The Evolution of Genetic Algorithms:
Towards Massive Parallelism

Shameet Baluja
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
baluja@cs.cmu.edu

Abstract

One of the issues in creating any search technique is balancing the need for diverse exploration with the desire for efficient focusing. This paper explores a genetic algorithm (GA) architecture which is more resilient to local optima than other recently introduced GA models. and which provides the ability to focus search quickly. The GA uses a fine-grain parallel architecture to simulate evolution more closely than previous models. In order to motivate the need for fine-grain parallelism, this paper will provide an overview of the two preceding phases of development: the traditional genetic algorithm, and the coarse-grain parallel GA. A test set of 15 problems is used to compare the effectiveness of a fine-grain parallel GA with that of a coarse-grain parallel GA.

1 INTRODUCTION

The effectiveness of heuristic techniques used in machine learning, search, and function optimization, resides in the heuristic's ability to balance the need for a diverse set of sampling points with the ability to focus search quickly upon potential solutions. Genetic algorithms (GAs) are general purpose optimization tools designed to search irregular. poorly characterized search spaces. GAs are based upon the ideas of natural selection and genetic recombination. GAs combine the principles of survival of the fittest with a randomized information exchange. Although the information exchange is randomized, GAs are far different than simple random walks. having the ability to recognize trends toward optimal solutions, and to exploit such information by guiding the search toward them.

Genetic algorithms have evolved through three phases, their development motivated by the goal of balancing exploration with focusing. Each of these phases will be individually discussed in the next three sections. The first phase presented the traditional genetic algorithm. A single population of potential solutions is evolved through a series of generations. The second phase introduced the concept of parallel subpopulations. Only a limited amount of swapping of potential solutions is allowed between subpopulations. Each subpopulation evolves independently, with swapping at infrequent intervals. This has been termed coarse-grain parallelism. or the "island" model. The third phase, an extension of the second, introduced fine-grain parallelism. The GAs in this category differ from the previous by relaxing the boundaries between subpopulations. Swapping between subpopulations occurs frequently, and significantly contributes to the effectiveness of this form of genetic algorithm. This phase also differs from the previous two by evolving numerous small subpopulations; in the previous two phases, only a few large subpopulations are evolved.

2 TRADITIONAL GAs

Traditional genetic algorithms maintain a single population of potential solutions for the objective function being optimized. Although a large portion of GA research has been conducted with potential solutions encoded in binary notation, any encoding scheme can be used. The initial group of potential solutions is randomly selected. These potential solutions, termed "chromosomes", are allowed to evolve over a number of generations. At every generation, the fitness of each potential solution in terms of the objective function is calculated, and pairs of solutions are recombined to create the subsequent generation. Recombination is the method by which the parent chromosomes of the current generation donate parts of their potential solutions to the "children" chromosomes which appear in the subsequent generation. The probability that a solution will participate in this recombination increases with its fitness. Thus, although "good" chromosomes are more likely to be

chosen for recombination, they are not guaranteed to be chosen. Further, the "children" chromosomes produced are not necessarily better than their parents. Nevertheless, because of the selective pressure applied through a number of generations, the overall trend is toward better chromosomes.

In order to perform expansive search. genetic diversity must be maintained. When diversity is lost. it is possible for the GA to settle into a local optimum. There are two fundamental mechanisms which the traditional GA uses to maintain diversity. The first, mentioned above, is a probabilistic scheme of selecting chromosomes for recombination. This insures that schemata, or common recurring patterns, other than those represented in the best chromosomes, appear in subsequent generations. Exclusively recombining good chromosomes will quickly converge the population without extensive exploration, thereby increasing the possibility of settling into a local optimum. The second mechanism, mutation, is a random change. For example, in binary encoded chromosomes, it is usually a random bit flip. In the traditional GA. the mutation *rate* is kept at a very small constant.

This algorithm is typically allowed to continue for an arbitrary number of generations. Upon completion, the best chromosome in the final population. or the best chromosome ever found, is returned.

Unlike the majority of other optimization heuristics, genetic algorithms do not work from a single point in the search space. Methods which only use a single point are susceptible to local optima. GAs continually maintain a population of points from which the search space is explored. This aids in searching multidimensional spaces, in which many variables must be optimized, and in locating global optima.

3 COARSE-GRAIN PARALLEL GAs

A coarse-grain parallel genetic algorithm (cgpGA) is based upon the theory of punctuated equilibria. In the paper Disrributed ***Genetic Algorithm for rhe Floor*** Plan ***Design Problem,*** Cohoon et. al. describe the theory [Cohoon, 1988]:

> Punctuated Equilibria is based upon two principles: allopatric speciation and stasis. Allopamc speciation involves the rapid evolution of new species after a small set of members of species. peripheral isolates. becomes segregated into a new environment. Stasis. or stability, of a species, is simply the notion of lack of change. It implies that after equilibria is reached in an environment, there is very little drift away from the genetic composition of species. Ideally, a species would persist until its environment changes (or the species would drift very little). Punctuated Equilibria stresses that a powerful method for generating new species is to thrust an old species into a new environment, where change is beneficial and rewarded. For this reason we should expect a genetic algorithm approach based upon punctuated equilibria to perform better than the typical single environment scheme.

By implication, after some period of evolution, a large portion of the chromosomes in a single population will represent very similar schema. The children chromosomes produced thereafter will be similar to each other and to their parents, thereby rendering recombination operators largely ineffective for further search space exploration, One method of resolving this problem is *to* partition a single large population into separate subpopulations, each evolving its chromosomes independently from others. The fitness used to determine the probability of selection for recombination is measured relative only to the other members within the subpopulation. Independent evolutions, in separate subpopulations, should yield closely competitive, yet possibly unique results. In the context of a single subpopulation, in order to continue evolution after convergence has started, members of species from outside subpopulations can periodically be introduced. To ensure thorough mixing of chromosomes throughout the population, the swapping does not always occur between the same subpopulations.

Although the sudden injection of new material is an important aspect of these simulations, it is not always effective. It is possible that the subpopulation into which new material is introduced is entirely settled into an quilibrium state. If this is the case. new infomation may not be incorporated because of its incompatibility with the existing information.

Despite the problems associated with the sudden introduction of new material, parallel subpopulations have proven their effectiveness in two areas. The first is, as mentioned above, to preserve diversity and to ensure perpetual novelty in the population's "gene pool". Through the use of parallel subpopulations, GAs have been able to solve problems which could not be solved in a reasonable amount of time by single population GAs [Whitley & Starkweather, 1990] [Liepins & Baluja, 1991] [Muhlenbien, 1989]. The second use of the subpopulation structure is to emphasize various characteristics in the chromosome. For example, in multi-objective functions, the evaluations in each subpopulation can be used to emphasize different objectives: When members of separate subpopulations are mixed, the genetic information may be combmed to reveal chromosomes which are strong with respect to more than a single objective. An exploration of multi-objective function optimization with parallel subpopulations can be found in [Husbands. 1991). *An* interesting method of multi-objective optimization using only a single population with multiple fitness measures can be found in [Schaffer & Grefensteae. 19851.

4 FINE-GRAIN PARALLEL GAs

4.1 OVERVIEW & MOTIVATION

Unlike cgpGAs, the fine-grain parallel genetic algorithms (fgpGAs) examined here evolve numerous small, constantly interacting subpopulations. Variants of fgpGAs have been explored by [Davidor, 1991], [Spiessens and Manderick, 1991], [Muhlenbein, 1989] and [Schleuter, 19901. One way in which to view the modified form of parallelism in fgpGAs is to conceptualize the populations as overlapping, as shown in Figure 1.

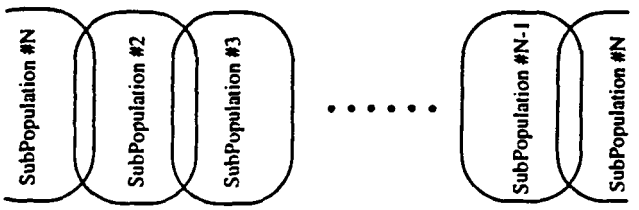

Figure 1: Overlapping Populations in a Fine Grain Parallel Genetic Algorithm.

The fine-grain population architecture offers three benefits which the other two models do not. First, the transfer of genetic information from one population to another is inherent in the architecture. In the traditional GA, there is no flow of information because there is only one population. In the cgpGA, the information is shared between subpopulations by swapping potential solutions at infrequent intervals. As pointed out earlier. the effectiveness of sudden infrequent swapping between subpopulations is limited because each subpopulation has the potential to evolve to incompatible local optima. The assimilation of genetic information across subpopulations is easier in the fine-grain parallel architecture as the exchange of information between subpopulations is continuous. A subpopulation cannot exist in a state of equilibrium until each subpopulation reaches equilibrium.

Second, the fine-grain model provides a more accurate representation of evolution. Instead of maintaining a large population from which any two chromosomes may recombine. recombination must be between two chromosomes from within the same neighborhood. This feature reveals a similarity to biological natural selection. in which a population is typically composed of relatively independent subpopulations which interact. Although the similarity with natural selection may not be important directly within the context GAs are used, the constant interaction can significantly help subpopulations escape from local minima.

The third benefit is a gain of speed. In the traditional genetic algorithm each potential solution must be assigned a fitness value with respect to all other potential solutions within the current population. This value is used to determine probabilistically which chromosomes will be chosen for recombination. Global ranking slows the GA considerably. Although the cgpGA does not require a global ranking of each potential solution, it requires a ranking within each individual subpopulation. In the fine-grain parallel model. the ranking required is local; ranking is relative to the other chromosomes within the neighborhood. As the population and neighborhood size can be kept small. the time required to perform local ranking is not as severe as in the two earlier models.

The fine-grain parallel organization and the cgpGA organization share the advantages of the model presented in the theory of punctuated equilibria: subpopulations which are a large distance apart will evolve comparatively unique chromosomes in a manner similar to simple. disjoint, parallel subpopulations. However, the fgpGA's higher connectivity allows all subpopulations within a close proximity to each other to have a greater influence on each other than those a large distance apart. As with cgpGAs, the larger the number of subpopulations. the greater the potential diversity in individual evolutions.

The danger of reaching a suboptimal state in fine-grain parallel genetic algorithms can be greater than that in cgpGAs for two reasons. First, the fgpGA structure employs a greater degree of swapping between subpopulations than the cgpGA's. Therefore, strong local optima can quickly spread throughout the entire population. Second, because there are significantly fewer chromosomes per subpopulation in the fgpGA model than in the cgpGA model, the diversity of infomation within subpopulations is more limited. In order to address this problem, fgpGAs rely upon the size of the group of genetic algorithms and the controlled degree of overlap to allow unique evolutions in different portions of the total population.

One question which must be answered under the fgpGA model is the extent and nature of overlap between subpopulations [Muhlenbein, 1989] [Schleuter, 1990] [Baluja, 19921. If the overlap spans many subpopulations. good chromosomes could rapidly flow throughout the GA structure. However. with a fast flow, the advantages of punctuated equilibria may be lost, and niche formation may be hindered. Many different issues regarding topology need to be addressed, such as whether the subpopulations should be connected in a linear manner or whether the overlapping subpopulations should be virtual, almost simulating neural network connections. Furthermore, should the connections between subpopulations be fixed, or time-varying? These topics are among those to be addressed by future research within this area.

4.2 fgpGA: POPULATION ARCHITECTURE

As described in the previous section. there are many possible population topologies. The fgpGA examined throughout the remainder of this paper uses a two dimensional toroidal array of subpopulations. See Figure 2. Each sub-

population evolves only 2 chromosomes per generation. The 2 parent chromosomes are chosen from a group of 10 chromosomes. The group of 10 is comprised of 1 chromosome from each of the 8 surrounding neighbors and the 2 chromosomes which were evolved in the previous generation. The chromosome selected from a neighbor is chosen randomly from the 2 evolved at the neighbor. The fitness of each chromosome is determined relative to the other chromosomes in the group of 10. Two chromosomes from this set of 10 are probabilistically chosen for recombination based upon their fitness; the other 8 are discarded. In the subsequent generation, the 2 "children" chromosomes produced, through crossover and mutation (described in the next section) of the selected parents, are available for recombination. either within the population in which they are located, or by its neighbors.

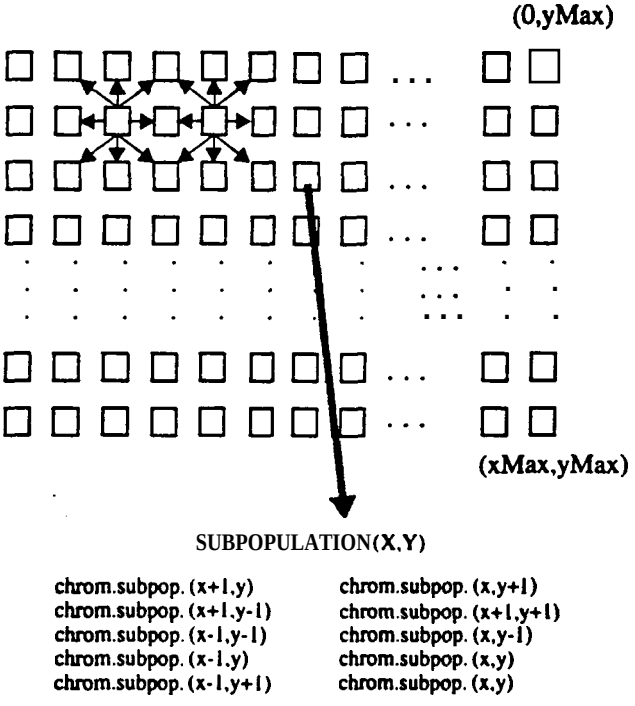

Figure 2: Each subpopulation contributes one of its two chromosomes to each of its 8 nearest neighbors. The composition of subpopulation (x,y) is shown. Both of the chromosomes evolved at population(x,y) arc included. The subpopulations form a toroid.

4.3 fgpGA: IMPLEMENTATION SPECIFICS

This section describes the recornbination. mutation, and elitist selection mechanisms used. The recombination operator used for testing this algorithm is a standard two point crossover, in which a randomly chosen subsection of one parent chromosome is swapped with the corresponding subsection of the second parent chromosome. This is a general crossover operator that has been shown to be effective in GA literature. For a discussion of the merits and drawbacks of two point crossover, the reader is referred to [DeJong, 19901 [Syswerda 1990]. Mutation is implemented as a simple random bit flip which occurs in the "children" chromosomes after the crossover has taken place with the parents, and before the chromosome is evaluated. The mutation rate is kept at a constant 1%.

Elitist selection is a commonly employed tool to ensure that the progress made by a GA is not lost due to random chance. Because a GA's selection of parent chromosomes is probabilistic, it is not guaranteed that the best chromosome in a particular generation will survive to the subsequent generation. It is also possible that if the chromosome is selected for recombination. some of the good genetic material may not survive through the crossover and mutation operators. A modest form of elitist selection is used to address this problem. Elitist selection carries the best chromosome, from the population of 10 candidates for recombination, from generation g to g+1. This does not imply that the best chromosome will be selected for recombination; rather, it means that the chromosome will be in the population of 10 which are candidates for recombination. Since the population size is limited to 10, the best chromosome from the previous generation replaces the chromosome with the lowest relative fitness in the current generation. Although this aids in preventing repetitive search, it may be detrimental when the GA is caught in a local optima, as elitist selection may preserve the local optima in the population's candidates for recombination.

5 FINE VS. COARSE-GRAIN PARALLELISM

The remainder of this paper is devoted to comparing fine-grain and coarse-grain parallel GAs. This paper does not compare results with traditional GAs since a large amount of work has already been conducted towards quantifying the differences in structure and performance between cgpGAs and traditional GAs [Pettey et. al. 1987] [Cohoon et. al. 19881 [Whitley & Starkweather, 1990] [Tanese, 19871.

5.1 ALGORITHMS TESTED

Two GAs were tested, the fgpGA described in the previous section and the cgpGA described below. The cgpGA was very loosely based upon the cgpGA described in (Whitley & Starkweather, 1990]. Two point crossover and a 1% mutation rate were used. Like the fgpGA described above, the cgpGA employed modest elitist selection, in which the single best chromosome from each generation was carried to the subsequent generation. The best chromosome replaced the worst chromosome in the subsequent generation. Forty subpopulations were evolved. Each sub population contained 100 chromosomes. for a total of 4000 chromosomes evaluated simultaneously.

The cgpGA implemented a small amount of communication between the subpopulations. Assuming a circular ordering of subpopulations, after every 100 generations, the best chromosome from each subpopulation migrated to a subpopulation e subpopulations away, where e is defined to be the number of generations divided by 100 that have passed. Because the population was a set size, the migrating chromosome replaced the worst chromosome in the target subpopulation.

Although the parameters for both the cgpGA and the fgpGA were not optimized for any single problem, they were chosen to work satisfactorily on a variety of different problems. When comparing results, it should be considered that the fgpGA evaluated 8192 chromosomes per generation, and the cgpGA evaluated 4000. The discrepancies in the number of evaluations per generation can be attributed to the mapping of the algorithm onto the hardware architecture on which these tests were attempted.

5.2 TEST PROBLEMS

This section presents the test problems which are used to compare the effectiveness of fine and coarse-grain parallel genetic algorithms. Fifteen test problems were attempted: DeJong's five function test suite, three subset-sum problems, two orderings of fully and partially deceptive order 4 problems, and three versions of all-ones problems.

5.2.1 DeJong's Test Suite

DeJong's test suite is comprised of five minimization problems commonly used to test the effectiveness of GAs [DeJong, 19751. The test suite was designed to incorporate functions with the following characteristics: continuous/discontinuous, convex/nonconvex, unimodal/multimodal, quadratic/nonquadratic, low dimensionality/high dimensionality, and deterministic/stochastic [Goldberg, 19891. The functions were encoded in standard binary notation.

5.22 Subset-Sum

The problems can be stated as follows: given S elements, each of a possibly unique weight, is there a subset of S that adds up exactly to an arbitrary number, T. The subset sum problems are NP-complete. This problem was implemented as a 120 bit chromosome. Each bit represented a unique object, with weight randomly assigned between 1 & 200. The weight T was selected to be either 1/4, 1/20, or 1/40 of the sum of the weights of the objects. The only addition to the problem was that the sum of the weights was guaranteed to be divisible by 4.20 & 40, respectively.

5.2.3 Fully and Partially Deceptive Order 4

The fully deceptive problem is a 40 bit maximization problem. The problem was defined in Whitley & Starkweather's paper GENITOR II [Whitley & Starkweather, 1990]. The problem is comprised of 10 subproblems, each 4 bits long. The subproblems use the lookup table shown in Table 1. The partially deceptive problem uses the same evaluations. with the exceptions of the evaluations corresponding to 1111 and 0101, which are reversed.

Table 1: Order 4 Fully Deceptive Problem

Chrom.	Eval.	Chrom.	Eval.
1111	30	0110	14
0000	28	1001	12
0001	26	1010	10
0010	24	1100	08
0100	22	1110	06
1000	20	1101	04
0011	18	1011	02
0101	16	0111	00

Both the fully deceptive and partially deceptive problems were attempted using two orderings of bits. The first encoding is block encoding; the placement of the 4 bits which comprise a subproblem are located next to each other. The second encoding is interleaved; the bits which comprise each subproblem are uniformly spread throughout the chromosome. With the use of two point crossover, the first encoding is much easier for the GA to solve than the second. The encodings are shown in Figure 3.

Block Encoding: aaaabbbbccccddddeeeeffffgggghhhhiiiijjjj

Interleaved: abcdefghijabcdefghijabcdefghijabcdefghij

Figure 3: Two encodings of the order 4 deceptive problem & partially deceptive problems.

5.24 Three "All-Ones" Problems

Three versions of the all-ones problem were tried [Syswerda, 19901. The first version was the straight all-ones problem. The objective of this problem is to find the chromosome which contains a 1 in each bit position. .

The second version of the all-ones problem contains bits which are meaningless. This problem was encoded as a 180 bit problem, but only the first 120 bits were counted toward the evaluation.

The third all-ones problem is the contiguous bits problem. In evaluating the chromosome, points are only given for 1s which also have at least one other neighbor which has a value of 1. If there exists a 1 which has zeros as its two neighbors, no points are given for the bit.

5.3 RESULTS & DISCUSSION

The results are shown for the 15 test problems in Table 2. They are the average of 10 runs per problem for each algorithm. Each run was started with a different randomly chosen initial population of chromosomes. The maximum number of allowed generations for the cgpGA is 3000; after 3000 the attempt is considered a failure. The maximum number of generations before failure for the fgpGA is 1400.

For many problems, especially the all-ones problem, on which the cgpGA did comparatively poorly, cgpGAs can do significantly better if the population size, mutation rate, and swap interval parameters are tuned for the problem. For example, because the all-ones problem is relatively "easy" for the GA to solve, it can be solved very efficiently using a population size of 10 rather than 100. However, to measure the ability of the algorithms to perform on a variety of problems without parameter tuning, the parameters were held constant throughout all of the test runs.

Table 2: Results for the 15 test problems. Each entry represents the average number of generations before the optimal solution was found. The fgpGA evaluated 8192 chromosomes per generation, the cgpGA 4000.

Test Function	Size (Bits)	fgpGA	cgpGA 40 subpop.
DeJong Function #1	30	29.8	79.0
DeJong Function #2	24	38.6	111.8
DeJong Function #3*	50	19.5	64.5
DeJong Function #4	240	See Figure 4	
DeJong Function #5**	34	18.0	18.0
Subset Sum (1/4)	120	14.0	35.6
Subset Sum (1/20)	120	55.0	344.5
Subset Sum (1/40)	120	76.8	629.0
Partially Deceptive (A)	40	32.0	95.1
Partially Deceptive (B)	40	53.0	252.5
Fully Deceptive (A)	40	57.5	305.9
Fully Deceptive (B)	40	742.5	1634.7
All-Ones	120	90.5	648.2
Sparse All-Ones***	180	94.8	342.0
Contiguous All-Ones	120	90.8	609.1

* The stopping criterion for DeJong's F3 was an evaluation of -30.

** The stopping criterion for DeJong's F5 was an evaluation of 0.998004.

*** Due to memory restrictions, this problem was attempted with 90 significant bits, and 30 extra bits (cgpGA only). The fgpGA was attempted with 120 significant bits and 60 extra.

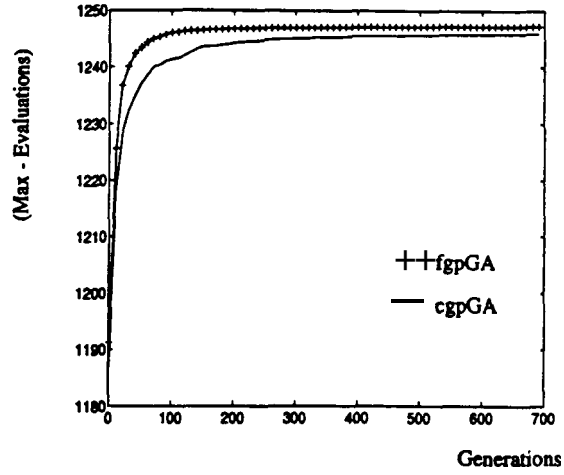

Figure 4: Average evaluations for 10 runs of the cgpGA and fgpGA on DeJong's F4. Due to memory restrictions, the cgpGA was run with 80 subpopulations with 50 chromosomes per population. F4 is shown graphically to aid in quantifying performance in the presence of the random gaussian factor [DeJong, 1975], [Ingber, 1992]. Note that generations 200-700 yield only very small improvements. (Samples taken at every 10 generations.)

One of the reasons the fgpGA performed better than the cgpGA is the ability of good chromosomes to spread rapidly through the population. A large portion of the population has access to the best chromosome very shortly after it is found. A sample run, shown in Figure 5, displays the number of populations which "have seen" the best chromosome found in each generation. The term "has seen" does not imply that the populations have selected the chromosome for recombination, rather that the chromosome is a candidate for selection. The sudden drops of the number of populations, in Figure 5, represent generations in which a better chromosome is found.

The fgpGA implementation allows a good chromosome to be accessed immediately by the 8 surrounding subpopulations. In order for more than the original 9 populations to incorporate the chromosome, it must again be selected for recombination. Assuming that it is selected, valuable schemata must not be destroyed by crossover or mutation operators. Although the populations which surround the original 9 will incorporate the resultant chromosomes into their population, for it to spread further, they must also select them for recombination based upon their evaluation, which may not be as good as their parents. Further, if the crossover and mutation operations have destroyed valuable schemata, the children produced may not be preserved by elitist selection. If the important schemata are a small part of the total chromosome, the chances of the chromosomes spreading throughout the population with the schemata intact is much greater, since in this case,

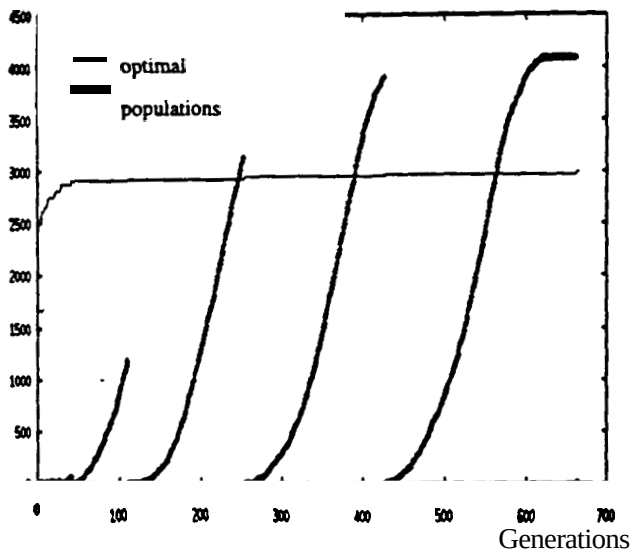

Figure *5:* The Number of populations which contain the best chromosome using the fgpGA to optimize the order 4 fully deceptive problem, interleaved. Also plotted is the best value • 10, for comparison. The sudden drops in the number of populations correspond to a new best solution found in one of the populations.

crossover and mutation operators have a smaller chance of destroying valuable schemata.

The choice of how subpopulations should overlap plays a significant role in how fast the chromosomes spread through the subpopulations. For certain classes of problems, it may be important to ensure that the flow of chromosomes is slow. in order *to* allow for extremely different evaluations in different portions of the model. However, in other applications. in which the search space is not deceptive and does not contain many local minima, a fast flow may yield good answers quickly, as good chromosomes can rapidly dominate the "gene pool".

6 SUMMARY & FUTURE DIRECTIONS

Preliminary results on fifteen test problems have shown the fgpGA to be able to solve problems more efficiently than a cgpGA. To evaluate fine-grain parallelism further, both harder test problems and different population topologies should be explored.

One of the difficulties inherent in comparing parallel genetic algorithms with each other. and with traditional GAs, is choosing the best criteria. Although the number of evaluations is commonly used, parallel evolutions show extensive overlap in the evaluations performed. For problems which are "easy" for the GA to solve, parallel subpopulations may perform too broad a search, as the search performed by a traditional. single population GA may be enough. However, on harder problems. it often happens that parallel populations find solutions both more often and faster than traditional GAs. Another measure, generations to find optimal solutions, has a bias in favor of parallelism because more chromosomes are evaluated per generation. Another candidate criterion is cpu-speed; however, implementations of different algorithms may use the same hardware with varying degrees of effectiveness. In this paper, both evaluations and generations were presented. Although the results were in favor of the fgpGA. the parameters in the cgpGA and the fgpGA were not tuned per problem. As stated earlier. it is suspected that with a little tuning, both GAs could improve performance.

Perhaps the most interesting topic for future research is the need to design subpopulation topologies which maximally exploit the parallelism inherent in these GAs. Two interesting *structures* to examine in the future would be one in which each population is only connected to 1 of its nearest neighbors and a second in which the connections are made randomly, perhaps with a set maximum reaching distance.

One of the important applications of parallel genetic algorithms, which was not explored here, is the use of sets of populations to optimize different objectives in multi-objective functions. For this type of problem, each sub population evolves under the pressures of individual subgoals of the complete problem. As stated in [Husbands, 19911 "...the solution to a complex problem is allowed to emerge from the simultaneous solution of a number of simpler, related subproblems. Using this variation of divide and conquer, the inherent parallelism in a problem is brought out and thoroughly exploited." This method is directly applicable to the fgpGA described here: individual populations can work towards individual sub-goals. It will be interesting to determine the role that the position of sub-goal assignment to individual populations has on the overall effectiveness of the GA.

Acknowledgments

I would like to thank Dean Pomerleau, Chuck Thorpe, Stephen Smith. Dayne Freitag. and Todd Jochem for their helpful comments and suggestions throughout the development of this paper.

This research was partly sponsored by Defense Advanced Research Projects Agency, under contracts "Perception for Outdoor Navigation" (contract number DACA76-89-C-0014, monitored by the US Army Topographic Engineering Center) and "Unmanned Ground Vehicle System" (contract number DAAE07-90-C-R059, monitored by TACOM). It was also partially sponsored by the National Science Foundation, under NSF Contract BCS-9120655, titled 'Annotated Maps for Autonomous Underwater Vehicles", and the NSF grant titled "Massively Parallel Real-Time Computer Vision". The views and conclusions contained in this document are those of the author and

should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the National Science Foundation, or the U.S. Government.

References

Baluja, S. (1992) A Massively Distributed Parallel Genetic Algorithm. CMU-CS-92-196R. School of Computer Science, Carnegie Mellon University.

Caruana, R. and J. Schaffer (1988) Representation and Hidden Bias: Gray Vs. Binary Coding for Genetic Algorithms. *Proceedings of the 5th International Conference on Machine Learning.* Morgan Kaufmann. Los Altos. CA. June 1988 152-161

Cobb, H. (1990) An Investigation Into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having continuous. Time Dependent Nonstationary Environments. NCARAI Library. AIC-90-001.

Cohoon, J.P, S.U. Hedge, W.N. Martin & D. Richards (1988), Distributed Genetic Algorithms for the Floor Plan Design Problem. Technical Report TR-88-12. School of Engineering and Applied Science. Computer Science Department, University of Virginia.

Davidor, Y (1991) A Naturally Occurring Niche & Species Phenomenon: The Model and First Results. *Proceedings of the Fourth International Conference on Genetic Algorithms.* Morgan Kaufmann. San Mateo. CA.

DeJong, K.A. (1975) *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* (Doctoral dissertation, University of Michigan). Dissertation Abstracts International 36-10, 5140B.

DeJong, K.A. and W. Spears (1990) An Analysis of Multi-Point Crossover. NCARAI Library. AIC-90-014.

Eshelman, L. (1990). The CHC Adaptive Search Algorithm: How to have safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorirhms,* Bloomington, IN.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Husbands, P., F. Mill & S.Warrington (1991). Genetic Algorithms, Production Plan Optimisation and Scheduling. *Parallel Problem Solving from Nature,* H.P. Schwefel & R. Manner Eds. Springer Verlag. Berlin.

Ingber, L and B. Rosen (1992) Genetic Algorithms and Very Fast Simulated Reannealing: A comparison. To be published in *Mathematical* and *Computer Modelling.*

Liepins. G.E. and S. Baluja (1991) apGA: an Adaptive Parallel Genetic Algorithm. *Computer Science* and *Operations Research, New Developments in Their Interfaces.* Balci, Sharda & Zenios eds. Pergamon Press.

Liepins, G. E. and M. D. Vose (1990). Representational Issues in Genetic Optimization, *Journal Expt. Theor. Artificial Intelligence.,* 2. 101-115

Muhlenbein. H (1989) Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. *Proceedings of the Third International Conference on Genetic Algorithm.* Morgan Kaufmann. San Mateo, CA.

Petty, C, M. Leuze, J. Grefenstette (1987) A Parallel Genetic Algorithm, *Proceedings of the Second International Conference on Genetic Algorithms.* Lawrence Erlbaum Associates. NJ.

Schaffer, J.D. and J.J. Grefenstette (1985) Multi-Objective Learning Via Genetic Algorithms, *Proceedings of Ninth International Joint Conference on Artificial Intelligence.*

Schaffer, J.D., R.A. Caruana, L.J. Eshelman, and R. Das (1989). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization, *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo. CA.

Schleuter, M.G. (1990), Explicit Parallelism of Genetic Algorithms through Population Structures. *Parallel Problem Solving from Nature,* H.P. Schwefel & R. Manner Eds. Springer Verlag. Berlin.

Spiessens, P. and B. Manderick (1991). A Massively Parallel Genetic Algorithm: Implementation and First Results. *Proceedings of the Fourth International Conference on Generic Algorithms.* Morgan Kaufmann. San Mateo, CA.

Syswerda, G. (1989) Uniform Crossover in Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann. San Mateo, CA.

Tanese, R. (1987) A Parallel Genetic Algorithm for a Hypercube. *Pmceedings of the Second International Conference on Genetic Algorithms.* Lawrence Erlbaum Associates. NJ.

Whitley, D. and T. Starkweather (1990). GENITOR II: a Distributed Genetic Algorithm, *Journal Expt. Theor. Artificial Intelligence,* 2, 189-214.