Chapter 6

Shumeet Baluja
School of Computer Science
Carnegie Mellon University
Pittsburgh
Pennsylvania 15213-3890

baluja@cs.cmu.edu

Structure and Performance of Fine-Grain Parallelism in Genetic Search

Abstract

Within the parallel genetic algorithm framework, there currently exists a growing dichotomy between coarse-grain and fine-grain parallel architectures. This chapter attempts to characterize the need for fine-grain parallelism, and to introduce and compare three models of fine-grain parallel genetic algorithms (GAs). The performance of the three models is examined on seventeen test problems and is compared to the performance of a coarse-grain parallel GA. Preliminary results indicate that the massive distribution of the fine-grain parallel GA and the modified population topology yield improvements in speed and in the number of evaluations required to find global optima.

6.1 Introduction

Since Holland's pioneering work [Holland, 1975], there have been many variations of the simple genetic algorithm. The development of genetic algorithms has been driven by the goal of maintaining the balance of diverse sampling and efficient focusing. With regards to parallelism, there have been two stages of development beyond the genetic algorithms (GAs) proposed by Holland. The first is the coarse-grain parallel genetic algorithm, in which several large populations are evolved in parallel with very little interaction. The second stage is the fine-grain parallel genetic algorithm, in which numerous small, constantly interacting populations are evolved in parallel.

6.1.1 The Motivation Behind Parallelism

Explained simply, a parallel genetic algorithm (pGA) divides a single large population into smaller subpopulations. Each of the subpopulations runs a separate genetic algorithm either independently or with limited interactions with other subpopulations. One motivation for this division is the potential increase in speed through the assignment of each processor, of a multi-processor system, to evolve a single population. However, a more interesting motivation stems from the observation that, after some period of evolution, the majority of the chromosomes in a single population will become very similar. Genetic diversity will be lost, and recombination thereafter may not be productive. One method of addressing this problem is to evolve subpopulations independently. As GAs are randomized algorithms, independent evolutions are likely to explore different portions of the search space. If the functions to be optimized for each subpopulation are the same, each subpopulation should reveal closely competitive, yet unique results.

The amount of interaction between subpopulations can be a critical factor in determining a pGA's effectiveness. Eliminating interaction between subpopulations effectively makes dividing a larger population similar to performing several GA runs with smaller populations. With too much interaction, the benefits of subpopulations are lost. Good chromosomes from one subpopulation quickly spread to other subpopulations, and the evolutions no longer remain independent. Cohoon et al. suggest that members of subpopulations be swapped after the subpopulations begin to reach equilibrium [Cohoon, 1988]. The results of Whitley and Starkweather, Tanese, and Grosso, also support limited interactions [Whitley and Starkweather, 1990] [Tanese, 1989] [Grosso, 1985].

Although some of the pGAs differ in many parameter settings, [Whitley and Starkweather, 1990] [Pettey, 1989] [Cohoon, 1988], several important factors remain consistent in the majority of them: they evolve a relatively small number of subpopulations, and each subpopulation contains a large number of chromosomes. The pGAs described to this point are referred to as coarse-grain parallel genetic algorithms (cgpGAs). One of the drawbacks of cgpGAs is that after a subpopulation converges to an equilibrium state, the introduction of new material may not be effective. The new material may not be incorporated because of its incompatibility with the existing information. A reason for incompatibility may be as simple as two subpopulations may evolve answers to opposite sides of a large hamming cliff, or in more general terms, that two subpopulations may find good solutions which, when combined, reveal a worse solution.

6.1.2 Massive Parallelism

Massive Parallelism in genetic algorithms has been used in at least two different contexts. In the first context, parallelism refers to the machine architecture on which the GA is run. Parallelism is employed to achieve a gain in speed, and to allow much larger population sizes to be evolved in reasonable amounts of time [Forrest and Perelson, 1990]. The second context, and the one explored throughout the remainder of this paper, is one in which numerous small, constantly interacting subpopulations are evolved in parallel, with localized mating rules. Another related application of massive parallelism can be found in [Hillis, 1990]. Hillis used a massively parallel architecture to co-evolve parasites with chromosomes.

Fine-grain parallel genetic algorithms (fgpGAs) addressed some of the problems found in cgpGAs. One way to conceptualize the modified form of parallelism is to view the populations as overlapping, with a portion of the constituents of one population also being constituents of one or more other subpopulations. In an analogous manner to biological natural selection, in which a population is typically composed of relatively independent subpopulations which interact, recombination occurs between two chromosomes from within localized neighborhoods [Davidor, 1991] [Spiessens and Manderick, 1991] [Muhlenbein, 1989] [Schleuter, 1990]. The constant interaction between subpopulations helps to alleviate the problems of recombining incompatible solutions. It is difficult for a subpopulation to exist in a state of equilibrium until all of its neighboring subpopulations reach equilibrium.

A potential drawback of the fgpGA architecture is that local optima can quickly spread through the entire population. Since there is constant swapping between subpopulations, the possibility of independent evolutions may be hindered. Further, because the size of subpopulations is small, the schemata represented in strong local optima can quickly dominate all of the genetic information in individual subpopulations. In practice, this problem is partially overcome by limiting the amount of swapping between subpopulations. Another factor which can reduce the detrimental effects of constant swapping is the large number of subpopulations. Subpopulations which are a large distance apart may evolve unique chromosomes in a manner similar to cgpGAs. This has been termed isolation by distance [Collins and Jefferson, 1991]. In the next section, three fgpGA structures are examined which vary with respect to how swapping between subpopulations is implemented. The effects of the speed of information flow, which is dependent upon the amount of interaction between subpopulations, will be discussed throughout the remainder of this paper.

6.2 Three Fine-Grain Parallel GA Topologies

Using the fine-grain parallel subpopulation structure, three topologies were examined. The first implementation uses a circularly linked linear ordering of subpopulations. Each subpopulation evolves only 2 chromosomes per generation. These 2 chromosomes are chosen from a group of 10 chromosomes. The group of 10 is comprised of 1 chromosome from each of the four immediate left and 1 chromosome from each of the four immediate right subpopulations, and the 2 chromosomes which were evolved in the subpopulation during the previous generation. See Figure 6.1. Each chromosome selected from the neighbors is chosen *randomly* from the two evolved at each neighbor. The fitness of every chromosome is calculated relative to the other chromosomes in the group of 10. Two chromosomes from the set of 10 are probabilistically chosen for recombination, based upon their relative fitness. The other 8 chromosomes are discarded. In the subsequent generation, the two "children" chromosomes produced (through crossover and mutation of the selected parents) are available for recombination, either by the subpopulation on which they are located, or by its neighbors.

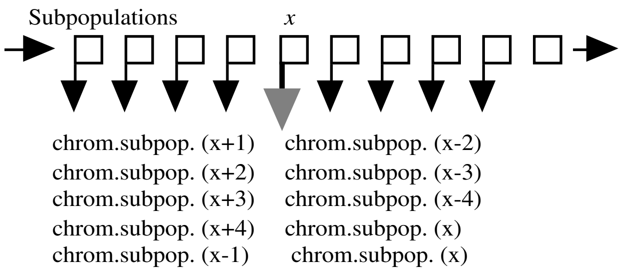
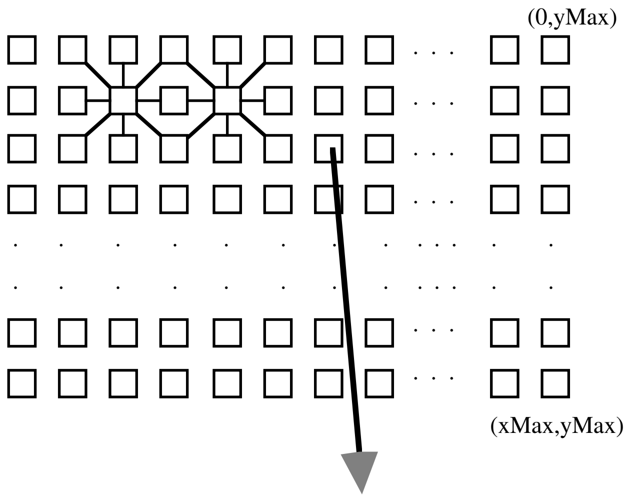


Figure 6.1: The architecture of subpopulations, arrangement #1. Subpopulation *x* shown enlarged. The subpopulations form a circular list.

The second implementation uses a two dimensional toroidal array of subpopulations. As in the previous implementation, each of the subpopulations evolve 2 chromosomes which are chosen from a group of 10. The 8 immediate neighboring subpopulations donate to the group of 10. See Figure 6.2. The remainder of the procedure is completed in the same manner as described above.

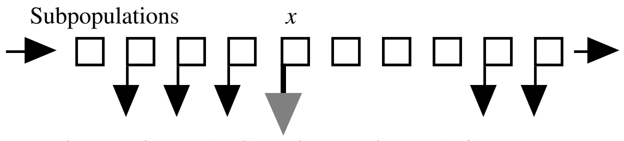
In the third implementation, a linear ordering of subpopulations is used once again. One chromosome from each of the 3 immediate left and one chromosome from the 4, 5, 6 subpopulations from the right contribute to the group of 10 chromosomes. See Figures 6.3 and 6.4. Figure 6.4 gives a pictorial example of how the genetic information flows through the series of GAs. Unlike the first implementation, the immediate right subpopulations are not used. Since there are 4 remaining positions in the group of 10, they are filled with 2 copies of each of the chromosomes evolved in the previous generation. This structure includes only 6 neighbours to examine the effects of reducing the spread rate of chromosomes and introducing a bias to the chromosomes evolved within each subpopulation. After the selection of 10 chromosomes, the fgpGA proceeds in the same manner as described earlier. This model was chosen because it allows a faster spread of chromosomes than the first implementation, and a slower rate than the second implementation.



SUBPOPULATION (X,Y)

chrom.subpop. (x+1,y)	chrom.subpop. (x,y+1)
chrom.subpop. (x+1,y-1)	chrom.subpop. (x+1,y+1)
chrom.subpop. (x-1,y-1)	chrom.subpop. (x,y-1)
chrom.subpop. (x-1,y)	chrom.subpop. (x,y)
chrom.subpop. (x-1,y+1)	chrom.subpop. (x,y)

Figure 6.2: 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 subpopulation(x,y) are included. The subpopulations form a toroid.



chrom.subpop. (x+4)	chrom.subpop. (x-3)
chrom.subpop. (x+5)	chrom.subpop. (x)
chrom.subpop. (x+6)	chrom.subpop. (x)
chrom.subpop. (x-1)	chrom.subpop. (x)
chrom.subpop. (x-2)	chrom.subpop. (x)

Figure 6.3: The architecture of subpopulations arrangement #3, subpopulation *x* shown enlarged. The subpopulations form a circular list.

Making the large assumption that the best chromosomes are not lost during crossover or mutation, the maximum spread rate of a good chromosome varies per implementation. In arrangement #1, the linear ordering, assuming 4096 subpopulations, the lower bound on the number of generations for all of the subpopulations to see the best chromosome is 512-1, or 511 generations. In arrangement #2, the toroid, the minimum number of generations to get from any subpopulation to the furthest away, is half the diagonal of the square. Assuming a 64 * 64 toroid, within approximately 31 (32-1) generations, the chromosome could be in all of the subpopulations. Using arrangement #3, a compromise between the first two with regard to speed, the number of generations is approximately 455. All of the -1 factors arise since in the first generation a good chromosome is found, it can be included in its neighbor's selection of 10 chromosomes. These estimates are only the lower bound of the spread of the best chromosome. The chromosomes will not generally spread this fast. For these speeds to be achieved, the chromosome must be reselected for recombination at each generation, and none of the valuable schemata can be destroyed by crossover and mutation. Further, this also assumes that no better chromosomes are found before full spreading. In the experiments performed, the chromosome in the neighboring subpopulations was selected randomly from the two which were evolved in the neighbor. However, always selecting the best, or using a probabilistic scheme of selection may also work well. Another topology, termed the "ladder" population structure, has been explored by Muhlenbein and Schleuter [Muhlenbein, l989] [Schleuter, 1990].

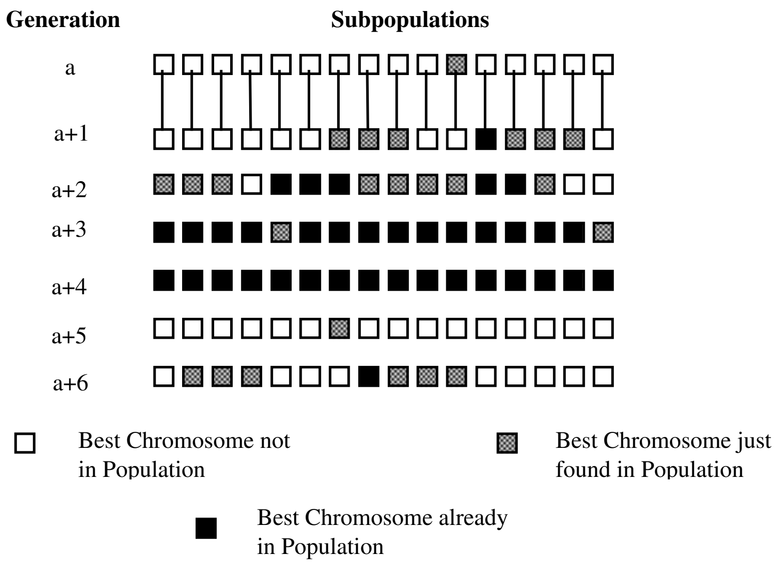

Generation
Subpopulations
a
a+1
a+2
a+3
a+4
a+5
a+6
Best Chromosome not in Population
Best Chromosome just found in Population
Best Chromosome already in Population

Figure 6.4: Making the large assumption that the best chromosome is not lost during crossover or mutation, the above diagram depicts how the best chromosome could spread through the populations, using fgpGA configuration #3. In generation a+5, a new best chromosome is found. The connections shown are the subpopulations from which a chromosome is included. These connections do not change through the run of the GA. Each subpopulation has similar connections to those shown.

6.3 Performance of fgpGAs and cgpGAs

6.3.1 Description of Algorithms Compared

Four algorithms were tested, the three implementations of fgpGAs described in the previous section, each with 4096 subpopulations, and a 40 subpopulation cgpGA described in this section [Baluja, 1993]. All used a constant 1% mutation rate and two point crossover. The evolution was generational, and crossover took place with each set of parents. An alternative to the generational model, the steady state model, is explored in [Whitley and Starkweather, 1990]. All algorithms also used a modest form of elitist selection, in which the single best chromosome in generation *a* replaced the worst chromosome in generation *a+1*. Elitist selection was performed within each subpopulation. In the fgpGAs, the best chromosome was selected from each group of 10 chromosomes, and replaced the worst chromosome from the group of 10 in the next generation. Elitist selection does not ensure that a particular chromosome will be selected for recombination, only that it will be a candidate for selection.

The cgpGA was very loosely based upon the cgpGA described in [Whitley and Starkweather, 1990]. Forty subpopulations were evolved. Each subpopulation contained 100 chromosomes, for a total of 4000 chromosomes evaluated simultaneously. 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.

In an attempt to efficiently map these algorithms onto the hardware architecture on which these tests were attempted, the MasPar MP-I, the fgpGA evaluated 8192 chromosomes per generation, while the cgpGA evaluated only 4000.

6.3.2 The Problems Attempted

DeJong's Test Suite: This test suite is comprised of five minimization problems commonly used to test the effectiveness of GAs [DeJong, 1975]. The functions were encoded using standard binary code.

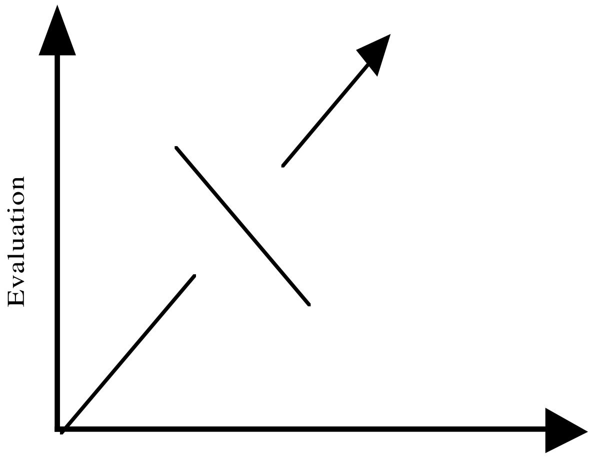
Subset Sum: The problem can be stated as follows: given S elements, each of a possibly unique weight, is there a subset of S that adds up exactly to an arbitrary number, T? This problem was implemented as a 120-bit chromosome. Each bit represented a unique object, assigned a random integer weight between 1 and 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 object was to find the group of sets whose weights add exactly to T. The sum was guaranteed to be divisible by 4, 20 and 40, respectively; note that this does not guarantee a set of weight T.

All Ones: Three versions of the all-ones problem were attempted [Syswerda, 1989]. The first was the straight all-ones problem. The objective is to find the chromosome which contains a 1 in each bit position. The second version 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 optimal solution to this problem is 120. The third is the contiguous bits problem. Points are only given for 1's which have at least one other neighbor which also has a value of 1.

Fully and Partially Deceptive Order 4: The fully deceptive problem is a 40 bit maximization problem. The problem was defined in Whitley and Starkweather's paper GENITOR II [Whitley and Starkweather, 1990]. The problem is comprised of 10 subproblems, each 4 bits long. The subproblems use the lookup table shown in Table 6.1. The partially deceptive problem uses the same evaluations, with the exceptions of the reversed evaluations for 1111 and 0101.

Chrom	Evl	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

Table 6.1: Order 4 fully deceptive problem.



o(x): The Number of Ones in Chromosome

Figure 6.5: The gap problem. o(x) is the number of ones per chromosome. The gap size is Y. The starting point of the gap is P.

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. For example, the first subproblem has bits in position 1,2,3,4. The second encoding is interleaved: the bits in each subproblem are uniformly spread throughout the chromosome. For example, the first subproblem has bits in positions 1, 11, 21, 31. Using two point crossover, the first encoding is easier for the GA to solve than the second

The Gap Problem: This is a maximization problem [Liepins and Baluja, 1991] shown in Figure 6.5. The gap function f(x), with gap of size Y, starting at point P, with o(x) being the number of ones in the bit string, is defined by:

$$f(x)=\begin{cases}2P+Y-o(x)-1, if(P\leq o(x)\leq P+Y-1)\\ o(x), if((o(x)<P)\vee(o(x)>P+Y-1))\end{cases}$$

This problem was tested on a 120 bit chromosome string. Gap sizes of 20 and 25 were tried with the starting gap point, P=60.

6.3.3 Results and Discussion

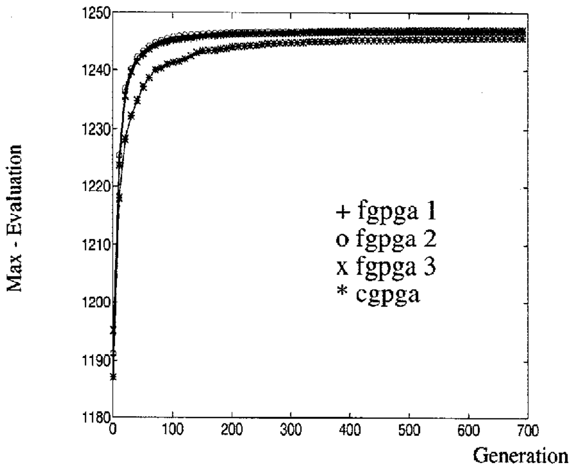
Table 6.2 shows the results of the 17 test problems; they are the average of 10 runs per problem for each algorithm. One of the difficulties inherent in comparing parallel genetic algorithms with each other, and with traditional GAs, is choosing the best criteria [Baluja, 1993]. Criteria which measure performance of the GA by the fitness of the best individual through the run of the algorithm are biased in favor of larger parallel GAs. If the number of evaluations performed is chosen as the criterion, parallel GAs often do not perform well, as parallel GAs may perform a lot of repetitive search. However, the quality of solutions evolved by pGAs have been shown empirically to be better than single population GAs in a variety of problems [Petty, 1989] [Tanese, 1989]. The measure used in this study is the number of generations to find the optimal solution and the number of evaluations per generation. However, using the optimal solution as a stopping criterion raises another issue: GAs find regions of good performance very quickly; the majority of the time is spent locating relatively small improvements in search of the optimal solution. For example, when the evaluation curves of DeJong f4 are examined, it is clear that the vast majority of the time between generations 200-700 is spent making very small improvements, see Figure 6.6. As stated by Forrest and Mitchell "it could be argued that the GA is more suited to finding good solutions quickly rather than finding the absolute best" [Forrest and Mitchell, 1993]. The results in this study certainly agree with this.

The ability of good chromosomes to spread rapidly through the population contributed to the success of the fgpGAs. A sample run, shown in Figure 6.7, displays the number of subpopulations that contain chromosomes which have evaluations equal to the best chromosome in the entire population. These chromosomes are candidates for selection in their respective groups of 10. This does not imply that all the chromosomes are exactly the same, nor does it imply that they will be chosen for recombination. The sudden drops of the number of populations, in Figure 6.7, represent generations in which a better chromosome was found. The actual spread rate does not match the fastest possible spread rates mentioned in Section 6.2. Although a good chromosome can be immediately accessed by its neighbors as soon as it is found, for more than the immediate neighbors to incorporate the chromosome, it must again be selected for recombination. Further, if it is selected, valuable schemata must not be destroyed by crossover or mutation operators. Although the populations which surround the immediate neighbors will incorporate the children chromosomes into their population, for them to spread the chromosome further, they must also select the children chromosomes for recombination. However, the evaluation of the children chromosomes may not be as good as the original chromosome. Further, if the crossover and mutation operations have destroyed valuable schemata, the children produced may not be preserved by elitist selection.

Test Function	***fgpGA*** Linear Order	***fgpGA*** 64*64 Array	***fgpGA*** Linear Skip	***cgpGA*** 40 sb pop.
DeJong Function #1	32.0	29.8	30.6	79.0
DeJong Function #2	40.0	38.6	43.9	111.8
DeJong Function #3*	22.0	19.5	20.4	64.5
DeJong Function #4	See Figure 6.6			
DeJong Function #5**	17.9	18.0	17.8	18.0
Subset Sum (1/4)	21.0	14.0	12.0	35.6
Subset Sum (1/20)	68.0	55.0	65.0	344.5
Subset Sum (1/40)	95.4	76.8	87.8	629.0
All-Ones	114.0	90.5	107.7	648.2
Sparse All-Ones***	134.0	94.8	113.3	342.0
Contiguous All-Ones	131.0	90.8	111.0	609.1
Fully Deceptive (A)	90.0	57.5	78.4	305.9
Fully Deceptive (B)	1220 (4)	742.5	942.2 (5)	1634.7
Partially Deceptive (A)	39.0	32.0	38.0	95.1
Partially Deceptive (B)	70.0	53.0	75.0	252.5
Gap Problem (Size 20)	161.2	126.3	164.3	675.9
Gap Problem (Size 25)	816.4 (5)	441.2	699.1 (9)	776.0

Table 6.2: Results for the 17 test problems. Each entry represents the average number of generations to find the optimal solution. The fgpGA evaluated 8192 chromosomes per generation, while the cgpGA evaluated 4000. A number in parentheses indicates that the optimal solution was only found the specified number of times, out of 10. The fgpGA was allowed 1400 generations, the cgpGA was allowed 3000.

* 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 runs were full size (120 significant bits, 60 extra bits).
Portions of this table appear in [Baluja, 1993].



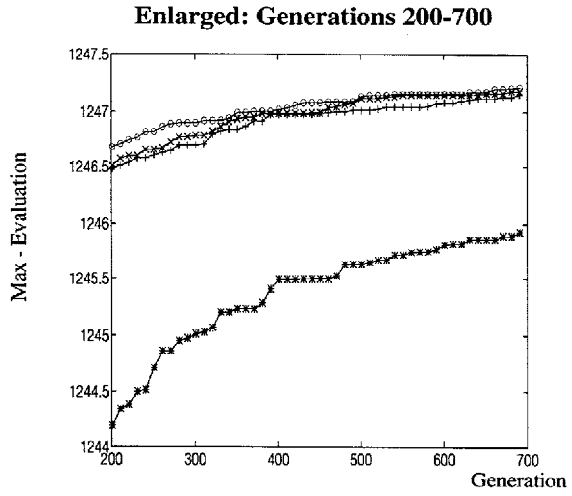


Figure 6.6: Average evaluations for 10 runs of the cgpGA and the fgpGAs on DeJong's F4, including the random Gaussian factor. The cgpGA was run with 50 chromosomes per subpopulation and 80 subpopulations. In the last 500 generations, very little improvement was made. [Baluja, 1993].

The different success rates of the fgpGAs on the Deceptive - Order 4 problem and the Gap(25) problem for the three fgpGAs illustrate the significant role subpopulation interaction has in performing successful search. It is interesting to note that in both of these problems, the cgpGA and the fgpGA (implementation 2) did the best; the other two implementations of the fgpGA did poorly. A possible explanation is that the structure of these problems benefits from larger population sizes. Since the fgpGA-2 has the fastest spread rate, it simulates a larger population more closely than the other implementations. One of the immediate plans for future research is to examine the performance on these two problems in greater detail.

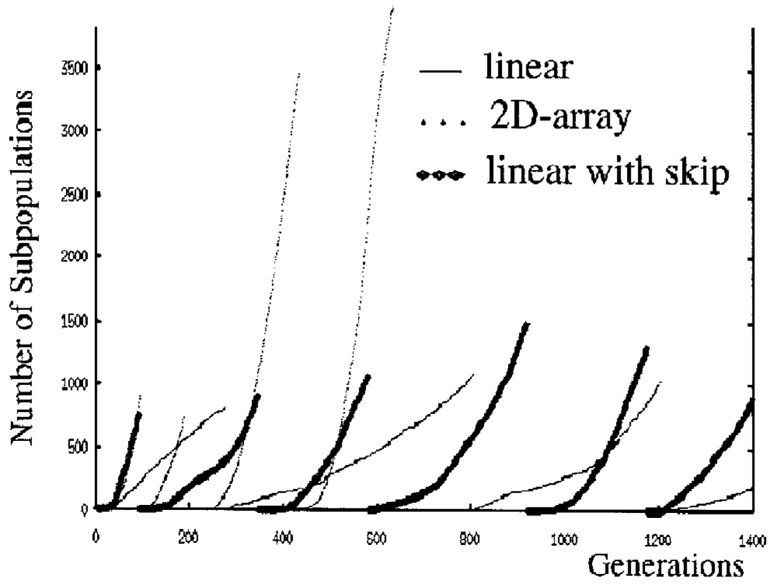


Figure 6.7: The number of populations which contain the best chromosome using the fgpGAs to optimize the order 4 fully deceptive problem, interleaved. The sudden drops in the number of subpopulations represent a new best solution found in one of the subpopulations. There are a total of 4096 subpopulations. The 2D array architecture found the optimal at approximately generation 600. The others did not find the optimal in 1400 generations.

The parameters in the cgpGA and fgpGA were not tuned per problem. It is suspected that with a little tuning, both types of GAs could significantly improve performance. 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.

6.4 Future Directions

To evaluate fine-grain parallelism in more detail, both harder problems and different population topologies should be explored.

6.4.1 Test Problems

The test problems attempted in this study comprise a fairly standard test suite of problems which aid in quantifying the effectiveness of GA models. However, many of these problems were designed to test the abilities of single population GAs, and do not reveal the potential of parallel GAs. For future testing of the fgpGA topologies, both harder problems ahd multi-objective problems should be attempted.

One of the harder problems tested should be the Traveling Salesperson Problem. This would help quantify the differences in performance of this system and the systems developed by Muhlenbein and Schleuter [Muhlenbein, 1989] [Schleuter, 1990]. They have extensively explored the TSP problem with parallel GA systems and have achieved very promising results. The functions termed *Tanese Functions* by S. Forrest and M. Mitchell [Tanese, 1989][Forrest and Mitchell, 1993] should also be attempted. These functions have proven to be very hard to optimize genetically, but are susceptible to hill climbing techniques.

Parallel GAs lend themselves to multi-objective optimization problems. The evaluation criterion of each population can reflect different objectives. When members of separate subpopulations are mixed, the children produced may be strong with respect to more than a single objective. Multi-objective problems have been explored in variants of cgpGAs by [Husbands, 1991] and [Cohoon, 1988]. Fine-grain parallel GAs also offer the ability to perform multi-objective optimization. It will be very interesting to see how the placement of objectives in subpopulations affects the abilities of the GAs. For example, all of the subpopulations with one objective could be placed close to each other, so that 'inner' subpopulations are surrounded only by others which have the same objective. Alternatively, the objectives could be assigned to the subpopulations in an interleaved manner. The formation and assimilation of niches will certainly play an integral role in the abilities of the GA to successfully optimize each of the objectives. Niche formation has been studied in massively parallel architectures by [Davidor, 1991].

6.4.2 Subpopulation Interaction

The massive distribution of the fgpGAs allows flexibility in the design of the interactions between populations. Three important issues which need to be resolved are: with which other subpopulations each subpopulation should interact, what the interaction should be, and how often the interactions should occur. For the problems which were tested, the 2D array topology worked well. This topology allowed for a rapid flow of genetic information, which is desirable in easy problems as good solutions can rapidly propagate. However, for harder problems, fast flow may not be a desirable property. A slower flow may prove its worth in the cases in which independent evolutions are needed to successfully optimize the function. Experimenting with time-varying and adaptive flows might also achieve impressive results; however, this may add another level of complexity to fgpGA design.

The three fgpGAs presented vary with respect to with which subpopulations may interact. The frequency and type of interactions (simply selecting one at random from the two evolved at the neighbor) have remained constant. However, the configuration used may be far from optimal. Fine-grain parallel genetic algorithms, and parallel genetic algorithms in general, still encompass a level of complexity which is not fully understood. The empirical results here are presented with the hope that they may help form insights into more rigorous models of the interactions in parallel GAs.

Acknowledgments

I would like to thank Dean Pomerleau, Stephen Smith, Todd Jochem, and Chuck Thorpe for their many helpful comments and suggestions throughout the development of this paper. This paper is dedicated to the memory of Dr. Gunar Liepins.

This research was partly sponsored by Defense Advanced Research Projects Agency, under contracts "Perception for Outdoor Navigation" (contract number DACA76-89-C0014, monitored by the U.S. Army Topographic Engineering Center) and "Unmanned Ground Vehicle System" (contract number DAAEO7-90-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 RealTime 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. (1993) The Evolution of Genetic Algorithms: Towards Massive Parallelism. To Appear in P.E. Utgoff, ed., *Machine Learning: Proceedings of the Tenth International Conference.* Morgan Kaufmann Publishers, San Mateo, CA.

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. AlC-90-00 l.

Cohoon, J.P., S.U. Hedge, W.N. Martin and 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.

Collins, R. and D. Jefferson (1991) Selection in Massively Parallel Genetic Algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA.

Davidor, Y (1991) A Naturally Occurring Niche and 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 MultiPoint 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 Algorithms,* Bloomington, IN.

Forrest, S. and A. Perelson (1990) Genetic Algorithms and the Immune System. *Parallel Problem Solving from Nature,* H.P. Schwefel and R. Manner, Eds. Springer-Verlag, Berlin.

Forrest, S. and M. Mitchell (1993) What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation. To Appear in *Machine Learning.*

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley. Grosso, P. (1985) *Parallel Subcomponent Interaction in a Multilocus Model.* Ph.D. Dissertation. Computer and Communication Sciences, University of Michigan.

Hillis, D. (1990) Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure. *Physica D.* 42. 228-234. North-Holland, Amsterdam.

Holland (1975) *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press.

Husbands, E, E Mill and S.Warrington (1991) Genetic Algorithms, Production Plan Optimisation and Scheduling. *Parallel Problem Solving from Nature,* H.P. Schwefel and 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 and Zenios, Eds. Pergamon Press, 1992.

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 Algorithms.* Morgan Kaufmann, San Mateo, CA.

Schaffer, J.D., R.A. Caruana, L.J. Eschelman, and R. Das (1989). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization, In J.D. Schaffer (Ed.) *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 and 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 Genetic Algorithms.* Morgan Kaufman, San Mateo, CA.

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

Tanese, R. (1989). Distributed Genetic Algorithms. In J.D. Schaffer (Ed.) *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA.

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