INT. J. PROD. RES., 1995, VOL. 33, NO. 7, 1911–1923

A genetic algorithm approach to curve fitting

M. GULSEN†, A. E. SMITH†‡ and D. M. TATE†

We present a genetic algorithm approach to fitting curves with multiple parameters to data points. Using data from three functions with two to ten coefficients and one to ten variables with linear, polynomial and transcendental functional forms, we varied the number of data points, the range searched over, the distance metric, the distribution for sampling, and the parameters of the genetic algorithm to examine the robustness of the approach. We developed a sequential evolution mechanism to overcome premature convergence to local minima when some terms dominate others in magnitude. Comparison with previously published work is favourable with regard to both goodness of fit and computational effort.

1. Introduction

Curve fitting is the process of approximating a closed form function from a given data set. To express the data in a closed form equation is particularly useful for analysis and interpretation of the observed data. Curve fitting is a preliminary activity to many techniques used to model and solve production problems, such as simulation, predictive modelling, and statistical inference. Although there are many ways to approach curve fitting, and there are methods that work well on data with few variables, there are few general techniques for complex functions with no derivative information and/or non-linear forms.

Genetic algorithms (GAs), a type of parallel heuristic search method, have been applied to least squares curve fitting in three previous papers (Karr *et al.* 1991, Manela *et al.* 1993, Rogers 1991). In this paper we extend the GA approach to larger problems and alternative error metrics, and perform more complete analysis of the robustness of the GA. We have developed a unique sequential evolution mechanism to overcome problems with premature convergence. Below we give a description of our algorithm and then present the results of the algorithm on three different test problems, two of which are taken from the literature (Friedman 1988, Karr *et al.* 1991, Rogers 1991). We conclude with general observations on the GA method.

2. Genetic approach to curve fitting

The objective of curve fitting is to select functional coefficients (i.e. parameter values) which minimize the total error over the set of data points being considered. Once a functional form and an error metric have been selected, curve fitting becomes an optimization problem over the set of given data points. Since genetic algorithms have been used successfully as global optimization techniques for both continuous

Revision received July 1994.

†1048 Benedum Hall, Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA.

‡To whom correspondence should be addressed.

0020-7543/95 $10·00

functions and combinatorial problems, they seemed suited to curve fitting when it is structured as a parameter selection problem. We present a brief introduction to genetic algorithms as search and optimization techniques, then describe our genetic formulation of curve fitting.

2.1. *Overview of genetic algorithms*

Genetic algorithms were described by John Holland and Kenneth DeJong in their pioneering works (DeJong 1975, Holland 1975). GAs are global optimization heuristics motivated by the process of natural selection in biological systems. Under the GA paradigm, rather than generating a sequence of candidate solutions one at a time, a population of candidate solutions is maintained. The primary distinguishing features of GA are an *encoding*, a *selection* mechanism, a *crossover* mechanism, a *mutation* mechanism, and a *culling* mechanism. The encoding is a data structure which describes a unique solution to the optimization problem, in a compact representation. To give a biological analogy, the encoding is similar to a chromosome, on which specific genes describe various aspects of the individual in question. The selection mechanism takes the encoding of an individual solution as input and computes the corresponding value of that solution with regard to the objective(s) of the optimization. This value is then used to select solutions for crossover. The crossover mechanism produces one or more new encodings using two 'parent' encodings selected from the current population as the source of 'genes'. The new encoding describes a solution with some features in common with those of each parent. The mutation mechanism is an algorithm for perturbing an encoding to produce a 'nearby' encoding, and is used to help maintain a diverse gene pool from which to construct new solutions. The culling mechanism is a procedure for removing known solutions from the population and replacing them with new solutions. This can be done by replacing parents with their children, or by deleting the worst solutions in the population when new individuals are created, or by some combination of the two.

For a complete overview of GAs and their uses see Goldberg (1989). We very briefly summarize the generic search process:

(1) Randomly generate an initial solution set (population) and calculate the fitness value (i.e. objective function value) for each member of the population.
(2) Repeat{
    Select Parents for crossover.
    Generate offspring.
    Mutate some of the members of the original population.
    Merge mutants and offspring into the population.
    Cull some members of the population to maintain constant population size.
    }
(3) Continue until desired termination criterion is met.

The distinctive aspects of genetic search are:

(1) The search is highly parallel, with each population member defining many different possible search directions. Potentially, GA search could be implemented extremely efficiently on massively parallel hardware.

(2) No special information about the solution surface, such as gradient or local curvature, need be identified. The objective function need not be smooth, continuous or unimodal.
(3) Genetic algorithms have proven to be fairly robust under varying parameter settings and problem particulars. Given even a weak correlation between encodings and objective function values, genetic algorithms usually find near-optimal solutions.

Since Holland's and DeJong's works, genetic algorithms have been applied to many combinatorial optimization problems, including job shop scheduling (Bean 1994, Davis 1985, Storer *et al.* 1992), the travelling salesman problem (Jog *et al.* 1991, Tate *et al.* 1994, Whitley *et al.* 1991), bin packing (House and Dagli 1992), pallet loading (Juliff 1993), quadratic assignment problem (Levitin and Rubinovitz 1993, Nissen 1994, Tate and Smith 1994), assembly line balancing (Anderson and Ferris 1994) and facilities layout (Tate and Smith 1993). These efforts are highly diverse in their use of the genetic approach, but often report favourable results compared to other, problem-specific heuristics. The encouraging results seem to stem from the GA's ability to escape local minima or maxima by breeding (i.e. crossover and mutation), the GA's robustness to evolution parameters and encodings, and the GA's efficiency in global search.

2.2. *Our genetic algorithm for curve fitting*

We used a real value encoding for our GA, so that each solution is encoded as a vector of real valued coefficients. We first experimented briefly with a traditional binary encoding, where all coefficients are represented in base 2; however this proved to be cumbersome, and performed no better. Our evolution algorithm uses two basic operations: crossover and mutation. Crossover generates new coefficients by taking the arithmetic mean of two 'parent' solutions of coefficients. Mutation perturbs a solution by altering a coefficient over a large range. Solutions are evaluated using the difference between the sampled data points and their fitted values. This was a measure of the magnitude of the residuals; any error metric could be used and we tried three common ones. The next sections describe population generation, evolution, evaluation, selection, and culling in detail.

2.2.1. *Generating the initial population* Each member of the population is a vector of real coefficients. In all of the experiments we used a population size of 50 solutions. Using the example of the family of curves given by equation 1, there are two variables, x_1 and x_2, and four coefficients. Therefore each solution (population member) has four real numbers, C_1, C_2, C_3, C_4.

$$y = C_1 + C_2 x_1^3 + C_3 \cos x_2 + C_4 x_1 x_2 \tag{1}$$

The initial estimates for each coefficient are generated randomly from a uniform distribution over a pre-specified range for each coefficient. However, the specified range need not include the true value of the coefficient (though, of course, this is permissible). We generated the initial coefficient values from a single uniform distribution for all test problems studied.

2.2.2. *Evolution mechanism* A fitness (objective function) value, which measures how well the proposed curve fits the actual data, is calculated for each member of the population. We considered three different distance metrics in our study: squared

error, absolute error and maximum error. The fitness value is used to select solutions for breeding.

We did not use the popular 'biased roulette wheel' (Goldberg 1989) selection method because of the large variations in the fitness values among the members of the population. A biased roulette wheel method would tend to encourage premature convergence by focusing the search completely in the neighbourhood of the first good solution, and the entire population would become identical. We selected parents for breeding uniformly from the best 25 members of the current population, i.e. the superior half of the population. This is a stochastic form of truncation selection originally used deterministically in evolution strategies for elitist selection (see Bäck and Schwefel 1993 for a good discussion of selection in evolution strategies). Mühlenbein and Schlierkamp-Voosen later used the stochastic version of truncation selection in their Breeder Genetic Algorithm (Mühlenbein and Schlierkamp-Voosen 1993). The values of the offspring's coefficients are determined by calculating the arithmetic mean of the corresponding coefficients of two parents. This crossover is known as extended intermediate recombination with $\alpha=0{\cdot}5$ (Mühlenbein and Schlierkamp-Voosen 1993). This retained identical coefficient values of the parents in the child, while averaging the non-identical coefficients. We show an example below for equation 1:

Parent A:	45·876	32·958	12·098	−3·892
Parent B:	12·988	35·832	0·234	−12·984
Offspring:	29·432	34·395	6·166	−8·438

Mutation perturbs solutions so that new regions of the search space can be explored. For the mutation process, first the range for each coefficient is calculated by determining the maximum and minimum values for that particular coefficient in the current population. Then the perturbation value is obtained by multiplying the range with a factor. This factor is a random variable which is uniformly distributed between $\pm k$ (coefficient range). We used $k=2$ for our studies. Our mutation scheme is related to that used in Mühlenbein and Schlierkamp-Voosen (1993)'s Breeder Genetic Algorithm. However, their range was static where our range adapts according to the extreme values found in the current population. We mutated all of the coefficients of 25 members of the population of 50, selected with uniform probability.

After mutation and crossover, the newly generated mutants and offspring are added to the population, so that 100 total solutions are present (50 from the previous population, 25 offspring and 25 mutants). The population of 100 is ranked according to fitness value, and the 50 lowest ranked members are culled, maintaining a population size of 50.

2.2.3. *Sequential evolution mechanism* The GA described thus far is fairly conventional, however we were compelled to add a unique aspect to our search—that of sequential evolution. In our algorithm, both breeding and mutation processes are carried out selectively. Since the contribution of each element (i.e. coefficient * variable) to the fitness value varies significantly, the algorithm concentrates on the coefficients which have the greatest effect on the fitness value. As an example, a higher level polynomial term would usually have a larger contribution to the final function value, y, than a linear term or a constant. The GA would concentrate efforts on the precise identification of the higher level terms' coefficients at the

expense of imprecision of the lower level terms' coefficients. In other words, the GA is sensitive to the combined effect of the coefficient values and their variable terms. We observed this effect as we applied GA search to larger test problems.

To overcome this sensitivity, we developed a sequential evolution mechanism. At each generation, only some of the coefficients are subject to crossover and mutation, while the values of the remaining coefficients are kept constant. The coefficients involved in evolution during a given generation are grouped so they have similar magnitude effects on the resultant, y. This grouping is rather coarse, and accomplished by identifying terms with similar contribution to the error measure. Establishing the groupings will be dependent on the error metric, the range of the data set, and the number and kind of functional terms. If it were not possible to group terms prior to evolution, arbitrary groups could be formed and altered during evolution to even out the contribution to the error. In the limit, the sequential evolution could take place in groups of one, i.e. each term evolves separately from the others.

The coefficient groupings are activated sequentially. Crossover and mutation work on the activated group exclusively for a set number of generations (although the fitness is always calculated using the entire solution). The groups are activated equitably so that all receive equal generations of evolution. More complicated strategies could be adopted where slower evolving groups are allocated more generations, or terminating a group's evolution could be based on an improvement metric rather than a preset number of generations.

3. Test problems and results

Three different test problems were used in this study to evaluate the performance of the genetic algorithm approach. Two of these three test problems were taken from previous GA studies on curve fitting.

The first test problem involves the estimation of two coefficients (C_1 and C_2) of a straight line given in equation 2. This was studied by Karr *et al.* (1991) with their GA approach.

$$y = C_1 x + C_2 \tag{2}$$

The second test problem, which is given in equation 3, has 10 independent variables and 9 coefficients to be estimated. This was studied for spline fitting using a GA by Rogers (1991) and before that for conventional spline fitting by Friedman (1988). A set of 200 randomly selected data points was used, and the problem is designed in such a way that the response depends only on the first five variables. The remaining variables, x_6 through x_{10}, have no effect on the response function, and they are included in the data set as 'red herring' noise. Therefore the optimal values of C_5 through C_9 should be zero.

$$y = C_1 \sin(\pi x_1 x_2) + C_2 (x_3 - 0{\cdot}5)^2 + C_3 x_4 + C_4 x_5 + \sum_{n=6}^{10} C_{n-1} x_n \tag{3}$$

The third test problem is a general second level equation of three variables. The function, which is given in equation 4, has 10 coefficients to be estimated, and is larger and more complex than any we found in the previous literature on using genetic algorithms for curve fitting. We employed the sequential breeding mechanism on this problem by dividing the coefficients into two groups. The first group—

the first four coefficients of equation 4—consists of the first degree terms and the intercept of the curve. The second group includes the six second degree coefficients of the equation.

$$y=C_1+C_2x_1+C_3x_2+C_4x_3+C_5x_1^2+C_6x_2^2+C_7x_3^2+C_8x_1x_2+C_9x_1x_3+C_{10}x_2x_3 \quad (4)$$

3.1. *Results from the first test problem*

The first test problem, its functional form, its coefficients, its initial search limits and its data set were taken directly from Karr *et al.* (1991). In this test problem six data points were used and true values of C_1 and C_2 were set as 1 and 0, respectively. The initial upper and lower search limits for the coefficients were set to ± 32. The algorithm was run four times with different random number seeds. We used the squared error metric in the objective function.

The final numerical values for C_1 and C_2 and squared error after the sixth generation are given in Table 1. Also included in Table 1 are the coefficients and squared error found after six generations of the binary GA used by Karr *et al.* (1991). Although both our approach and Karr's gave very good coefficients after only a few generations, our squared error was more than an order of magnitude smaller than Karr's and our population size (and thus solutions searched) was half of Karr's 100.

3.2. *Results from the Second Test Problem*

The function modelled (equation 3) has ten variables with nine coefficients. The function is dependent only on first five variables, and the coefficients of remaining five variables are set to zero. Thus, the remaining five variables are only used to add noise to the data. Roger's work on this function (Rogers 1991) was aimed at determining whether his GA (G/SPLINE) could set the last five coefficients to zero and how it compared to Friedman's MARS spline technique (Friedman 1988), a non-GA approach.

The performance of our GA over evolution in terms of $\log_{10}$ error is shown in Fig. 1 and the final coefficient values compared to the correct ones are shown in Table 2. Direct comparison of our GA with Rogers' is impossible, but a rough comparison can be made. In this test problem we used 100 data points, and the upper and lower limits of the initial search area were set to ± 10; Rogers used 200 data points with noise, with a signal to noise ratio of 4·8, added to the response, y. Rogers ran his GA for about 10 000 solutions which, for our population size, translates to about 200 generations. Rogers' squared error was 1·17 and Friedman's was 1·12,

Trial	C_1	C_2	Squared error
1	0·994865	0·054508	0·010059
2	1·006897	−0·01569	0·003333
3	1·012757	−0·06632	0·013703
4	1·004273	−0·02298	0·001614
Mean of 4	1·004698	−0·012621	0·007177
(Karr *et al.* 1991)	0·9688	0·0000	0·127

Table 1. Comparative results of the first test problem.

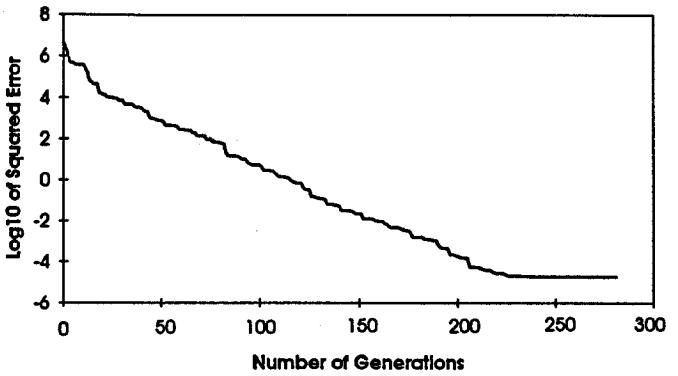

Figure 1. Log of error of test problem 2 over evolution.

however Friedman's took computational effort roughly equivalent to 1600 generations of our GA. Since they used data with an error term, the optimal that the techniques of Rogers and Friedman could do was 1·08 (resulting in a net squared error of 0·09 (Rogers) and 0·04 (Friedman)). This compares to our squared error after 200 generations of 0·000158. Therefore, we can conclude that our GA achieves orders of magnitude improvement over Rogers' G/SPLINE for the same effort, and achieves both far smaller errors and quicker convergence than Friedman's MARS. Table 3 summarizes this comparison.

3.3. *Results from the third test problem*

This is a general second order function of three variables. We chose this test problem to perform more detailed analysis of the GA, and its robustness. We tested robustness over (1) initial range for coefficients, (2) error metrics, (3) randomly selected data set, (4) number of data points, (5) range of the data points, and (6) underlying distribution of the data points. These last four criteria had to do with the robustness with respect to the characteristics of the data set to which the curve is to be fitted. After experimentation, we selected a rotation period of 20 generations for two groups of coefficients for the sequential evolution.

As a first step we evaluated the robustness of the GA with respect to the initial search intervals on all of the coefficients. The initial search intervals that we used

Coefficient number	True value	GA value
1	10	10·000023
2	20	19·999958
3	10	10·000037
4	5	4·999990
5	0	0·000012
6	0	0·000000
7	0	−0·000003
8	0	0·000013
9	0	0·000019

Table 2. Final results of test problem 2 after 250 generations.

Technique	Gross error	Net error*	≈Generations
(Rogers 1991)	1·17	0·09	200
(Friedman 1988)	1·12	0·04	1600
Our GA	0·000158	0·000158	200

*Net error for Rogers and Friedman is a result of their noisy data.

Table 3. Summary of comparison of results on test problem 2.

were (1) -120 to -100, (2) -500 to $+500$, (3) -100 to $+100$ and (4) 0 to $+20$. Apart from the first trial, all initial search intervals covered the true values of the coefficients. The performance of the algorithm for different search ranges is shown in Fig. 2. As expected, the fastest convergence is achieved when the search interval is [0, $+20$], a small interval centred about the true values of the coefficients. Similarly, the algorithm is slowest when the interval is between [-120, -100], which does not cover the true values of coefficients. Note that the GA did not stall in a local minimum in any instance, and again, seems quite robust to initial selection of coefficient ranges in regard to final solution quality, if not search speed.

For error metrics, we minimized total squared error (SE), total absolute error (AE) and maximum absolute error. Figure 3 illustrates the best-of-generation error for the three different fitness criteria used. For both AE and SE the algorithm converges similarly. However, the rate of convergence is greater when SE is used as a fitness criterion, as opposed to AE. The primary reason for faster convergence is due to the nature of the curve fitting problem: the range of fitness values observed is much wider for SE as compared to AE. Thus, in SE problems, members with worse fitness values are heavily discriminated against, and they have very little chance to breed into the next generation. The maximum error metric also converged quickly.

After setting each coefficient of equation 4 equal to 10, we created six separate randomly drawn data sets of 50 points each. The values for coefficients found and the squared error are given in Table 4. As can be seen, the GA was extremely robust across whatever data set was chosen randomly. Coefficient one, the intercept, merits further discussion. The precision attained on this coefficient was less than other

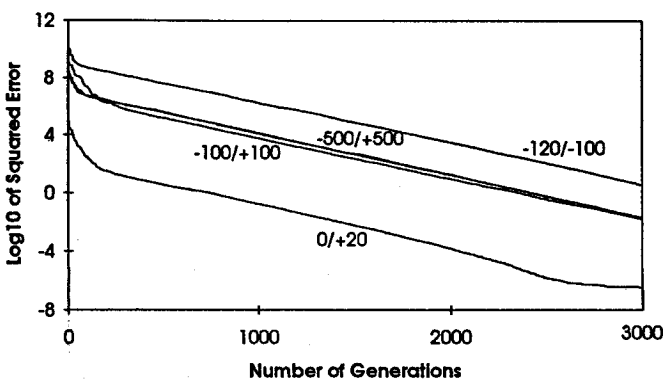

Figure 2. Log of squared error over evolution for different initial coefficient ranges.

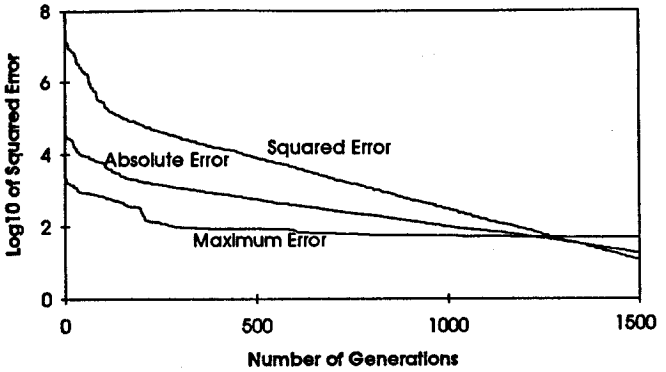

Figure 3. Log of error over evolution for different error metrics.

coefficients. This was because it was difficult to fine tune a parameter which is not dependent on x and has a very small contribution to the overall squared error relative to the other terms in its sequence. A potential improvement in the accuracy of fitting the intercept would be to group the intercept coefficient by itself for the sequential evolution.

As a final step, we set up a full factorial design to investigate the performance of the algorithm under three different sampling parameters. The first parameter was the number of data points on which the curve is fitted. We used 25 and 100 data points. The second parameter was the range which is described by upper and lower limits of data points. We used four different ranges: ± 500, ± 100, ± 50 and ± 10. The third parameter was the nature of distribution of the independent random variables from which the data points were selected. We used three types of distributions: a uniform distribution between the prescribed upper and lower limits, an exponential distribution and a normal distribution. The mean of the exponential distribution was selected to be the upper bound of the range, then we shifted the distribution to include negative numbers by subtracting off the upper bound. The mean of the normal distribution was selected to be the midpoint of the range, and the standard

Coeff.	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Mean	Std. dev.
1	9·986	9·998	10·002	10·000	9·996	10·001	9·997	0·005
2	9·999	10·000	10·000	10·000	10·000	10·000	10·000	0·000
3	10·000	10·000	10·000	10·000	10·000	10·000	10·000	0·000
4	10·000	10·000	10·000	10·000	10·000	10·000	10·000	0·000
5	10·000	10·000	10·000	10·000	10·000	10·000	10·000	0·000
6	10·000	10·000	10·000	10·000	10·000	10·000	10·000	0·000
7	10·000	10·000	10·000	10·000	10·000	10·000	10·000	0·000
8	10·000	10·000	10·000	10·000	10·000	10·000	10·000	0·000
9	10·000	10·000	10·000	10·000	10·000	10·000	10·000	0·000
10	10·000	10·000	10·000	10·000	10·000	10·000	10·000	0·000
Sq. err.	0·0017	0·000	0·000	0·000	0·000	0·000	0·000	—

Table 4. Results of test problem 3 across randomly drawn data sets.

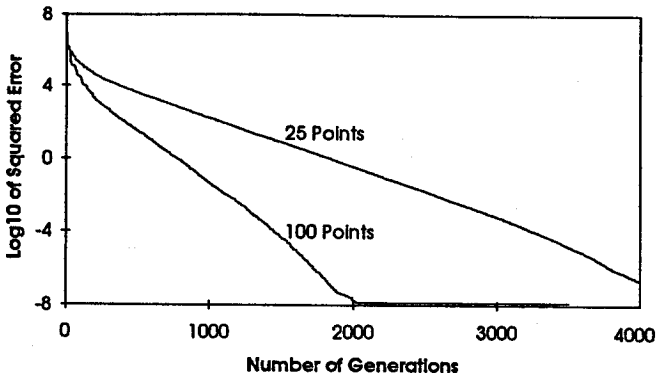

Figure 4. Log of squared error during evolution for different numbers of data points.

deviation was chosen to be 0·167 times the range. Thus, we ran 2*4*3 = 24 versions of the genetic search.

Examining the results of these 24 runs, we reached the following conclusions. First, when the number of data points was increased, the algorithm converged faster (Fig. 4). Second, when data points were selected from a narrower range, the algorithm converged faster (Fig. 5). Finally, the convergence process was not very sensitive to the distribution of the underlying random variables. We selected two extremes of our designed experiment—*hard* (25 data points with a data range of ± 500) and *easy* (100 data points with a range of ± 10)—and ran ten data sets of each distribution. Figures 6 and 7 show the mean of these ten runs over evolution. Since the uniform distribution is the only unbiased sample, one would expect that it would generally give the best results, however the normal distribution was slightly quicker in convergence than the uniform. The exponential distribution, a skewed

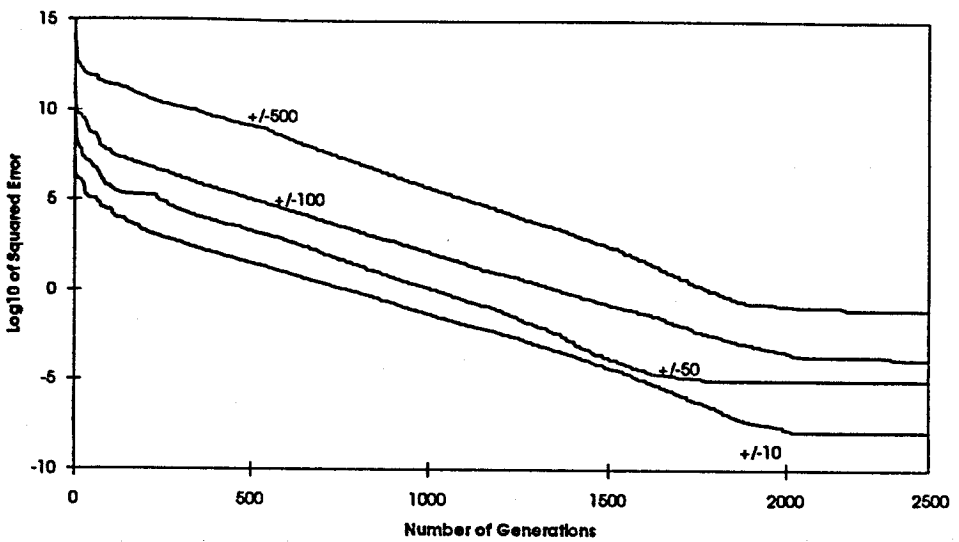

Figure 5. Log of squared error during evolution for different data ranges.

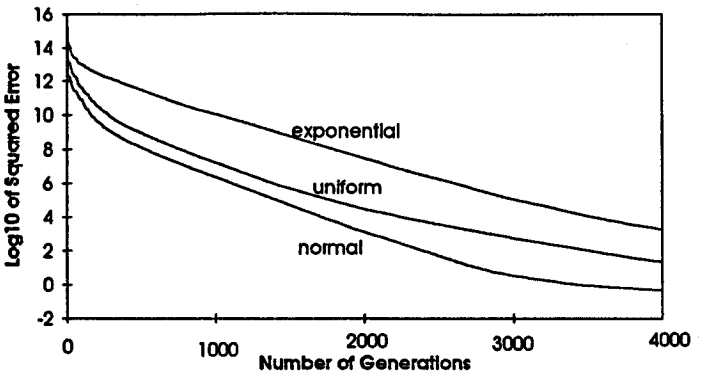

Figure 6. Log of squared error during evolution for different population distributions for data sampling for 25 data points with range = ± 500-*Hard*.

distribution, converged slower than the two symmetric distributions, but still achieved error rates similar to the symmetric distributions. While this designed experiment has encouraging results, clearly more work is needed to fully describe the relationship between the rate of GA convergence and functional form, number of coefficients, number of data points, and method of data point sampling.

4. Conclusions

In this study, we implemented a GA to perform curve fitting. We observed that the GA was fairly robust with respect to problem instance, data set size, population distribution and interval for sampling, error metric, initial coefficient range and random number seed. This robustness, as well as the ease with which search and problem parameters can be altered, encourages us to believe that a GA approach to curve fitting is both viable and versatile. Our results do indicate that the computational effort to reach an accurate solution is dependent on the complexity of the functional form that is to be fitted to the data. Sequential evolution seems to offer a workable and efficient way to attain equitable precision on the coefficients of the

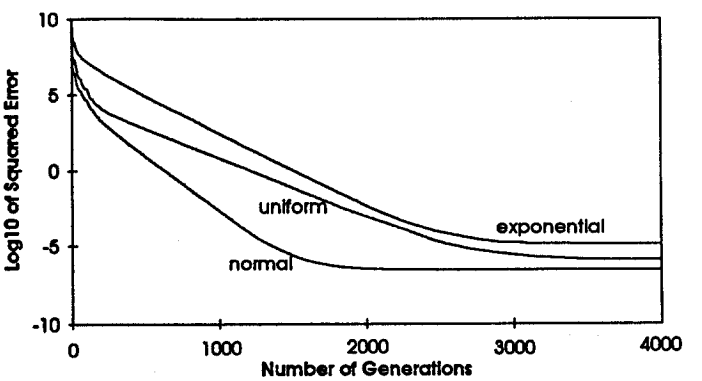

Figure 7. Log of squared error during evolution for different population distributions for data sampling for 100 data points with range = ± 10-*Easy*.

function. More research on the scale-up of the GA approach is needed. In particular, the sequential evolution mechanism could be made more sophisticated as it is adapted to larger, more complex problems.

Attention should also be given to the possibility of the selection of functional forms by the GA. Ideally, a curve-fitting approach would select not only coefficients, but also functional form. Our GA did this in a primitive manner for test problem 2, where the last five coefficients were not selected (i.e. set to zero). Manela *et al.* (1993) also attempted something similar when they used the GA to seek not only spline coefficients but also the degree of polynomial. Function selection for data was also discussed by Koza (1992). He termed this 'symbolic regression' and briefly illustrated the problem by selecting functional terms and their interactions on two small problems. We are currently working on expanding our method to include selection of functional form and term interaction.

References

ANDERSON, E. J., and FERRIS, M. C., 1994, Genetic algorithms for combinatorial optimization: the assembly line balancing problem. *ORSA Journal on Computing*, **6**, 161–173.

BÄCK, T., and SCHWEFEL, H.-P., 1993, An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, **1**, 1–23.

BEAN, J. C., 1994, Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, **6**, 154–160.

DAVIS, L, 1985, Job shop scheduling with genetic algorithms. *Proceedings of an International Conference on Genetic Algorithms*, pp. 136–140.

DEJONG, K. A., 1975, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. thesis, University of Michigan.

FRIEDMAN, J., 1988 (revised 1990), Multivariate adaptive regression splines. *Department of Statistics Technical Report 102*, Stanford University.

GOLDBERG, D. E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Reading, MA: Addison-Wesley).

HOLLAND, J. H., 1975, *Adaptation in Natural and Artificial Systems* (Ann Arbor: University of Michigan Press).

HOUSE, R. L., and DAGLI, C. H., 1992, An approach to three-dimensional packing using genetic algorithms. In *Intelligent Engineering Systems Through Artificial Neural Networks, Volume 2* (C. H. Dagli, L. I. Burke and Y. C. Shin, eds) (New York: ASME Press), pp 937–942.

JOG, P., SUH, J. Y., and VAN GUCHT, D., 1991, Parallel genetic algorithms applied to the traveling salesman problem. *SIAM Journal of Optimizaton*, **51**, 515–529.

JULIFF, K., 1993, A multi-chromosome genetic algorithm for pallet loading. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 467–473.

KARR, C. L., STANLEY, D. A., and SCHEINER, B. J., 1991, Genetic algorithm applied to least squares curve fitting. *U.S. Bureau of Mines Report of Investigations* 9339.

KOZA, J. R., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (Cambridge, MA: MIT Press).

LEVITIN, G., and RUBINOVITZ, J., 1993, Genetic algorithm for linear and cyclic assignment problem. *Computers and Operations Research*, **20**, 575–586.

MANELA, M., THORNHILL, N., and CAMPBELL, J. A., 1993, Fitting spline functions to noisy data using a genetic algorithm. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 549–556.

MÜHLENBEIN, H., and SCHLIERKAMP-VOOSEN, D., 1993, Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. *Evolutionary Computation*, **1**, 25–49.

NISSEN, V., 1994, Solving the quadratic assignment problem with clues from nature. *IEEE Transactions on Neural Networks*, **5**, 66–72.

ROGERS, D., 1991. G/SPLINES: A hybrid of Friedman's multivariate adaptive regression splines (MARS) algorithms with Holland's genetic algorithm. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 384–391.

STORER, R. H., WU, S. D., and VACCARI, R., 1992. New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, **38**, 1495–1509.

TATE, D. M., and SMITH, A. E., 1993. Genetic algorithm optimization applied to variations of the unequal area facilities layout problem. *Proceedings of the Second IIE Research Conference*, Los Angeles, CA, pp. 335–339.

TATE, D. M., and SMITH, A. E., 1994. A genetic approach to the quadratic assignment problem. *Computers and Operations Research* (in press).

TATE, D. M., TUNASAR, C., and SMITH, A. E., 1994. Genetically improved presequences for euclidean traveling salesman problems. *Journal of Mathematical and Computer Modelling*, **20**, 135–143.

WHITLEY, D., STARKWEATHER, T., and SHANER, D., 1991, The traveling salesman and sequence scheduling: quality solutions using genetic edge recombination. In *Handbook of Genetic Algorithms* (L. Davis, ed) (New York: Van Nostrand Reinhold), pp. 350–372.