Should Every Man be an Island?

Island number and population size for popular benchmark functions

Neal Holtschulte
Dept. of Computer Science
University of New Mexico
Albuquerque, NM 87131
+1 (505) 277-8432
neal.holts@cs.unm.edu 2nd. author

Melanie Moses
Dept. of Computer Science
University of New Mexico
Albuquerque, NM 87131
+1 (505) 277-9140
melaniem@cs.unm.edu

ABSTRACT

We evaluate 7 optimization algorithms on 21 benchmark functions. A surprisingly robust island-model evolutionary algorithm exhibits excellent performance on all the benchmarks. We then empirically determine optimal **population size** and **island number** in island-model genetic algorithms, where optimality is defined by solution quality under the constraint of limited fitness function calls. We conclude that there are specific population sizes and island numbers that are optimal. These values are constant across benchmark functions. Our results suggest that there are non-intuitive rules of thumb for parameterizing island-model GAs and that a priori optimization of island model parameters is possible on new problems. All code, configuration files, and data are made available online along with detailed descriptions of all benchmarks and experiments.

General Terms

Algorithms

Keywords

Genetic Algorithm, Parallel Genetic Algorithm, Population Size, Fitness Landscape, Benchmark, Island GA

1. INTRODUCTION

Genetic Algorithms (GAs) encompass a variety of search heuristics inspired by biological evolution. GAs apply mutation, crossover, and selection to a population of candidate solutions, also called individuals, in order to evolve better solutions to a given problem.

There are a number of ways to parallelize genetic algorithms. In this paper we consider coarse-grain parallel GAs, called **island-model GAs**, in which the population is divided up into sub-populations known as islands or demes. Individuals from separate islands cannot cross over with each other. However, individuals periodically migrate between islands over the course of GA search. Island-model GAs can be implemented on any hardware. We do not analyze the speed-up gained from parallelizing a GA across multiple processors. See [3, 29] for speed-up analysis.

Island-model GAs are commonly found to be as good as or better than **single-population** GAs [3, 13, 23, 30, 32]. However, single-population GAs have been known to outperform islands under certain circumstances depending on the problem being studied and the parameters used [32].

Traditional GAs have parameters such as crossover rate, mutation rate, and population size. The island model has additional parameters: number of islands, migration size (the number or percentage of individuals to migrate), migration interval (number of generations between migrations), and topology, specifying to which islands individuals migrate. 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.

GAs continue to be popular optimization algorithms for problems with unknown or uncharacterized solution spaces as evidenced by the following sparse sample of GA application papers from 2012 alone: [4, 6, 7, 17, 31].

Single-population GAs are used in all of these papers and little or no justification of the chosen population size is given, with the two exceptions being Gupta [17], which evaluates 4 different population sizes ranging from 5 to 20, and Creaco et al. [7]. Even though careful population sizing and the use of island models have a substantial impact on GA performance, they are rarely used in applied work.

Our empirical study shows an optimal island number and population size that is surprisingly consistent across multiple benchmark functions. We also describe a GA variant we call "every man is an island" which outperforms every optimization algorithm evaluated. These results provide general rules of thumb for researchers who may not be experts in evolutionary algorithms, but who wish to apply GAs.

2. RELATED WORK

Whitley et al. [32] also varies population size and island number in island-model GAs. They focus on whether island models perform particularly well on linearly separable problems. They run experiments with and without mutation using 4 different benchmarks, and 1, 5, 10, 50, and 100 islands. The island model performs as well as or better than the single-population model when run with mutation regardless of the separability of the problem.

GECCO'13, July 6-10, 2013, Amsterdam, The Netherlands.

Fernandez et al. [13], Punch [26], and Goldberg et al. [5, 14, 15, 19] agree that island models can improve performance over single-populations, but emphasize that per-island populations must be large enough so that individual islands can make progress in order for migration to be of benefit. Fernandez et al. reach this conclusion in the context of varying island number, population size, communication topology, number and type of migrating individuals, and migration frequency for a variety of genetic programming (GP) benchmarks. Punch [26] reaches this conclusion while studying the effects of solution number and problem deception on the relative performance of island and single-population models. Goldberg et al. reach this conclusion after presenting an equation for conservatively estimating the population size needed to achieve a solution of a particular quality. They determine that a phase transition exists as a function of population size. Below the critical population size, the GA is dominated by noise and may converge to a local optima, but above the critical size, the GA reliably converges to the global optimum. Goldberg et al. explicitly address the population sizing question in the context of the island model in [14].

Nakano et al. [25] propose a statistical model to gain insight into population size optimality and investigates optimal population size under constant computation cost for a job shop scheduling problem. They verify their model and prove the existence of an optimal population size in the case of the job scheduling problem.

A number of researchers vary population size in an attempt to improve performance but do not make population size itself the focus of study such as [8, 12, 18, 20, 33] and [28] for multiobjective problems. Eiben [10] compares a handful of on-the-fly population resizing mechanisms for best performance. Dynamic population size models are introduced in [1, 11, 18, 28]. Grefenstette [16] uses a meta-GA to search for optimal GA parameters for a number of different problems.

Jumonji et al. [21] introduces the Variable Island GA (VIGA), which dynamically adapts its island number and population size in order to increase its efficiency. The authors evaluate VIGA on nine of the same benchmarks we use and show that VIGA performs as well as or better than both parallel and single-population GAs.

These studies show a continued interest in population size and island number. Taken together, they recommend first determining the smallest population size, n, that can make progress on the problem of interest. Next, create as many islands, m, with n individuals per island as time and computational power allow. Finally, run this setup with no more than 5% of the population migrating, and migration occurring no more than once every 5 generations. How well this rule generalizes, or if there exists an adaptive algorithm that unequivocally beats it remains unknown. We aim to evaluate population size and island number while holding migration size and interval constant in a non-adaptive GA.

3. ALGORITHM COMPARISON

We evaluate 7 optimization algorithms on 21 benchmark functions. We then select 7 benchmarks from the 21 that distinguish well between the algorithms. That is, benchmarks for which algorithm performance varies given the limit of 50,000 fitness function calls. All the algorithms are run with a population size of 64, and with parameters and operators as described in Section 3.3 unless the implementation of the algorithm dictates otherwise.

The set of benchmark functions includes all combinations of multimodal/unimodal, asymmetric/symmetric, and separable/inseparable functions. There is one noisy function and two function generators. The functions are converted to maximization functions and run in 20 dimensions. Each of these benchmarks except the generators are static mathematical functions. That is, the different random seeds for each run only impacts search, not the shape of the fitness landscape.

Detailed information including function name, references, 2d and 3d landscape visualizations, python implementations, and mathematical descriptions of each benchmark function can be found at `cs.unm.edu/~neal.holts/GECCO2013`.

3.1 Implementation Details

3.2 Algorithms

We evaluate 6 GA variants and a hill climber. Parameters and operators of these algorithms are described in Section 3.3 and Table 1. Each algorithm is run until the stopping criterion of 50,000 fitness function calls is met. Brief descriptions follow.

Genetic Algorithm (*GA*) - There are numerous variants of the genetic algorithm. We attempt to make our implementation as generic as possible. Our implementation is based on the GA described in [2]. It is also very similar to the GA described in [23], but we use tournament selection instead of proportional selection, and we use elitism.

A population of individuals is generated randomly. One generation of *GA* runs as follows: Tournament selection with replacement chooses two individuals from the current population. These individuals cross over with probability equal to the crossover rate. The resulting children, or the parents if no cross occurred, are mutated. The two mutants are added to the next generation. This process is repeated until the next generation has been filled up to the population size. Finally, elitism checks to see if the best individual seen so far is in the population, if not, then the worst fitness individual in the population is replaced by the best.

GA without crossover (*gaNoX*) - The same GA with the crossover rate set to zero.

Select Down GA (*select*) - This algorithm differs from *GA* in that parents are selected uniformly at random with replacement instead of via tournament selection. Additionally, the population is **select**ed **down** to size: at the end of each generation, the population of children for the next generation and the current generation's population are merged and tournament selection with replacement is used to select which individuals to pass forward to the next generation.

Steady State GA (*steady*) - This GA is steady state meaning that there are no generations. It differs from *GA* in that tournament selection does not replace the selected individuals in the population, and instead of adding the children of the selected parents into the next generation, the two best individuals out of the two parents and two children are added back into the population so that the population size remains constant.

Every man is an island (*emIsland*) - Each individual in this algorithm is like an island population of one. First, a population of individuals is generated randomly. Each indi-

vidual is mutated and if the mutant is superior, it replaces the original. Each individual crosses over with its neighbor in a one-directional ring topology with 64 nodes and if either child of crossover is superior to the original, the child replaces the original. This algorithm is similar to the PGA described in [23], but with a directed-ring neighborhood arrangement so that each individual has only one neighbor.

Lonely mutant (*mut*) - In this algorithm, a single individual is initialized randomly. Mutation is applied and whenever the mutant is superior to the original, the mutant replaces the original. No crossover occurs. The only algorithms that do not have a population size of 64 are this one and hill climber.

Hill climber (*hill*) - This is a deterministic hill climbing algorithm. An individual is initialized randomly. While the individual is not at a local optimum, the algorithm takes a "step" (increments or decrements one of its genes by the step size). If the resulting individual has better fitness, it replaces the original and the step size doubles. If the result is worse, the step size halves. Once a gene is at a local optimum (holding all other genes constant) then stepping begins again on the next gene. When the individual reaches a local optimum, a new solution is randomly generated and hill climbing begins again. The best prior solution is remembered.

3.3 Parameters

All of our optimization algorithms use the same parameters and operators, unless the nature of the algorithm deems otherwise (for example, *hill* "mutates" its genes by steps, instead of using the bitwise mutation operator). The parameters and operators used for the algorithms are detailed in Table 1 and described below.

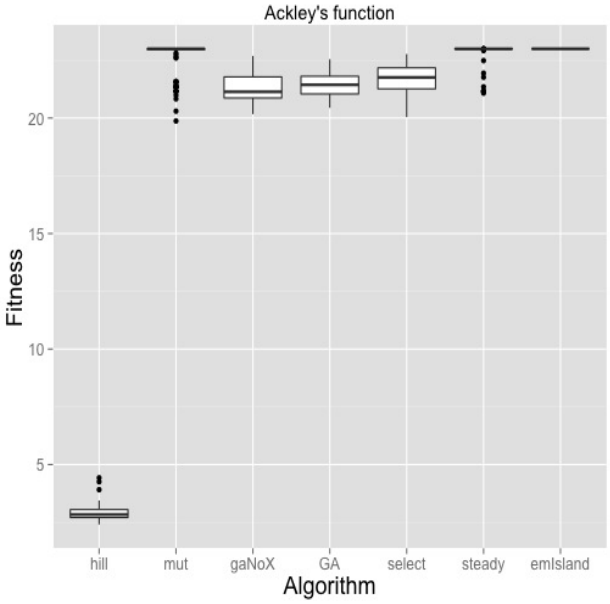
We test different crossover rates, mutation rates, problem dimensionality, and precisions of representation to ensure that the values in Table 1 are reasonable. Population size and island number are investigated in the experiment. All other parameters such as fitness function calls, migration size, migration interval, and elitism size are fixed based on a survey of these variable values in the literature.

We investigate a range of mutation and crossover rates using the generic GA (*GA*). The crossover rates explored ranged from 0.1 to 0.8. We found no significant differences in resulting fitness values, suggesting that almost any nonzero crossover rate is effective. The mutation rates ranged from 0.001 to 0.5. For all of the benchmarks, *GA* found significantly better solutions for mutation rates in the range 0.001 to 0.015. Detailed results are available at `cs.unm.edu/~neal.holts/GECCO2013`.

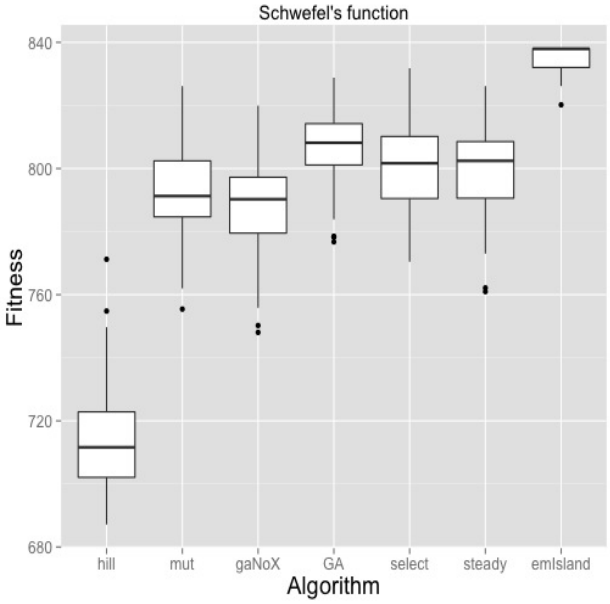
Tournament selection uses replacement, meaning that an individual can be selected to be crossed over multiple times in a single generation. Two individuals are selected uniformly at random to compete in each tournament and the higher fitness individual is returned.

The mutation rate is fixed at 0.01 per gene. That is, each gene has a 1% chance of being mutated per generation. With 320 bits per individual, multiple mutations per individual per generation are likely.

The conclusions of an experiment using optimization algorithms can be highly dependent on what cutoff is used. Kennedy and Spears [22] use a cutoff of 20,000 fitness function calls. The authors of [32] use as many as 400,000 calls in some experiments. We constrain all runs to 50,000 unique



(a)



(b)

Figure 1: The hill climber (*hill*) has the worst performance on all of the benchmarks while *emIsland* stands apart as a superior optimization algorithm. Other more common variants of the genetic algorithm such as *GA* , *gaNoX* , *select* , and *steady* are harder to distinguish. A single mutating individual (*mut*) has middle-of-the-road performance on Schwefel's function, but performs quite well on Ackley's.

Population size:	varied
Crossover rate:	0.7
Bitwise mutation rate:	0.01
Crossover operator:	Two point
Mutation operator:	Bit flip
Selection:	Size 2 Tournament selection w/ replacement
Elitism:	Size 1
Individual:	320 bits
Encoding:	Gray *
Fitness function evaluation limit:	50,000 evaluations
Runs:	100
Dimensionality:	20
Migration interval:	10 generations
Migration amount:	5% of population rounded up
Topology:	One-way ring
Emigrant selection:	Elitism
Replacement policy:	None

Table 1: The double-line in the table above separates general GA parameters above the line from island model-specific parameters below it.

fitness function calls. By unique, we mean that if the fitness of a specific individual is evaluated once, any subsequent evaluations of the same individual (as in a population with low diversity, for example) are not counted towards the 50,000 call limit. Once an algorithm has called the fitness function on 50,000 unique individuals, that algorithm is halted, even if this event occurs in the middle of a generation.

We optimize each benchmark function in 20 floating point dimensions, representing each dimension by 16 bits for a total of 320 bits per individual. Whitley et al. [32] uses 10 dimensions and 10 bits per dimension. However, others claim that for some benchmarks 40 dimensions is too few [9].

Since many of the benchmark functions have different ranges for their inputs (for example, Ridge's function expects values in the range -64 to 64, but the exponential function expects values in the range -5.12 to 5.12), the number of significant digits varies. However, 16 bits is sufficient to map the significant topological features of the fitness landscapes produced by the benchmark problems used.

3.4 Island parameters

Only the generic genetic algorithm (*GA*) is used for the population size and island number experiments. The GA is parameterized as described in Section 3.3.

Fernandez et al. [13] find that migration of 10% of each island's population every 5-10 generations is optimal, but Skolicki [27] finds 10% to be unnecessarily large. Muhlenbein [24] uses a migration interval of 10. In our implementation, the top 5% of the population (rounded up to the next largest whole number) migrates once every 10 generations.

The 5% that migrates is not removed from the source population. The migrants are appended to the destination population rather than replacing any individuals there. This temporarily increases the population size, but this increase is corrected by selecting only *population size* individuals for the next generation. The migration topology used is a ring, with each island passing individuals to one neighbor in a clockwise fashion. Fernandez et al. [13] finds topology to have little effect on island-model genetic programming, but ring and random topologies perform slightly better than mesh topologies.

4. EXPERIMENTAL SETUP

We evaluate all combinations of 11 population sizes and 7 numbers of islands for the 7 selected benchmarks. We evaluate population sizes of: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024. We evaluate 1, 2, 4, 8, 16, 32, and 64 islands. Every combination is evaluated 100 times. Combinations for which there are fewer than one individual per island are ignored.

For the population size and island number experiments, we choose 7 of the 21 benchmark functions to use including: Ackley's function, Langermann's function, Lunacek's function, Schwefel's function, 20 gaussian GRUNGE, 200 gaussian GRUNGE, and Whitley's function. GRUNGE is designed specifically for addressing the issue that many benchmarks are not comparable to real world problems[9]. The remaining benchmarks are chosen based on their ability to distinguish between the 7 optimization algorithms. We perform pairwise T-tests on the fitness distributions of the algorithms and consider two algorithms distinguished if their means differ with 95% confidence.

5. RESULTS

Out of the 7 optimization algorithms evaluated on the 21 benchmark functions the "every man is an island" algorithm (*emIsland*) is statistically undefeated. That is to say, for all the benchmarks, *emIsland* either has the highest mean best fitness or is statistically indistinguishable from the algorithm that does. *EmIsland* is the only algorithm that performs this well.

The next best algorithms are the steady state GA (*steady*) and the "lonely mutant" (*mut*). The common feature of *emIsland* , *steady* , and *mut* is an implicit elitism that ensures that an inferior individual is never promoted over a superior individual. The hill climbing algorithm performs poorly on virtually every benchmark. All other GA variants fall somewhere in between *emIsland* and *hill* . Some of these results can be seen in Figure 1. In Figure 1a *emIsland* , *mut* , and *steady* distinguish themselves from the other GA variants. In Figure 1b, *emIsland* yields superior solutions to all the other algorithms. It is harder to distinguish between the GA variants in this figure. Figure 1 shows representative boxplots for the majority of the benchmark functions, but complete results can be viewed online at `cs.unm.edu/~neal.holts/GECCO2013`.

It is surprising that *emIsland* performs so well on such a wide variety of benchmarks since *emIsland* is effectively an island-model GA in which every island has a population size of 1. In order to investigate this further, we next look at the results of varying population size and island number in the island-model version of the generic GA.

5.1 Population size and number of islands

We create heat maps of all the results with population size on the x-axis, island number on the y-axis, and the mean of the best fitness values from each run plotted as temperature.

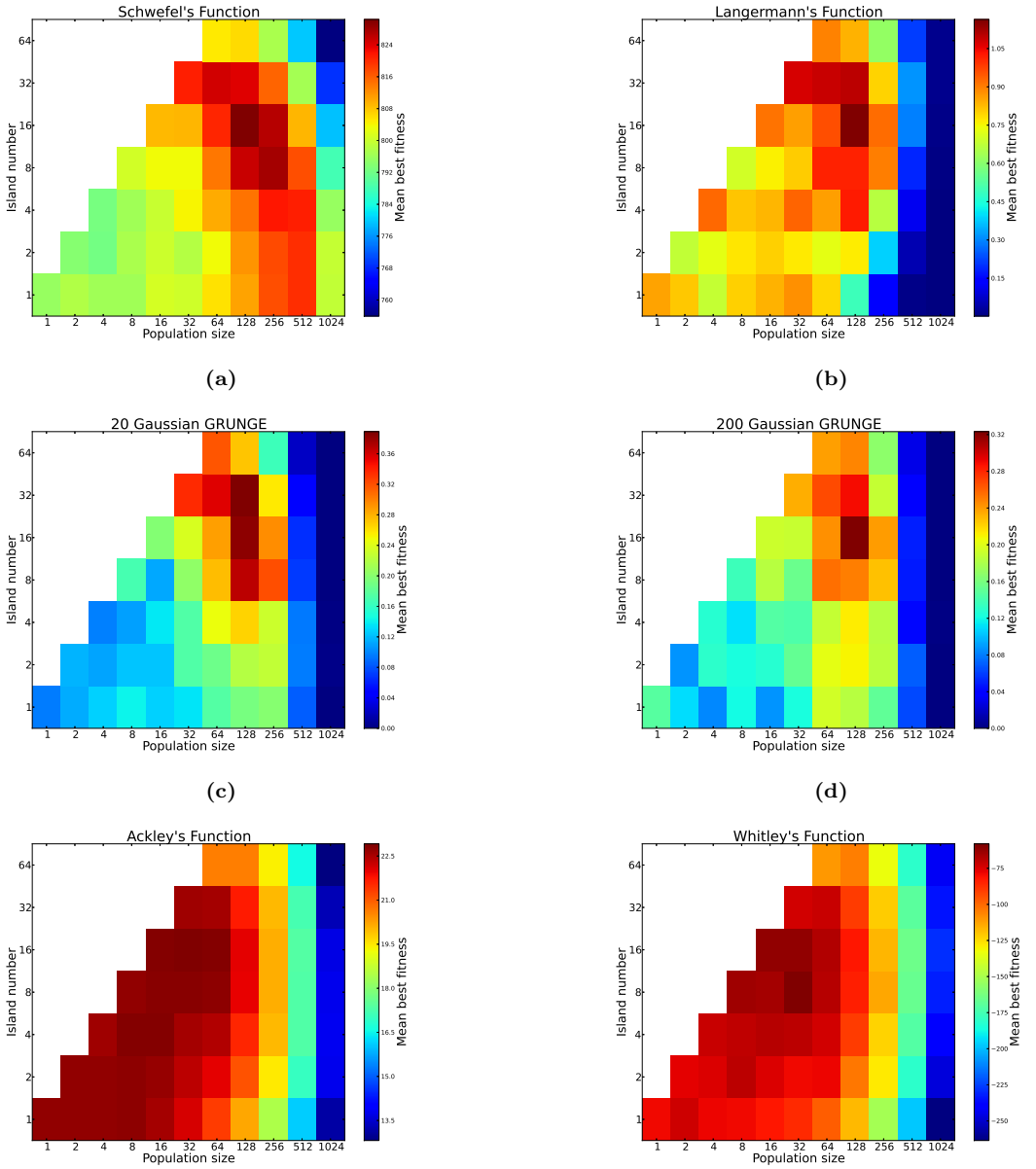


Figure 2: The x-axis is population size. The y-axis is island number. Temperature is the mean best fitness of 100 runs of the GA for the given benchmark. The blank triangle in the upper left corner of each figure is where the population size per island is less than one. No data was collected for these points.
The most exciting feature of these heat maps is the hot spot at population size roughly 128 to 256 and island number 16 to 32 for many of the benchmarks. This is the optimal population size and island number for the particular parameterizations of benchmark and GA. The similarity between the optimal points suggests that they are problem independent and can be generalized for a priori parameter optimization of other problems.
Lunacek's function omitted due to space constraints.

	Schwefel	Whitley	Langermann	Lunacek	Ackley	GRUNGE 20	GRUNGE 200
β_1 (pop^2)	-8.463e-05**	1.077e-04***	7.459e-08	6.875e-06	4.278e-06***	-6.124e-07.	-2.397e-07
β_2 (pop)	6.581e-02***	-2.872e-01***	-1.066e-03	-4.428e-02**	-1.384e-02***	4.053e-04	5.178e-06
β_3 (isn^2)	-1.589e-02**	-2.061e-02*	-7.438e-04**	-6.717e-03	-1.011e-03***	-5.306e-05	-6.162e-05
β_4 (isn)	8.491e-01*	1.321*	4.927e-02**	4.783e-01.	5.571e-02**	4.416e-03	5.418e-03
Adjusted R^2	0.430	0.901	0.309	0.600	0.962	0.106	0.106
Optimum of p	388.810	NA	NA	NA	NA	NA	NA
Optimum of n	26.718	32.048	33.120	NA	27.552	NA	NA

Table 2: $p < 0.001 =$ ***, $p < 0.01 =$ **, $p < 0.05 =$ *, $p < 0.1 =$.
General linear model fit of $fitness = \beta_0 + \beta_1 pop^2 + \beta_2 pop + \beta_3 isn^2 + \beta_4 isn$ where pop =population size and isn =island number.
The optimal number of islands is quite consistent across the benchmark functions, 26 to 33. The coefficient on the squared island number, β_3, is uniformly negative, indicating a peak in fitness at that value. The coefficient on the squared population size, β_1, is negative for some benchmarks, indicating a peak in fitness at the population size inflection point. β_1 is positive and significant for Whitley's and Ackley's functions. For these functions the inflection points are $pop = 1333.3$ and $pop = 1617.6$ respectively. These are fitness minima and are therefore not reported in the table.

The benchmarks fall into two categories. In category A is Schwefel's function, Langermann's function, and 20 and 200 gaussian GRUNGE. These benchmarks exhibit optimal values of total population size at 128 divided up into islands such that each island contains about 8 individuals. These optimal values show up as a dark red spot on the heat maps in Figures 2a, 2b, 2c, and 2d.

Ackley's and Whitley's functions fall into category B, which exhibits a heat "ridge" of optimal values staying close to one individual per island, Figures 2e and 2f. Lunacek's function (not shown due to space contraints) is intermediate between category A and category B, featuring a hot spot with a skew towards fewer individuals per island.

We do not have an explanation for the distinction between category A benchmarks and category B benchmarks, but note that the "lonely mutant" (*mut*) from section 3.2 performs better on the category B benchmarks than it does on the category A benchmarks relative to the other algorithms. For example, compare *mut* 's relative performance in Figure 1a to its relative performance in Figure 1b. This suggests that category B consists of benchmarks for which larger population sizes are merely a waste of computational effort.

Next we fit a general linear model to the results for each benchmark. The model we choose to fit is a quadratic function of two variables: population size and island number. Our goal in fitting this model is to provide further evidence of an optimal population size and island number pair, and to calculate the value of this optimal point by setting the derivative of the model equal to zero with respect to each variable. Coefficients of the fitted models along with significance levels are shown in Table 2 along with R^2 values. The optimal population size and island number are only displayed for models for which the coefficients were significant and negative on the squared terms.

Negative coefficients on the squared terms imply local maxima. Positive coefficients on the squared terms imply local minima. The coefficient on the squared island number, β_3, is uniformly negative, indicating a peak in fitness at the island number inflection point. The optimal number of islands is quite consistent across the benchmark functions, ranging from 26 to 33 islands.

The coefficient on the squared population size, β_1, is not uniformly negative. β_1 is negative and significant for Schwefel's function indicating an optimal population size, but β_1 is positive and significant for Ackley's and Whitley's functions suggesting a fitness minimum at the population size inflection point with fitness increasing as the population size moves away from this value. This is interesting because these are the category B benchmarks that also exhibited different heat map patterns than the other benchmark functions.

The nadir of fitness for Ackley's and Whitley's functions occurs at population sizes of 1617 and 1333 respectively. These values are larger than the maximum population size tested, 1024. Therefore, we hesitate to conclude that the GA would perform better on these functions at much larger population sizes, but the heat maps in Figure 2 suggest that the GA does perform better for much smaller population sizes.

The heat map results and the general linear model results are in agreement. There are two categories of benchmark functions, those for which GAs with a specific intermediate population size and island number are optimal (category A) and those for which GAs with minimal population size and intermediate island number are optimal.

The results from the category A benchmarks beg the question: Why is one individual per island not optimal for the GA on these benchmarks since "every man is an island" (*emIsland*) did so well? One answer is that the GA with islands does not precisely simulate *emIsland* for small population sizes. One difference is the migration interval which is every generation for *emIsland* , but once every ten generations for the island-model GA.

In order to determine which is better, *emIsland* or the island-model GA with optimized population size and island number (*imGA*), we compare best fitness distributions from *emIsland* and *imGA* on the same benchmark functions as we use in the population size and island number experiments. We first optimize *emIsland*'s island number, evaluating island numbers ranging as powers of 2 from 1 to 1024, in order

to make a fair comparison. Since *emIsland* uses one individual per island, island number is synonymous with total population size for this algorithm

Optimized *emIsland* outperforms optimized *imGA* on 5 out of 7 benchmark functions and ties on 2. *EmIsland*'s optimal island number ranges from 64 to 256 on these benchmarks. *EmIsland* has a simple implementation and finds high quality solutions robustly across a great variety of benchmark functions. We recommend that researchers looking for a GA variant to apply to their problem of interest use "every man is an island" with 128 islands.

Conclusion

We evaluate 7 optimization algorithms on 21 benchmark functions with respect to fitness under limited fitness function evaluations. An island-model GA with minimal population size, "every man is an island," reliably finds the best solutions across all benchmarks. We then empirically evaluate optimal population size and island number in the island-model GA on 7 benchmarks, analyzing the results using heat maps and by fitting a general linear model. We find two categories of benchmarks, one category for which intermediate population size island-model GAs perform better and the other for which minimal population size island-model GAs perform better. Multiple islands perform better on all benchmarks compared to single-population models.

A commonly stated assumption is that spreading a population too thinly among island populations results in poor performance [13, 14, 25, 26]. The assumption that island populations must be sufficiently large to make progress on their own before an island-model GA can accrue benefits from migration makes intuitive sense, but we find the opposite. Contrary to this intuition, we find that island-model performance is optimal with surprisingly few individuals per island.

By examining heat maps of our data (Figure 2) and by fitting a quadratic general linear model (Table 2) we distinguish between category A benchmarks and category B benchmarks. Both categories have a unique optimal island number around 26 to 33 islands. The categories differ with respect to population size. The two categories can be roughly summarized as follows: for category A benchmarks, the GA performs better with roughly 8 individuals per island. For category B benchmarks, the GA performs better with one individual per island.

The consistency between the optimal parameter values for the different benchmarks is both surprising and encouraging. It suggests that the optimal island number, and to a lesser extent the optimal population size, generalize to other benchmarks and perhaps to real world problems.

The results of the population size and island number experiments for category B benchmarks are consistent with our finding that the "every man is an island" algorithm is highly effective on a wide variety of benchmarks. This algorithm performs as well as or better than every other algorithm on every single benchmark by simulating an island model with islands of size one.

We hypothesize that *emIsland*'s superior performance is due to maintenance of diversity and preservation of fit individuals. Though migration occurs every generation in *emIsland*, the low connectivity prevents relatively fit individuals from quickly dominating all the islands. For example, it would take one individual 64 generations to reach every island in the 64-island *emIsland* due to the topology of the one-directional ring. Fit individuals are preserved due to the fact that a mutant or child of crossover takes over an island only if its fitness is superior to the individual already on the island. These hypotheses will be investigated in future work.

In conclusion, we have shed light on key design decisions affecting evolutionary algorithms, uncovered a talented little optimization algorithm in "every man is an island", and taken important steps towards a priori parameter optimization in island-model genetic algorithms.

6. FUTURE WORK

In future work, we will evaluate additional benchmarks to see if they fall into our informal categorization scheme, A or B. We will also determine if features of the benchmark functions can be used to determine beforehand which category the benchmarks will fall into.

The optimal island number appears to generalize across at least 7 benchmarks. The optimal population size appears to fall into one of two categories. In future work, we will investigate how sensitive these optimal values are to the variance of GA parameters, number of fitness function calls, problem dimensionality, and precision of representation. Our goal is to present an empirically based formula for determining optimal population size and island number on previously unseen optimization problems.

7. ACKNOWLEDGMENTS

The authors would like to acknowledge Kenneth Letendre and Stephanie Forrest for their advice and guidance. A large thanks goes out to Ben Edwards for his irreplaceable statistics and python support. This work is supported by the DARPA CRASH grant (P-1070-113237).

8. REFERENCES

[1] J. Arabas, Z. Michalewicz, and J. Mulawka. Gavaps-a genetic algorithm with varying population size. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 73 –78 vol.1, jun 1994.

[2] T. Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA, 1996.

[3] T. C. Belding. The Distributed Genetic Algorithm Revisited. In *eprint arXiv:adap-org/9504007*, page 4007, May 1995.

[4] V. Bilolikar, K. Jain, and M. Sharma. An annealed genetic algorithm for multi mode resource constrained project scheduling problem. *International Journal of Computer Applications*, 60(1):36–42, 2012.

[5] E. Cantu-Paz and D. E. Goldberg. Efficient parallel genetic algorithms: theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):221 – 238, 2000.

[6] C.-C. Chiu, C.-H. Chen, S.-H. Liao, and K.-C. Chen. Bit error rate reduction by smart uwb antenna array in indoor wireless communication. *Journal of Applied Science and Engineering*, 15(2):139–148, 2012.

[7] E. Creaco and M. Franchini. Fast network multi-objective design algorithm combined with an a posteriori procedure for reliability evaluation under various operational scenarios. *Urban Water Journal*, 0(0):1–15, 0.

[8] J. Denies, B. Dehez, F. Glineur, and H. Ben Ahmed. Genetic algorithm-based topology optimization: Performance improvement through dynamic evolution of the population size. In *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*, pages 1033 –1038, june 2012.

[9] J. M. Dieterich and B. Hartke. Empirical review of standard benchmark functions using evolutionary global optimization. *CoRR*, abs/1207.4318, 2012.

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

[11] C. Fernandes and A. Rosa. A study on non-random mating and varying population size in genetic algorithms using a royal road function. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 60 –66 vol. 1, 2001.

[12] C. Fernandes and A. Rosa. Self-regulated population size in evolutionary algorithms. In T. Runarsson, H.-G. Beyer, E. Burke, J. Merelo-Guervos, L. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 920–929. Springer Berlin / Heidelberg, 2006.

[13] F. Fernández, M. Tomassini, and L. Vanneschi. An empirical study of multipopulation genetic programming. *Genetic Programming and Evolvable Machines*, 4:21–51, March 2003.

[14] D. Goldberg, H. Kargupta, J. Horn, and E. Cantu-Paz. Critical deme size for serial and parallel genetic algorithms. *Technical Report ILLiGAL 95002*, 1995.

[15] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *COMPLEX SYSTEMS*, 6:333–362, 1991.

[16] J. Grefenstette. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122–128, Jan. 1986.

[17] R. Gupta. Genetic algorithm based approach for obtaining alignment of multiple sequences. *complexity*, 3(12), 2012.

[18] P. A. Gutiérrez, C. Hervás, and M. 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.

[19] G. Harik, G. Harik, D. E. Goldberg, D. E. Goldberg, B. L. Miller, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. In *Evolutionary Computation*, pages 7–12. IEEE Press, 1997.

[20] R. 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.

[21] T. Jumonji, G. Chakraborty, H. Mabuchi, and M. Matsuhara. A novel distributed genetic algorithm implementation with variable number of islands. In *IEEE Congress on Evolutionary Computation*, pages 4698–4705, 2007.

[22] J. Kennedy and W. Spears. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 78 –83, may 1998.

[23] H. Muhlenbein. Evolution in time and space - the parallel genetic algorithm. In *Foundations of Genetic Algorithms*, pages 316–337. Morgan Kaufmann, 1991.

[24] H. Muhlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17(6-7):619 – 632, 1991.

[25] R. Nakano, Y. Davidor, and T. Yamada. Optimal population size under constant computation cost. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Parallel Problem Solving from Nature*, volume 866 of *Lecture Notes in Computer Science*, pages 130–138. Springer Berlin / Heidelberg, 1994.

[26] W. Punch. How effective are multiple populations in genetic programming. *Genetic Programming*, 98:308–313, 1998.

[27] Z. Skolicki and K. 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.

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

[29] R. Tanese. Parallel genetic algorithms for a hypercube. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 177–183, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[30] R. Tanese. *Distributed genetic algorithms for function optimization*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1989. AAI9001722.

[31] S. Wenger and M. Magnor. A genetic algorithm for audio retargeting. In *Proc. ACM Multimedia 2012*, June 2012.

[32] D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47, 1998.

[33] W. Zhu, J.-a. Fang, Y. Tang, W. Zhang, and W. Du. Digital iir filters design using differential evolution algorithm with a controllable probabilistic population size. *PLoS ONE*, 7(7):e40549, 07 2012.