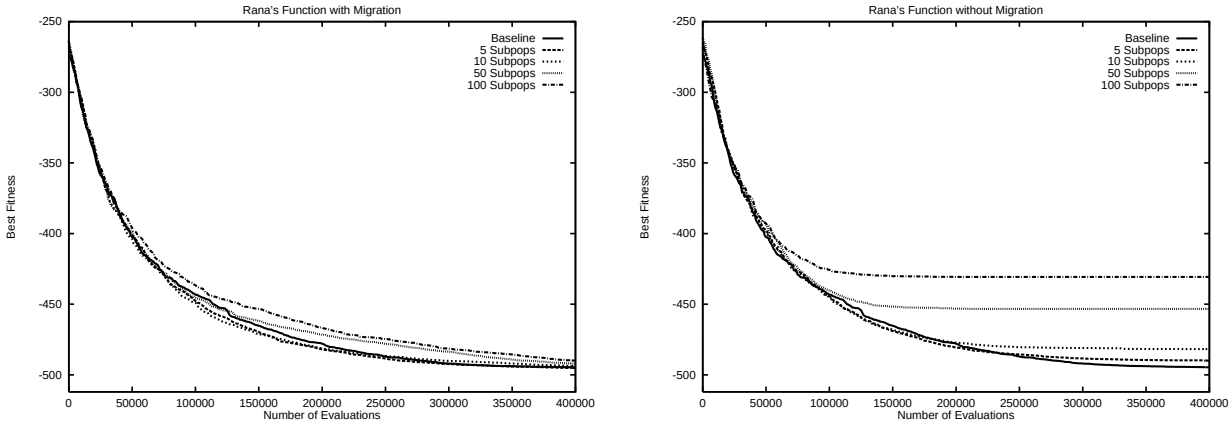

Figure 6: Rana's Function with and without Migration.

Island Population	Number of Islands	Number of Emigrants	Migration Interval
50	10	2	250
100	5	2	500
500	1	N/A	N/A

Table 2: Experimental Population and Migration Sizes for a population size 500.

rable functions, it is still possible for useful genetic information to be transferred across populations in Island Model genetic algorithms.

Population size may also be more of a factor on other test problems than many researchers may realize. Thus, what we are looking at in Figs. 3, 4, 5, and 6 are how readily subsolutions spread in the populations and how much the solutions are disrupted by crossover. Of course, the "ugly" deceptive problem is designed so that crossover is disruptive. However, by looking at the experiments involving Rastrigin's function, it can clearly be seen that the various subsolutions are distributed across the population (or subpopulations) and monotonically increase their distribution in the population over time. Distributing the total population across islands in this case only makes it harder for the subsolutions to spread. Although there are dependencies between parameters in Powell's and Rana's function, migration is still effective at transferring useful genetic material across populations and crossover is effective at combining that material to produce better individuals. However, in these experiments the single large population still performs best on the nonseparable problems.

4.1 Experimental Results for Small Populations

In the previous experiment, the population size was so large that the convergence of the genetic algorithm was very slow. In fact, when migration was used, many of the genetic algorithms were still making progress when they reached the cutoff for the maximum number of evaluations. However, the purpose of those experiments was to estimate what the convergence behavior of a genetic algorithm using an infinite population might be with and without migration.

The use of migration categorically improved the performance of the Island Model genetic algorithms in the large population experiments. In some cases, the convergence behavior of the Island Model genetic algorithms were approaching or surpassing that of a single population. But in practice, genetic algorithms are used with relatively small populations and mutation. It is important to determine what effect migration coupled with mutation has on the performance Island Model genetic algorithms using a smaller total population size. In the following experiments we use a smaller total population size of 500.

Table 2 lists the population sizes, number of islands, migration rate and migration interval used for these experiments. The mutation rate for all test problems was set to $\frac{2}{L}$ where L is the length of the string. The cutoff for each test problem is set to $200,000$ evaluations or 400 generations for the separable problems and $400,000$ or 800 generations for the nonseparable problems. The results presented here were generated by averaging the evaluations of the overall best individuals from 30 different runs of each algorithm configuration.

Figures 7, 8 and 9 are the results of running the three genetic algorithm configurations on Rastrigin, Powell, and Rana's functions respectively. The plots on the right correspond to the genetic algorithms run with mutation but without migration.

The results for the deceptive function were not included in the figures because the Island Model genetic algorithms exhibited similar behavior for the small population experiments as they did for

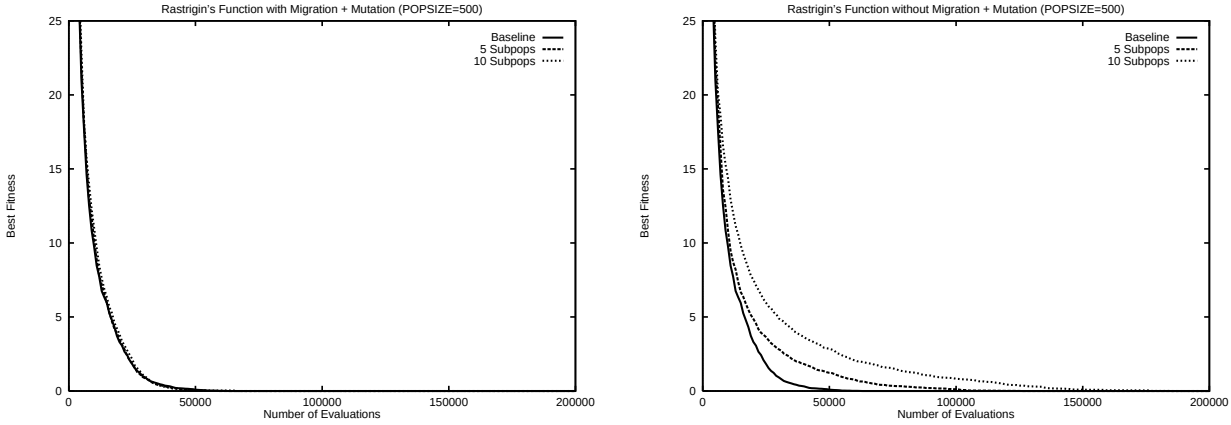

Figure 7: Rastrigin's Function with and without Migration Using a Total Population of 500.

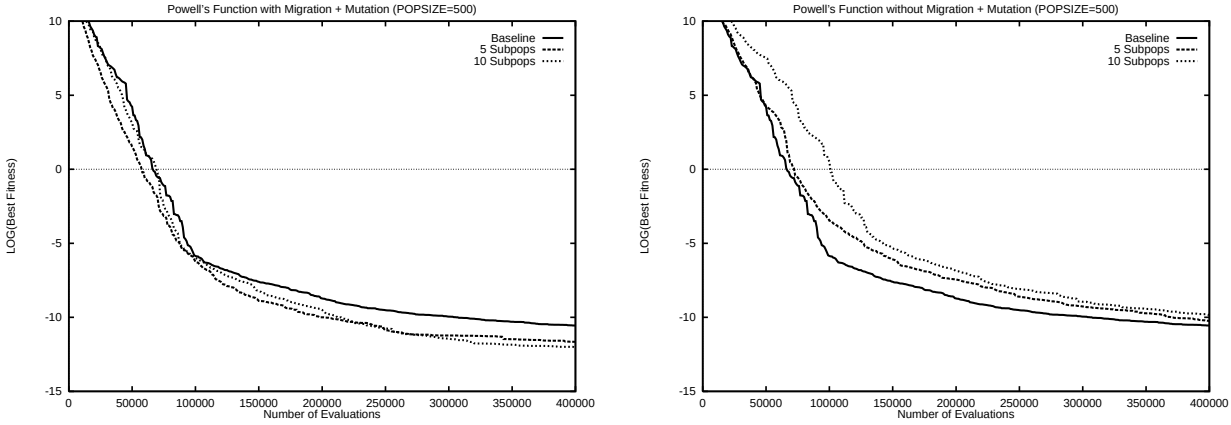

Figure 8: Powell's Function with and without Migration Using a Total Population of 500.

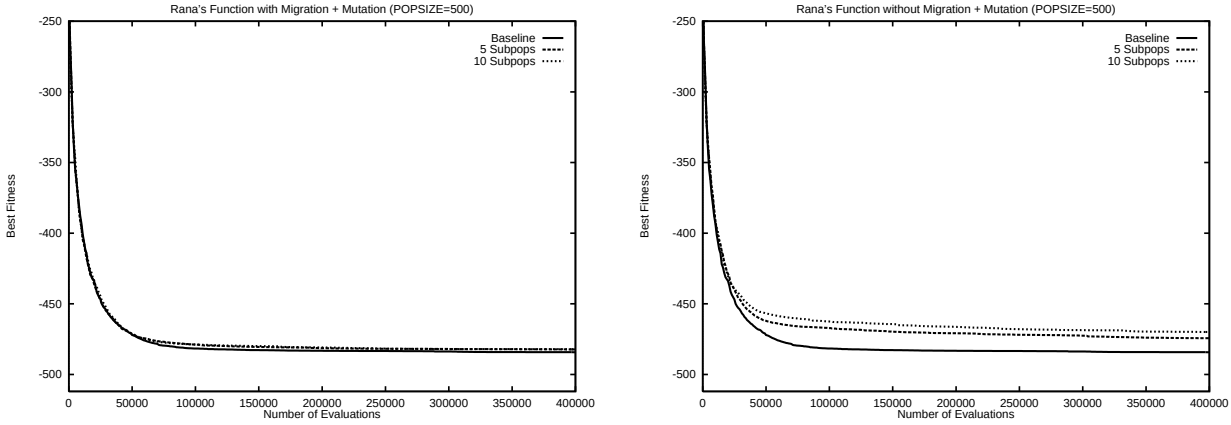

Figure 9: Rana's Function with and without Migration Using a Total Population of 500.

the large population experiments. With or without migration, the Island model genetic algorithms performed better than a single population on the deceptive problem. Furthermore, the performance improves as the number of populations increases.

The smaller population sizes mean that it is less likely that optimal subsolutions for each parameter appear in the initial population as they did for the population size of 5000. For example for problems with 10 bit subproblems and a population size of 100 there is only a 9.3% chance of a specific optimal subsolution appearing the population versus 99.2% for a population of 5000.

The graphs illustrate that the use of mutation had an equalizing effect on the partitioned genetic algorithm (i.e. when migration was not used). While the partitioned genetic algorithms still perform worse than a single large population, the use of mutation improves the performance on the smaller populations.

When migration is used, the performance of all configurations of the genetic algorithm becomes even more similar. We compared the performance of the Island Model genetic algorithms to the single large population using a one-tailed Student's T-Test with $p \leq 0.05$ to determine when the Island Model genetic algorithms outperformed the single population genetic algorithm. When migration is introduced, the performance of all Island Model genetic algorithms improves so that they perform as well or better than the single population. For Rastrigin's function, all three genetic algorithm configurations solve the problem in approximately $50,000$ evaluations. However, the Island Model genetic algorithm using 5 subpopulations of 100 solves the problem with significantly fewer evaluations than the single population when migration is used. For Powell's function, both variants of the Island Model arrive at significantly better solutions than the single population. There is no significant difference between the solutions found by the Island Model genetic algorithms and the single population genetic algorithm for Rana's function.

While the large population experiments serve to illustrate what the convergence behavior of an infinite population might be for the different configurations of Island Model genetic algorithms, they may not be characteristic of what happens when a more traditional set of parameters are used. These experiments illustrate what the performance of a practical genetic algorithm – using a small population and mutation – might be for separable and nonseparable parameter optimization problems. In the context of our abstract model of Island Model genetic algorithms, the use of mutation just alters the probability that subsolutions will be found by some population. When mutation is used, isolated populations tend to improve their performance to approach that of a large population. When migration is used with mutation, the performance of the Island model genetic algorithms is as good or better than the performance of the single population.

5 Discussion

We present an abstract model of when we might expect Island Model genetic algorithms to outperform single population models on linearly separable problems. Our model suggests that Island Models may be at an advantage when increasing the population size does not help to solve the problem. There are many factors that our model does not consider however. Our empirical results suggest that Island Model genetic algorithms can still display very complex and unexpected behavior even when applied to separable problems.

We are well aware of the limitations of using separable problems for testing optimization algorithms and have argued for more diverse types of test problems with different degrees of nonlinearity [20, 9]. Island Model genetic algorithms typically use larger total population sizes; 10 subpopulations of size 500 would not be an unusual or unreasonable design. Yet, for separable functions larger overall population sizes will mean that there is an increased chance of the optimal solution

being in the initial population–which can produce anomalous experimental results. On the other hand, separable functions are a special subclass of problems and studying how they are processed in Island Models may still prove useful despite anomalies. In particular, separable functions are nice for studying migration because the subproblems are independent.

Our experiments have also included nonseparable functions. Although none of the genetic algorithms solved either of the nonseparable functions, it would appear that the performance of the Island Model genetic algorithm is still consistent with our abstract model at a high level. When the performance of the partitioned genetic algorithm is similar to that of the large population, the use of migration causes the performance to equal or surpass that of the large population. The use of migration transfers information between populations and that information is used effectively to improve the performance of each subpopulation.

References

[1] Theodore C. Belding. The distributed genetic algorithm revisited. In L. Eshelman, editor, *Proc. of the 6th Int'l. Conf. on GAs*, pages 114–121. Morgan Kaufmann, 1995.

[2] R. Collins and D. Jefferson. Selection in Massively Parallel Genetic Algorithms. In L. Booker and R. Belew, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 249–256. Morgan Kaufmann, 1991.

[3] Kalyanmoy Deb and Samir Agrawal. Understanding Interactions among Genetic Algorithm Parameters. In W. Banzhaf and C. Reeves, editors, *Foundations of Genetic Algorithms - 5*, Morgan Kaufmann, 1999.

[4] R. Fisher. *The Genetical Theory of Natural Selection.* Dover, New York, 1958.

[5] D. Goldberg, B. Korb, and K. Deb. Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems*, 4:415–444, 1989.

[6] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[7] V. Scott Gordon and D. Whitley. Serial and Parallel Genetic Algorithms as Function Optimizers. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 177–183. Morgan Kaufmann, 1993.

[8] Martina Gorges-Schleuter. Explicit Parallelism of Genetic Algorithms through Population Structures. In H.P. Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 150–159. Springer/Verlag, 1991.

[9] R. B. Heckendorn and D. Whitley. A Walsh Analysis of NK-landscapes. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 41–48. Morgan Kaufmann, 1997.

[10] Ken De Jong, William Spears, and Diana Gordon. Using Markov Chains to Analyze GAFOs. In D. Whitley and M. Vose, editors, *FOGA - 3*. Morgan Kaufmann, 1995.

[11] Jorge J. More, Burton S. Garbow, and Kenneth E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, March 1981.

[12] H. Mühlenbein. Evolution in Time and Space: The Parallel Genetic Algorithm. In G. Rawlins, editor, *FOGA -1*, pages 316–337. Morgan Kaufmann, 1991.

[13] A. Nix and M. Vose. Modeling Genetic Algorithms with Markov Chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.

[14] Timothy Starkweather, L. Darrell Whitley, and Keith E. Mathias. Optimization Using Distributed Genetic Algorithms. In H.P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, pages 176–185. Springer/Verlag, 1990.

[15] Reiko Tanese. Distributed Genetic Algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 434–439. Morgan Kaufmann, 1989.

[16] M. Vose and G. Liepins. Punctuated Equilibria in Genetic Search. *Complex Systems*, 5:31.44, 1991.

[17] M. Vose. Modeling Simple Genetic Algorithms. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms - 2*, pages 63–73. Morgan Kaufmann, 1993.

[18] D. Whitley, R. Das, and C. Crabb. Tracking Primary Hyperplane Competitors During Genetic Search. *Annals of Mathematics and Artificial Intelligence*, 6:367–388, 1992.

[19] Darrell Whitley. Modeling Hybrid Genetic Algorithms. In G. Winter, J. Periaux, M. Galan, and P. Cuestra, editors, *Genetic Algorithms in Engineering and Computer Science*, pages 191–201. Wiley, New York, 1995.

[20] Darrell Whitley, Keith Mathias, Soraya Rana, and John Dzubera. Evaluating Evolutionary Algorithms. *Artificial Intelligence Journal*, 85, August 1996.

[21] L. Darrell Whitley. The GENITOR Algorithm and Selective Pressure: Why Rank Based Allocation of Reproductive Trials is Best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121. Morgan Kaufmann, 1989.

[22] L. Darrell Whitley. Fundamental Principles of Deception in Genetic Search. In G. Rawlins, editor, *FOGA -1*, pages 221–241. Morgan Kaufmann, 1991.

[23] L. Darrell Whitley. A Genetic Algorithm Tutorial. *Statistics and Computing*, 4:65–85, 1994.

[24] L. Darrell Whitley and Timothy Starkweather. GENITOR II: A Distributed Genetic Algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:189–214, 1990.

[25] Sewell Wright. The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in Evolution. In *Proceedings of the Sixth International Congress on Genetics*, pages 356–366, 1932.