GEOPHYSICAL RESEARCH LETTERS, VOL. 18, NO. 12, PAGES 2177-2180, DECEMBER 1991

GENETIC ALGORITHMS: AN EVOLUTION FROM MONTE CARLO METHODS FOR STRONGLY NON-LINEAR GEOPHYSICAL OPTIMIZATION PROBLEMS

Kerry Gallagher[1], Malcolm Sambridge[2] and Guy Drijkoningen[3]

Abstract In providing a method for solving non-linear optimization problems Monte Carlo techniques avoid the need for linearization but, in practice, are often prohibitive because of the large number of models that must be considered. A new class of methods known as Genetic Algorithms have recently been devised in the field of Artificial Intelligence. We outline the basic concept of genetic algorithms and discuss three examples. We show that, in locating an optimal model, the new technique is far superior in performance to Monte Carlo techniques in all cases considered. However, Monte Carlo integration is still regarded as an effective method for the subsequent model appraisal.

Introduction

Techniques for multi-parameter non-linear optimization may be conveniently classed into two groups. Methods in the first group use an iterative approach, and rely on local information on the gradient of some objective function to improve upon some appropriate starting model. Included in this class are the well known matrix inversion and single gradient methods. The second class of methods require no derivative information, avoiding a linearization of the problem, and instead use random processes to search the model space and find models which have smaller value of the objective function. The most well known of these global methods is the Monte Carlo (MC) technique, effectively a memoryless random walk. Thus, when generating a new model, MC neglects to make use of the information gained from the sampling of previous models, and instead relies totally on random exploration of the model space. The local methods, in contrast, rely on exploiting the limited information derived from a comparatively small number of models and avoid extensive exploration of the model space. In practice, many geophysical optimization problems are non-linear and result in irregular objective functions. Consequently, the local methods can depend strongly on the starting model, are prone to entrapment in local minima and can often become unstable. In addition the calculation of derivative information can become difficult and costly. The global methods avoid nearly all of the limitations of the local methods and are therefore more attractive for problems which are not too labour intensive in forward modelling. However, using MC to locate a near optimal solution always involves a significant exploration of unfavourable regions of the model space. Nevertheless one can use this exploration to build up an estimate of the a-posteriori covariance matrices and from these obtain confidence limits on the model parameters and assess the resolving power of the data (e.g. Tarantola and Valette, 1982). Ideally, one would like a more efficient technique to locate the solution, and then MC or another appropriate technique must still be used for the error analysis stage.

Genetic algorithms (GA), like MC methods, are completely non-linear, use random processes and require no derivative information, yet they have potential for significant increases in efficiency over the random walk strategy for the location stage of the inversion process. The original development of GA is attributed to Holland (Holland 1975) and recent summaries of progress in this field have been given by Grefenstette (1987), Goldberg (1989) and Davis (1990). Genetic algorithms are related to Simulated Annealing methods in that they are both stochastic search techniques, employing probabilistic approaches to improve the solution. In this paper we shall consider GA only. Comparisons between the two can be found in Davis (1990) and Scales, Smith and Fischer (1991). The basis of the GA approach is that large and complex models are represented by simple binary strings. These bit strings can then be manipulated using simple procedures which have an analogy with the way biological systems evolve to produce more successful organisms. The ability of GA to use the information contained in successful models, while still exploring the model space, suggests that they would be more efficient than MC methods in solving optimization problems. Although GA are simple to use, to date they have received little attention from geophysicists.

In this paper we outline the basic methodology of GA, however we restrict ourselves to considering the GA solely as an optimization technique and, at present, do not consider its potential for the error analysis stage of an inversion. We assume that this would be performed about the solution with the most appropriate technique available.

Genetic algorithms in non-linear optimization

The optimization problem to be considered is as follows: suppose we have a set of M unknowns x_i, denoted by the model vector $\mathbf{m}$, and an objective function, $\phi(\mathbf{m})$. For each parameter we have a pair of hard bounds, a_i and b_i, such that $a_i \leq x_i \leq b_i$, and some discretization interval d_i so that all allowable models, $\mathbf{m}$, represented by the set of parameters x_i, are restricted to the set $x_i = a_i + j \times d_i$, where $j = 0, \ldots, N_i$. Usually the objective, or cost, function, $\phi(\mathbf{m})$, represents the misfit between some observed data and the corresponding predictions of the model, and one is interested in examining the range of models that give a value of $\phi(\mathbf{m})$ less than some specified limit.

MC is implemented by randomly selecting models from the finite model space and calculating each value of $\phi(\mathbf{m})$ in turn. In contrast, GA work with a group of Q models simultaneously, initially chosen at random, and code each into a binary string. For example, the three parameter sequence (18, 28, 6) may be replaced by the binary string (1,0,0,1,0,1,1,1,0,0,0,0,1,1,0) where each sub-string of five elements are the binary representation of the respective base ten values. The patterns in the binary string repre-

[1] Dept. of Earth Sciences, O.U., Milton Keynes, and Dept. of Geological Sciences, U.C.L., UK

[2] Inst. of Theoretical Geophysics, Univ. of Cambridge.

[3] Dept. of Mining and Petroleum Engineering, TU Delft

Paper number 91GL02368
0094-8534/91/91GL-02368$03.00

sent different characteristics of the original three parameter model. The GA exploit this feature by using the sub-strings of the better data fitting models as building blocks in the generation of new models. The value of the cost function is used to control the likelihood that characteristics of individual strings will propagate into later generations of models. As the algorithm proceeds, the more successful models evolve and reproduce at the expense of the poorer models, similar to the survival of the fittest, where the fitter models produce lower data misfits.

A single iteration of GA proceeds in three stages:

i) Reproduction step: From a randomly selected initial population of Q bit strings and their cost functions $\phi(\mathbf{m}_k)$ $(k = 1, \ldots, Q)$, which we write as ϕ_k for short, an interim population of Q parents is generated by selecting models from the original group, with the likelihood of selection determined by the reproduction probability, $P_r(\phi_k)$, where $\sum P_r = 1$. This is achieved by dividing the unit interval (0,1) into Q segments with lengths equal to $P_r(\phi_k)$. Any real number between 0 and 1 can then be assigned to a particular model and by generating Q random numbers one produces Q models whose likelihood of selection is controlled by $P_r(\phi_k)$. Two common forms for $P_r(\phi_k)$ are linear, $P_r(\mathbf{m}_k) = a + b\phi(\mathbf{m}_k)$ and exponential, $P_r(\mathbf{m}_k) = A\exp\{B\phi(\mathbf{m}_k)\}$. Common choices for the constants are $b = -Q^{-1}(\phi_{\max} - \phi_{\text{avg}})^{-1}, a \geq -b.\phi_{max}$ and $B = -(\phi_\sigma)^{-1}, A = \left[\sum_j \exp\{B\phi_j\}\right]^{-1}$, where $\phi_{\max}, \phi_{\text{avg}}$ and ϕ_σ are the maximum, mean and standard deviation of the distribution of ϕ values in the current population, respectively.

ii) Crossover step: From the parent population we create a new generation of offspring models, each of which is derived from a mixing, or cross-over, of the bit strings from two parents. Initially, all parents are randomly paired off to produce Q/2 couples and then each pair are selected in turn for a possible crossover. A random number between 0 and 1 is generated to determine whether the current pair are to be crossed over. If the value is less than the constant P_c (the probability of crossover) then a position is chosen at random along the bit strings and two new strings are created by the cutting and transposing the two segments created by the cut. For example, when the two parents (1,0,0,1,0,1,1,1,0,0,0,0,1,1,0) and (1,0,0,0,0,1,1,1,1,0,0,0,0,1,1) are cut between the 4th and 5th site counted from the left, the strings (1,0,0, 1,0,1,1,1,1,0,0,0,0,1,1) and (1,0,0,0,0,1,1,1,0,0,0,0,1,1,0) are produced as offspring. If the random number is greater than P_c then the two parents are not selected for crossover and pass through to the offspring population unaffected.

iii) Mutation step: A mutation probability denoted by P_m is introduced and is defined as the probability that a single, randomly chosen, bit is altered in parity. P_m is usually rather small. For example if $P_m = 1/N_B$, where N_B is the number of bits per string, then one would expect on average that only one bit would be altered per string.

Each stage has an input of Q models and an output of Q models. Combining all three gives a new population of Q bit strings which may be used as input to the next iteration. The three bit string processes have different roles in the performance of the algorithms. The reproduction step effects the survival of the fittest between generations while the crossover step controls the degree of mixing and sharing of information that occurs between the models. The purpose of the mutation step is to keep a certain amount of diversity or randomness in the population, which would otherwise be gradually exhausted by the action of the previous two steps. Since mutation affects only a single model parameter, a relatively low value perturbs the whole model in a very restricted manner, comparable to a random local search about the original model. As the mutation probability is increased, and therefore the degree of randomness is increased, the algorithm becomes more like MC.

To apply the algorithm to a particular problem one must decide on the type of bit string encoding, the nature of the fitness function to be used, the size of the working population, Q, and the probabilities of crossover, P_c, and mutation, P_m. In practice it is usual to tune these parameters somewhat according to the problem being addressed. Grefenstette (1987), Davis and Steenstrup (1990) and Booker (1990) illustrate a variety of adaptations for different problems. However all of these modifications have at their core the basic algorithm described above.

To demonstrate the general mechanism consider the trivial problem of finding the maximum of the function $\phi = x^2$, in the range $0 \leq x \leq 127$. Table 1 summarises the performance of a genetic algorithm with the parameters $Q = 4, P_c = 1.0, P_m = 0.0$ and using a binary string of length seven. In this maximization problem we simply use the relative cost functions to control the reproduction likelihood, i.e. $P_r(m_k) = \phi(m_k)/\sum\phi(m_k)$. Note that during the reproduction step a model with the average fitness ϕ_{avg} would have an expectation value of 1 and so models with less than average fitness will tend to die off while those with above average fitness will survive. This property also holds for the linear probability function and is approximately valid for the exponential function, both de-

Table 1. Details of a simple genetic algorithm

It	i	x_i	Bit-string	$\phi(x)$	$P_r(x)$	i_{parent}	Cross point	Offspring	x_i
1	1	15	0001111	225	0.012	1	000—1111	0000101	5
	2	70	1000110	4900	0.265	3	110—0101	1101111	111
	3	101	1100101	10201	0.553	2	1—000110	1111000	120
	4	56	0111000	3136	0.170	4	0—111000	0000110	6
		ϕ_{avg}		4616					6696
2	1	5	0000101	25	0.001	2	110111—1	1101110	110
	2	111	1101111	12321	0.460	3	111100—0	1111001	121
	3	120	1111000	14400	0.538	2	1101—111	1101000	68
	4	6	0000110	36	0.001	3	1111—000	1111111	127
		ϕ_{avg}		6696					11874

scribed above. In this particular example we actually reach the optimal solution after generating two new populations. The likelihood of the same performance being achieved by MC is less than 1/10. Generally, however, we do not expect to reach an optimal solution so easily, and the real power of GA is their ability to generate near optimal solutions rapidly. In this simple example one can observe the rapid movement towards regions of good solutions by the progressive increase in the average fitness at each iteration.

Examples of Genetic Algorithm versus MC optimization

Although the multi-parameter minimization of a quadratic function is straightforward to perform using a local method, it provides a convenient test problem to compare the relative efficiencies of GA and MC in locating an optimal solution. Figure 1 shows the results of a minimization problem of the form $\phi(x_j, j = 1, N) = \sum_j a_j(x_j^* - x_j)^2$ where a_j is a constant and x_j^* is the solution, for $N = 3$ and 10. In both cases the performances have been averaged over 500 separate trials with different random sequences. The total length of the bit strings were 23, and 72 in the $N = 3$ and 10 problems respectively. A population size (Q) of 32 was used together with a linear probability function, and parameters in the range $0.8 < P_c < 1.0$ and $P_m = 0.001$. These choices were made with little exploration of alternatives and may be far from optimal. Nevertheless, it is clear that the GA show a dramatic improvement in performance relative to MC as the size of the problem increases. By calculating the ratio of the number of models sampled by GA and MC, that are required to achieve the same reduction in cost function, one may get a measure of the relative speeds of the two algorithms. This calculation shows that the speed of the GA increases exponentially relative to MC as the number of models sampled is increased in both cases. Lower bounds on the speed ratio can be gained by linear extrapolation of the speed curves. For the $N = 10$ case a speed ratio of 200 was found after 1920 sampled models. In this simple example the speed of GA over MC increases non-linearly as the dimension of the problem increases.

Although this example illustrates some of the differences in performance of GA and MC, the shape of the cost function is regular and varies smoothly in model space. An application to a geophysical problem involving real data provides a more convincing test. We take an example from the work of Sambridge and Drijkoningen (1991) who have compared the performance of GA to MC in non-linear seismic waveform inversion for 1–D marine velocity profiles. The data set used is the FF2 marine seismic refraction data from the 1959 Fanfare cruise of the Scripps Institution of Oceanography which consists of a suite of 25 seismograms. The waveform data is a high quality data set that has previously been interpreted by several authors, including Cary and Chapman (1988). They used 10 parameters to represent a 1–D seismic model down to a depth of 11.5 km below sea level. MC was used to identify the correct 'valley' in parameter space where the global minimum lay and linearized waveform inversion was used to refine the best model using 21 parameters.

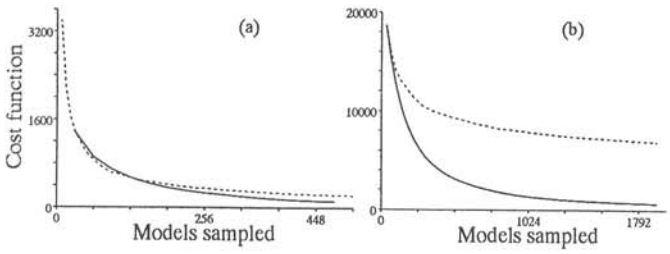

Fig 1. Decrease in N dimensional quadratic cost function against the number of models sampled averaged over 500 trials. MC (dashed) line and GA (solid line), (a) N=3, T=8.4 $\times 10^6$, (b) N=10, T=6.0 $\times$ 10^{23}, where T = total number of possible models.

Figure 2 shows the results of two different GA against an MC search. The same form of misfit function used by Cary and Chapman was used here (their best 21 parameter model was re-calculated using our misfit and gave $\phi_{wf} = 0.79$). Comparison of synthetics show that good waveform fits correspond to misfit values less than 0.8. In both cases the two GA use a linear mapping from waveform misfit to probability function and have bit strings of length 74. The first GA (solid line) has parameters $Q = 26$, $P_c = 1.0$, $P_m = 0.025$, while the second one (dotted line) has $Q = 100$, $P_c = 0.6$, $P_m = 0.01$. The total number of models in the discrete model space is 1.7×10^{17}. Clearly, the MC search results in an inferior misfit reduction than both GA, even after sampling more than 50 times as many models. Interestingly, the character of the two sets of curves are distinctly different. The GA show a cascade of small improvements to the best model (especially in the solid curve), while the memoryless MC shows only 4 or 5 improvements over the entire range of the curve. Curiously both solid line GA show periods of little improvement followed by an accelerated reduction in waveform misfit (seen at $\approx$ 2500 and 7000 models in the first case and 6000 in the second case). A possible explanation for this is that during the quiet periods, the population has effectively exhausted most of the information contained in the genetic patterns of its strings. The onset of improved performance is due to the introduction of extra information, through mutation, which is quickly incorporated into the new population. The best waveform misfit achieved by the GA for an 11 parameter model is $\phi_{wf} = 0.73$, slightly lower than that of the final 21 parameter model obtained by Cary & Chapman using linearized inversion. Although the differences between the two solutions is not thought to be very significant, the GA approach avoids the need for large scale matrix inversion. Both lie in the same region of model space and consequently one would expect that non-linear error analysis would yield similar results about either model. The important point here is that the GA have been able to identify this region of model space more efficiently than the more complex techniques used in previous studies.

Discussion

The major advantage of GA seems to lie in their ability to generate near optimal solutions rapidly. Their superior performance in the waveform inversion problem would suggest that they provide an efficient alternative for the location stage of a non-linear inverse problem. However it is important to follow the location stage with a non-linear error analysis and we have not explored the possibilities of GA in this area. At present MC integration seems to be the most comprehensive approach to this problem. Nevertheless it is worthwhile noting that in GA each successive population of models is derived using information obtained from previous populations. Therefore, one may monitor the refinements to the optimal solution, or the entire popula-

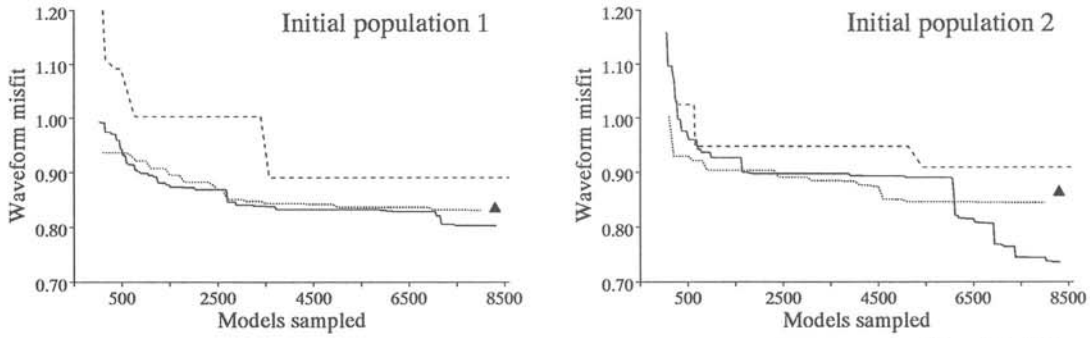

Fig 2. Performance of two different GA against MC. The solid and dotted lines are the GA with different values of Q, P_c, P_m, and the dashed line is MC. The two panels are for different random sequences. The triangles show the waveform misfits obtained by MC after 4.4×10^5 sampled models.

tion, to examine the structural changes in the 'best' model or even to detect competing classes of solutions.

Although GA have worked successfully in these examples we must, however, add a note of caution as this is still a relatively young field. There are some practical difficulties which should be highlighted and two problems are common. The first arises when a relatively good model makes multiple copies of itself early on in the evolution of the algorithm. Even if this model is not very close to the optimal model it may still reproduce itself at a fast enough rate to overwhelm the rest of the population, a problem known as premature convergence. Usually this occurs if the population size Q is too small and is easily avoided by increasing Q. The second problem arises when no model in the population is particularly good compared to any other model and so the cost functions are all about equal and driving force of the algorithm is lost. This is essentially the opposite case to the first problem and the usual way to cure it is to adjust, or re-scale, the cost function so that the reproduction probabilities of the models have a greater range. Other more sophisticated mechanisms for these problems are discussed in Davis (1990).

The tuning of the GA's free parameters Q, P_c and P_m is also rather *ad hoc*. Indeed, most of the research into GA seems to be centred on providing minor improvements to efficiency without much theoretical justification. It seems that a clearer theoretical framework is needed for a more thorough understanding of the subject. It is conceivable that a quantitative statistical basis for GA may be found in Bayesian Inference (see, for example, Backus 1988), and this is an area worthy of future study. In our opinion it is the underlying concepts of the GA (i.e. the use of the model fitness to control the likelihood of model survival, and the manipulation of the model at a binary level) that are its most important aspects, rather than the particular mechanism described here. There may be other formulations which are equally suited to geophysical problems but lend themselves more easily to theoretical analysis. The present formulation is nevertheless a very useful optimization tool and we expect that it will find many applications in geophysical problems.

Acknowledgements The authors are indebted to Chris Chapman & J. Huw Davies for some helpful discussions and encouragement, and to an anonymous reviewer for suggesting the relevance of Bayesian Inference.

References

Backus, G.E., Bayesian inference in geomagnetism, Geophys. J., 92, 125-142, 1988.

Booker, L., Improving search in genetic algorithms In L. Davis (Ed.), Genetic algorithms and simulated annealing, 61–73, Pitman London, 1990.

Cary, P.W., and Chapman, C. H., Automatic 1-D waveform inversion of marine seismic refraction data, Geophys. J., 93, 527–546, 1988.

Davis, L., (Ed.), Genetic algorithms and simulated annealing, Pitman, London, 1990.

Davis, L. and Steenstrup, M., Genetic algorithms and simulated annealing: An overview. In L. Davis (Ed.), Genetic algorithms and simulated annealing, 1–11, Pitman London, 1990.

Goldberg, D.E., Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading Massachusetts, 1989.

Grefenstette, J.J., (Ed.), Proc. of the 2nd int. conference on genetic algorithms and their applications, Hillsdale, NJ: Lawrence Erlbaum associates, 1987.

Holland, J.H., Adaptation in natural and artificial systems, Ann Arbor: The University of Michigan press, 1975.

Sambridge, M.S., and Drijkoningen, G., Genetic algorithms in seismic waveform inversion, Geophys. J. Int. (submitted), 1991.

Scales, J.A., Smith, M.L., and Fischer, T. L., Global optimization methods for highly nonlinear inverse problems, J. Comp. Phys., (submitted), 1991.

Tarantola, A., and Valette, B., Generalized nonlinear inverse problems solved using the least squares criterion, Rev. Geophys. Space Phys., 20, 219–232, 1982.

K. Gallagher, Dept. of Earth Sciences, Open University, Walton Hall, Milton Keynes, MK7 6AL, UK.

(Received: June 13, 1991;
Revised: August 19, 1991;
Accepted: September 11, 1991)