Optimal Population Size and the Genetic Algorithm

STANLEY GOTSHALL
School of Engineering
University of Portland
Portland, Or 97203
U.S.A.
sgotshal@up.edu

BART RYLANDER
School of Engineering
University of Portland
Portland, Or 97203
U.S.A.
rylander@up.edu

Abstract: - We conduct experiments to determine the optimum population size for problems as the instance size varies. We show that increasing the population size increases the accuracy of the GA. Increasing population size also causes the number of generations to converge to increase. The optimal population for a given problem is the point of inflection where the benefit of quick convergence is offset by increasing inaccuracy. Finally, we propose a method that might be used for determining the optimum population size for a given problem instance. This method holds for all three of the dissimilar problems that were used to conduct the experiment. It seems possible that it may hold for all GA applicable problems.

Key-Words: - Genetic Algorithm, Population, Optimization, Evolutionary Computation

1 Introduction

The genetic algorithm (GA) is a method of computation that simulates biological evolution [1][2]. This method is typically used to optimize functions that are intractable or have large or unknown search spaces. Despite years of successful application to a wide array of problems, little consensus has been generated concerning the optimum population size that should be used [3]. The reasons are multiple but stem from a lack of recognition of the difference between problems and problem instances.

Typically, a practitioner will try a range of population sizes before settling on a size that seems to work best with his problem instance. Since problems and their instances vary greatly, so it would seem do the optimum population sizes.

We need to carefully distinguish between a problem, and a problem instance. We will usually refer to the latter simply as an "instance". A *problem* is a mapping from problem instances onto solutions. For example, consider the Maximum Clique (MC) problem, which is to find the largest complete subgraph H of a given graph G. An instance of MC would be a particular graph G, whereas the *problem* MC is essentially the set of all pairs (G, H) where H is a maximal clique in G.

The goal of our research is to determine the optimum population size for a GA as a function of the instance size. As an example, consider the problem Sort (i.e. given an unsorted array of size n, return a sorted array). We want to empirically determine the optimum population size (by optimum, we mean the size that causes the quickest convergence) for an instance of size n. Then, see if this optimum holds as we increase the size of n. We want to know if the optimum population size remains constant; remains a constant percentage of the search space; or if some other consistent characteristic arises.

2 Test Parameters

Our experiment was conducted using 50% elitism, single point crossover, and 1% mutation. For each problem we began with instances of size 4 and then increased size until the problem was untractable. For each instance size we began with a population of 4 chromosomes and then increased until a recognizable pattern emerged or the optimum population was determined. We then collected data for each problem across all instance sizes to see if there was a pattern. To give the greatest insurance that our results were not problem specific we selected three dissimilar problems for conducting our research.

The first problem is Maximum 1's. See definition 1.

Definition 1, Maximum 1's.

Given: A string of length n.
Find: Maximum number of 1's that can be in that string.

This is probably the easiest known problem for a GA to solve. For all but very large n, the GA converges in only a few generations.

The next problem is Sort. See definition 2.

Definition 2, Sort.

Given: An unsorted array of length n.
Find: The sorted array for some predefined condition.

This problem was selected because of its uniform familiarity and because it is a problem known to be in P (i.e. the class of problems solvable in polynomial time with a Turing Machine)[6].

The last problem to be examined in our experiment is Maximum Clique (MC). See definition 3.

Definition 3, Maximum Clique.

Given: A graph G.
Find: The largest complete subgraph H in G.

This problem was chosen because it is a known NP-Complete Hard problem (i.e. NP is the class of problems solvable in polynomial time with a Nondeterministic Turing Machine)[4][5].

Given this range of very different problems we conjecture that any emerging population characteristics may hold for all problems that GAs can be successfully applied to.

3 Results

After conducting our experiments, we tabulated them on a per problem basis. Later, we grouped the results as a whole to see if there were commonalties.

3.1 Maximum 1's

The first problem we examined was Maximum 1's. We observed the generations to convergence for populations of 4 and 8 is within 2 and 10 generations for all instance sizes tested. We conjecture this quick convergence is due to the high probability that the initial state of a small population will find a sub optimal solution and converge to that solution. Of the 250 tests per population, the percent of incorrect solutions remained above 98%. Using an instance size of 12 bits and a population of 200, convergence occurs in an average of 20 generations. However, this population size yields an average of 60 incorrect solutions out of 250. The percentage of correct solutions appears to increase steadily as population size increases.

Although accuracy increases with population size, there is a tradeoff with how many generations the population needs to converge. With larger populations, the advantage of converging in a reasonable number of generations diminishes. See figure 1.

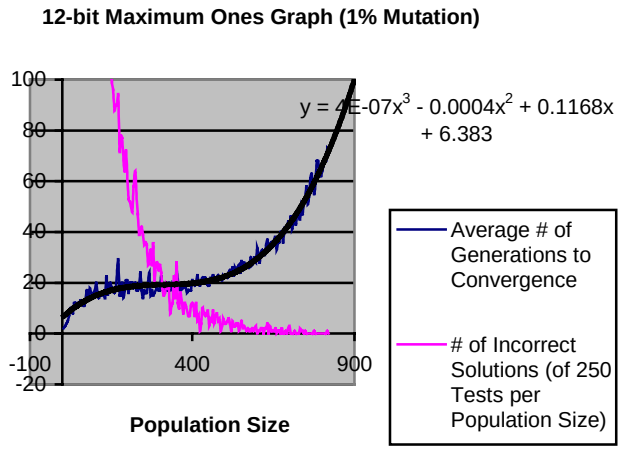

Fig. 1, Average Generations to Convergence for 12-bit Instance Size for Maximum Ones

As can be seen, the trend for accuracy increases as the population increases. Also, the number of generations to converge increases as the population increases. We suspect the increase of generations to convergence is probably due to the overall increase in probability that a mutation will take place in the population and in the increased time it takes for a greater number of chromosomes to completely converge. Given a population of 500 chromosomes, on average, 5 chromosomes will have a mutation per generation. Thus, more generations are needed to discard these chromosomes to let the population converge. This seems to explain the relationship between population size and generations needed to converge with large sets of populations.

Further, we noticed the number of generations to convergence grows as a third order polynomial with respect to the population size. In fact, this trend is common among instance sizes of 4, 8, 12, and 16

bits. Although the number of generations to convergence increases quickly with respect to population size, the limit of incorrect solutions as the population size goes to infinity appears to be 0 for all instance sizes.

3.2 Sort

Although GA's are more useful with difficult or intractable problems, they can be designed to sort a list of numbers. In the representation of sort, the optimal solution is a string of binary numbers in numerical order. For example, if the GA was evolving to sort 5 numbers, the optimal solution would be 001|010|011|100|101. To sort n numbers, the chromosome length is the minimum binary representation of n multiplied by n. Fitness is determined by counting the number of correct bits when compared to the predetermined optimal solution.

We observed that the behavior of sort is similar to that of Maximum 1's. For populations of 4 and 8, regardless of instance size, convergence occurs within 3 and 10 generations. The resulting accuracies are equally similar. However, given a population and instance size, sort converges in fewer generations and has a higher rate of accuracy. For an instance size of 12 bits and a population of 300, maximum ones converges after 20 generations with 91% accuracy while sort converges in 11 generations with accuracy > 99.5%.

Sort also exhibits a tradeoff between generations to convergence and accuracy. For large populations, the number of generations to convergence increases without much increase in accuracy. See figure 2. Although the number of incorrect solutions appears to reach zero at a population size of 100, it is reasonable to assume that the percentage of incorrect solutions is always approaching zero from the positive side, since GA's are never guaranteed to yield the optimal solution. This trend is consistent with Maximum 1's.

The generations to convergence curve grows as a third order polynomial similar to Maximum 1's, although it grows at a slower rate as indicated by the equations in figures 1 and 2. These equations are derived from the best-fit curve of each graph. The convergence trend is common among instance sizes of 6, 12, 15, 18, and 32 bits for sort.

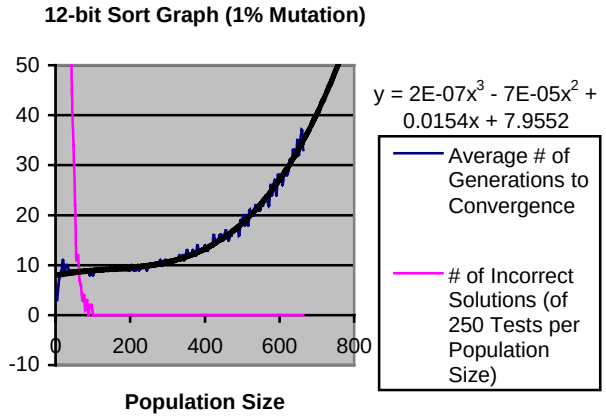

Fig. 2, Average Generations to Convergence for 12-bit Instance Size for Sort

3.3 Maximum Clique

Among the three problems introduced, Maximum Clique represents the greatest challenge for optimization. Unlike Sort and Maximum 1's, the problem of Maximum Clique is inherently difficult to solve for large instance sizes. In comparison, Sort can be solved with an algorithm which takes $O(n^2)$ time to execute. Maximum 1's is solvable with a simple loop taking O(n) to converge. Both these algorithms are guaranteed to yield the expected results. In contrast, a conventional brute-force implementation to solve Maximum Clique takes exponential time with respect to the instance size.

The chromosome representation used is one bit for each node of the input graph. So a chromosome of 01110 would represent a potential solution of nodes 1,2, and 3 (counting from the right beginning with node 0). Each chromosome represents a possible clique. The fitness assigned is equivalent to the size of the clique that the chromosome represents, if in fact it represents one. All chromosomes with a single 1 are counted as cliques of 1 because a single node is considered a clique.

Although several trends appeared common among all three problems, the convergence landscape of Maximum Clique differs from Maximum 1's and Sort. The average number of generations to convergence for small populations is large for instance sizes of 12 and 16 nodes. However, the behavior among the populations ranging from 4 to 12 for graphs of 12 and 16 nodes is interesting. For a graph size of 12 nodes and a population of 4 chromosomes, the population takes an average of 84 generations to converge. However, with a population

of 8, the convergence time jumps to 600 generations. After 8, an increase in population gradually decreases the convergence times until a minimum is reached. See figure 3. For a 16-node graph, this minimum occurs at an approximate population size of 400, taking 58 generations to converge.

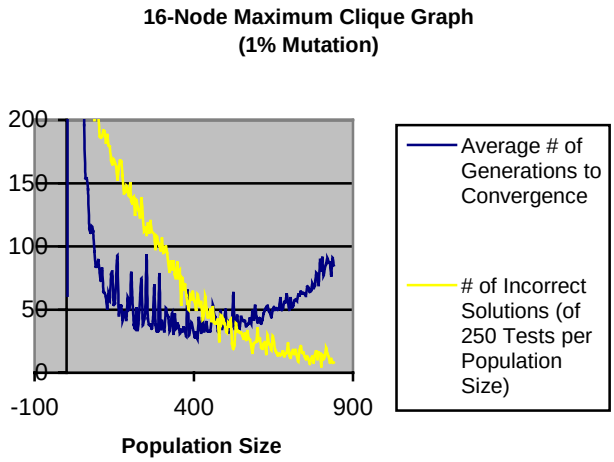

Fig. 3, Average generations to convergence for 16-bit instance size for Maximum Clique.

This erratic jump of generations to convergence seems due to a fundamental difference in the maximum clique fitness function, since nothing else is different among the three problems that would affect convergence times.

Disregarding the large convergence times for relatively small populations it is still difficult to determine the best-fit curve, however it resembles a quadratic. This quadratic resemblance is common among instance sizes of 7, 8, 12, and 16 nodes. The other trends such as percentage of correct solutions and behavior for arbitrarily large population sizes hold for all three problems.

4 Conclusions

After analyzing the preceding data, three conclusions seem warranted. First, for arbitrarily large population sizes the accuracy of the genetic algorithm approaches, but does not reach, 100%. The greater the population size the greater the chance that the initial state of the population will contain a chromosome representing the optimal solution.

Second, the increase in population size causes the number of generations to converge to increase. This appears to be due to the overall increase in probability that mutation will occur. If mutation occurs for large population sizes, more generations are needed to eliminate the mutated chromosomes.

Third, the optimal population for a given problem implementation is the point in which the benefit of low numbers of generations to convergence balances with the benefit of increased accuracy as the population size increases. This point is where the average rate of change of the number of generations to convergence is at a minimum. In other words, the optimal solution is where the average slope is closest to zero. In the case of the cubic graphs, this is the point of inflection, when the curve ceases to be concave down and becomes concave up or vice versa. In the case of quadratics, it is the global minimum. This is the crossover point in which the benefits of accuracy and convergence time balance. In each convergence graph, the average slope of the number of incorrect solutions changes at this point. See figure 3. Before the minimum at 400 the incorrect solutions trend is nearly linear, but after 400, it curves up as it approaches zero. Figures 1 and 2 also show a similar relationship, although figure 2 is slightly distorted because of the initial slope of the incorrect solutions line.

When this optimal point is observed over varying instance sizes, an interesting trend emerges. The optimal population appears to grow logarithmically with respect to the instance size for any of the three problems. Since this trend is common among all three dissimilar problems, it seems possible that all GA applicable problems share this trend. Note that these approximations, as on the previous graphs, are generated using Microsoft Excel.

Consider the trend of the Maximum 1's problem. See figure 4. Given the equation of the curve, the optimal population size can be calculated as a function of the instance size. For example, if we wanted to determine the optimal population for an instance size of 30, the solution would be 616. To increase the accuracy in this approximation more data points are needed in the graph.

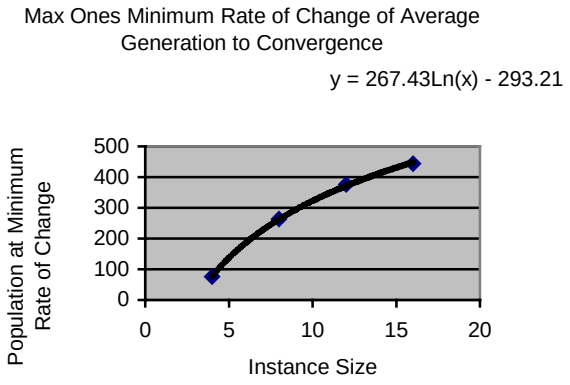

Fig. 4, Optimal population trend for Maximum 1's.

The trend for Sort and Maximum Clique can be generated and used in the same manner. Given a set of optimal, or transitional points, generate the function and use it to approximate the optimal population for a given instance size.

The results of this research indicate that we may have a general means for determining the optimal population considering the advantages of both convergence time and accuracy. This method seems useful in practice because it selects the population that optimizes the benefits of convergence time and accuracy.

Possible ideas for future work would be to conduct the same experiment with other problems. It may even be possible to group commonalties within complexity classes. In other words, it might be interesting to find out if all problems in P exhibit the same characteristics as Sort. If so, it might be that problems within the same complexity class exhibit the same population characteristics. This would provide value practical knowledge for the implementation of any GA.

References:

[1] Holland, J., *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI, University of Michigan Press, 1975.

[2]Rawlins, G.J.E. (1991). Introduction. Foundations of Genetic Algorithms, Morgan Kaufman. pp. 1-10.

[3] Goldberg, D.E. (1989b). Sizing Populations for Serial and Parallel Genetic Algorithms, Proceedings of the Third International Conference on Genetic Algorithms. San Mateo, CA: Morgan Kaufman. pp. 70-79

[4] Håstad, J. (1996). Clique is Indeed Hard ,*Proc.on STOC*, 1996.

[5] Rylander, B., Foster, J., GA-hard Problems, *Proc. On Genetic and Evolutionary Computation Conference*, 2000.

[6] Bovet, D., Crescenzi, P. *Computational Complexity*, Prentice Hall.1994.