FINAL SUMMARY REPORT

FOR

"STUDY OF GENETIC DIRECT SEARCH ALGORITHMS FOR FUNCTION OPTIMIZATION"

NASA Grant NGR23-005-602

Project Director:

Bernard P. Zeigler

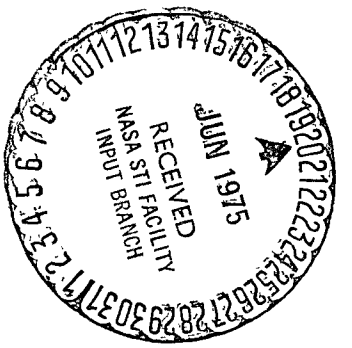
Associate Professor

Department of Computer and Communication Sciences

University of Michigan

Ann Arbor, Michigan

June, 1974



I INTRODUCTION

1.1 Objectives

The following is the final report for a research project on the "Study of Genetic Direct Search Algorithms for Function Optimization" conducted under NASA Grant NGR-23-005-047. The duration of this project totalled approximately three years, beginning on October 1, 1970 and running to September 30, 1972 then lapsing for one year before beginning again on May 1, 1973 and terminating on April 30, 1974.

The purpose of the project was to determine the performance of genetic direct search algorithms (employing techniques suggested by natural adaptive systems) in solving function optimization problems arising in the optimal and adaptive control areas. In particular we attempted to answer the following questions with respect to a particular class of algorithms to be studied:

1) How well could the particular genetic algorithms studied perform in comparison to standard direct search techniques currently available?
2) What are the essential parameters determining the behavior of genetic algorithms to be studied and what values should be assigned to these parameters for optimum performance?
3) Can relatively simple genetic algorithms be constructed while still retaining the efficiency of more complex ones?

These questions were investigated with the hope of being able to draw tentative conclusions for the following more long range questions:

How broad is the domain of useful application of genetic algorithms? Should one general form of algorithm be applied in all situations or are different configurations or parameter settings appropriate in different problem environments?

1.2 Methods

In investigating these questions our research took the following directions:

1) Design, construction and experimental testing of algorithms
2) Analysis of the data so generated with a view toward drawing empirical confirmation of intuitively based hypotheses.
3) Mathematical modelling of the algorithms and their environments.

Our major effort was concentrated in direction 1) and 2). We gathered large amounts of data from parametrically controlled experiments employing the IBM 1800 computer available to us on an open shop basis. It is from the reduction and the analysis of this data that our conclusions are mainly drawn. Part of our effort also went into direction 3). Our results in

this direction fell well short of providing predictive models but did succeed in clarifying some of the factors involved in the highly complex behavior displayed by our algorithms.

1.3 Documentation

The research performed under the grant was described in the following reports:

I. "Comparison of Genetic Algorithms with Congugate Graduent Methods" (J.L. Bosworth, N.Y. Foo, and B.P. Zeigler) NASA Contractor Report No. 2093, NASA, Washington, D.C. August, 1972.
(This paper was also presented at the Sixth Annual Princeton Conference on Information Sciences and Systems, Princeton, N.J., 1972).

II "Algebraic, Geometric and Stockastic aspects of Genetic Operators" (N.Y. Foo and J.L. Bosworth), The University of Michigan, Computer and Communication Sciences Report No. 003120-2-T, March, 1972.

III "Noisy Function Optimization by Genetic Algorithms and Conjugate Gradient Methods" (B.P. Zeigler, J.L. Bosworth, and A.D. Bethke) The University of Michigan, Computer and Communication Sciences Report No. 143, March, 1973.

IV "Convergence Properties of Simple Genetic Algorithms", (A.D. Bethke, D. Strauss and B.P. Zeigler), The University of Michigan, Computer and Communication Sciences Report, to appear.

In what follows we shall first provide a brief summary of the results discussed in these reports. Then we shall discuss the conclusions drawn in response to the short and long range questions raised above.

We shall refer to these reports by number in what follows.

II BACKGROUND AND RESULTS

2.1 Background

Our Report I outlines the importance of direct search optimization methods in optimal and adaptive control applications. In such applications it is desirable that the method firstly, converges (i.e., actually locates an optimum or near optimum) and secondly, converges rapidly (many algorithms such as random search are guaranteed to converge but do so much too slowly for practical application).

At the time of the project initiation, a host of direct search methods, which we refer to as classical methods, had been suggested and studied. The most successful of these, in terms of performance and analytic justification were the congugate gradient-variable metric methods. However, while the latter methods could be shown to converge superlinearly on quadratic functions, they were not guaranteed to converge on functions not well approximated by quadratic forms. Moreover, in practice the matrix updating techniques employed by the more sophisticated versions required complex computation. Also the matrices involved

can not be guaranteed to remain non-singular and positive definite as required in the justifying theory.

More generally, the classical methods were largely developed for the case of unimodal functions. Based on local information, these methods were subject to being trapped on false sub-optimal peaks when applied to non-unimodal functions. Moreover, reliance on gradient information, implies an erratic behavior when applied to noisy functions. (Recent work in the area has included attempts to broaden the class of functions on which conjugate gradient methods can be guaranteed to coverge, and some relatively unsophisticated proposals for employing second level supervision for co-ordinating local optimizers in the multipeak optimization problems).

Also at the time of project initiation, several promising results had been obtained employing genetic algorithms in artificial adaptation problems. These algorithms had the attractive feature of being applicable to functions defined on non-numerical spaces. For such situations, the usual euclidian topologies do not apply and hence genetic algorithms must, of necessity, employ non-local, combinatorial information gathering and search techniques. Applied to numerical spaces, this would appear to limit the performance of these methods since gradient information can not be utilized. On the other hand, because of the non-reliance on local information, the genetic methods appeared to be promising for application to the multi-peaked and noisy function optimization cases.

2.2 Construction of Genetic Algorithms

Our Report I describes several versions of genetic algorithms we constructed during the exploratory phase of our investigation. Many variations were incorporated and tested in the possibility that they could prove essential to algorithm performance. As could be expected, this lead eventually to a complex software package. Thus in the last year of the project the algorithm was refashioned into a much more simple and elegant form. This version is described in Report IV. It contains only the basic features which our experimentation with earlier versions suggested to be essential. As a consequence, it fits well within the 32K core of the IBM 1800 we employ and would thus be within the limited complexity range necessary for on line control applications.

2.3 Basic Structure of Genetic Algorithms

The basic structure of the genetic algorithm studied in the last year of the project can be briefly described as follows:

We are given a function $f:A^n \rightarrow R$ to be maximized; R is the set of real numbers, A is an arbitrary set called the alleles. When A=R we have the usual real valued functions with numerical arguments. In representing such functions on a computer, the arguments must be bounded between limits and quantised to discrete values between these limits; the set A can be taken to represent the resultant discrete set.

At any time, a population of points in the domain A^n of f is maintained. Each point is represented in the computer by an ordered string of the form $a_1,a_2,\ldots,a_n$ with the value of that string being given by $f(a_1,a_2,\ldots,a_n)$. To form a new generation of strings the following operations may be applied:

1) mutation: altering one or more of the alleles of a string.
2) crossover: combining two strings so that a pair of strings is produced whose allele values come from either one of the parents (the position information of alleles is retained in this operation i.e., at locus or position i, a daughter receives one the alleles a_i or a'_i which appear in the i-th position of the parents).
3) selection: the population size is reduced by eliminating points with low function values.

A generation consists of the application of one or more of the operators. Parameters control the probability that a particular operator will be applied in a generation and the exact nature of the operation involved. As the generations proceed, the populations are expected to improve in the sense of average function value or maximum function value of the population.

2.4 Additions to Basic Structure

As indicated, we initially investigated various additions to the basic structure just described. These additions included the following:

4) inversion: here we must distinguish between a point $(a_1,a_2,\ldots,a_n)$ in the domain of f and its representation as a string. One such representation is $(1,a_1)(2,a_2)\ldots(n,a_n)$; another is $(2,a_2)(1,a_1)\ldots,(n,a_n)$; indeed, there are n! such representations, each preserving the significance of allele a_i in its position i. Inversion is an operator which operates upon representations by altering the order of the pairs (i,a_i).
5) adaptation: the parameters governing mutation are controlled during an optimization run by a second level adaptation routine which basically determines the size of a mutation alteration according to an estimate of whether large or small mutation sizes have been fruitful in the recent past.

2.5 Structure of Operators and Search Effectiveness

In our Report II, the crossover and inversion operators are precisely defined and characterized algebraically and geometrically. The operation of crossover as a combinatorial operator was studied by characterizing the smallest sphere which encloses a set of points subjected to arbitrary crossover in n-dimensional euclidian space. It was shown that such a sphere could grow exponentially with successive generations to a maximum of n times its original radius. Monte Carlo simulations however showed that more typically the points produced by successive crossovers remain inside a sphere close to the original one.

Thus conceptually, the crossover operator implements a search of a well defined type. We had hoped to employ these results in analysis of genetic algorithm search efficiency but were unable to achieve reportable progress by project's end.

2.6 Comparison of Genetic Algorithms with Conjugate Gradient Methods

In our Report I, we present results obtained with genetic algorithms employing all five of the operators outlined above. Versions of these algorithms were run against the Fletcher-Reeves conjugate gradient algorithm. Performance was measured by the number of function evaluations required to achieve a given level of function value. The results can be summarized as follows:

1. The genetic algorithms were markedly inferior to the conjugate gradient algorithm when applied to "easy quadratic functions" (functions with spherical contours). This was a confirmation of the expected limitations of the genetic algorithms in exploiting gradient information.

2. The genetic algorithms were markedly superior to the conjugate gradient algorithm when applied to a function with multiple peaks. This was a confirmation of the expected limitation of the local optimizer on a non-unimodal domain. But more significantly, it established that genetic algorithms can locate the global optimum of certain multimodal functions.

3. Applied to other unimodal standard-test functions (including for example the Wood function) the genetic algorithms proved superior in converging closer to the true optimum while the conjugate gradient algorithm was superior in rapidity of convergence. The progress curves of the two kinds of algorithms were qualitatively very distinct. The conjugate gradient method made very rapid progress at the beginning of a run but when a given value was reached no further progress was achieved. In contrast, the genetic algorithm made relatively slow but steady progress until it too reached a value where no further progress was possible. In all cases, the final value attained by the genetic algorithm was orders of magnitude closer to the optimum than that of the other method.

2.7 Noise Behavior

We hypothesized that the impediment to further progress of the conjugate gradient method was due to the effect of round off noise generated in the function value computation to which the genetic algorithms were less sensitive. This was confirmed when we investigated the behavior of the algorithms given functions corrupted by noise. In our Report II, we describe the following result:

4. The genetic algorithms are much less affected by increasing additive noise levels then are the conjugate gradient methods. This was a confirmation of the expected limitations of local optimizers in noisy environments. More significantly, it showed that genetic algorithm performance need not be degraded by reasonably low noise levels.

2.8 Basic versus Augmented Structure

In Report I we describe experiments in which the effect of adding inversion and adaptation (section 2.4) to the basic structure is explored. This augmentation considerably complicates the software realization since it requires more sophisticated data structures (for inversion) and memory storage and processing (for adaptation). We found that while inversion did improve performance, the improvement was not very marked. On the other hand, the second level adaptation routine considerably improved performance, by enabling the sizes of mutations to be adjusted as the optimum was approached.

2.9 Covergence Properties of the Basic Structure

As indicated in section 2.2, we concentrated our final year of effort on the study of the basic genetic algorithm structure. We constructed a software package which enabled us to conveniently control 8 parameters specifying the algorithm structure and the objective function difficulty. We ran experiments for about 80 experimental points in this parameter space, each consisting of 3 to 5 randomly initialized runs. Various statistics were gathered but of central importance was the plotting of mean value (over runs) attained versus number of function evaluations for each experiment. The details of method, results and analyses are given in Report IV.

Our main objectives were to describe the progress curves obtained as a parametrically specified family, and to assess the effect of structure parameters on these descriptive parameters. We hoped to draw conclusions concerning optimal parameter settings and to develop possible predictive models.

Our main results concerned maximization of linear functions, and are summarized as follows:

1) All progress curves could be described by an exponential approach to final value of the form

$$V(n) = (V_0 - V_f)e^{-rn} + V_f$$

where $V(n)$ is the mean value achieved after n sample evaluations

V_0 is the mean value at the beginning of a run

V_f is the mean value which appeared to be asymptotically being approached

r is the exponential decay factor

Thus the properties of most interest in the study of convergence viz. the final value converged to, and the rate of convergence could be simply described by the parameters V_f and r respectively according to our results.

2) There is a trade off between final value attained V_f and rate of convergence r in the following cases:

   a) Increasing population size tends to increase V_f but decrease r,

   b) V_f for mutation operating without crossover is higher than that for crossover without mutation but the reverse is true for r,

   c) For small populations (10 strings), V_f for crossover-mutation combined is greater than that of either alone, but r is lower than the r's of either alone. For large populations (50 strings) the combined curve approximates that of mutation alone.

3) The sensitivity of performance (V_f,r) to parameter settings (such as crossover rate and mutation rate) decreases as the population size increases and the objective function difficulty (number of co-ordinates) decreases. With small populations, equal crossover to mutation ratio is optimum.

4) The computer time required for a given run length increases faster than linearly with population size from around 1 hour for 10 strings to 12 hours for 50 strings.

The results are consistent with the following explanation: Mutation is the dominant operator in determining the progress curve. The exponential form can be derived from a model in which mutation randomly samples a space with uniform density of good (in the sense of yielding an improvement) points. Increasing the

population size and adding crossover tend to increase the "effective" space accessible to mutational sampling. As a consequence, the final value attainable increases while the rate of achieving this final value decreases.

III CONCLUSIONS

With regard to the questions which motivated the project investigation (raised in section 1.1), our results suggested the following conclusions:

1) Genetic algorithms can outperform standard algorithms in multimodal and/or noisy optimization situations, but suffer from lack of gradient exploitation facilities when gradient information can be utilized to guide the search.

2) For large populations, or low dimensional function spaces, mutation is a sufficient search operator. However, for small populations or high dimensional functions, crossover applied in about equal frequency with mutation is an optimum combination.

3) Complexity, in terms of storage space and running time, is significantly increased when population size is increased or the inversion operator, or the second level adaptation routine is added to the basic structure.

   Each of these additions tends to increase the closeness to which an optimum can be approached. Thus, in this sense, simple algorithms can not replace more complex ones. However, the rate of approach may also be decreased by these additions so that if this is the criterion of efficiency, simple algorithms can perform, as well or better, than the more complex ones.

These conclusions, of course, apply strictly speaking only to the algorithms and problem situations actually studied. However, with respect to long range, more global questions posed we venture the following conclusions:

Genetic algorithms may be usefully applied to function optimization in which one or a combination of the following features are significant:

a) the function domain is non-numerical and has no obvious topology in which gradient information can be efficiently utilized.

b) the function evaluation is confounded by a high level of noise contamination.

c) the function has a numerical domain but is multimodal.

Genetic algorithms should not be applied in situations where sufficient information is available about the structure of a numerical function to enable classical exploitation of gradient information.

The form of the algorithm employed depends on the criteria of performance and the difficulty of the optimization problem. The simple algorithm structure (small population, crossover and mutation) is best when quick response is essential or in easy (linear, low dimensional) function optimization environments. Larger populations, inversion and/or second level adaptation significantly increase, the prospect of ultimate convergence. However the price to be paid is a significant increase in both the number of function evaluations required to achieve a given suboptimal level and the complexity of computation involved.