ABSTRACT

In this paper we evaluate 2 cellular genetic algorithms (CGAs), a single-population genetic algorithm, and a hill-climber on the Black Box Optimization Benchmarking testbed. CGAs are fine grain parallel genetic algorithms with a spatial structure imposed by embedding individuals in a connected graph. They are popular for their diversity-preserving properties and efficient implementations on parallel architectures. We find that a CGA with a uni-directional ring topology outperforms the canonical CGA that uses a bi-directional grid topology in nearly all cases. Our results also highlight the importance of carefully chosen genetic operators for finding precise solutions to optimization problems.

Categories and Subject Descriptors
G.1.6 [Numerical Analysis]: Optimization—global optimization, unconstrained optimization; F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems

General Terms
Algorithms

Keywords
Benchmarking, Black-box optimization

1. INTRODUCTION

Parallel genetic algorithms (PGAs) are genetic algorithms in which the population is divided into semi-isolated subpopulations. PGAs take advantage of the speed afforded by parallel or multicore architectures. The isolation of individuals in different subpopulations has been shown to be advantageous even when running on a single CPU [3, 10, 8].

Coarse grain PGAs divide their population amongst few subpopulations. Fine grain PGAs divide it amongst many. Fine grain PGAs maintain diversity better than coarse grain PGAs, but pay a steep communication cost when they are scaled up to a large number of subpopulations [10].

Cellular genetic algorithms (CGAs), in which each individual is its own subpopulation, are the most fine of the fine grain PGAs. A spatial structure is imposed on CGAs by locating each individual on a sparse, connected graph. Individuals crossover with others from a small, connected neighborhood. The sparsity of the graph and lack of global communication allows CGAs to scale more efficiently than other fine grain PGAs.

Mühlenbein and Gorges-Schleuter introduced one of the earliest cellular genetic algorithms in 1989 with an asynchronous parallel genetic algorithm called ASPARAGOS. The algorithm uses a ladder structure where each individual has 3 neighbors at one Manhattan distance away from itself. ASPARAGOS was shown to be effective at solving traveling salesman and quadratic assignment problems. [5, 9]

The most common graph structure for cellular GAs is a two-dimensional grid with wrapping edges such that each individual has 4 neighbors, one in each cardinal direction. This structure mimics the topology of interconnected processors common to many parallel systems [10].

In addition to scalability and efficiency on GPUs, CGAs preserve diversity and avoid premature convergence because individuals in a CGA are isolated by distance and the best solutions in the population spread gradually from neighborhood to neighborhood. CGAs emphasize the exploration side of the exploration/exploitation tradeoff. [1, 5]

We benchmark and compare 2 CGA variants, a single-population GA, and a hill-climbing algorithm. Comparisons to a single-population GA benchmarked by Tran and Jin [12] are also discussed. Data and source code from these experiments can be found on the GECCO Black-Box Optimization Benchmarking (BBOB) 2013 webpage.
1: GenerateInitialPopulation(cga.pop);
2: Evaluation(cga.pop);
3: while ! StopCondition() do
4: for individual ← to cga.popSize do
5: neighbors ← CalculateNeighborhood(cga, position(individual));
6: parents ← Selection(neighbors);
7: offspring ← Recombination(cga.Pc, parents);
8: offspring ← Mutation(cga.Pm, offspring);
9: Evaluation(offspring);
10: Replacement(position(individual), auxiliary_pop, offspring);
11: end for
12: cga.pop ← auxiliary_pop;
13: end while

Figure 1: The above pseudocode for the canonical genetic algorithm is duplicated from [1].

The second CGA evaluated on the benchmarks differs from the canonical CGA only in its neighborhood. We implement a one-directional, ring CGA (ring) in which each individual has one neighbor. The selection of a mate is deterministic since there is only one other individual in each neighborhood.

A generational, single-population genetic algorithm using rank selection (ga) is implemented to test whether CGAs are superior to single-population GAs.

A hill-climber (hill) is also benchmarked for comparison. Hill-climbers have a population of one and take steps along the fitness landscape. Our hill-climber uses the same mutation operator as the GA and CGAs for its step function. Our hill-climber does not restart if it reaches a local optimum.

3. EXPERIMENTAL DESIGN

Both CGAs update synchronously. The CGAs, GA, and hill-climber use a per-gene mutation rate of 1/dimensionality such that one mutation occurs per individual per generation on average. Two point crossover with a crossover rate of 90% is used for the GA and CGAs.

All the algorithms we implement use a Gaussian mutation operator. The Gaussian mutation operator replaces a value, $x$, in an individual with a value selected from a Gaussian distribution with mean $x$ and variance 2. 2 is 20% of the range of a gene since genes range from -5 to 5. A smaller variance would result in more localized search. Algorithms benchmarked with a uniform mutation operator are included in the source code and data associated with this paper, which is available on the BBOB website, but are not included in this paper due to their poor performance.

We benchmark each of the CGAs with three different population sizes: 100, 49, and 16. These values are used because the individuals can be laid out in a square grid. These values and the neighborhood differences between ring and grid CGAs are the only experimentally varied parameters. The results for population size 49 are omitted from the paper, but included in the associated data. The GA is benchmarked with a population size of 100.

None of our algorithms restart. The number of function evaluations is limited to 50,000 * $D$ where $D$ is the number of dimensions. This limit is the same as the limit used by other researchers on this benchmark set [11, 12].

4. RESULTS

Results from experiments according to [6] on the benchmark functions given in [4, 7] are presented in Figures 3, 4 and 5 and in Tables 1 and 2.

Uni-directional, one-dimensional “ring” CGAs (ring) outperform the canonical bi-directional, two-dimensional CGA (grid) with very few exceptions, such as the f8 and f19 benchmarks, for which grid with population size 16 is competitive with ring. Furthermore, the population 16 grid outperforms grid with population 49 and 100. Since the only difference between ring and grid is the spatial structure of the populations, these results suggest that the canonical CGA struggles to diffuse superior solutions through the population. Such diffusion occurs faster with a smaller population. Since the canonical CGA uses neighborhoods with size greater than one and rank selection to choose which neighbor to crossover, inferior neighbors can be selected, further slowing the diffusion of superior solutions. Future work could test the hypothesis that slow diffusion of superior solutions hampers the canonical CGA by using an elitist selection scheme.

Population size has less of an impact on the ring CGA than it has on the grid CGA. The population size 100 ring algorithm (ring100) outperforms all others on the weakly-structured multi-modal functions in Figure 5, but is outperformed by ring16 on all multi-modal functions in 5 dimensions (Figure 4). In all other cases, the difference between ring100 and ring16 is small.

The genetic algorithm with population size 100, ga100, is superior to or competitive with the grid CGAs. Ga100 is inferior to or competitive with the ring CGAs. This is a surprising result since CGAs are generally considered to be superior to single-population GAs.

Figure 3 shows our algorithms reaching the maximum function evaluation limit before finding a solution within 10^{-3} of most benchmark problems. The algorithms scale quadratically with respect to dimensionality on all benchmarks except f2 through f5, on which they scale linearly.

Tables 1 and 2 along with Figures 4 and 5 suggest that while separable problems are amenable to hill-climbing, the hill algorithm has difficulty getting within 10^{-7} of the final solution. We suspect that the unchanging variance of the Gaussian mutation operator made it difficult for the hill-climber (and the CGAs as well) to close the distance to the optimal solution for these benchmarks.

The testbed format permits easy comparison of algorithms. We compare our algorithms to Tran and Jin's Real-Coded GA (rcga) [12], but do not include their results in this paper due to space constraints.

Rcga outperforms hill, ga, and the CGAs we implement on most of the benchmarks, some notable exceptions being the weakly-structured, multi-modal functions f20, f21, and f22, on which the CGAs outperform rcga. It may be that the diversity-preserving properties of the CGAs improve search by emphasizing exploration over exploitation on these difficult landscapes that exhibit weak global structure and have many local optima. However, the superior performance of rcga on most other functions suggests that the non-uniform mutation operator and arithmetical crossover operator rcga uses are superior to the operators our algorithms use for many benchmarks. Non-uniform mutation uses a variable step size such that the magnitude of mutation tends to decay over time. This results in increasingly local search as
time progresses, allowing algorithms using such an operator to close in on optimal values [2]. Future work can test whether using arithmetic crossover and non-uniform mutation, as rega does, in a ring CGA, can further improve the performance of the ring CGA.

5. CONCLUSION

Cellular GAs are a popular solution to the scaling problems faced by Fine Grain PGAs. The toroidal grid structure of the canonical CGA reflects underlying architectures such as GPUs. However, CGAs with uni-directional ring topologies demonstrate faster convergence and a superior final solution compared to the canonical CGA on all benchmark functions. The canonical CGA with a population size of 16 was superior to, or competitive with, both its larger population counterparts, but population size has less effect on the ring CGA. We posit that rank selection should be replaced with a more elitist selection scheme to improve the performance of the canonical CGA by facilitating more rapid spread of high quality solutions through the population.

Additionally, we find that hill-climbing algorithms are robust and effective for solving some simple benchmark functions provided that the right step operator is chosen. The hill-climber exhibits rapid convergence and competitive final solutions for separable functions, even in higher dimensions. A standard, single-population GA implementation is surprisingly competitive with the CGAs, though it typically has slightly worse performance than the ring CGA. This suggests that the superior performance of parallel GAs, when run on sequential CPUs, may be overstated in the literature.

Though none of the algorithms presented were competitive with the best 2009 optimization algorithm, non-uniform mutation and arithmetic crossover could greatly improve CGA performance. Our results also show that ring CGAs perform better than the more common grid CGAs on these benchmarks.

6. ACKNOWLEDGEMENT

The authors would like to thank the Black Box Optimization Benchmarking team and the GECCO Workshop for Real-Parameter Optimization organizers for providing the benchmark suite and analysis tools that made this paper possible. This work is supported by DARPA CRASH P-1070-113237 and NSF EF 1038682.

7. REFERENCES

Figure 3: Expected running time (ERT in number of $f$-evaluations) divided by dimension for target function value $10^{-3}$ as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of $f_1$ and $f_2$. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Horizontal lines give linear scaling, slanted dotted lines give quadratic scaling. Black stars indicate statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: $\circ$:grid16, $\triangledown$:grid100, $\star$:hill, $\blacklozenge$:ring16, $\triangle$:ring100, $\diamond$:ga100
Figure 4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $[1^{18-2}]$ for all functions and subgroups in 5-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.
Figure 5: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{1-8.2}$ for all functions and subgroups in 20-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.
<table>
<thead>
<tr>
<th>Δ opt</th>
<th>f1</th>
<th>f2</th>
<th>f3</th>
<th>f4</th>
<th>f5</th>
<th>f6</th>
<th>f7</th>
<th>f8</th>
<th>f9</th>
<th>f10</th>
<th>f11</th>
<th>f12</th>
<th>f13</th>
<th>f14</th>
<th>f15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>4.0</td>
<td>5.0</td>
<td>6.0</td>
<td>7.0</td>
<td>8.0</td>
<td>9.0</td>
<td>10.0</td>
<td>11.0</td>
<td>12.0</td>
<td>13.0</td>
<td>14.0</td>
<td>15.0</td>
<td>16.0</td>
<td>17.0</td>
<td>18.0</td>
</tr>
</tbody>
</table>

Table 1: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 (given in the respective first row) for different Δ values in dimensions 5. The central 80% range divided by two is given in braces. The median number of conducted function evaluations is additionally given in italics, if ERT(10⁻⁷) = ∞.

#succ is the number of trials that reached the final target.
Table 2: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 (given in the respective first row) for different $\Delta f$ values in dimension 20. The central 80% range divided by two is given in braces. The median number of conducted function evaluations is additionally given in italics, if $ERT(10^{-7}) = \infty$. $\#succ$ is the number of trials that reached the final target $f_{opt}$ + $10^{-5}$. Best results are printed in bold.