Optimal Population Size in Island Model Genetic Algorithms

Neal Holtschulte

American Proverb: Two's company, three's a crowd. Genetic Algorithm Proverb: One's a hill climber and a thousand's random search. Population size is one of the key parameters affecting the success of genetic algorithms (GAs). Assuming a limited number of fitness evaluations (the most time-intensive factor in virtually all optimization problems), there exists an optimal population size for a genetic algorithm for a given application. Intuitively, a GA with population size one is a hill climber and a GA with maximal population size performs random search. Somewhere in between lies the sweet spot. The Island Model GA divides a single population into semi-isolated subpopulations connected by migration. On the extreme of high migration, the subpopulations function as a single large population. On the extreme of no migration, the subpopulations mights as well be independent runs of smaller population size GAs. Somewhere in between lies the sweet spot. In this paper we propose to explore the dynamics of optimal population size as a function of migration in island model GAs.

This is more of a research proposal than anything else. I am seeking constructive criticism.

Introduction

Genetic Algorithms (GAs) encompass a variety of search heuristics based on biological evolution. The traditional GA applies the operators: selection, crossover, and mutation to a population of candidate solutions in order to evolve better solutions to a given problem. Traditionally, population size is an input parameter to the GA and remains fixed throughout the course of search.

Modifying the population size can have a large impact on the performance of a GA. Understanding how and why pop size has such an impact will lead to improved performance for heuristic search algorithms and a better understanding of how they scale. Such insights may also be broadly applicable to populations of organisms or

collections of intelligent agents such as swarm robots.

For a fixed number of evaluations there exists an optimal population size that is problem dependent. That is, the quality of solutions discovered by a genetic algorithm in a fixed number of fitness evaluations varies based on the population size.

This is not surprising when one considers the extreme cases. On the small population end, a genetic algorithm with population size one is essentially a hill climber. Hill climbers ascend the nearest peak in the fitness landscape reachable by mutation but are then stuck atop the peak which may be a local, but not global, optimum. On the large population end, a genetic algorithm with population size equal to the total number of allowed fitness evaluations merely performs random search since the GA would only be able to initialize the population and determine the members' fitness values within the fitness evaluation limit. No selection, crossover, or mutation would be performed.

In Island Model GAs, individuals are divided up into subpopulations (islands). Individuals are then periodically exchanged between the islands (migration) over the course of the GA run.

The parameters added by the island model include: number of islands, migration size (the number or percentage of individuals to migrate), migration interval (number of generations between migrations), and topology, specifying which islands individuals migrate to. Further implementation decisions such as how migrants are selected and whether or not they replace individuals in the population to which they migrate must also be determined.

Related work

To our knowledge there has been no extensive empirical study of optimal population sizes in genetic algorithms across a wide variety of benchmark problems as a function of migration. However, many others have been interested in varying the population size for genetic algorithms.

Fernandez et. al. empirically studied population size for a fixed computational effort on a set of genetic programming benchmarks [3]. They found that for each benchmark there was a point of diminishing returns, after which, adding individuals to the population had no effect or degraded the search results.

Surprisingly, they found that for island GP models with migration the optimal population size remained constant regardless of the number of islands into which

the population was divided. For example, on one of the benchmarks, for a migration of 10% of individuals between the populations once every ten generations, two populations of 5,000 was more effective than two populations of 10,000. And, one population of 10,000 was more effective than one population of 15,000 (for a limited number of fitness evaluations, of course).

Though the authors varied the migration parameters, there was no follow up experiment to determine the effects of the different migration parameter values on optimal population size. In the extreme case of zero migration it seems unlikely that two populations of 5,000 or to be even more extreme, 5,000 populations of two would be optimal. Out goal is to tease apart the relationship between migration rate and optimal pop size in this paper.

Skolicki and De Jong [8] studied migration size and migration interval on island GA performance. They found that moderately large migration intervals and small migration size were optimal and that these parameter settings maintained high diversity, which was important for finding novel solutions. They fixed the population size throughout their experiments as they were not interested in the interaction of pop size with the other parameters.

A number of papers vary population size in an attempt to improve performance but do not make the population size itself the focus of study such as [4, 5] and [9] for multiobjective problems. [2] compares a handful of on-the-fly population resizing mechanism for best performance.

The authors of [10] are interested in both population size and island models, but their conclusions are general and their results are difficult to compare to the results in this paper due to differing methodologies. For example, they used a binary representation for all of their experiments. The binary string was converted to a float for problems such as F5 Rastrigin's function, which we also study, but chose to use a floating point chromosome for. Also, in some of their other experiments they either removed mutation to eliminate it as a complicating factor or used much larger population sizes and effort limits (ex: pop 5000, effort limit: 300,000).

They acknowledge that using a smaller population with mutation is more standard and follow up by performing such experiments.

Setup

The Problems

Without superstition, 13 problems are evaluated for optimal population sizes including: F1 through F9 (table 1), Royal Road, a simple deceptive problem, a one-dimensional optimization problem, and an NP-complete thread mapping problem.

F1 through F9 are used as benchmark problems in [6]. They are minimization problems exhibiting every combination of the binary properties: (uni/multi)modal, (a)symmetric, and (in)separable (with one repeat to make 9). We explored these problems in 10 dimensions using floating point numbers. More details on these problems can be found in table 1. The fitness landscape for F5 is shown in figure 4 and the landscape for F7 is shown in 6.

The Royal Road is a class of functions introduced in [7] to study the types of fitness landscapes on which GA's perform well. Our royal road is identical to the one described in [7]. With a 64 bit chromosome, 4 tiers, and an optimal fitness of 256.

A simple, piecewise linear, deceptive problem:

$$f(x_i) = \sum_{i=1}^{n} \begin{cases} (\frac{-x_i}{d} + 1)/dim, & \text{if } x_i < d \\ ((x_i - d) \cdot \frac{0.7}{1-d})/dim, & \text{otherwise} \end{cases}$$

where d is the deceptiveness parameter (counterintuitively, smaller d corresponds to greater deceptiveness) and dim is the number of dimensions. Thus the global optimum has fitness of 1.0 and the local optimum has fitness 0.7. The chromosome for this problem consists of an array of 10 floating point numbers.

A one-dimensional optimization problem:

$$f(x) = 2^{-2((x-0.1)/0.9)^2} sin(5\pi x)^6$$

For this problem, 32 bit binary chromosomes are used and are translated to integers using a gray code. Finally, the integers are divided by $2^32 - 1$ to get a float in the range 0 to 1.

The last problem is an NP-complete thread mapping problem in which 64 threads are mapped one-to-one to 64 cores on a multicore processor so as to minimize the communication cost between the cores. The cores lie in a 16x16 grid (without wrapping) and communication cost is the sum of the manhattan distances between communicating threads. Not all communication between threads is created equally. A fixed communication graph describes which threads communicate and the energy cost associated with the communication.

The representation of this problem is an array containing a permutation of the values 0 through 63. Special crossover and mutation operators are used to maintain the invariant that a chromosome is a permutation of 0-63 without duplicates.

Genetic Algorithm Settings

Pseudocode for the genetic algorithm used in all experiments is given in figure 2. Parameter settings used for most benchmark problems are summarized in table 3. Genetic operators are described in more detail and exceptions are also described below.

Tournament selection uses replacement. Meaning that an individual can be selected to be included in the next generation multiple times. Two individuals compete per tournament.

Gaussian mutation consists of replacing a gene with a value randomly selected from a gaussian distribution with mean equal to the gene value and standard deviation equal to 20% of the range (so if a gene has min 0 and max 1 then the standard deviation is 0.2).

The mutation rate was fixed at 0.01 per gene. That is, each gene had a 1% chance of being mutated per generation. More than one mutation could occur in the same chromosome in a single generation though such an event is unlikely.

The crossover rate was fixed at 0.9 per individual, meaning that each individual had a 90% chance of reproducing by crossover with another randomly selected individual per generation. Note that the number of generations varied based on the population size due to the fact that the number of fitness evaluations (effort) was chosen as the limiting factor.

Though these rates are relatively standard, parameter sweeps for a fixed population size of 80 individuals were performed on F7 Schwefel's function to verify that the rates are optimal. The parameter sweeps confirmed this.

Exceptions:

The mutation operator for the one dimensional optimization problem and Royal Road problem (which use binary chromosomes) flips the bit.

Mutation and crossover for the locality optimization problem must preserve the invariant that each integer value 0 through 63 appears exactly once in each chromosome. So inversion is used as the mutation operator and a special order crossover operator designed for traveling salesman-like problems (see [1]) is used for crossover.

F1 Sphere Model unimodal symmetric separable $F(x_i) = \sum_{i=1}^{n} x_i^2$ $-5.12 \leq x_i < 5.12$ $F(x_i)_{min} = F(0, 0, \cdots, 0) = 0$
F2 Ridge's Function unimodal symmetric inseparable $F(x_i) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$ $-64 \leq x_i < 64$ $F(x_i)_{min} = F(0, 0, \cdots, 0) = 0$
F3 Exponential Function unimodal asymmetric separable $F(x_i) = \sum_{i=1}^{n} e^{ix_i} + \alpha$ $\alpha = \sum_{i=1}^{n} e^{-5.12i}$ $-5.12 \leq x_i < 5.12$ $F(x_i)_{min} = F(-5.12, -5.12, \cdots, -5.12) = 0$
F4 Modified Double Sum unimodal asymmetric inseparable $F(x_i) = \sum_{i=1}^{n} (\sum_{j=1}^{i} (x_j - j)^2)$ $-10.24 \leq x_i < 10.24$ $F(x_i)_{min} = F(1, 2, 3, \cdots, n) = 0$
F5 Rastrigin's function multimodal symmetric separable $F(x_i) = 10n + \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i))$ $-5.12 \leq x_i < 5.12$ $F(x_i)_{min} = F(0, 0, \cdots, 0) = 0$
F6 Griewank's function multimodal symmetric inseparable $F(x_i) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \frac{x_i}{\sqrt{i}}$ $-512 \leq x_i < 512$ $F(x_i)_{min} = F(0, 0, \cdots, 0) = 0$
F7 Schwefel's function multimodal asymmetric separable $F(x_i) = \frac{1}{n} \sum_{i=1}^{n} (-x_i sin(\sqrt{\lvert x_i \rvert}) + \alpha$ $\alpha = 418.982887$ $-512 \leq x_i < 512$ $F(x_i)_{min} = F(420.968746, 420.968746, \cdots, 420.968746) = 0$
F8 Rosenbrock's saddle multimodal asymmetric inseparable $F(x_i) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2)$ $-2.048 \leq x_i < 2.048$ $F(x_i)_{min} = F(1, 1, \cdots, 1) = 0$
F9 Whitley's function multimodal asymmetric inseparable $F(x_i) = \sum_{i=1}^{n} \sum_{j=1}^{n} (\frac{100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2}{4000} - cos(100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2) + 1)$ $-10.24 \leq x_i < 10.24$ $F(x_i)_{min} = F(1, 1, \cdots, 1) = 0$

Figure 1

```
#Elitism
best = getBestIndividual(population)

#Cull the expanded population down to pop_size
population = tournamentSelection(population, pop_size)

children = []
for p in population:
  if random < prob_crossover:
    cross p with a random individual in population
    children.append(resulting solutions)

mutants = []
for p in population:
  mutate p
  mutants.append(resulting solution)

for c in children:
  mutate c
  replace c with the mutated result

evaluate_fitness(children)

evaluate_fitness(mutants)

population = population + children + mutants + best
```

Figure 2

Population size:	varied
Crossover Probability:	90%
Mutation Probability:	1%
Crossover Operator:	One point
Mutation Operator:	Gaussian
Selection:	Tournament with replacement

Figure 3

Finding the Sweet Spot

> The vulgar crowd values friends according to their usefulness.
> - Ovid

We attempted to establish the optimal population sizes on the benchmark problems. For each problem we ran the genetic algorithm until 4000 unique fitness evaluations had been performed. By unique, we mean that solution, fitness pairs are cached, and later evaluation of a cached solution did not count towards the effort limit.

Population sizes tested include: 8, 40, 80, 120, 240, and 480 individuals. We ran 100 trials on each of the 13 benchmarks, recording the best final fitness value for each population size and then averaging.

> The intelligence of the creature known as a crowd, is the square root of the number of people in it.
> - Terry Pratchett

Surprisingly we found that for a large number of problems the smallest population size had the best performance. We verified this by running additional experiments with a true hill climber and found that it outperformed genetic algorithms of all population sizes on every problem except for the Royal road, deceptive problem, and F7 Schwefel's function.

The best average fitness values for 100 runs across a range of population sizes are shown for F5 Rastrigin's function (figure 5), F7 Schwefel's function (figure 7), and deceptive (figure 8). The GA with the smallest population size outperformed all others on the F5 function despite the rugged landscape as shown in figure 4. The fitness landscape of F7 is shown in figure 6.

Hill climbers outperforming genetic algorithms on so many difficult problems flies in the face of every intuition. Follow up experiments ferreted out the flaw in the experimental design: the effort limit was too low. Figure 9 shows a single run of a hill climber and multiple GA population sizes on F7 Schwefel's function. For this run, the effort limit was increased from 4000 to 40,000. Figure 9 shows that the hill climber and smaller GA populations rapidly converge and successfully exploit local optimum. The GA with population size 480 takes longer to converge, but when it finally does so, it finds a much higher quality (lower fitness since this is a minimization problem) than any of the other search strategies.

Clearly much larger population sizes need to be used for higher effort limits, at least for F7 Schwefel's function and probably for other benchmark problems as well.

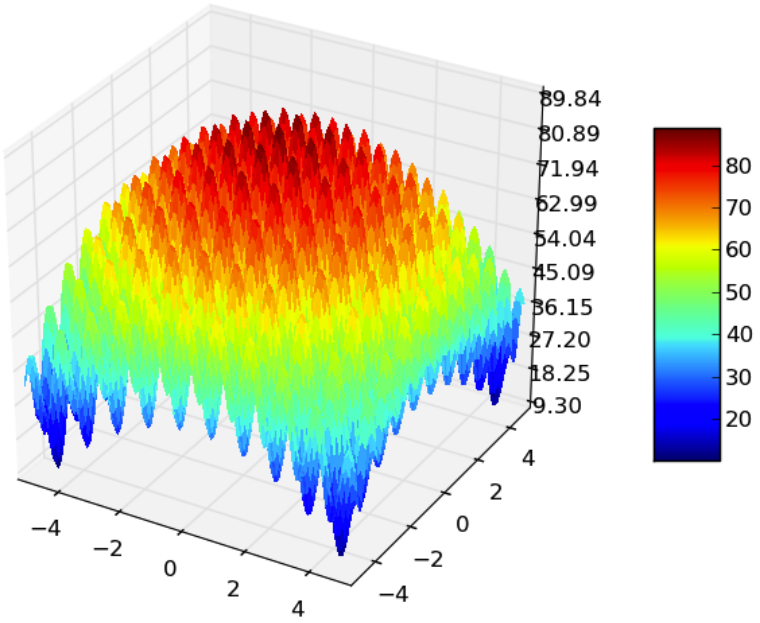

Figure 4: Fitness landscape of F5 Rastrigin's, a minimization problem.

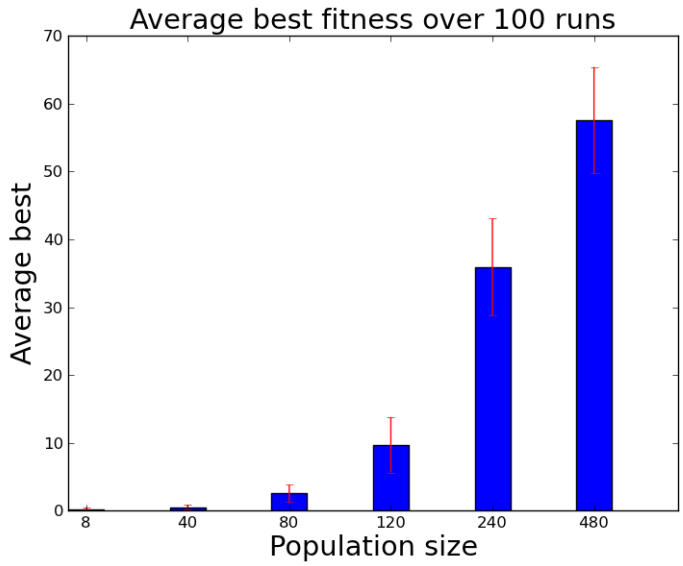

Figure 5: Average best fitness with standard deviation for varying population sizes for F5 Rastrigin's function, a minimization problem.

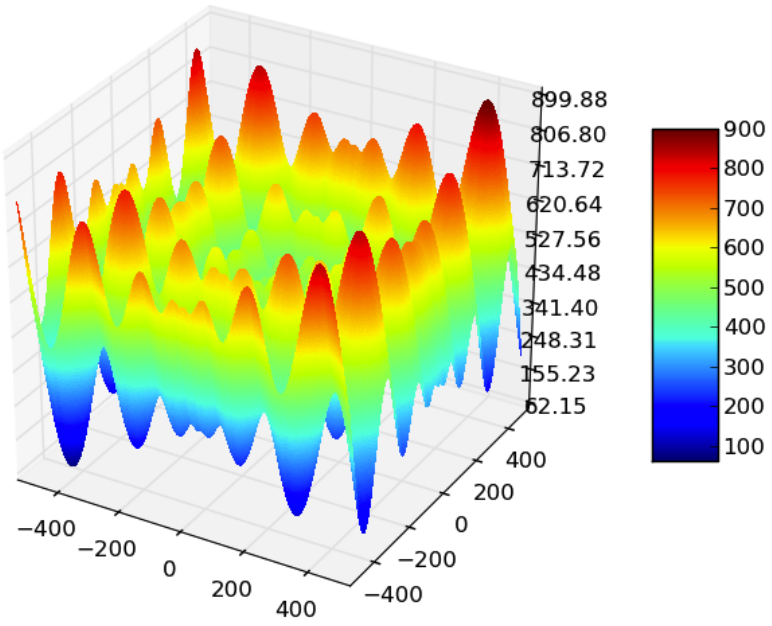

Figure 6: Fitness landscape of F7 Schwefel's, a minimization problem.

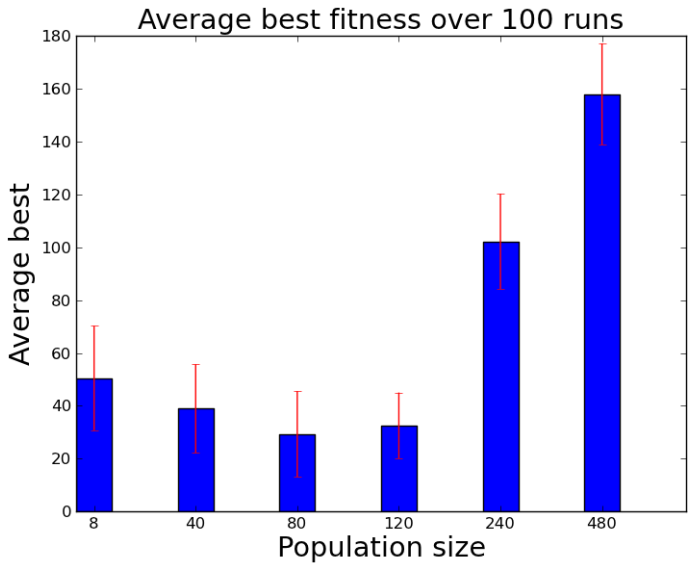

Figure 7: Average best fitness with standard deviation for varying population sizes for F7 Schwefel's function, a minimization problem.

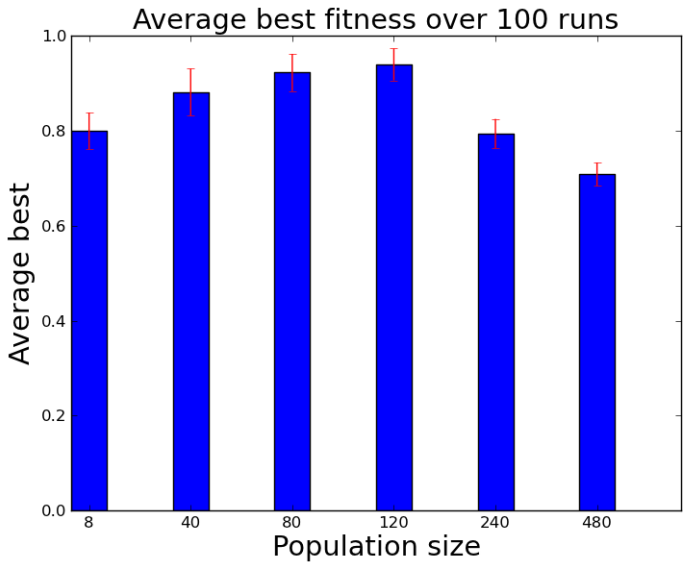

Figure 8: Average best fitness with standard deviation for varying population sizes for the deceptive problem, a maximization problem.

To avoid the problem of a specific effort limit creating a bias towards population sizes that converge just prior to reaching the limit we will instead evaluate the quality of a given population size based on how many fitness evaluations the population takes to reach the optimal fitness value.

Population sizes that get stuck in a local optimum, never to reach optimal fitness will be ended once the population's diversity passes below a certain threshold and after no improvements in best fitness are found after a specified number of fitness evaluations. Both of these cut-off parameters will have to be carefully crafted to prevent them from prematurely terminating actively searching GAs.

Some optimal fitness solutions are virtually impossible to find due to the high precision of floating point numbers. For such problems, the fitness landscape will be adjusted so that a GA gets credit for finding the optimal solution if it finds a solution within some epsilon of the true optimal.

Finally, we will get back to the main question, "What is the relationship between optimal population size and migration in island model genetic algorithms?"

All feedback is welcome.

Special thanks to Ben Edwards, Kenneth Letendre, and Professor Melanie Moses

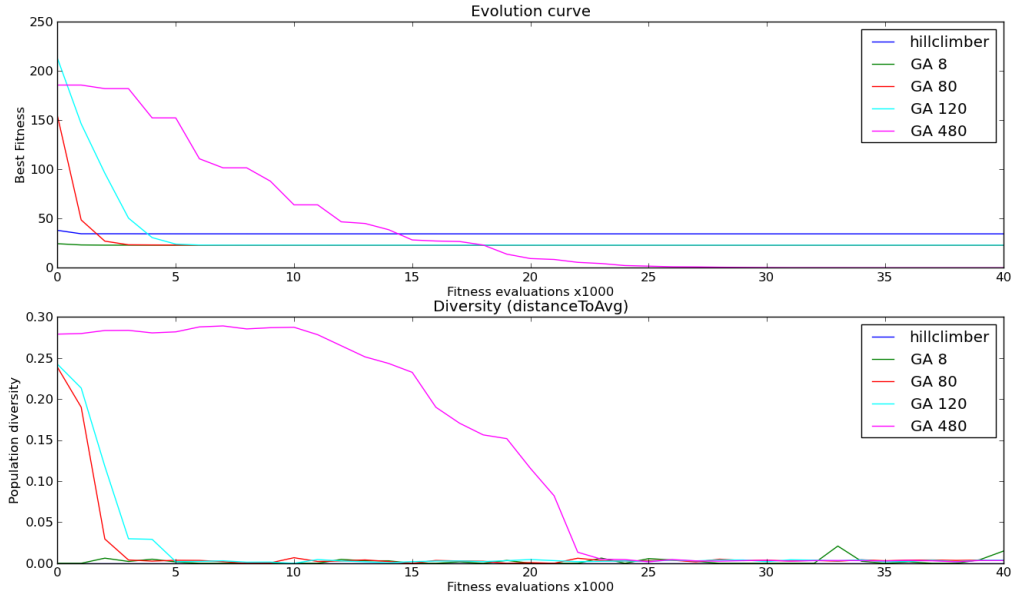

Figure 9: Top: Best fitness after a given number of fitness evaluations. Bottom: Population diversity measured by distance to average point.

for advice on this project.

References

[1] Buthainah Fahran Al-Dulaimi and Hamza A Ali. Enhanced Traveling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA). In *Proceedings of World Academy of Science Engineering and Technology*, volume 28, pages 296–302, 2008.

[2] A. E. Eiben, E. Marchiori, and V. A. Valk. Evolutionary algorithms with on-the-fly population size adjustment. In *Parallel Problem Solving from Nature PPSN VIII, LNCS 3242*, pages 41–50. Springer, 2004.

[3] Francisco Fernández, Marco Tomassini, and Leonardo Vanneschi. An empirical study of multipopulation genetic programming. *Genetic Programming and Evolvable Machines*, 4:21–51, March 2003.

[4] Pedro Antonio Gutiérrez, César Hervás, and Manuel Lozano. Saw-tooth algorithm guided by the variance of best individual distributions for designing evolutionary neural networks. In *Proceedings of the 8th international conference on Intelligent data engineering and automated learning*, IDEAL'07, pages 1131–1140, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] R.L. Haupt. Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors. In *Antennas and Propagation Society International Symposium, 2000. IEEE*, volume 2, pages 1034 –1037 vol.2, 2000.

[6] Takuma Jumonji, Goutam Chakraborty, Hiroshi Mabuchi, and Masafumi Matsuhara. A novel distributed genetic algorithm implementation with variable number of islands. In *IEEE Congress on Evolutionary Computation*, pages 4698–4705, 2007.

[7] Melanie Mitchell, Stephanie Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Proceedings of the First European Conference on Artificial Life*, pages 245–254. MIT Press, 1991.

[8] Zbigniew Skolicki and Kenneth De Jong. The influence of migration sizes and intervals on island models. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1295–1302, New York, NY, USA, 2005. ACM.

[9] K.C. Tan, T.H. Lee, and E.F. Khor. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 5(6):565 –588, dec 2001.

[10] Darrell Whitley, Soraya Rana, and Robert B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47, 1998.