Evaluation of a Randomized Parameter Setting Strategy for Island-Model Evolutionary Algorithms

Ryoji Tanabe and Alex Fukunaga
Graduate School of Arts and Sciences
The University of Tokyo

***Abstract*—This paper presents a large-scale, empirical evaluation of a Random, Heterogeneous Island-Model (RHIM) for evolutionary algorithms (EAs), where the control parameter values are independently, randomly assigned for each island that has recently been proposed by Gong and Fukunaga as a method for configuring island-model evolutionary algorithms in situations where it is not possible to expend the resources to carefully tune control parameters for a particular application.**

We apply RHIM to standard DE, JADE (an adaptive DE), and real-coded genetic algorithms. Evaluations are performed on standard black-box function optimization benchmarks, as well as combinatorial optimization problems (the TSP and QAP). The search efficiency of RHIM is compared to manual tuning of parameter settings for each benchmark problem. Our results with up to 256 islands, show that the search efficiency of RHIM, a method which does not involve any parameter tuning, tends to becomes increasingly competitive with manual parameter tuning as the number of islands increases. The consistent, relatively good performance of RHIM when applied to a variety of EAs on numerous, different benchmark problems suggest that it can be an effective, default method for configuring island-model EAs.

I. Introduction

Evolutionary algorithms (EAs) are "embarrassingly parallel" algorithms that are a natural fit for increasingly ubiquitous, parallel computers, including multicore machines, and distributed clusters (including cloud/grid/P2P resources). While there are numerous models for mapping population-based methods to parallel processors, including master-slave, island, fine-grained, and hierarchical models, parallelization remains an active area of research.

An *island-model* EA consists of a set of islands, where each island executes an independent EA instance (i.e., subpopulation or *deme*), and selection and other evolutionary operators are applied in each island independently from the other islands. The islands may communicate with each other via some migration mechanism. The island model is a particularly simple, natural method of implementing a parallel EA, e.g., each island could executes as a separate thread (on a shared-memory, multicore system), or as a separate process on a distributed system. The idea of largely independent, possibly communicating demes can also be applied to a sequential, single-processor EA as a way to impose an artificial population structure to promote/control diversity.

In the simplest possible island-model EA, each island is configured to use the same control parameters (e.g., population size), so that each island essentially executes the same algorithm, except for differing initial populations. We refer to this type of island-model EA as a *homogeneous* island-model EA. In contrast, in a *heterogeneous* island-model EA, it is possible to configure each island independently so that a different set of control parameters is used on each island.

In this paper, we consider the problem of configuring an island-model EA in a *resource-constrained, "black-box" optimization scenario*: The goal is to optimize the parameters to an unknown, poorly understood objective function, e.g., a simulation which takes hours or even days to execute just once (c.f., [1]). Suppose that a practitioner is given 1 week to find a good (but not necessarily optimal) solution for a problem where a single objective function evaluation (i.e., simulation run) requires 1 CPU hour. This allows only $24 \times 7 = 168$ objective function evaluations per core during the week. Even on a cluster with 100 CPU cores this is only enough time to execute 16800 objective function evaluations – barely enough for one (or several) runs of an EA. It is well-known that the performance of EAs is significantly influenced by control parameter settings. Unfortunately, in this kind of resource-constrained scenario, parameter tuning is infeasible.

How can one configure the control parameters of an island-model EA under these severe, resource constraints? The problem of setting control parameters for an EA has been extensively studied (c.f., [2] for a recent survey). However, the vast majority of the literature on parameter setting techniques has been for single-deme EAs, and while there has been some work on parameter setting in island-model EAs (e.g., [3]), parameter tuning for multi-island EAs, particularly for heterogeneous island-model EAs, is still not well-understood.

Recently, Gong and Fukunaga proposed the *Random, Heterogeneous Island Model* (RHIM), a simple approach to parameter setting where each island is assigned a set of control parameter values that are selected randomly (uniformly) from a range of plausible values [4]. An evaluation of this simple technique to a standard, binary-coded island-model GA showed promising results for three standard benchmark problems and a sorting network generation problem.

In this paper, we present a large-scale, evaluation of the Random Heterogeneous Island Model. The approach is evaluated on a set of 13 standard optimization benchmark functions, as well as the Traveling Salesperson Problem (TSP) and the Quadratic Assignment Problem (QAP). While previous work on RHIM was quite limited in scope and used a simple, binary-coded GA, this work evaluates RHIM on a range of EA's, including standard Differential Evolution [5] (including several migration topologies and policies), a real-coded GA, an adaptive DE (JADE) [6], and problem-specific GAs for the TSP and QAP. Our results indicate that RHIM should be considered as a simple, effective, *default method* for setting

control parameters in an island-model EA in situations where careful parameter tuning is not feasible.

The paper is organized as follows. In Section II, we describe the Randomized, Heterogeneous Island Model for configuring island control parameters. Section III describes alternative island model configuration models which are used for comparison in the empirical evaluation of RHIM. Section IV presents the empirical evaluation of RHIM applied to differential evolution on standard, black-box function optimization benchmarks. In Section V, we evaluate RHIM on 4 alternative migration topologies and policies. The evaluation of RHIM on real-coded GA and JADE, an adaptive DE are presented in Sections VI and VII. We compare our work with related work in Section IX, and conclude with a discussion of our results and directions for future work in Section X.

II. Randomized, Heterogeneous Island-Model

The two key features of a Randomized, Heterogeneous, Island-Model (RHIM) EA are:

- Heterogeneity - each island can be assigned a different set of control parameter values, and
- Randomization - the parameter values are selected randomly.

The first, basic feature is heterogeneity, i.e., islands can have varying control parameter values, rather than a homogeneous island-model EA where all islands have the same control parameter values. For any particular problem, it might be the case that the best performance is obtained with a homogeneous island configuration, but a heterogeneous set of control parameters should be more robust (in the sense of alleviating worst-case performance) when the solver is applied to a wide range of test problems. Of course, if the scope of test problems is completely unconstrained, then the No Free Lunch Theorems imply that all configurations will exhibit the same average performance [7]. However, if we restrict ourselves to broad classes of benchmark optimization problems heterogeneous configurations offer the possibility of robustness compared to homogeneous configurations.

The second key feature is the randomized selection of parameter values. More precisely, the control parameter values are selected uniformly and independently from $D(p_1), ..., D(p_n)$, the domains of parameters $p_1, ..., p_n$. For example, in a GA, the parameters might be the population size $pop \in [2, 500]$, crossover rate $p_{cross} \in [0.0, 1.0]$, and mutation rate $p_m \in [0.0, 1.0]$.

It is important to note that this space of parameter values is *not* unconstrained, and it is *not* completely arbitrary. Randomly selecting parameter values from a completely unconstrained parameter space is clearly not a good idea. For example, if the range of population sizes is $[1, 10^{10}]$ and we uniformly sample the population size from this range, the expected value is $10^9/2$. Even assuming that there is enough memory for a population of this size, this will cause the EA to spend all of the available time evaluating the initial population, behaving in effect like a random generate-and-test algorithm, regardless of the values of the other control parameters. Thus, we assume that the parameter space is constrained such that "obviously bad" parameter values (i.e., values that most practitioners would reject out of hand) are excluded. The space represents the set of possibilities that the programmer believes to be the *feasible* parameter settings.

Algorithm 1: Island EA Execution Model

```
   //  Initialization phase
1  Initialize island population and control parameters;
   //  Main loop
2  while The termination criteria are not met do
       //  Advance each island 1 'step'
3      for i = 1 to ♯ of islands do
4          Evolve1Step(Island_i);
5      end
       //  Migration phase
6      for i = 1 to ♯ of islands do
7          if The best individual on Island_i was updated then
8              Island_r = a randomly selected destination island;
9              if fitness(best_individual(Island_i)) <
                  fitness(worst_individual(Island_r)) then
10                 Replace(worst_individual(Island_r),
                       best_individual(Island_i));
11             end
12         end
13     end
14 end
```

This paper focuses on a *static* approach where the control parameters that are determined according to random sampling do not change during an EA run. For example, if population size is a control parameter which is set according to the RHIM approach, then it remains constant through the EA run. However, this allows for the possibility that each island is executing a self-adaptive EA – in that case, the control parameters we consider are meta-level control parameters to the self-adaptive EA. For example, in Section VII, we use RHIM to set the meta-level control parameters for an island-model version of JADE [6], an adaptive DE.

A. Island-Model EA Execution Model

Island-model evolutionary algorithms are naturally suited for parallel processing, and the original work on RHIM GAs by Gong and Fukunaga evaluated a parallel GA implemented on a cluster [4]. However, large-scale evaluation of parallel code is challenging due to resource allocation/cost issues – our experiments use up to 256 islands. Thus, while our work is motivated by the problem of configuring a parallel EA, the experiments are performed using a serial implementation of an island-model EA. While experiments are actually executed on a cluster with 48 cores, the island-model EA code is implemented serially, and we executed multiple runs in parallel.

Algorithm 1 shows the (simulated) parallel execution model used for all of our experiments. The default migration topology is a fully connected topology where the destination is randomly selected island. After each "step" (defined below), if the local best-so-far individual M has been updated, M', a copy of M, is sent to some destination according to the topology described above. At the destination island D, we check whether the migrant M' is more fit than the worst member of D. If so, then M replaces the worst member of D; otherwise, M' is discarded.

A "step" is a discrete batch of objective function evaluations. Migrations can only occur in the window after all islands have finished executing a step. In the case of a homogeneous island-model EA, a "step" is simply the number of individuals

per island. On the other hand, in a heterogeneous island-model EA, each island subpopulation can have a different number of individuals, so a "step" is *not* necessarily equivalent to a complete generation. In general, a step is defined as P_{max} objective function evaluations, where P_{max} is the number of individuals in the island with the largest population. Thus, in this model, across each step, each island performs the same number of evaluation function evaluations.

III. Alternative Island Model Configuration Models

The main hypothesis of this paper is: given a feasible parameter space, and *a sufficiently large number of islands*, assigning control parameters to each island by sampling this space is a relatively robust method for configuring the parameters of an island-model EA.

We evaluate RHIM by comparing it to several natural, alternative methods for configuring an island-model EA.

1) Best Homogeneous-Multi (BH-M): Another approach to configuring an island-model EA which is common in practice is brute-force parameter tuning. The BH-M configuration models such a brute-force parameter tuning effort, is intended to generate a "highly-tuned" configuration that is tuned *specifically for each particular target problem*. 200 parameter sets are randomly generated. For each parameter set S, we create a homogeneous, I-island-model EA by replicating S on all processors. We evaluate each of these 200, I-island configurations on the target problem, and use the configuration with the best average performance. [1]

2) Median Homogeneous-Multi (MH-M): It should be expected that the BH-M configuration, which is the best of 200 configurations, will perform significantly better than the RHIM. To investigate how RHIM compares to a less intensive, brute-force tuning effort, we also evaluate the configuration which had the median (100th best) performance out of the 200 I-island configurations evaluated while generating the BH-M configuration above. The MH-M can be considered a model of a brute-force parameter tuning effort where insufficient time/resources are allocated to the tuning effort.

3) Best Homogeneous-Single (BH-1): The basic idea of the BH-1 configuration strategy is to find a parameter set that works well for a single-island EA, and then create a multi-island EA by replicating that single island. This is a naive, but plausible method for practitioners who have limited time/resources for parameter tuning, but decide to invest some effort for parameter configuration. 200 parameter sets are randomly generated. Each set is evaluated by instantiating a single-island EA that is configured to use the set, and running the EA on a target problem. The parameter set which has the best utility is selected. This parameter set is then copied to all of the I islands.

Although tuning a island-model EA based on the performance of a single island has the advantage that it can require significantly less computational effort to evaluate many different parameter settings, the obvious disadvantage is that replicating a good, single-island configuration may not lead to a good multi-island configuration.

Note that the BH-1 and BH-M configuration processes was applied separately to every benchmark problem instance used in this paper. We did *not* seek to generate configurations that worked well "on average" – the BH-1 and BH-M configurations represent substantial, brute-force efforts to tune a homogeneous EA *for each, particular benchmark problem*.

While migration is implemented as described in Section II-A for RHIM, BH-M, and MH-M, we do not use migration in BH-1, because preliminary experiments showed that BH-1 performed best without migration.

IV. Evaluating the Scalability of RHIM based Differential Evolution

We first evaluate the scalability of RHIM as the number of islands is varied. The RHIM is applied to Differential Evolution (DE) [5] for black-box function optimization. DE is a simple, effective method which has been successfully applied to a wide variety of applications [8]. Like other EAs, the performance of DE depends significantly on its control parameters. The basic control parameters for island-based DE are the population size P, scaling factor F, and crossover rate CR, and it has been shown that the best settings for these parameters vary according to the problem [5], [9], [10]. In practice, it is necessary to tune these parameters in order to obtain good performance.

The control parameters for the DE are the following:

- Scaling factor $F \in [0, 1]$, crossover rate $CR \in [0, 1]$
- The population is set to the dimensionality of the problem multiplied by a parameter $P \in [1, 5]$.
- The mutation strategy is selected from among the 7 strategies in table II, where K_i is a uniform random number in $[0, 1]$. After mutation, Binomial Crossover is applied to mutant vector and parent vector for generating a trial vector.

We use the widely used, 13 classical benchmark functions [11]. Table I shows some features of each function, as well as the maximum number of objective functions and *target accuracy* (defined below). Note that the maximum number of fitness evaluations is *per island*, e.g., a 4-island EA would execute $4 \times 1.5\text{e}+05 = 6\text{e}+06$ total evaluations. A search algorithm terminates ("succeeds") if it finds a solution whose error relative to the known, optimal score is within the target accuracy. For each problem, we executed 50 independent trials, and measure the success rate SR (% of trials that found a solution within the target accuracy), as well as the number of fitness evaluations (NFE) executed on the successful trials. Statistical tests for significance were performed using the Wilcoxon rank-sum test with a significance threshold of $p < 0.05$. While we use all 13 functions in Section IV-A, the rest of the experiments with black-box objective functions use 6 out of the 13 functions ($f_1, f_3, f_5, f_8, f_9, f_{11}$) .

A. Scalability of RHIM as the Number of Islands Varies

Table III compares RHIM, MH-M, BH-1, and BH-M on 128 and 256 islands. For ease of comparison, the NFE values

[1] The utility of each parameter set is evaluated by running 10 trials, where on each trial, max. fitness evaluations per trial of $D \times 10,000$), i.e., max. 300,000 fitness evals/trial.

TABLE III. ISLAND-BASED DE CONFIGURED USING RHIM, MH-M, BH-1, AND BH-M ON 128 AND 256 ISLANDS. 50 RUNS/PROBLEM

f	128 islands								256 islands							
	RHIM		MH-M		BH-1		BH-M		RHIM		MH-M		BH-1		BH-M	
	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE
f_1	100	1.3e+04	100	2.8^-	100	1.2^-	**100**	**0.27**$^+$	100	1.2e+04	100	3.4^-	100	1.3^-	**100**	**0.29**$^+$
f_2	100	1.8e+04	82	1.4^-	100	1.1^-	**100**	**0.24**$^+$	100	1.7e+04	100	3.8^-	100	1.2^-	**100**	**0.31**$^+$
f_3	100	4.8e+04	26	10^-	100	1.3^-	**100**	**0.23**$^+$	100	4.3e+04	100	7.7^-	100	1.5^-	**100**	**0.23**$^+$
f_4	**100**	**8.0e+04**	100	3.2^-	100	1.5^-	98	0.19^+	100	6.5e+04	100	4^-	100	1.9^-	**100**	**0.17**$^+$
f_5	**100**	**8.4e+04**	88	5.3^-	100	1.7^-	96	0.32^+	100	7.7e+04	100	5.6^-	100	1.8^-	**100**	**0.33**$^+$
f_6	**100**	**4.7e+03**	100	2.1^-	100	1.3^-	98	0.24^+	100	4.4e+03	100	2.6^-	100	1.3^-	**100**	**0.26**$^+$
f_7	100	4.0e+03	100	3.5^-	100	3.4^-	**100**	**0.22**$^+$	100	3.5e+03	100	3.3^-	100	3.6^-	**100**	**0.19**$^+$
f_8	100	1.7e+04	4	1.5^-	100	2.8^-	**100**	**0.37**$^+$	100	1.6e+04	0	N/A	100	3.1^-	**100**	**0.32**$^+$
f_9	**100**	**2.0e+04**	0	N/A	100	2.7^-	98	0.83^+	100	**1.8e+04**	0	N/A	100	3^-	84	0.25^+
f_{10}	100	2.1e+04	100	2.9^-	100	1.2^-	**100**	**0.27**$^+$	100	1.9e+04	100	3.5^-	100	1.3^-	**100**	**0.29**$^+$
f_{11}	100	1.4e+04	82	4^-	100	2^-	**100**	**0.27**$^+$	100	1.3e+04	100	5.9^-	100	2.1^-	**100**	**0.29**$^+$
f_{12}	100	1.1e+04	76	0.27^+	100	1.3^-	**100**	**0.28**$^+$	100	1.1e+04	100	3.7^-	100	1.4^-	**100**	**0.3**$^+$
f_{13}	100	1.3e+04	100	2.9^-	100	1.7^-	**100**	**0.27**$^+$	100	1.2e+04	100	3.4^-	100	1.8^-	**100**	**0.29**$^+$
Avg. SR	**100.0**		73.7		**100.0**		99.2		**100.0**		84.6		**100.0**		98.8	
Avg. Q	2.7e+02		∞		4.5e+02		**8.0e+01**		2.4e+02		∞		4.4e+02		**6.5e+01**	

TABLE IV. ISLAND-BASED DE CONFIGURED USING RHIM, MH-M, BH-1, AND BH-M ON 4, 8, 16, 32 64 ISLANDS. 50 RUNS/PROBLEM

Strategies	4 islands		8 islands		16 islands		32 islands		64 islands	
	SR	Avg. Q	SR	Avg. Q	SR	Avg. Q	SR	Avg. Q	SR	Avg. Q
RHIM	76.6	1.9e+03	84.2	1.0e+03	89.7	5.6e+02	93.4	4.1e+02	98.6	3.2e+02
MH-M	39.2	∞	45.4	∞	64.3	∞	72.6	∞	76.0	∞
BH-1	**100.0**	4.9e+02	**100.0**	4.8e+02	**100.0**	4.7e+02	**100.0**	4.6e+02	**100.0**	4.5e+02
BH-M	92.8	**3.2e+02**	95.5	**2.2e+02**	96.9	**1.5e+02**	98.2	**1.2e+02**	99.4	**1.0e+02**

TABLE I. 13 CLASSICAL BENCHMARK FUNCTIONS [11]

f	Name	Max NFE ($D = 30$)	Target accuracy	Separable	Unimodal
f_1	Sphere	1.5e+05	1.0e-08	Y	Y
f_2	Schwefel 2.22	2.0e+05	1.0e-08	Y	Y
f_3	Schwefel 1.2	5.0e+05	1.0e-08	N	Y
f_4	Schwefel 2.21	5.0e+05	1.0e-08	N	Y
f_5	Rosenbrock	2.0e+06	1.0e-08	N	N
f_6	Step	1.5e+05	1.0e-08	Y	Y
f_7	Noisy Quartic	3.0e+05	1.0e-02	Y	Y
f_8	Schwefel 2.26	9.0e+05	1.0e-08	Y	N
f_9	Rastrigin	5.0e+05	1.0e-08	Y	N
f_{10}	Ackley	1.5e+05	1.0e-08	Y	N
f_{11}	Griewank	2.0e+05	1.0e-08	N	N
f_{12}	Penalized1	1.5e+05	1.0e-08	Y	N
f_{13}	Penalized2	1.5e+05	1.0e-08	Y	N

TABLE II. DE STRATEGIES

Strategies	Definitions
rand/1	$\boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})$
rand/2	$\boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) + F \cdot (\boldsymbol{x}_{r_4} - \boldsymbol{x}_{r_5})$
best/1	$\boldsymbol{x}_{best} + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$
best/2	$\boldsymbol{x}_{best} + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}) + F \cdot (\boldsymbol{x}_{r_3} - \boldsymbol{x}_{r_4})$
current-to-best/1	$\boldsymbol{x}_i + F \cdot (\boldsymbol{x}_{best} - \boldsymbol{x}_i) + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$
current-to-best/2	$\boldsymbol{x}_i + F \cdot (\boldsymbol{x}_{best} - \boldsymbol{x}_i) + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}) + F \cdot (\boldsymbol{x}_{r_3} - \boldsymbol{x}_{r_4})$
current-to-rand/1	$\boldsymbol{x}_i + K_i \cdot (\boldsymbol{x}_{r1} - \boldsymbol{x}_i) + F \cdot (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3})$

for MH-M, BH-1, and BH-M are scaled values which are *multiples* relative to RHIM. For example, in Table III, on problem f_1, 256 islands, the NFE for MH-M was 3.4 times higher than the NFE for RHIM, i.e., the NFE for MH-M was $3.4 \times 1.2\text{e}+04 = 40,800$. The best result for each function is shown in bold. The $+, -, \approx$ symbols indicate whether result is significantly better than, significantly worse than, or not significantly different (according to the Wilcoxon ranked-sum test, threshold $p < 0.05$), compared to RHIM result for the same problem. At the bottom of the table, the average SR value for all problems are shown. The average Q score is also shown, where Q = NFE/SR. A low Q score indicates that the search finds an acceptable solution quickly and reliably. In case a method completely fails on for some particular problem (SR = 0), the average Q across all problems is defined to be ∞.

The results for 4, 8, 16, 32, and 64 islands are shown in Table IV. Due to limited space, only the mean success rate SR and mean Q-scores (defined above) across all problems are shown.

Table III shows that for 128 and 256 islands, RHIM outperforms MH-M and BH-1 on almost all of the 13 benchmark problems. Tables IV and III show that regardless of the number of islands, MH-M has the worst performance and completely fails to find acceptable (fitness below the target threshold) solutions for several problems (SR=0). Thus, the average Q-score for MH-M is ∞ in all cases. Even when MH-M succeeds in finding a solution, the amount of search required (NFE) is significantly higher than that of RHIM.

Although BH-1 outperforms RHIM for 4-16 islands, the Q-score for RHIM is better than BH-1 for 32 or more islands. For 128 and 256 islands, RHIM achieves a success rate of SR=100% for all problems, and for all problems, the NFE for RHIM is lower than that of BH-1. Thus, with sufficient parallelism (number of islands), the trivial, RHIM configuration method consistently and significantly outperforms BH-1, the result of tuning an island-model EA based on tuning the performance of a single island for each problem.

As expected, BH-M, which is the result of tuning a I-island, homogeneous DE for each benchmark problem separately, achieves the best performance overall, regardless of the number of islands. The Q-score is consistently the highest, and it can be seen from Table III, the NFE for BH-M is signifi-

TABLE V. DE WITH NO MIGRATION. 6 TEST FUNCTIONS ($f_1, f_3, f_5, f_8, f_9, f_{11}$). 50 RUNS/PROBLEM

Strategies	16 islands		128 islands	
	SR	Avg. Q	SR	Avg. Q
RHIM	91.7	2.2e+03	**100.0**	6.9e+02
MH-M	3.0	∞	16.7	∞
BH-M	**100.0**	**5.2e+02**	**100.0**	**4.8e+02**

TABLE VI. DE WITH RING/BEST MIGRATION TOPOLOGY/POLICY. 6 TEST FUNCTIONS ($f_1, f_3, f_5, f_8, f_9, f_{11}$). 50 RUNS/PROBLEM

Strategies	16 islands		128 islands	
	SR	Avg. Q	SR	Avg. Q
RHIM	79.3	8.7e+02	**100.0**	4.4e+02
MH-M	41.3	∞	66.0	∞
BH-M	**96.7**	**2.7e+02**	99.3	**1.3e+02**

TABLE VII. DE WITH RING/PERIODIC MIGRATION TOPOLOGY/POLICY. 6 TEST FUNCTIONS ($f_1, f_3, f_5, f_8, f_9, f_{11}$). 50 RUNS/PROBLEM

Strategies	16 islands		128 islands	
	SR	Avg. Q	SR	Avg. Q
RHIM	98.7	1.0e+03	**100.0**	5.7e+02
MH-M	55.0	∞	98.7	2.5e+03
BH-M	**99.0**	**2.3e+02**	**100.0**	**2.0e+02**

TABLE VIII. FULL/PERIODIC MIGRATION TOPOLOGY/POLICY. 6 TEST FUNCTIONS ($f_1, f_3, f_5, f_8, f_9, f_{11}$). 50 RUNS/PROBLEM

Strategies	16 islands		128 islands	
	SR	Avg. Q	SR	Avg. Q
RHIM	97.3	9.2e+02	**100.0**	4.7e+02
MH-M	47.0	∞	59.0	7.9e+04
BH-M	**97.7**	**2.2e+02**	99.3	**2.0e+02**

cantly lower than that of RHIM on all 13 problems. This shows that clearly, carefully tuning an EA for each problem results in good performance. However, as noted in Section I, tuning an optimization algorithm for a given problem is frequently not feasible in many real-world, black-box optimization scenarios, where each fitness function evaluation takes a very long time. The RHIM is intended for such scenarios, where parameter tuning is infeasible due to time/resource constraints.

We have also evaluated island-model DE using RHIM, BH-1, BH-M, and MH-M on harder problems ($D = 60$ dimensions and $D = 90$ dimensions) for 16 and 128 islands. Although details are omitted due to space constraints, the results for 60 and 90 dimensions were qualitatively similar to the 30-dimension, 16 and 128 island results shown here.

V. EVALUATION OF RHIM USING VARIOUS MIGRATION TOPOLOGIES AND POLICIES

In all of our experiments *except* this section, we use a fully connected migration topology, and the migration policy is to generate a migrant and send it to a randomly selected neighbor when the local best-so-far individual is updated. In this section, we evaluate RHIM for function optimization using 4 alternative migration topologies and policies:[2]

- No migration; islands evolve independently.
- Ring/Best - Ring Topology; Migrant sent when local best-so-far is sent.
- Ring/Periodic - Ring topology; Migrant is sent every 5 steps (where "step" is defined as in Section II-A).
- Full/Periodic - Fully connected topology; Migrant is sent every 5 steps.

For each migration strategies, we evaluated RHIM, MH-M, BH-M for DE on 16 and 128 islands, using 6 function optimization problems ($f_1, f_3, f_5, f_8, f_9, f_{11}$). BH-1 was not evaluated in this comparison because it performs significantly better without migration. The results are shown in Tables V (no migration), VII (Ring+Periodic), VI (Ring+BestUpdate), VII (Ring+Periodic), and VIII (Full+Periodic).

The results show that RHIM consistently outperforms MH-M for all migration topologies/policies, and for both 16 and 128 islands. Thus, the robustness of RHIM does not depend on any particular migration topology/policy.

[2]Using this naming scheme, the default migration topology/policy used in all of the other experiments would be called "Full/Best".

VI. RHIM-BASED REAL-CODED GA

In principle, RHIM can be applied to any EA with control parameters. In this section, we apply RHIM to a real-coded genetic algorithm (GA). The PBX-α [12] crossover operator and G3 [13] generation alternation model are used. In order to be tunable, the G3 generation alternation model is modified as follows: While the original G3 always selects the best individual among the parents as the centric parent, our implementation selects the best individual among a subpopulation of size $P \times p$ ($p \in [0, 1]$) which is selected randomly from the full population in the current generation.

The control parameters for the GA are the following: The population size was set by multiplying the dimensionality of the problem ($D = 30$) by a multiplier $m \in [2.0, 10.0]$, the number of offspring per mating was selected from $[1, 10]$, the expansion rate α for the PBX-α was selected from $[0.5, 1.5]$, and greediness parameter for G3 (described above) was $p - rate \in [0.0, 1.0]$.

Table IX shows the summary of the results of comparing RHIM, MH-M, BH-1, BH-M using 16 and 128 islands on 6 problems from Table I ($f_1, f_3, f_5, f_8, f_9, f_{11}$). For benchmarks f_8, f_9, the island-model RCGA was unable to find solutions within the target accuracy threshold used for our DE experiments within the given computational limits. Thus, for these two benchmarks, we changed the target accuracy from 10^{-8} to 2.5×10^3 and 5.0×10^1, respectively, for these RCGA experiments. The results are similar to the DE results – with a large number of islands, the performance of RHIM is substantially better than MH-M and BH-1, and similar to the performance of BH-M.

VII. RHIM-BASED ADAPTIVE DE

While we have shown so far that the RHIM is a robust method for configuring island-model evolutionary algorithms including standard DE and GAs, there is a large body of work on self-adaptation mechanisms for evolutionary algorithms (c.f. [2]), mostly for single-deme EA's. Most of these self-adaptation mechanisms are not entirely parameter free – while some of the traditional control parameters are automatically adapted, there are new, "meta-level control parameters" that control, for example, the rate of self-adaptation.

TABLE IX. REAL-CODED GENETIC ALGORITHM: 6 FUNCTIONS ($f_1, f_3, f_5, f_8, f_9, f_{11}$). 50 RUNS/PROBLEM

Strategies	16 islands		128 islands	
	SR	Avg. Q	SR	Avg. Q
RHIM	62.0	6.3e+02	95.7	4.2e+02
MH-M	73.0	3.9e+02	98.3	8.7e+02
BH-1	**100.0**	8.4e+02	**100.0**	7.1e+02
BH-M	91.0	**3.7e+02**	95.3	**1.9e+02**

TABLE X. ADAPTIVE DIFFERENTIAL EVOLUTION ALGORITHM (JADE): 6 FUNCTIONS ($f_1, f_3, f_5, f_8, f_9, f_{11}$) . 50 RUNS/PROBLEM

Strategies	16 islands		128 islands	
	SR	Avg. Q	SR	Avg. Q
RHIM	97.8	3.6e+02	99.7	2.9e+02
MH-M	93.0	5.6e+02	**100.0**	5.2e+02
BH-1	**100.0**	4.4e+02	**100.0**	4.1e+02
BH-M	89.5	**3.1e+02**	99.5	**2.4e+02**

TABLE XI. GA WITH ORDER-BASED OPERATORS FOR THE TSP: 6 TSP INSTANCES (ATT48, EIL51, PR76, RAT99, KROA100, LIN105) FROM TSPLIB. 50 RUNS/PROBLEM

	RHIM	MH-M	BH-1	BH-M
16 islands	+	0	3	1
	-	2	1	0
	≈	4	2	5
128 islands	+	0	0	2
	-	2	4	0
	≈	4	2	4

TABLE XII. GA WITH ORDER-BASED OPERATORS FOR THE QUADRATIC ASSIGNMENT PROBLEM: 7 QAP INSTANCES (TAI25A, BUR26A, KRA30A, TAI35A, TAI35B, LIPA40A, TAI50B) FROM QAPLIB. 50 RUNS/PROBLEM

	RHIM	MH-M	BH-1	BH-M
16 islands	+	2	7	5
	-	0	0	0
	≈	5	0	2
128 islands	+	0	5	5
	-	3	0	0
	≈	4	2	2

RHIM is complementary to self-adaptation, because a self-adaptive mechanism can be used to automatically tune base level control parameters on each island, while the meta-level control parameters are set randomly according to the random heterogeneous model.

JADE [6] is one of the state-of-the-art, adaptive DE algorithm which dynamically self-adapts the F and CR parameters. JADE has been shown to outperform standard DE on a wide variety of benchmark problems. JADE has two meta-level parameters which affects performance: c, which controls the rate of parameter adaptation, and p, which determines the greediness of the mutation strategy. Although it has been argued that the performance of standard single-deme JADE should be insensitive to the values of c and p [6], this has not been evaluated in an island-model setting.

We applied RHIM to JADE, with the following 4 parameters m, c, p, a: The population size was set by multiplying the dimensionality of the problem ($D = 30$) by a multiplier $m \in [1.0, 5.0]$. Following the parameter study of JADE [6], $c \in [0.05, 0.2]$, and $p \in [0.05, 0.2]$. In addition, the size of the archive used by JADE was set to the population size multiplied by an archive multiplier $a \in [0.0, 1, 0]$.

Table X shows the results for 6 benchmark problems (30 dimensions) using 16 and 128 islands. RHIM performs relatively well, and has a significantly better average Q-score than MH-M and BH-1. This shows that RHIM can be applied successfully to a self-adaptive EA.

VIII. EVALUATION ON COMBINATORIAL OPTIMIZATION PROBLEMS

So far, our evaluation has focused on the application of RHIM to various evolutionary algorithms for black-box function optimization benchmark problems. In this section, we evaluate RHIM on two representative, combinatorial optimization problems: the Traveling Salesperson Problem (TSP) and Quadratic Assignment Problem (QAP) (see, e.g., [14] for definitions).

For both problems, we used a Steady State GA (SSGA) with problem-specific genetic representation and genetic operators. The control parameters for the SSGA are the following (N is the size of the problem instance).

- $P \in [1, 10]$ is a parameter that control the population size (the population size is set to $P \times N$).
- Tournament size $TS \in [2, N/5]$
- The number of children $C \in [2, N/5]$
- Mutation rate $m_r \in [0, 1]$

A standard order-based representation, along with problem-specific crossover and mutation operators, was used for both the TSP [15] and QAP [16]. Specifically, for the TSP, Order Crossover (OX) and a 2-Edge Exchange Mutation operator that swaps 2 randomly selected edges were used, and for the QAP, the Cycle Crossover (CX) operator and a Swap Mutation operator that swaps 2 randomly selected nodes was used.

In these experiments, we used 6 TSP instances (att48, eil51, pr76, rat99, kroA100 and lin105) from TSPLIB [1], and 7 QAP instances (tai25a, bur26a, kra30a, tai35a, tai35b, lipa40a and tai50b) from QAPLIB [2]. The numerical portion of each instance indicates the problem size N (e.g., att48 is a TSP instance with 48 cities), and The maximum number of fitness evaluations (NFE) was $N \times 1,000$. Unlike the previous, black-box function optimization instances, it was difficult to set an appropriate target accuracy, so for the TSP and QAP, we evaluate the quality of the best solution found after NFE evaluations.

Tables XI-XII summarize the TSP and QAP results for 16 and 128 islands. The number of problems for which each method performed significantly better ($+$), worse ($-$), and not significantly different ($\approx$) compared to RHIM are shown.

Table XI shows that on the TSP, RHIM outperforms MH-M for both 16 and 128 islands, and also outperforms BH-1 with 128 islands. On the QAP, Table XII shows that while RHIM performs worse than MH-M on 16 islands, RHIM outperforms MH-M with 128 islands.

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[2] http://www.opt.math.tu-graz.ac.at/qaplib/

IX. Related Work

Miki, Hiroyasu, and Hatanaka proposed DEGA, a heterogeneous, island model GA where each island was assigned a different mutation rate and crossover rate [17]. This scheme was evaluated using 9-islands, where the specific values for the crossover and mutation rates were manually selected. They showed that DEGA outperformed hand-tuned configurations of a standard, binary-coded GA for the Rastrigin, Schwefel, Griewank, and Rosenbrock functions in 10 dimensions, on a 9-island distributed GA [17], [18].

Herrera and Lozano proposed a heterogeneous, island-model GA where each island uses a different crossover operator [19]. They used a 8-island model with a 3-D hypercube topology, and manually assigned a gradually increasing/decreasing crossover expansion rate and selection pressure to each island, in order to design an environment where the balance between exploitation and exploration gradually increased as the islands were traversed according to their topology.

Dorronsoro and Bouvry investigated a wide variety of distributed DE implementations, including a two-island, heterogeneous model where both islands were manually configured (HdDE) [20]. On a single processor, this was shown to be competitive with the state-of-the-art, single-threaded DE.

Weber, Neri, and Tirronen proposed a heterogeneous, distributed DE with two classes of islands. The first class of islands are uniformly configured (i.e., homogeneous), and are intended to explore the search space, and the second class of islands are dynamically, heterogeneously configured and intended to perform local search [21]. This approach was evaluated on a large set of function optimization benchmarks using 2-3 subpopulations.

Peng et al., investigated algorithm portfolios composed of 2-4 different evolutionary algorithms (e.g., CMA-ES, GA, DE and PSO) [22]. Each algorithm was allocated one subpopulation, and the control parameters for each constituent algorithm was manually selected. These portfolios were evaluated extensively on standard function optimization benchmarks.

While these previous papers on heterogeneous, island-model EAs showed the potential of a distributed, heterogeneous model, in various settings, our work differs in two respects: First, in contrast to previous work, where the control parameters used for each subpopulation/island were manually determined, RHIM selects control parameters for individual subpopulations by randomly sampling a space of plausible parameter settings. Second, in contrast to previous work which focused on a fairly specific class of EA for a relatively small number of islands (or combinations of EAs, as in [22]), this paper presents a broad set of empirical evaluations on a wide range of benchmark problems, including both function optimization and combinatorial optimization problems, and evaluates the applicability of RHIM to a variety of state-of-the-art evolutionary algorithms, using up to 256 islands.

Bianzzini et al [23] proposed a parallel hyper-heuristic where each island (processor) executes an iteration of some heuristic (EA), and passes the result of executing heuristics among the islands. Bianzzini et al [23] use a set of 8 EAs (6 different configurations of differential evolution, a particle swarm optimization, and a random search algorithm). If there are more than 8 processors, some processors will be executing duplicate EAs – this duplication is not a problem because hyper-heuristics operate by very rapidly passing the output of each heuristic as the input of another heuristic. They investigate various strategies for dynamically reassigning the 8 candidate heuristics to each processor. Similarly, Leon et al apply a dynamically reconfiguring, parallel hyper-heuristic to 2D packing [24]. The RHIM is at the opposite end of the spectrum as this line of work. Our focus is on an extremely simple, *static*, randomized strategy. We statically assign a randomly generated GA configuration to each processor, so there is greater diversity in the algorithmic configurations (compared to [23], [24]), but the configurations are not changed during the run. An empirical comparison of these contrasting approaches is an interesting avenue for future research.

The parameter-less GA is an approach to eliminating parameter tuning in GAs [25]. In this approach, Harik and Lobo first "eliminate" some parameters by arguing (based on schema theory) that selection rate and crossover rate should be set to a constant setting for all problems, and turning off mutation completely. The remaining, single parameter is population size. They execute a race among multiple populations of various sizes. Smaller populations are killed and replaced by larger populations when it no longer seems worthwhile to continue running the small population. Although this work was implemented sequentially, the racing populations could be implemented in parallel. The RHIM can be considered a different type of "parameter-less", parallel GA. While Harik and Lobos fixed selection rate and crossover rates at particular values, we sample from a range of reasonable values.

Numerous approaches to self-adaptive genetic EAs have been proposed (c.f., [26], including adaptive methods for some island-model EAs (c.f., [27]). However, most of these approaches introduce meta-level control parameters which themselves must be tuned for the adaptation to work well, so they are not quite parameter-free. In contrast, we do not try to actively automatically tune the system, but take a completely passive approach: we rely on random assignment of parameters to assign good control parameter settings to *some* of the islands, and exploit the fact that this becomes increasingly likely as the number of islands increases. While the bounds on the ranges are meta-level parameters, the purpose of these parameters is not to bias the parameter space in a particularly favorable way – rather, the purpose is simply to rule out values that are *a priori* believed to be extremely poor.

X. Discussion and Conclusions

This paper presented a large-scale, empirical evaluation of the Random Heterogeneous Island Model, a method for configuring island-model evolutionary algorithms by assigning independently selected randomly parameter values to each island [4]. While the previous work that introduced RHIM [4] evaluated RHIM on only 3 benchmark functions and sorting network generation problem, all on a simple, binary-coded GA, this paper evaluates the RHIM on a much larger set of benchmarks (13 function optimization benchmarks, 6 TSP and 7 QAP benchmarks), and also evaluates RHIM applied to a broad range of modern EAs (differential evolution, real-coded GA, adaptive DE, and problem-specific GAs for the TSP and

QAP). We have also shown that RHIM is a robust method for various migration topologies and policies.

Not surprisingly, RHIM can not compete with a serious attempt to tune the control parameters of a homogeneous island-model EA for each benchmark problem (i.e., RHIM is not competitive with BH-M). However, with a sufficiently large number of islands, RHIM consistently performs better than a weak attempt to tune the control parameters of a homogeneous EA (i.e., RHIM outperforms MH-M in all settings for ≥ 128 islands), and RHIM with enough islands is also better than the naive approach of tuning a multi-island EA by tuning a single-island EA for each problem and replicating its parameters on all of the islands (i.e., RHIM outperforms BH-1 in many settings for ≥ 128 islands).

We re-emphasize that MH-M, BH-M, and BH-1 were tuned for every single benchmark problem, for every experimental setting (these are models for tuning parameters for each problems and environment, *not* tuning for good average case performance). The fact that RHIM, which randomly assignments of parameter values independently to each island can be competitive at all with any of these tuning methods is remarkable.

The RHIM is not only robust. It is also trivial to implement, and it is as parameter-free as possible – as explained in Section II the only "parameters" are bounds on the ranges for the randomly generated control parameter values, which were set very broadly in our experiments.

Therefore, our results suggest that RHIM should be considered as the default method for setting control parameters in an island-model EA, when there is insufficient time/resources for careful parameter tuning. We have shown across many benchmarks and settings that the RHIM is substantially better than naive methods that practitioners commonly resort to when tuning an island-model EA (single-island based tuning, or tuning with insufficient resources).

We have focused so far on demonstrating that the simplest possible implementation of RHIM is an effective, robust approach to island-model EA configuration. There are many directions for future work. One particularly interesting avenue is the investigation of a more expressive approach to defining how parameter values are sampled. So far, we have only considered uniform sampling from a range of parameters. Other natural sampling distributions (e.g., normal) should be investigated. Also, while parameters have been generated completely independently, expressing natural/obvious constraints on combinations of parameters (e.g., "don't generate combinations where the mutation rate and crossover rate are both over 0.9") could help prune completely unproductive parameter combinations and further improve the robustness of RHIM.

REFERENCES

[1] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Tran. Evol. Comput.*, vol. 14, no. 3, pp. 329 –355, 2010.

[2] L. F, L. C, and M. Z, Eds., *Parameter setting in evolutionary algorithms*, ser. Studies in Computational Intelligence. Springer, 2007.

[3] E. Cant'u-Paz, "Parameter setting in parallel genetic algorithms," in *Parameter setting in evolutionary algorithms*, L. F, L. C, and M. Z, Eds. Springer, 2007, pp. 259–276.

[4] Y. Gong and A. Fukunaga, "Distributed island-model genetic algorithms using heterogeneous parameter settings," in *IEEE CEC*, 2011, pp. 820–827.

[5] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[6] J. Zhang and A. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945 –958, 2009.

[7] D. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comp.*, vol. 1, no. 1, pp. 67–82, 1997.

[8] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Tran. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2011.

[9] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, 2002, pp. 293–298.

[10] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *GECCO*, 2006, pp. 485–492.

[11] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.

[12] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," *Evol. Comput.*, vol. 12, no. 3, pp. 273–302, Sep. 2004.

[13] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evolutionary Computation*, vol. 10, no. 4, pp. 371–395, Dec. 2002.

[14] M. R. Garey and D. S. Johnson, *Computers and intractability*. Freeman San Francisco, CA, 1979, vol. 174.

[15] T. Starkweather, S. Mcdaniel, D. Whitley, K. Mathias, and D. Whitley, "A comparison of genetic sequencing operators," in *Proc. Int. Conf. on Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 69–76.

[16] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 337–352, Nov. 2000.

[17] M. Miki, T. Hiroyasu, and K. Hatanaka, "A parallel genetic algorithm with distributed environment scheme," in *Proc. IEEE Systems, Man, and Cybernetics*, 1999.

[18] M. Kaneko, M. Miki, and T. Hiroyasu, "A parallel genetic algorithm with distributed environment scheme," in *PDPTA*, 2000, pp. 695–700.

[19] F. Herrera and M. Lozano, "Gradual distributed real-coded genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 1, pp. 43–63, 2000.

[20] B. Dorronsoro and P. Bouvry, "Improving classical and decentralized differential evolution with new mutation operator and population topologies," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 67–98, 2011.

[21] M. Weber, F. Neri, and V. Tirronen, "Distributed differential evolution with explorative—exploitative population families," *Genetic Programming and Evolvable Machines*, vol. 10, no. 4, pp. 343–371, 2009.

[22] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-based algorithm portfolios for numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 782–800, Oct. 2010.

[23] M. Biazzini, B. Bánhelyi, A. Montresor, and M. Jelasity, "Distributed hyper-heuristics for real parameter optimization," in *GECCO*, 2009, pp. 1339–1346.

[24] C. León, G. Miranda, and C. Segura, "A memetic algorithm and a parallel hyperheuristic island-based model for a 2D packing problem," in *GECCO*, 2009, pp. 1371–1378.

[25] G. R. Harik and F. G. Lobo, "A parameter-less genetic algorithm," in *GECCO*, 1999, pp. 258–265.

[26] S. Meyer-Nieberg and H.-G. Beyer, "Self-adaptation in evolutionary algorithms," in *Parameter setting in evolutionary algorithms*, ser. Studies in Computational Intelligence, L. F, L. C, and M. Z, Eds. Springer, 2007, pp. 121–142.

[27] J. Tang, M.-H. Lim, and Y.-S. Ong, "Adaptation for parallel memetic algorithm based on population entropy," in *GECCO*, 2006, pp. 575–582.