Geophys. J. Int. (1992) **109**, 323–342

Genetic algorithms in seismic waveform inversion

Malcolm Sambridge[1] and Guy Drijkoningen[2]

[1]*Institute of Theoretical Geophysics, Department of Earth Sciences, Downing St., Cambridge* CB2 3EQ, *UK*
[2]*Department of Mining and Petroleum Engineering, TU Delft, Mijnbouw Straat,*120, 2628 *RX Delft, The Netherlands*

Accepted 1991 November 9. Received 1991 November 8; in original form 1991 August 8

SUMMARY

Recently a new class of methods, to solve non-linear optimization problems, has generated considerable interest in the field of Artificial Intelligence. These methods, known as genetic algorithms, are able to solve highly non-linear and non-local optimization problems and belong to the class of global optimization techniques, which includes Monte Carlo and Simulated Annealing methods. Unlike local techniques, such as damped least squares or conjugate gradients, genetic algorithms avoid all use of curvature information on the objective function. This means that they do not require any derivative information and therefore one can use any type of misfit function equally well. Most iterative methods work with a single model and find improvements by perturbing it in some fashion. Genetic algorithms, however, work with a group of models simultaneously and use stochastic processes to guide the search for an optimal solution. Both Simulated Annealing and genetic algorithms are modelled on natural optimization systems. Simulated Annealing uses an analogy with thermodynamics; genetic algorithms have an analogy with biological evolution. This evolution leads to an efficient exchange of information between all models encountered, and allows the algorithm to rapidly assimilate and exploit the information gained to find better data fitting models.

To illustrate the power of genetic algorithms compared to Monte Carlo, we consider a simple multidimensional quadratic optimization problem and show that its relative efficiency increases dramatically as the number of unknowns is increased. As an example of their use in a geophysical problem with real data we consider the non-linear inversion of marine seismic refraction waveforms. The results show that genetic algorithms are inherently superior to random search techniques and can also perform better than iterative matrix inversion which requires a good starting model. This is primarily because genetic algorithms are able to combine both local and global search mechanisms into a single efficient method. Since many forward and inverse problems involve solving an optimization problem, we expect that the genetic approach will find applications in many other geophysical problems; these include seismic ray tracing, earthquake location, non-linear data fitting and, possibly seismic tomography.

Key words: genetic algorithms, global optimization, waveform inversion.

1 INTRODUCTION

Techniques for multiparameter non-linear optimization may be conveniently classed into two groups. Methods in the first group rely on using local information on the gradient of the objective function to improve upon some starting model in an iterative fashion. Included in this class are the well-known matrix inversion methods such as least squares and its variants, and also the single gradient methods such as steepest descent, conjugate gradients and simultaneous iterative reconstruction techniques. The second class of methods requires no derivative information (thereby avoiding a linearization of the problem) and instead uses random processes to search the model space and find better models. The most well known of these global methods is the Monte Carlo technique which requires only misfit function evaluations. Another type of method that requires only function evaluations is a directed grid search, such as that

used by Sambridge & Kennett (1986) for earthquake hypocentre location. However strictly speaking this procedure should be classed with the local methods, since it consists of a series of searches within restricted regions of parameter space. In general, local methods rely on exploiting the limited information derived from a comparatively small number of models and avoid extensive exploration of the model space. The Monte Carlo search is truly global since it is effectively a memoryless random walk. As a consequence when Monte Carlo generates a new model it neglects to exploit the information gained from the sampling of previous models, and instead relies totally on random exploration of the model space. (In this paper we use the term Monte Carlo to describe a purely random search of the parameter space and stochastic to describe any method which relies on random processes but which need not necessarily result in an overall random search.)

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, the Monte Carlo approach always involves a large degree of computation in exploring unfavourable regions of the model space. This usually means that a large number of models must be sampled and so Monte Carlo becomes prohibitively slow for large-scale problems and in such cases local methods are commonly considered the only viable approach.

A new class of global optimization methods known as genetic algorithms have recently been developed in the field of Artificial Intelligence. Like Monte Carlo methods they 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. The original development of genetic algorithms 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 (Kirkpatrick, Gelatt & Vecchi 1983) in that they are both stochastic search techniques, employing probabilistic mechanisms to solve complex optimization problems with multiple minima. Comparisons between the two can be found in Davis (1990) and Scales, Smith & Fischer (1991). Simulated Annealing methods use an analogy with physical annealing in thermodynamic systems whereas genetic algorithms have a resemblance with the genetic evolution of biological systems. In the past the major difficulty encountered with Simulated Annealing has been in deciding on an appropriate annealing schedule for a given problem, although Scales *et al.* (1991) have suggested that this problem has more or less been solved with the advent of the statistical mechanical approach introduced by Nulton & Salamon (1988) and extended by Andresen *et al.* (1988). Nevertheless Simulated Annealing algorithms are still inhibited by the 'critical slowing down' phenomena described by Brower *et al.* (1989) which, it is suggested arise because of the inherently inefficient mechanism by which they gather information on the large-scale structure of the misfit function. The Simulated Annealing method is well known to geophysicists since the work of Rothman (1985, 1986) on residual statics estimation (for a discussion see also Tarantola 1987). Genetic algotithms, however, have largely been confined to problems of Artificial Intelligence, and have received little attention from geophysicists, although, recently, they have been applied to a synthetic 1-D acoustic inverse scattering problem by Scales *et al.* (1991). Gallagher, Sambridge & Drijkoningen (1991) have discussed their use in geophysical optimization problems and Bolt (1991) has presented results of their use in estimating the bulk modulus of a decompressed lower mantle by finite-strain theory.

The power of genetic algorithms is that the optimization is driven completely by stochastic means. Essentially this means that the search mechanism does not follow a deterministic set of rules which force it to move away from a single point in parameter space in a pre-defined manner. However the use of a stochastic process does not mean that the algorithm searches randomly. Indeed one of the most interesting features of the algorithm is how some relatively simple procedures, requiring only random decisions, can lead to such an efficient type of search mechanism. An important ingredient of the genetic approach is that large and complex models are represented by simple binary strings, so that the patterns of 1's and 0's represent characteristics of the original model. These bit-strings can then be manipulated in a manner which has an analogy with the way biological systems evolve at a genetic level to produce more successful, or fitter, organisms. Genetic algorithms exploit the structure of the better data fitting models by using their components as building blocks to develop new models. In this way they combine the two important objectives of exploitation and exploration in a very efficient manner which makes them inherently superior to random search Monte Carlo techniques. There is every indication that genetic algorithms can be applied successfully to much larger scale problems than Monte Carlo without becoming computationally prohibitive.

In this paper we outline the basic methodology of genetic algorithms, illustrated through a simple example. To demonstrate some of the characteristics of the method we consider a multidimensional quadratic optimization problem and show that as the number of unknowns increases, genetic algorithms exhibit a dramatic improvement in efficiency over a Monte Carlo search. As an example of the performance of the technique in a real data problem, we consider the non-linear inversion of marine seismic refraction waveforms for 1-D velocity profiles. Finally we discuss the relationship of genetic algorithms to Simulated Annealing methods. The numerical examples presented here demonstrate the efficiency of genetic algorithms over Monte Carlo procedures, and also illustrate well the character of the method as it converges to a near optimal solution. The results lend considerable encouragement for their application to other non-linear geophysical optimization problems.

2 GENETIC ALGORITHMS IN NON-LINEAR OPTIMIZATION

The optimization problem to be considered is as follows: suppose we have a set of unknowns x_i (for $i = 1, \ldots, M$),

denoted as the model vector $\mathbf{m}$, and a non-linear objective function, $\phi(\mathbf{m})$. For each parameter we have a pair of bounds, a_i and b_i, such that $a_i \leq x_i \leq b_i$, and some discretization interval d_i such that

$$d_i = (b_i - a_i)/N_i \tag{1}$$

so that all allowable models, $\mathbf{m}$, represented by the set of parameters x_i, are restricted to the set

$$x_i = a_i + jd_i \qquad (j = 0, \ldots, N_i). \tag{2}$$

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. However, since this can usually be achieved by searching about an optimal model, or group of competing models, the aim of the problem is to seek out the model producing an optimal (usually minimal) value of $\phi(\mathbf{m})$. (For a maximization problem we usually replace the objective function with the term 'fitness' function.) In formulating a geophysical problem in this way we have implicitly performed a double discretization. The first is due to the use of a discrete set of parameters to represent, what in most geophysical applications will be a continuous function. This process is common in many inversion studies. Its validity and consequences are not the topic of this paper. In most cases, one may expand the appropriate unknown field or function as a set of known basis functions and use their coefficients as the discrete set of parameters, x_i (Parker 1977). The second discretization arises because the continuous range of each parameter has been replaced by a set of discrete intervals, which results in a discrete model space containing a finite number of models, $\mathcal{N}$, where

$$\mathcal{N} = \prod_{i=1}^{M} N_i \tag{3}$$

Of course the value of $\mathcal{N}$ alone does not describe the complexity of the optimization problem. This is primarily controlled by the length-scales on which the objective function varies relative to the interval size d_i and range $(b_i - a_i)$ for each parameter. One should take all of these quantities into account when assessing whether the optimization problems is highly non-linear. In most problems an appropriate set of bounds and intervals can be chosen *a priori*.

If the objective function is highly non-linear, then Monte Carlo techniques are usually more robust than linearized methods. Monte Carlo works by randomly selecting models from the finite model space and calculating each value of $\phi(\mathbf{m})$ in turn. In contrast genetic algorithms work with a group of Q models simultaneously, initially chosen at random, and code each into a binary string. For example, the two three-parameter decimal models (18, 28, 6) and (16, 30, 3) are replaced by the binary strings shown in Fig. 1(a), where each substring of five elements are the binary representation of the respective decimal values, with the dark squares representing 1 and the white squares 0. This we will refer to as 'regular' bit-string coding. Clearly, however, this is only one possible coding scheme of many. An alternative representation of the two models is shown in Fig. 1(b), where the bit ordering is based on the magnitude

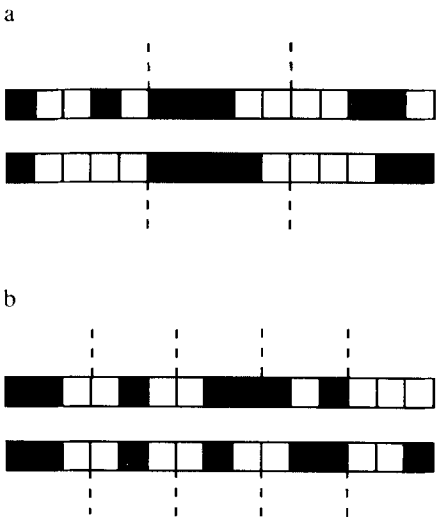


Figure 1. (a) Binary string representation of the three parameter decimal models (18, 28, 6) and (16, 30, 3). The dark squares represent a '1' and the light squares a '0'. (b) Same models as in (a) but re-ordered in terms of binary magnitudes.

components of the three parameters, i.e. the first three elements (1, 1, 0, . . .) indicate that the first two parameters, 18 and 28, are above 2^4 and the last, 6, is below, and so on for the next block of three. Many other types of coding are possible, each of which may have a different effect on the nature of the algorithm. Goldberg (1989) discusses a 'gray' coding scheme which has the property that any two sequential decimal values, say 18 and 19, are represented by bit-strings which differ by only a single element, i.e. a change in parity of any bit always results in a movement of one unit in a base 10 parameter. (Note: in the 'regular' and 'magnitude' coding schemes, described above, this property only occurs over limited ranges of decimal values.)

Genetic algorithms exploit the link between the model and its string representation by using the substrings present in the better data fitting models of the group to develop new models. At each stage the value of the cost function of each model is used to control the likelihood that characteristics of individual strings will propagate into later generations of models. An analogy with the evolution of biological systems arises through the fact that from one iteration to the next, the more successful models (with lower data misfits) will survive and reproduce themselves at the expense of the poorer models, in a manner akin to the survival of the fittest. The analogy continues in the nature of manipulations applied to the bit-strings at each iteration.

A single iteration of a minimization genetic algorithm proceeds in the following three stages.

(i) The reproduction step

From a randomly selected initial population of Q bit-strings and their cost functions $\phi(\mathbf{m}_k)$ $(k = 1, \ldots, Q)$, an interim population of Q parents is generated by selecting models from the original group, with the likelihood of selection determined by some measure of the cost functions. The probability of selecting the kth string is written as $P_r(\mathbf{m}_k)$

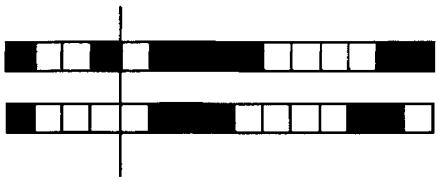


Figure 2. An example of crossover between the two models in Fig. 1(a). The first four elements of each pair have been transposed forming two new models.

and the two most common forms are linear,

$$P_r(\mathbf{m}_k) = a - b\phi(\mathbf{m}_k) \tag{4}$$

and exponential,

$$P_r(\mathbf{m}_k) = A \exp[-B\phi(\mathbf{m}_k)]. \tag{5}$$

Common choices for the constants are

$$b = Q^{-1}(\phi_{\max} - \phi_{\text{avg}})^{-1}, \qquad a \geq b\phi_{\max} \tag{6}$$

and

$$B = (\phi_\sigma)^{-1}, \qquad A = \left[\sum_j \exp(-B\phi_j)\right]^{-1}, \tag{7}$$

where $\phi_{\max}$, ϕ_{avg} and ϕ_σ are the maximum, mean and standard deviation of the distribution of ϕ values in the initial population, respectively.

(ii) The crossover step

From the parent population we create a new generation, referred to as offspring models, each of which is derived from a mixing (crossover) of the bit-strings from two parents. Initially, all parents are randomly paired off to produce $Q/2$ couples and then each pair is considered for a possible crossover. A random number between 0 and 1 is generated to determine whether the current pair is 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 in Fig. 1(a) are cut between the fourth and fifth sites from the left, the two strings in Fig. 2 are produced as offspring. These strings represent the models (16, 28, 6) and (18, 30, 3) and so the effect of crossover has been to swap a pair of the decimal parameters. This feature is relatively common when a pair of similar models become parents; however, in general decimals values may be both created and destroyed by crossover. If the random number is greater than P_c then the two parents are not selected for crossover and pass through the offspring population unaffected.

(iii) The mutation step

The final process applied to the offspring population of strings is the mutation of one or more randomly chosen bits. A mutation probability denoted by P_m is used to control the likelihood of an individual bit in each model being altered in parity. P_m is usually rather small (typically $P_m \leq 1/l$, where l is the length, or number of bits, per string). After the three steps are completed, a new population of Q bit-strings is produced which may be used as input to the next iteration.

The three bit-string processes perform different roles in the genetic algorithm. The reproduction step affects 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 two previous steps. In an example presented below it will be seen to be particularly useful in introducing new genetic material which, when beneficial, is quickly copied by other models. Since the mutation operation affects only a single model parameter, a relatively low value of P_m perturbs the model in a very restricted manner comparable to a random search in the neighbourhood of the original model. As the mutation probability is increased, the algorithm becomes more like Monte Carlo because an increasing degree of randomness is introduced, favouring exploration of the model space over exploitation of the information contained in the population. Before the method can be applied to a particular optimization problem several decisions need to be made. Specifically one must decide on the type of bit-string encoding, the nature of the reproduction probability to be used (i.e. linear, or exponential), 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 probelm being addressed. Grefenstette (1987), Davis & Steenstrup (1990) and Booker (1990) illustrate a variety of modifications to the algorithm which have been used for different problems. These include choosing a non-random initial population, variable selection methods and crossover probabilities, scaling of the cost function, re-ordering the bit-string and incorporating constraints on the model parameters. However, all of these modifications have at their core the basic algorithm described here.

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 summarizes the performance of a genetic algorithm using a regularly coded binary string of length seven ($l = 7$), and with the parameters $Q = 4$, $P_c = 1.0$, $P_m = 0.0$, i.e. crossover always performed for each set of parents and mutation never performed. In this case the problem is one of maximization and so we simply use the relative fitness functions to control the reproduction likelihood, i.e.,

$$P_r(x_k) = \frac{\phi(x_k)}{Q\phi_{\text{avg}}}. \tag{8}$$

Note that during the reproduction step a model with the average fitness ϕ_{avg} would have an expectation value of 1 copy, 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 (4) [with coefficients given by (6)] and is approximately valid for the exponential function (5) [with coefficients given by (7)]. In the example in Table 1 all four models in the initial population are selected to be parents during the first iteration, even though one model has a very small probability of selection ≈ 0.01. After crossover four new models are generated and the average fitness of the

Table 1. Progress of a simple genetic algorithm that maximizes $\phi = x^2$ for $0 \leq x \leq 127$, with $Q = 4$, $P_c = 1.0$, $P_m = 0.0$.

Iteration 1								
i	x_i	Bit-string	$\phi(x)$	$P_r(x)$	Parents	Crossover point	Offspring	x_i
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
Iteration 2								
i	x_i	Bit-string	$\phi(x)$	$P_r(x)$	parents	Crossover point	Offspring	x_i
1	5	0000101	25	0.0009	2	110111—1	1101110	110
2	111	1101111	12321	0.46	3	111100—0	1111001	121
3	120	1111000	14400	0.5378	2	1101—111	1101000	68
4	6	0000110	36	0.0013	3	1111—000	1111111	127
ϕ_{avg}			6696					11874

population is increased. During the second iteration the two poorer models are removed and two copies of models 2 and 3 make up the parent population. After crossover four new models are generated and again the average fitness of the population is increased. This second new population actually contains the optimal solution. It is interesting to note that the likelihood of the same performance being achieved by Monte Carlo is less than 1/10. Generally, however, we do not expect to reach an optimal solution so quickly, and the real power of genetic algorithms lies in 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 of the four models at each iteration. In general one could, if necessary, resort to using a local search method to refine a model after applying the genetic algorithm.

Although much progress has been made in the theoretical understanding of genetic algorithms since their inception, it is not known how to ensure an optimal implementation for a given problem. This is essentially because the algorithm applies a stochastic process to a finite population Q, and all performance estimates which are based on finite samples will inevitably have a sampling error associated with them. Repeated iterations of the algorithm compound the sampling error and lead to search trajectories which can be very different from those theoretically predicted (Booker 1990). This feature is evident in the first reproduction step of the example in Table 1, where the ratio of the highest to lowest selection probability is ≈ 63 and yet all four models have been copied once so that the initial group is unaltered. Because of the finite error in stochastic processes, most of the modifications to the standard genetic algorithm mentioned above are accompanied by semi-quantitative, or intuitive, reasoning as to why they should enhance performance. However, there is no guarantee that an improved performance will be achieved for a given application, and therefore many of these refinements appear to be rather *ad hoc*. Nevertheless some of the theoretical developments are useful for gaining insights into how genetic algorithms work and are worth a brief discussion.

Holland (1975) recognized that the success of the genetic algorithm is controlled by the way it processed 'schemata', which is the term used to describe classes of strings (or substrings) which share a common set of elements. For example, the schema represented by '1 **** 0' is used to represent all binary strings of six elements which begin with a 1 and end with a 0 (the * symbol is taken to represent either value). To quantify the growth and decay of schemata from generation to generation he introduced the 'fundamental theorem of genetic algorithms' (for a recent account see Goldberg 1989). The fundamental, or schema, theorem states that, for the genetic algorithm described in the maximization problem above, the expected number of copies m, of a schema H, at iteration $(t+1)$, is bounded by the expression

$$m(H, t+1) \geq m(H, t) \frac{\phi_s(H)}{\phi_{avg}} \times \left(1 - P_c \frac{d(H)}{l-1} - P_m o(H)\right), \quad (9)$$

where $d(H)$ is the defining length of the schema, i.e. the distance between the first and last fixed string position [$d(1 **** 0) = 5$], $o(H)$ is the order of the schema defined by the number of fixed elements in H[$o(1 **** 0) = 2$], ϕ_{avg} is the average fitness of the population, and $\phi_s(H)$ is the average fitness of all the strings in the current population which represent the schema H. [For a minimization problem the fitness function is given by the right-hand side of equation (4) or (5).] The schema theorem shows that the schema with above average observed performance, short defining length (i.e. small d) and low order (i.e. small o) will be sampled at exponentially increasing rates. Since each schema represents a complete class of bit-strings the theorem suggests that the generation of successful schema proceeds in a highly parallel fashion, leading to a rapid of exchange of information between the better and the poorer models of each generation. Holland (1975) has estimated that the number of schemata that are beneficially processed at each iteration is of the order of Q^3 even though only of the order of Q manipulations are performed. This process is commonly refered to as *Implicit Parallelism* in the Artificial Intelligence literature. At present, no optimization procedure used in geophysical applications can reasonably claim to benefit from such a process. Indeed there is no guarantee that genetic algorithms will take full advantage of their 'parallel' nature, but they seem, at least, capable of

exploiting it to some degree. In the next section we present two examples of the use of genetic algorithms, and in each case compare their performance to Monte Carlo methods. In the final section we discuss their relationship to Simulated Annealing methods.

3 EXAMPLES OF GENETIC ALGORITHM VERSUS MONTE CARLO OPTIMIZATION

3.1 Quadratic minimization

Although the multiparameter minimization of a quadratic function is straightforward to perform using a local method, it provides a convenient test problem to illustrate the relative efficiencies of genetic algorithms and Monte Carlo. Since the Monte Carlo technique, used here, involves nothing more than random search, a comparison with a simple genetic algorithm will show whether it is capable of anything more than just random exploration. In Fig. 3 we show the results from three quadratic minimization problems of the form

$$\phi = \sum_{j=1}^{M} a_j (x_j^* - x_j)^2 \qquad (10)$$

where the a_j are constants and x_j^* are the solution values, for $M = 3$, 6 and 10. In all cases the genetic and Monte Carlo algorithms have the same initial population of models, although, of course, the grouping of models together as a population has no effect on the Monte Carlo procedure. All curves in Fig. 3 are generated by averaging over 500 separate trials in order to eliminate the influence of the initial random population. A population size, Q, of 32 was used together with the linear probability function of equation (4), the regular bit-string coding and parameters in the range $0.8 < P_c < 1.0$ and $P_m = 0.001$. The total length of the strings, l, determined from the sum of the range of possible binary values for each model parameter, were 23, 40 and 72 in the $M = 3$, 6 and 10 problems respectively. These choices were made with little exploration of alternatives and may be far from optimal. Nevertheless, it is clear that the genetic algorithm shows a dramatic improvement in performance relative to Monte Carlo as the number of model parameters increases. This lends some support to the notion that genetic algorithms will remain practical for problems of much larger scale than are feasible with Monte Carlo.

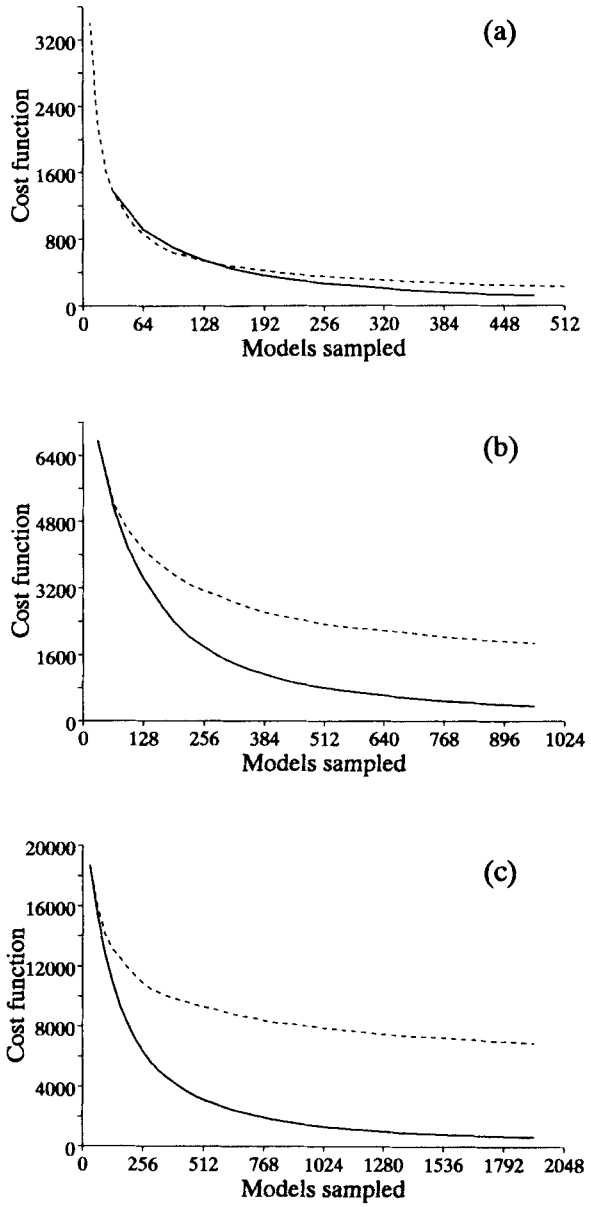


Figure 3. Reduction of the cost function of a genetic algorithm (solid line) and Monte Carlo (dashed line) in a quadratic minimization problem against the number of models sampled. The curves have been averaged over 500 trials with different initial random populations. (a) Three-parameter problem with a total of 8.4×10^6 possible models, (b) six-parameter problem with a total of 1.4×10^{14} possible models, (c) 10 parameter problem with a total of 6.0×10^{23} possible models.

3.2 Seismic waveform inversion

Although this simple example illustrates some of the differences in performance of genetic algorithms and Monte Carlo, 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 much more convincing test of the new method. The non-linear inversion of marine seismic refraction data for 1-D velocity profiles is a sufficiently complex problem, and is adequate for this purpose. The data set used in this example 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 using synthetic seismograms (Spudich & Orcutt 1980), linearized waveform inversion (Chapman & Orcutt 1985) and Monte Carlo together with linearized inversion (Cary & Chapman 1988, hereafter referred to as CC). The data were collected in a region of 15 Myr old oceanic crust east of Guadalupe Island in the Pacific Ocean using explosive charges as seismic sources with weights between 1 and 45 kg. The recordings were made from hydrophones suspended about 46 m below the sea surface. The frequency band of the data used in the inversion is

2.54–15.04 Hz with a peak frequency of approximately 7.23 Hz. A detailed account of the acquisition procedure and the instrumentation can be found in Spudich & Orcutt (1980). Forward modelling is achieved by means of Chapman's WKBJ seismograms (Chapman 1976, 1978; Dey-Sarkar & Chapman 1978). CC have compared the accuracy of Chapman's method with the more accurate reflectivity synthetics for the FF2 data set. They concluded that although the WKBJ synthetics do not model the entire signal in the observed seismograms (because of some reverberations) they are nevertheless sufficiently accurate for this type of study. Details of the topographical corrections and source time functions used during forward modelling are also given in Spudich & Orcutt (1980).

To cast the waveform inversion problem as one of optimization one requires a misfit function which describes the discrepancy between observed and synthetic seismograms. The function used in this work has exactly the same form as that used by CC, and is evaluated in the frequency domain using the expression

$$\phi_{\mathrm{wf}} = \frac{\sum_{j=1}^{J}\sum_{k=1}^{K}[d_{kj}\hat{A}(\omega_k)\hat{P}_{\mathrm{d}}(\omega_k, x_j) - cd_{kj}\hat{P}_{\mathrm{s}}(\omega_k, x_j)]^2}{\sum_{j=1}^{J}\sum_{k=1}^{K}[d_{kj}\hat{A}(\omega_k)\hat{P}_{\mathrm{d}}(\omega_k, x_j)]^2} \tag{11}$$

where $\hat{P}_{\mathrm{d}}(\omega_k, x_j)$ is the Fourier transform of the observed seismogram as a function of frequency ω_k and range x_j, $\hat{P}_{\mathrm{s}}(\omega_k, x_j)$ is the frequency spectrum of the corresponding synthetic seismogram and $\hat{A}(\omega_k)$ is a frequency filter,

$$\hat{A}(\omega) = \mathrm{sinc}^4(\omega/\omega_{\mathrm{N}}) \tag{12}$$

where ω_{N} is the Nyquist frequency ($\omega_{\mathrm{N}} = \pi/\Delta t$), and Δt is the time sampling of the observed seismogram. The frequency smoothing $\hat{A}(\omega)$ is required in order to stably implement Chapman's method. Since it is implicit in the synthetic seismograms, $\hat{P}_{\mathrm{s}}$, it must also be applied to the data. The constants d_{kj} control the frequency scaling of the components within each seismogram as well as the amplitude scaling between seismograms at different ranges. The original motivation for frequency scaling was to reduce the significance of the high-frequency data and thereby make the problem more tractable with linearized techniques (Chapman & Orcutt 1985). Since genetic algorithms do not make any linearizing assumptions, this issue would appear to be less important in the current study. Nevertheless it is still advisable to down weight the influence of the high frequencies, as Chapman & Orcutt (1985) point out, because these will be more susceptible to noise and consequently more difficult to fit. We therefore follow previous authors by assigning the weights

$$d_{kj} = \begin{cases} id_j/\omega_k, & \text{in the significant frequency band,} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The range-dependent weights d_j are chosen so that the misfit function (11) is not dominated by the higher amplitude seimograms at the expense of the lower amplitude seismograms. Again we follow previous authors and calculate the d_j so that the scaled spectral peaks of each observed seismogram are of approximately equal magnitude. The constant c controls the overall scaling between data and synthetics, and several different methods have been used for its calculation. In this paper we follow the approach of CC and determine c by comparing the envelopes of the data and synthetic waveforms. The appropriate expression for c is given by

$$c = \frac{1}{J}\sum_{j=1}^{J}\frac{E[P_{\mathrm{d}}(t_{\max}, x_j)]}{E[P_{\mathrm{s}}(t_{\max}, x_j)]} \tag{14}$$

where $E[\ ]$ is the envelope function (equivalent to the modulus of the complex trace, formed by an analytic extension of the real trace over the complex plane). The functions $P_{\mathrm{d}}(t_{\max}, x_j)$ and $P_{\mathrm{s}}(t_{\max}, x_j)$ are the observed and synthetic seismograms, in the time domain, both evaluated at the point were the synthetic envelope is a maximum, $t_{\max}$. This choice of scaling is more involved than simply matching the peak amplitude of the synthetics to the peak amplitude of the data, but avoids some of the problems that arise when the two peaks occur at very different times. Overall the misfit function (11) provides a reasonably sensitive measure of the mismatch between data and synthetics.

Finding the 1-D seismic P velocity model which produces a global minimum of (11) is a complicated multiple-minima optimization problem. The multiple minima will be of variable size and arise mainly when the synthetics are out of alignment with the observed seismograms but can also be caused by noise and triplications. For this particular data set CC found a significant local minimum in the misfit function nearby their final solution which corresponded to some of the synthetics arriving a period late with respect to the data. To locate the global minimum they found it necessary to use a two-stage method. Initially they employed a Monte Carlo technique to search for a reasonble velocity model within the valley of the global solution. This model was subsequently refined by performing a full linearized matrix inversion about the Monte Carlo solution. In the first stage the model depth was parametrized as a function of velocity $z(v)$ at 10 points. The velocity at any depth was represented by a series of linear gradient layers connecting each node. During the linearized inversion a more detailed model was possible and so 21 velocity parameters were used. In this work we parametrize the velocity model as a function of depth $v(z)$, down to a depth of 11.5 km, which allows the extra flexibility of incorporating low-velocity zones and interfaces if necessary. Initially we use 11 velocity parameters at fixed depths chosen to match those of the best model from their Monte Carlo search. The structure of the ocean/crust interface is assumed to be known, we use the model of Spudich & Orcutt (1980) down to a depth of 3.83 km and allow velocities to vary in the depth range $3.83 \leq z \leq 11.5$ km. The extremal velocity bounds of these authors have also been used as a guide in choosing the bounds of our velocity parameters, which are shown in Fig. 4, together with the homogeneous layer models of Spudich & Orcutt (1980) and the linear gradient model of CC. The velocity interval for each model parameter, dv, was set to $0.05\ \mathrm{km\,s^{-1}}$ which produced a total of 1.7×10^{17} possible models in the parameter space.

3.2.1 *Fitting traveltimes*

In the Monte Carlo search of CC all randomly generated models were initially tested by evaluating their traveltime

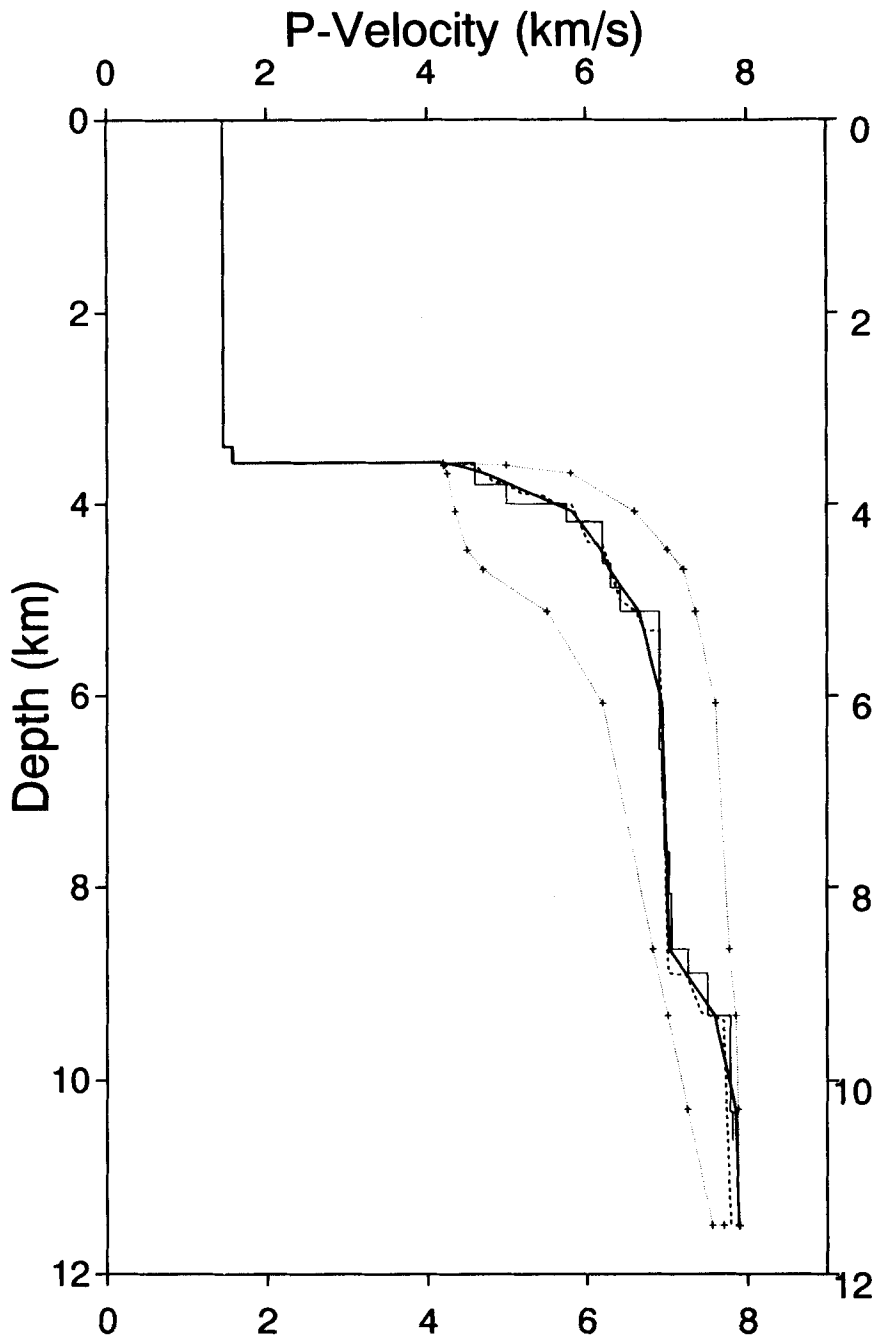


Figure 4. Velocity model bounds used in 11 parameter inversion. Crosses indicate the depths of each velocity parameter. The homogeneous layer model is from Spudich & Orcutt (1980), the linear gradient model (dashed) is the 21 parameter model of Cary & Chapman (1988), and the solid line is the best 11 parameter model obtained from the genetic algorithms in Fig. 6.

misfit ϕ_{tt}, where

$$\phi_{tt} = \frac{1}{N_A} \sum_{i=1}^{N_A} \frac{(T_i^o - T_i^s)^2}{(\Delta T)^2}. \quad (15)$$

This traveltime misfit differs slightly from that used by CC because it is normalized with respect to the traveltime bounds, ΔT. An average is taken over all N_A synthetic arrivals (in this case first P-wave arrivals) to avoid any problems arising from velocity models that do not produce geometrical arrivals (due to shadow zones). Only those models which had a synthetic arrival time within $\pm\Delta T$ were considered for waveform fitting, where $\Delta T = 0.15$ s. CC also impose bounds on the velocity gradients that may be generated within the model parametrization. This has the effect of biasing the search away from regions of the model space which may be considered unfavourable *a priori*. In this work we use a similar mechanism and constrain all velocity gradients to be in the range $0 \leq dv/dz \leq 16\,s^{-1}$. In practice these values exclude the possibility of low-velocity zones, which would create shadow zones and therefore no geometric arrivals, and also very high-gradient zones for which Chapman's method becomes inaccurate. The first

restriction is not likely to be too severe as none of the previous P velocity models for the regions has contained any low-velocity zones. The upper gradient bound is much higher than any realistic velocity gradient and so the maximum velocity gradients are effectively unbounded.

As a first illustration of the application of genetic algorithms to this problem we examined their performance in finding models that pass the traveltime test. In this case the traveltime misfit (15) was minimized using four different genetic algorithms with population sizes $Q = 26$, 50, 100 and 1000. In each case the reproduction probability was a linear function of the traveltime misfit (4), with coefficients given by (6), and the 1-D P velocity model was coded into a binary string using the regular coding scheme. The binary coding requires that the number of velocity intervals for each parameter is an integral power of 2. To achieve this with a fixed velocity interval $dv = 0.05\ \mathrm{km\ s^{-1}}$ it was necessary to widen some of the velocity bounds. Overall the number of intervals per velocity parameter varied between 32 and 128, and produced a total bit-string length of $l = 74$ for the 11 parameters. The probabilities of crossover, P_c, and mutation, P_m, were chosen to be in the range $0.6 \leq P_c \leq 1.0$ and $0.001 \leq P_m \leq 0.03$. We have applied only one modification to the simple genetic algorithm described in Section 2. This is to replace the worst model in the ith population, $\phi_{\mathrm{highest},i}$, with the best model from the previous iteration $\phi_{\mathrm{lowest},i-1}$ if, and only if, all copies of the model that produced $\phi_{\mathrm{lowest},i-1}$ were eliminated by the actions of crossover and mutation. In our experience this modification seems to improve performance in all cases.

The performances of the four genetic algorithms and a Monte Carlo procedure are illustrated in Fig. 5. This diagram shows the number of models passing the traveltime test, i.e. with arrival time within ± 0.15 s of the observed time, against the total number of models sampled, which does not include any copies generated during the genetic algorithm. It is clear that the genetic algorithms with $Q = 26$, 50 and 100 are an improvement on the Monte Carlo since they find more models that pass the traveltime test even though, strictly speaking, their goal is to minimize ϕ_{tt}. It is also evident that the efficiency of the genetic algorithms varies with population size, with a peak around $Q = 50$. If the population size is too small then not enough exploration of the parameter space is taking place, while if it is too high, the cross fertilization of information becomes very slow. (Note: in the $Q = 1000$ case the entire curve represents only seven iterations whereas in the $Q = 50$ nearly 160 iterations have been performed.)

3.2.2 Waveform fitting: genetic algorithm compared to Monte Carlo

The traveltime fitting is a straightforward way of finding a good mixture of models from which one can select an initial population for waveform fitting. It would be simple enough to select these models from those with the lowest value of ϕ_{tt}. However, CC have shown that the waveforms provide independent information from the traveltimes and so we do not wish to bias the procedure too heavily towards models that only produce good traveltime fits. Therefore, we prefer to select the initial population for the waveform fitting stage at random from the models that pass the traveltime test. In this way the minimizations of ϕ_{tt} produces an accelerated start to the waveform inversion. In the following examples a genetic algorithm with parameters $Q = 50$, $P_c = 1.0$,

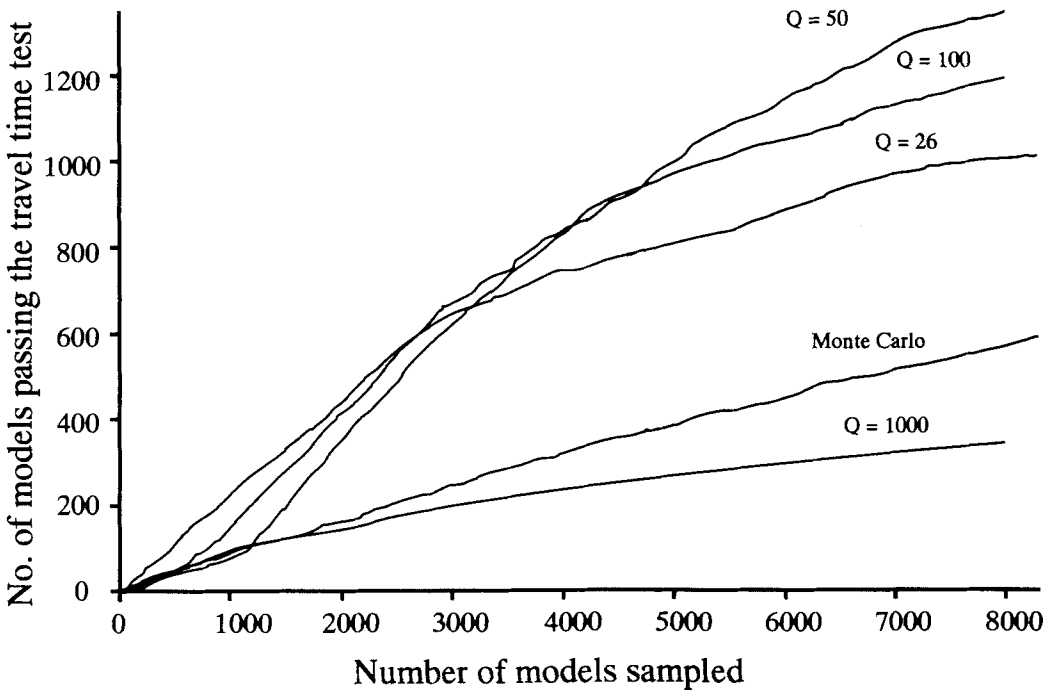


Figure 5. Performance of genetic algorithm and Monte Carlo in finding models that have traveltimes within ± 0.15 s of the observed values. Q is the population size in the genetic algorithm.

$P_m = 0.025$ is used for the ϕ_{tt} minimization and is halted after 160 iterations.

The results of two different algorithms applied to waveform inversion are shown for three different sets of initial populations in Figs 6(a), (b) and (c). Comparison of synthetic seismograms shows that reasonable waveform fits correspond to misfit values less than 0.85. The parameters controlling the genetic algorithms are given in Table 2. In each case a Monte Carlo search is shown which has the same initial sequence of models as in the corresponding minimization of ϕ_{tt}, i.e. the first Q models generated in the two procedures are exactly the same (remember that the initial population of the genetic algorithm is generated randomly). Therefore the differences between the two curves cannot be ascribed to a fortuitous initial population of models in the genetic algorithm. Clearly the performance curves show that the traveltime stage has given each genetic algorithm an advantage over the Monte Carlo. The relative efficiency of the two methods is also clear since all genetic algorithms achieve smaller values of ϕ_{wf} after 8000 models than the Monte Carlo searches do after 440 000 models (indicated by the solid triangles), although this is probably a severe underestimate of the relative efficiencies because each Monte Carlo would have to search many more models, possibly even orders of magnitude more, before they came across the solutions obtained by the genetic algorithms (which, of course, they eventually must because there are only a finite number of models in the entire space, i.e. 1.7×10^{17}).

It is reassuring to observe an increase in efficiency, even though we cannot gain any accurate estimates of it. However, what is perhaps more important is the manner in which the genetic algorithms reduce ϕ_{wf}. In all cases there are periods where the best model undergoes a cascade of improvements, which suggests that it is being refined by an efficient local search mechanism. In contrast each Monte Carlo search shows only five or six improvments during the first 8000 sampled models. (Indeed all three Monte Carlo searches achieve only five or six more reductions in ϕ_{wf} over the next 432 000 sampled models). The general character of the genetic algorithm performance trends appears to be one of a rapid initial decrease of misfit followed by quiet periods where little improvment is achieved, followed again by more cascade-like improvements [seen at about 2500 and 7000 models in Fig. 6(a) and 6000 in Fig. 6(c)]. 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 and the onset of improved performance is due to the random introduction of extra material, or information, through mutation, which is quickly copied and refined by other models in the population. The onset of secondary cascades after periods of little activity suggests that the genetic algorithm is also benefiting from a global-type search mechanism which allows it to jump out of the current local valley in the misfit surface. Once a new valley has been found the local nature of the algorithm takes over and efficiency finds the local minimum. Clearly the potential to take advantage of both a

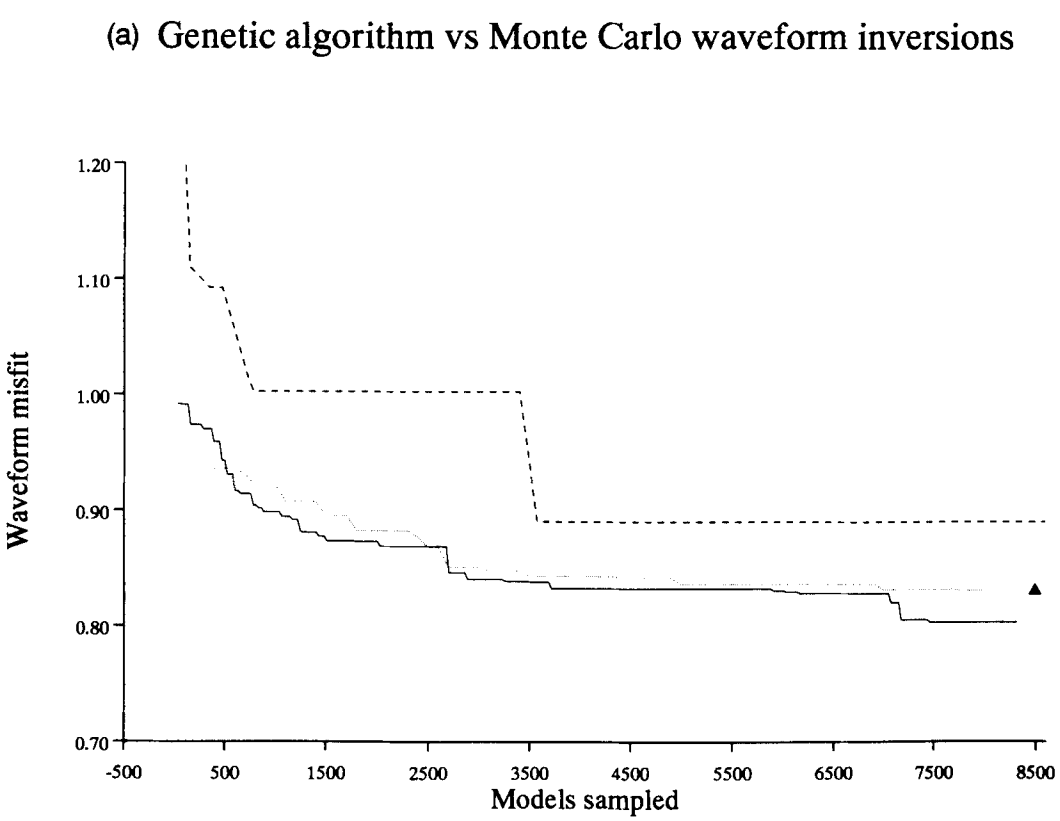
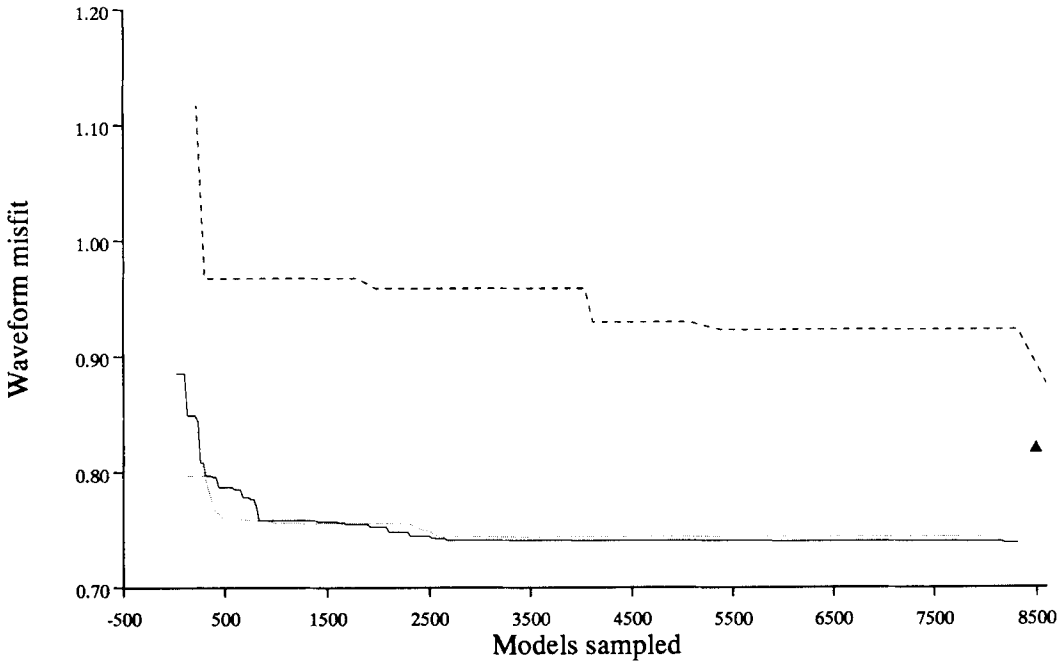


Figure 6. (a) Performance of two genetic algorithms (solid and dotted) against Monte Carlo (dashed) in minimizing the waveform misfit with an 11 parameter velocity model. The triangle indicates the value of the waveform misfit found by the Monte Carlo after sampling 440 000 models. The parameters controlling the genetic algorithms are given in Table 2. (b), (c) Same as in (a) but with different initial model populations.

(b) Genetic algorithm vs Monte Carlo waveform inversions



(c) Genetic algorithm vs Monte Carlo waveform inversions

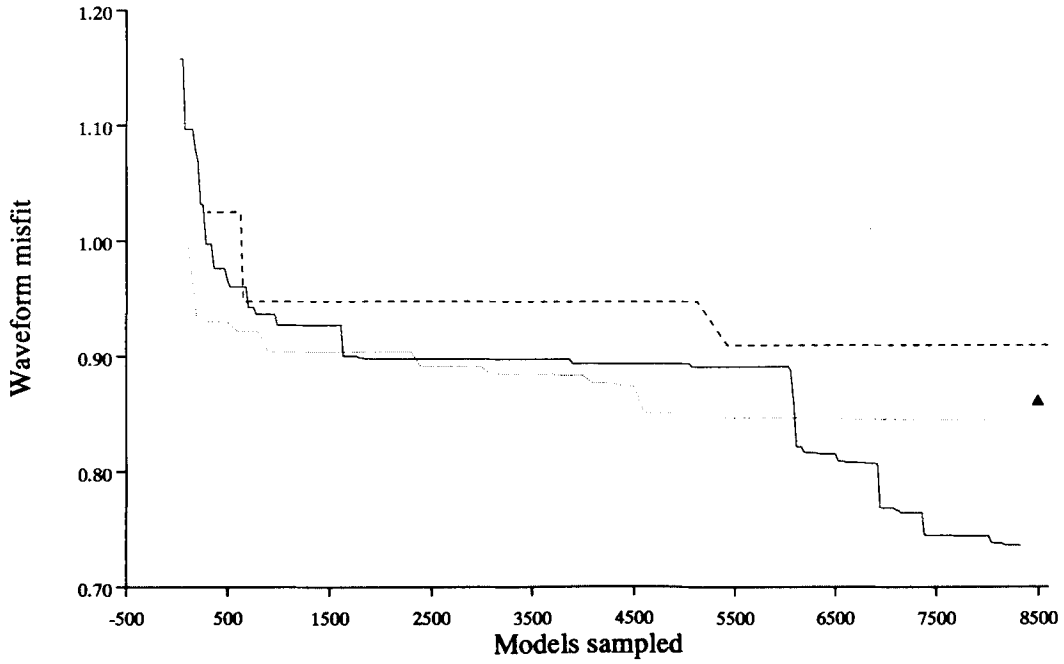


Figure 6. (*continued*)

Table 2. Control parameters and results of all genetic algorithms used in the 11 and 22 parameter waveform inversions.

11 parameter model ($N_B = 74, N_T = 1.7 \times 10^{17}$)							
Genetic algorithm parameters					min ϕ_{wf}		
	Q	P_m	P_c	N_S	*ran*1	*ran*2	*ran*3
ϕ_{tt}	50	.025	1.0	8000	–	–	–
*GA*1	26	.025	1.0	8000	0.804	0.738	0.736
*GA*2	100	.001	0.6	8000	0.831	0.744	0.845
MC	–	–	–	8000	0.891	0.923	0.909
MC	–	–	–	4.4×10^5	0.831	0.819	0.860

22 parameter model ($N_B = 131, N_T = 2.5 \times 10^{27}$)							
Genetic algorithm parameters					min ϕ_{wf}		
	Q	P_m	P_c	N_S	*ran*1	*ran*2	*ran*3
ϕ_{tt}	50	.025	1.0	200	–	–	–
*GA*1	50	.005	0.75	10000	0.717	–	–
*GA*2	50	.001	0.75	10000	0.726	–	–
*GA*3	50	.005	0.55	10000	0.737	–	–
MC	–	–	–	10000	0.763	0.763	0.799
MC	–	–	–	4.4×10^5	0.763	0.757	0.750

local and global mechanism is a very desirable feature when one wishes to find the global minimum of an irregular misfit surface containing multiple minima. The ability of the genetic algorithm to efficiently combine local and global components makes it inherently superior to techniques that rely on either one.

The best waveform fit achieved by the genetic algorithms, with 8000 models sampled, was $\phi_{\rm wf} = 0.736$. The optimum value of $\phi_{\rm wf}$ will not be zero chiefly because of the noise in the data and the inadequacies of the synthetic waveform calculation (e.g. it does not model reverberations). However, it is interesting to note that, even though the genetic algorithm was applied to an 11 parameter model, it is actually a lower misfit than that of the model obtained from the linearized inversion solution of CC which used 21 model parameters, $\phi_{\rm wf} = 0.79$ (the misfit of this model was recalculated with the misfit function used here so that a fair comparison can be made). The difference between these figures may, in part, be due to the differences in the type of model parametrization and also small numerical differences in the misfit function (11) due to slight differences in the range coefficients, d_j, and the synthetics scaling parameter c. Nevertheless it is clear that the efficient genetic algorithm has been able to find a velocity model which requires half as many degrees of freedom and fits the waveform data just as well. The best velocity model obtained by the genetic algorithm is shown in Fig. 4. A comparison of the synthetic and observed seismograms for the two linear gradient models are shown in Figs 7 and 8. Both sets of synthetics appear to be good fits to the data. The primary difference appears to be that the first arrivals in the two central panels are lined up better with the synthetics from the genetic algorithms model (Fig. 7). The 11 parameter model found by the genetic algorithm is similar to that of CC but, of course, it has less detailed structure indicating that the small-scale gradient variations in the latter model are not necessary to fit the observed waveform data.

A comprehensive understanding of the information contained in the observed seismograms requires more than just finding a global minimum of $\phi_{\rm wf}$. One must also perform

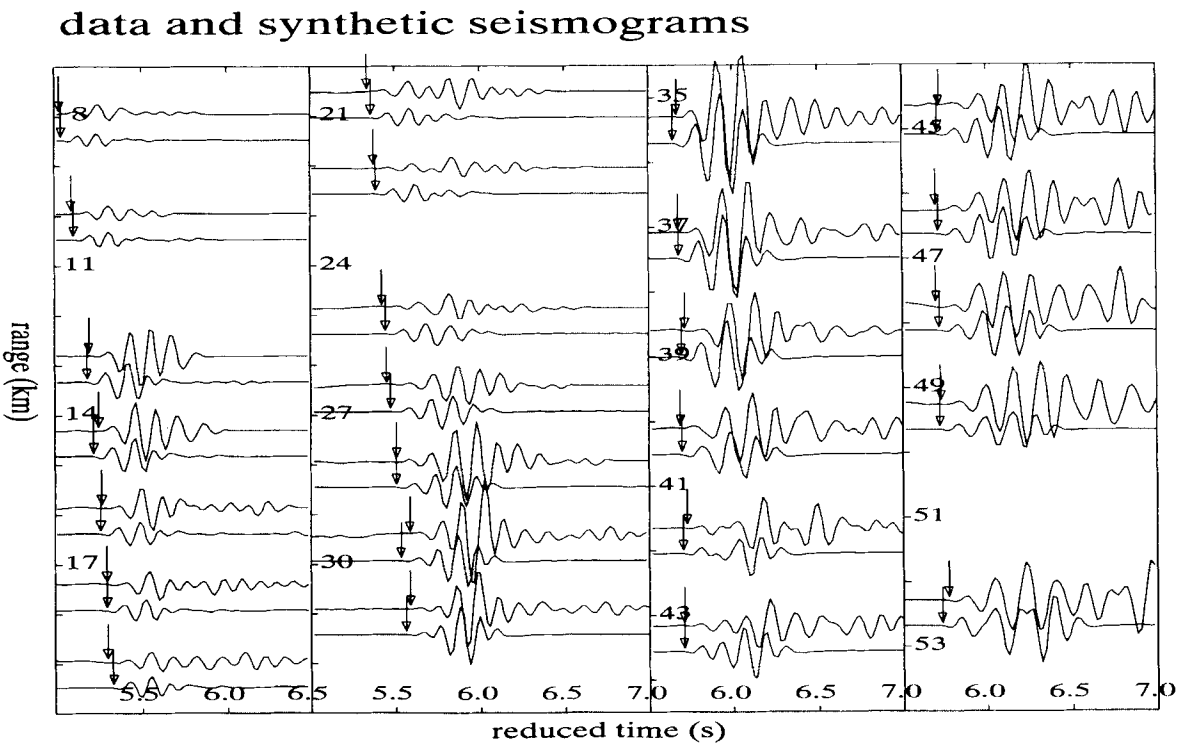


Figure 7. Observed and synthetic seismograms for the 11 parameter model obtained by the genetic algorithm. The data are plotted at their true range and the synthetics are shifted down slightly. Both data and synthetics have been scaled by a factor proportional to the square of the range, x^2.

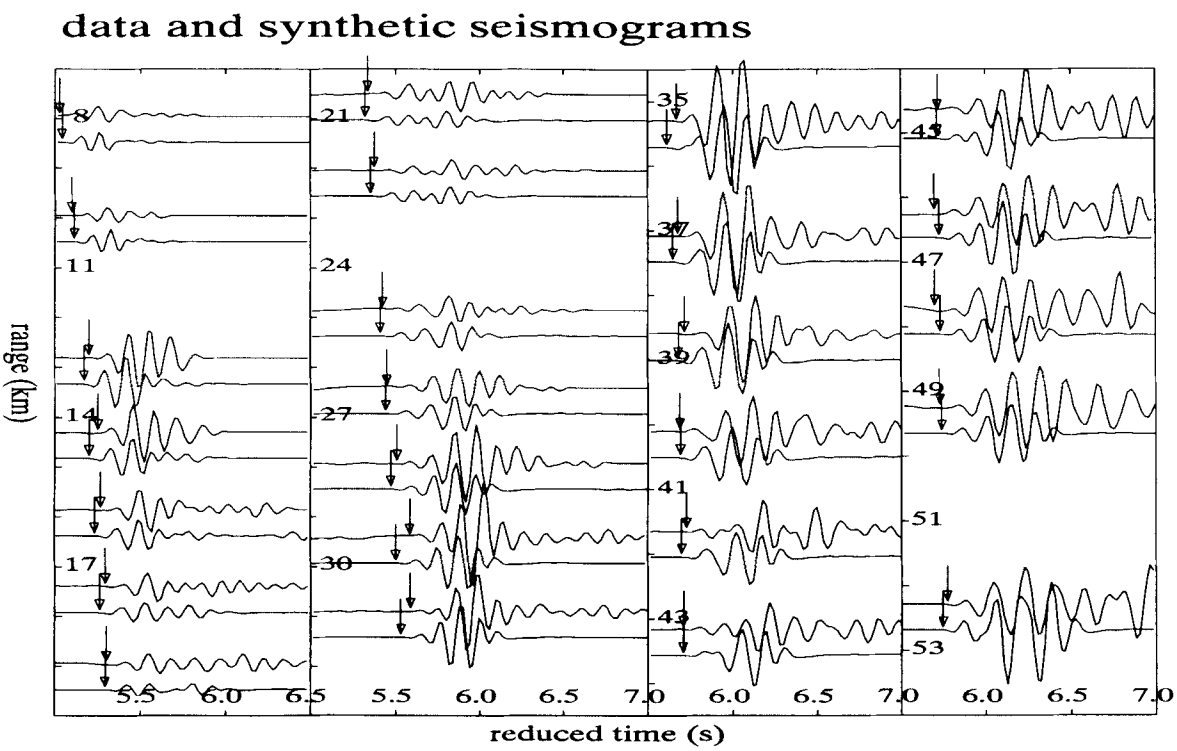


Figure 8. Observed and synthetic seismograms for the 21 parameter model obtained by Cary & Chapman (1988). All other details as for Fig. 7.

a detailed error analysis by examining the multidimensional neighbourhood of the global solution. The common approach is to try to assign confidence regions to contours of ϕ_{wf} about the solution and find the range of each parameter that lie on, say, the 95 per cent confidence contour. In addition one should also examine the trade-off between different parameters which can be estimated by means of a posterior covariance matrix (see Tarantola 1987). Since it is our interest here to examine the use of genetic algorithms for waveform inversion, it is something of a diversion to discuss non-linear error analysis in any depth. The procedure used by CC to estimate the covariance matrix via Monte Carlo integration could equally well be applied after using the genetic algorithm to locate the global solution.

3.3.3 Waveform fitting: genetic algorithm compared to linearized inversion

Even though the genetic algorithm has yielded a simple 11 parameter velocity model that fits the data well, it is still worthwhile examining its performance as the number of model parameters, and the size of the model space, are increased. In particular it is interesting to know whether the genetic algorithm can compete directly with the linearized matrix inversion scheme of Chapman & Orcutt (1985) when the number of model parameters is increased to 22 and the velocity interval is reduced. In addition we would like to know whether the genetic algorithm performs as well against Monte without the benefit of the preliminary traveltime fitting stage. Most waveform inversion procedures are basically local methods which require a good starting model within the valley of the global minimum. Several authors have begun waveform inversions by fitting the envelope function (Shaw & Orcutt 1985; Nolet, van Trier & Huisman 1986). This usually improves the robustness of the waveform inversion just as the Monte Carlo search for an 11-parameter model provides a reasonable starting point for the linearized inversion scheme of CC with 21 model parameters. In this algorithm a large-scale($\approx 3250 \times 21$) non-sparse matrix must be formulated and solved for a series of model perturbations. As Chapman & Orcutt (1985) point out, the computational effort required to do this is considerable, largely because the coefficients of the linear system must be determined by calculating differential seismograms for each depth variable, and each of these requires the same amount of work as solving the forward problem. Of course, one of the attractions of the genetic algorithm is that no matrix inversions need be performed, but perhaps equally importantly, for this problem, the differential coefficients that make up the linear system do not have to be evaluated. They give the number of synthetic seismograms per iteration of the linear system as $J(K+1+n)$, where n is the number of different damping parameters tested, J is the number of ranges and K is the number of velocity parameters. The 21 parameter model of CC was produced after eight iterations, and so assuming $n \approx 10$ then the number of synthetic seismograms required to perform the linearized inversion is approximately 6400. We can take this value as an approximate measure of the work required by the linearized inversion scheme used by CC and use it to gain a rough comparison with the genetic algorithm. (We note, however, that in practice the matrix inversion scheme will require additional labour to solve the linearized system at each iteration and also memory to store the equations.)

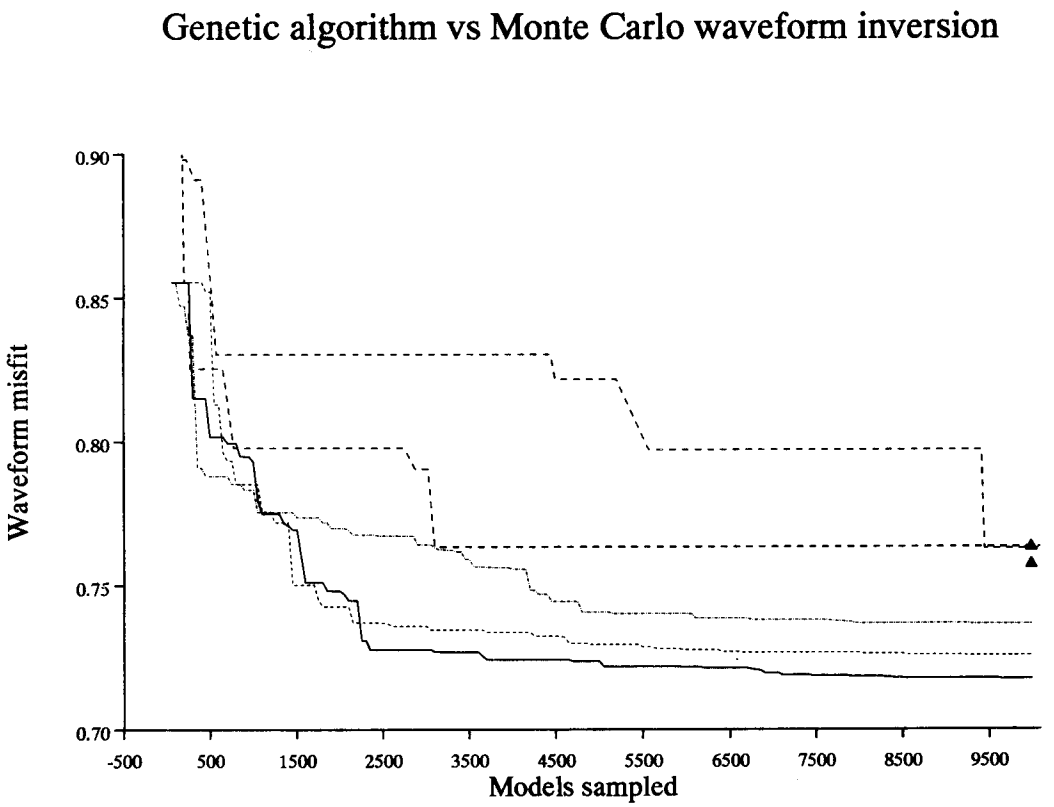


Figure 9. The performances of three genetic algorithms (solid, dotted and chain dotted) and two Monte Carlo searches (dashed) in minimizing the waveform misfit for a 22 parameter velocity model. The parameters controlling the genetic algorithms are given in Table 2. The triangles indicate the values of the waveform misfit found by the Monte Carlo searches after sampling 440 000 models.

The results of three different genetic algorithms and two Monte Carlo's for a 22 parameter velocity model are shown in Fig. 9. The controlling parameters for each genetic algorithm are given in Table 2. In all three cases the population size was 50 and the traveltime fitting stage was reduced to merely taking the first 50 models that passed the traveltime test (which was achieved after 200 randomly generated models). The depths of each velocity node are shown in Fig. 10 along with the best model found by the genetic algorithms and the model of Spudich & Orcutt (1980). The velocity bounds have been narrowed from that used in the 11 parameter case and the velocity interval, dv, was decreased to $0.02\,\mathrm{km\,s^{-1}}$. The total string length, l increased to 131 producing a model space containing a total of 2.5×10^{27} models. From Fig. 9 it is clear that the lack of any significant traveltime fitting stage has reduced the initial advantage of the genetic algorithms over the Monte Carlo searches. However one observes, as in the previous case, a rapid reduction in the misfit during the early stages and all three genetic searches quickly reach values which are better than the Monte Carlo after 440 000 sampled models (again represented by solid triangles). As in the 11 parameter case the relative efficiencies are difficult to estimate because of the exceedingly slow nature of the Monte Carlo. The waveform misfits achieved by the three genetic algorithms are in the range $0.717 \leq \phi_{\mathrm{wf}} \leq 0.736$ which are all less than the that of the CC model obtained by linearized inversion ($\phi_{\mathrm{wf}} = 0.79$).

A comparison of the best 22 parameter model with the CC model is shown in Fig. 11. In the shallow regions $z \leq 8.5$ km the models are very similar. In the deeper parts of the model, where the data constraint presumably begins to decrease, the models begin to differ in character. The new model has a slightly smoother velocity increase around 9 km compared to the model of CC. The synthetics for the 22 parameters solutions are displayed in Fig. 12. As in the 11 parameter case the first arrivals are lined up better in the two central panels. Clearly the difference between the waveform fits does not appear to be very significant, although this is, strictly speaking, a question for the error analysis. The important point from our point of view is that the genetic algorithm is certainly able to compete with the linearized waveform inversion for quality of data fit and also overall computational efficiency without the need for derivative information or a good starting model within the valley of the global minimum. It may be argued that the three genetic algorithms have apparently converged to different models and that therefore at least two of the three are probably in local minima. This is most likely true and demonstrates one of the current inadequacies of the genetic approach, i.e. that we have no guarantee of an optimal implementation. However, in answer to this criticism we must point out that it is not really correct to say that the algorithms have converged. As we have already seen in the previous example it is possible that new onsets of accelerated misfit reduction can begin, and so we cannot be

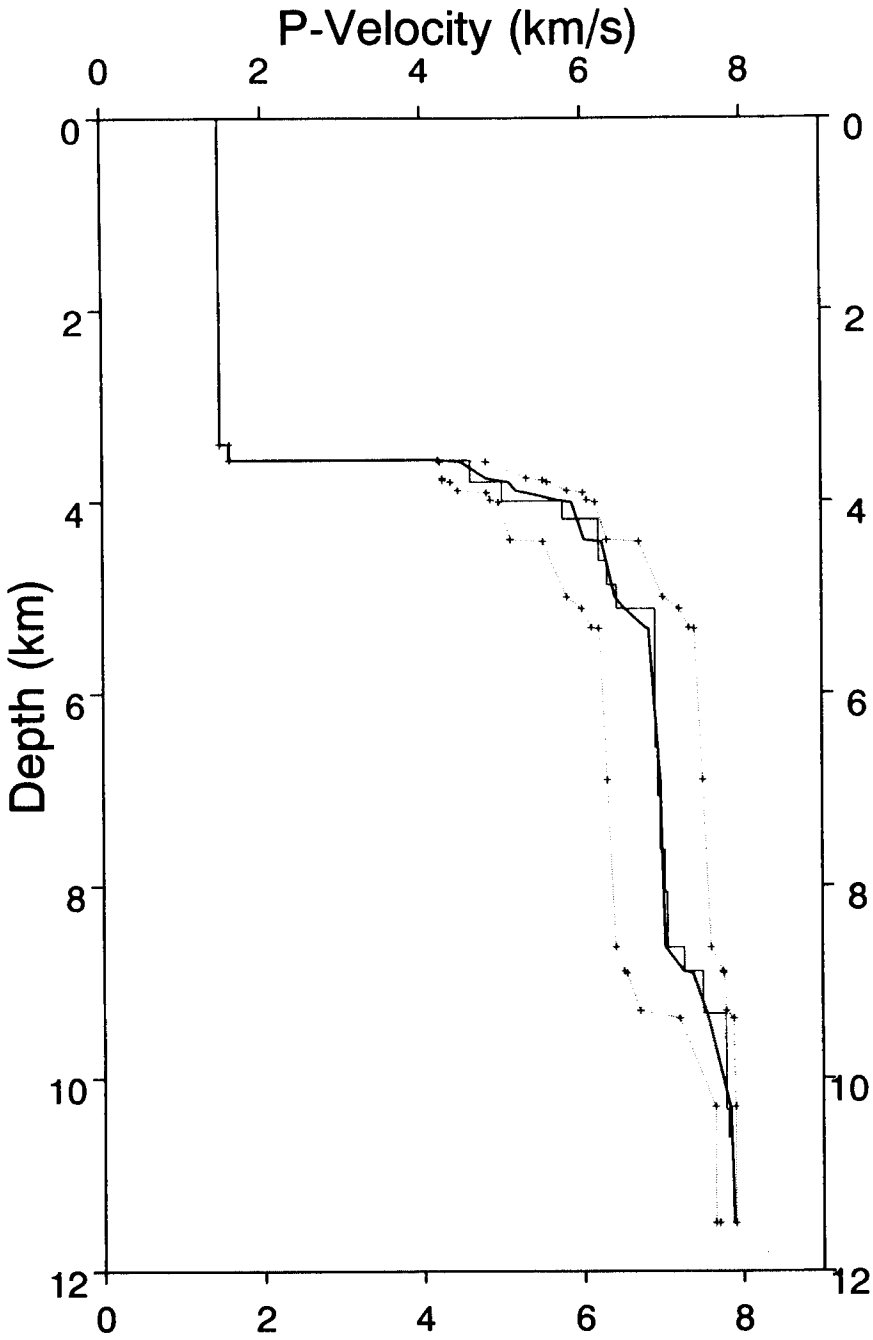


Figure 10. The best 22 parameter velocity model found by the genetic algorithm with the homogeneous layer model of Spudich & Orcutt (1980). The crosses indicate the depths of the velocity parameters used in the model.

sure that the two poorer genetic solutions will not improve later on. In addition, the difference between the three velocity models is considerably less than that for the secondary minima identified by CC, which occurred in more shallow regions of the model. The two poorer models are similar to the CC model in that they have larger velocity gradients at around 9 km depth but are very similar to the genetic solution at shallower depths.

A comparison of Figs 9 and 6 shows that, in the 22 parameter case, both the genetic algorithm and the Monte Carlo search begin at much lower waveform misfits than in the 11 parameter case. This indicates that the narrowing of the velocity bounds for the second inversion has assisted both procedures in locating models with smaller misfits. The new bounds were generated by examining the range of models found by the 11 parameter genetic algorithm that passed the traveltime test ($|\delta T| \leq 0.15$ s). Therefore all of the curves in Fig. 9 have in fact benefited from the earlier genetic algorithm and therefore we cannot really claim that the genetic algorithm in Fig. 9 has achieved its low misfit without the aid of some indirect traveltime fitting. Nevertheless in the 22 parameter case the Monte Carlo and

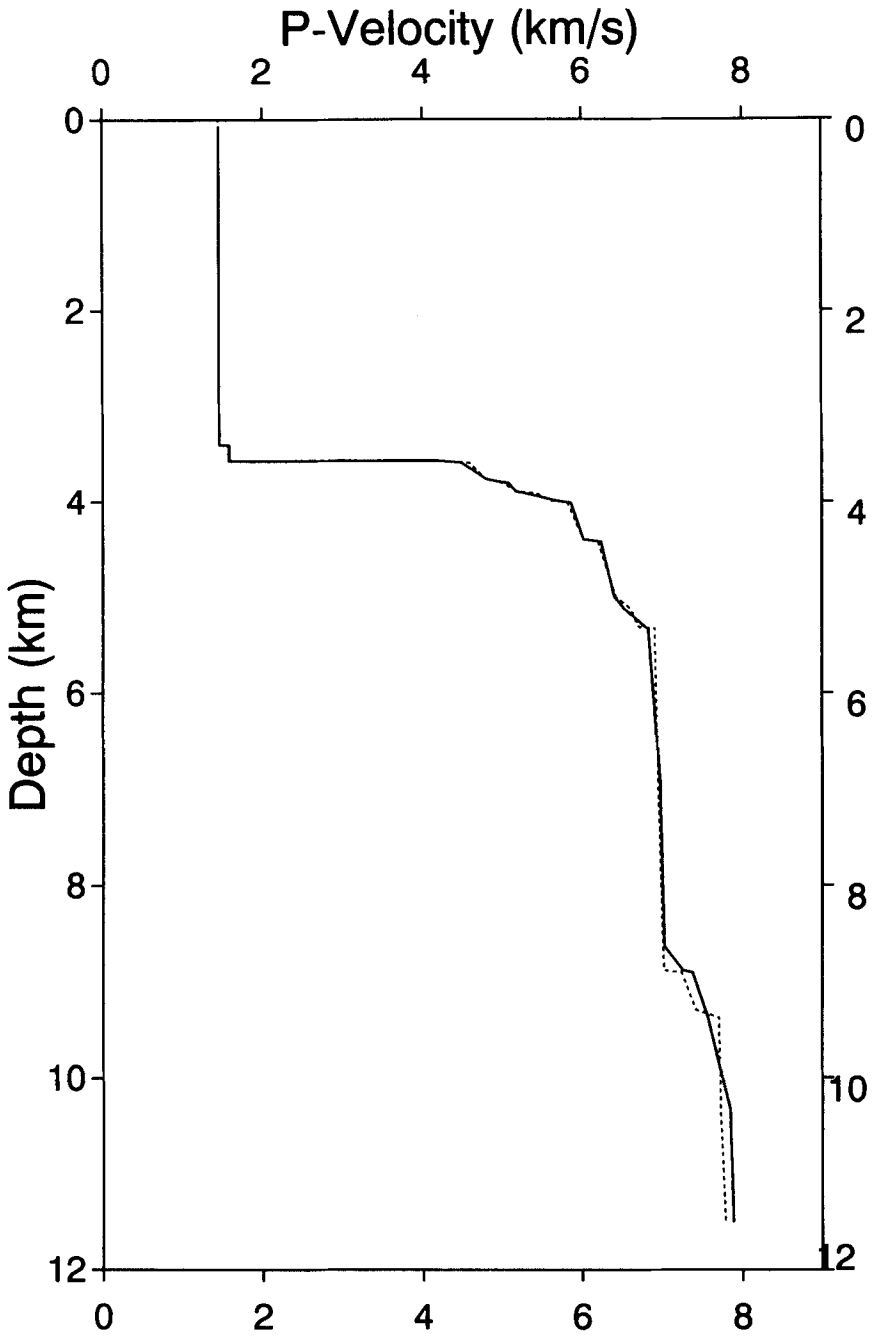


Figure 11. A comparison between the 22 parameter model obtained by the genetic algorithm and the 21 parameter model obtained from a linearized inversion by Cary & Chapman (1988). The two models differ only at depths greater than 8.5 km.

the genetic algorithm benefit equally from the narrower bounds and so a comparison between these is still valid.

The performance curves in Figs 6 and 9 are illuminating but, nevertheless, only represent the misfit of the best model achieved. They do not give any information about the range, or average, misfits within each population. It can be instructive to examine a range of models that fit the data well. The spread of ϕ_{wf} within the population gives us an indication of whether the best model is the result of some fortuitous interaction with a few models, or if information exchange is taking place among all of the models. Fig. 13(a) shows a scatter plot of ϕ_{wf} and ϕ_{tt} for the first 1229 models that passed the traveltime test in the Monte Carlo scheme for the 22 parameter inversion, and Fig. 13(b) is the corresponding diagram for the most successful genetic algorithm. The genetic scatter plot shows an intense concentration of models with relatively small ϕ_{wf} compared to the Monte Carlo case which is much more spread out by comparison. As the iterations proceed the genetic algorithm is able to exploit the information gained from previous populations and direct the entire population towards favourable regions of model space, thereby generating more and more models with smaller waveform misfits. This indicates that substantial and efficient exchange of

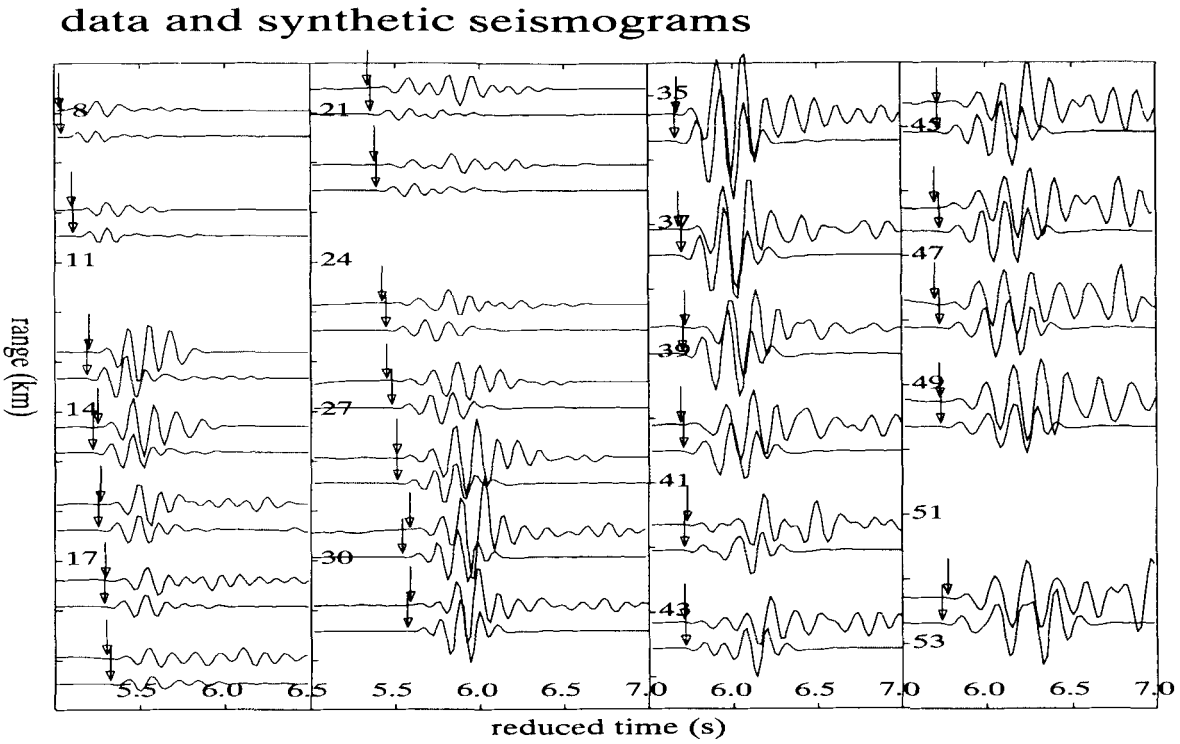


Figure 12. Observed and synthetic seismograms for the 22 parameter model obtained by the genetic algorithm. Other details are as for Fig. 7.

information has taken place among many of the models in the population.

As a final point it is worthwhile pointing out that in the work of Cary & Chapman (1988) considerable effort was exerted in performing non-linear error analysis about the final solution. In this way it was possible to place confidence bounds on the velocities and obtain information on the trade-offs between velocities at different depths. This is clearly important information which can only be obtained by performing a multidimensional numerical integration to estimate the posterior covariarance matrix (see Cary & Chapman 1988, or Tarantola 1987, for a description). In this work the genetic algorithm has only been used to solve the optimization part of the inverse problem and has not been applied to the error analysis stage. However, to arrive at a solution, the genetic algorithm samples many models throughout the parameter space, and the forward problem is solved for each of these models. Clearly it would be advantageous if this information could be used to perform the error analysis, without any extra solving of the forward problem. In addition an estimation of *a posteriori* covariance matrix based on these models is likely to be efficient because many of them are concentrated about the final solution. In theory this process is straightforward; however, in practice there are problems since it requires knowledge of the multidimensional probability density function which generated the genetic algorithm models. An accurate and efficient method of calculating this density function, from a finite number of samples, is a difficult problem, and this remains an area for further study.

4 GENETIC ALGORITHMS AND SIMULATED ANNEALING

The superiority of genetic algorithms over Monte Carlo is encouraging. However, in recent years Simulated Annealing techniques have also been shown to be a major improvement on Monte Carlo for global optimization problems. The question therefore arises of how they compare. The two techniques are similar in that both are modelled on processes found in nature, and both have been used to solve difficult and important problems, including the design of semi-conductor layouts, optimization of communication networks and signal analysis (see Grefenstette 1987, for a summary). Both technologies are developing rapidly, and so comparisons between the two on specific problems may quickly become dated. Nevertheless detailed comparisons on combinatorial problems (i.e. those which are naturally discrete, like the travelling salesperson problem) have been made (see Davis 1990). The general conclusion from these studies seems to be that neither technique is superior for all problems. Usually the more appropriate technique is the one that can be applied most easily to the particular problem at hand. For instance in the travelling salesperson problem (see Press *et al.* 1987) Simulated Annealing is easier to use because the crossover stage of the genetic algorithm can cause models to be generated which are not allowed in the problem, i.e. paths between cities are produced which omit some cities and visit others more than once. Modifications of the simple genetic algorithm are required, in order to avoid this problem, and

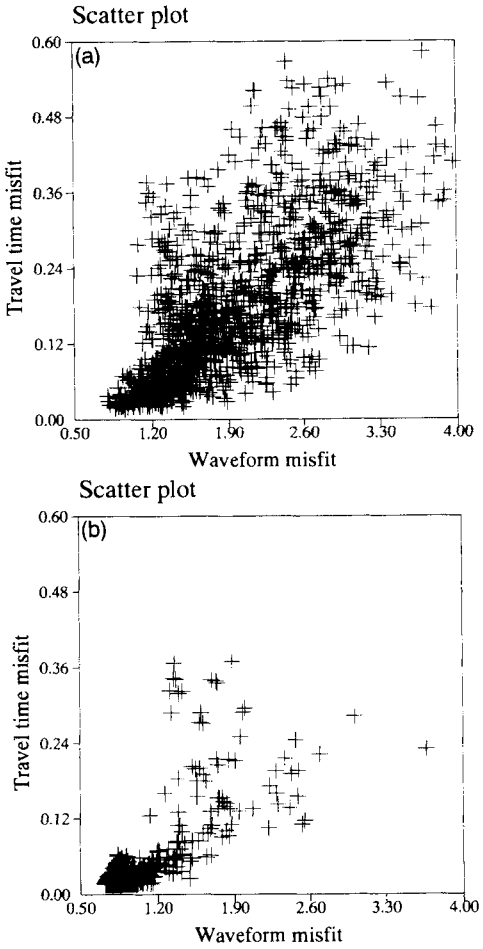


Figure 13. (a) Scatter plot of waveform against traveltime misfit for the first 1229 models, found by the Monte Carlo search, that passed the traveltime test. (b) The corresponding scatter plot for the 1229 models from the genetic algorithm. The lack of scatter in (b) indicates that all models are taking part in the information exchange during the iterations of the genetic algorithm.

these seem to inhibit its performance (Grefenstette 1990). In geophysical applications one might expect a similar situation to arise. Fortunately there is at least one detailed numerical comparison between the two methods available for a seismic inversion problem. Scales *et al.* (1991) compared genetic algorirthms with Simulated Annealing for a synthetic inverse scattering problem, where the model parameters were a series of reflection coefficients in a 1-D seismic model consisting of between 15 and 20 layers. Before we discuss the results of Scales *et al.* we must briefly sketch out the steps of the Simulated Annealing method.

Simulated Annealing differs from genetic algorithms in that it produces a sequence of models, each of which is generated from the previous model by one step of the algorithm. During each step a new model is selected from a pre-defined neighbourhood of the current model, and a decision is made on whether to accept or reject the new model by comparing their misfit values. Random processes play an important role in controlling both the generation and acceptance stages of the procedure. The sequence of models produced in this way has an analogy with a physical system in a heat bath. A probability, p, is used to decide whether to accept or reject the new model, where

$$p = \exp\left[(E_1 - E_2)/kT\right]. \quad (16)$$

Here E_1 and E_2 are the values of the misfit function of the two models, k is a constant and T is the temperature function which is lowered as the algorithm proceeds. If p is greater than 1 then the new model has a lower misfit than the old, and it is accepted; if it is less than 1 then it is accepted with the probability p. Many more detailed description of the technique exist (see for example van Laarhoven & Aarts 1987). Clearly the algorithm can move both uphill and downhill, and has the ability to climb out of local minima of the misfit function. Different versions of the algorithm have been suggested. The definition of the neighbourhood determines the class of models that can be produced at any one step. In many cases the neighbourhood is simply defined as an unrestricted variation of a single parameter in the model. In the next step a different model parameter is chosen either sequentially, or randomly. Variations also exist in the way a new model is generated within the neighbourhood. Most authors use the Metropolis algorithm (Metropolis *et al.* 1953) which is a completely random selection between the parameter's bounds. Szu & Hartley (1987) consider perturbations derived from a Gaussian and a Cauchy probability density function (pdf) centred about the current model, while Tarantola (1987) advocates using the marginal pdf of the parameter to be perturbed. The combination of neighbourhood and selection scheme is known as the 'move class' of the algorithm.

Scales *et al.* (1991) use the continuous version of the algorithm, where the neighbourhood is defined by an N-dimensional sphere about the current model (N is the dimension of the parameter space), and a new model is slected randomly on this N-sphere. They perform 10 trial runs of the algorithm at varying temperatures and calculate ensemble averages of thermodynamic properties of the system, which allows them to estimate the critical range of temperatures to be sampled. They then use the fast annealing schedule of Szu & Hartley (1987) to reduce the temperature as the iterations proceed. The use of thermodynamic properties of the system to control the annealing schedule has probably been the major theoretical advance in the method in recent years, and this approach is embodied in the 'constant thermodynamic speed' annealing schedule of Nulton & Salamon (1988). It is interesting to note that, like the genetic algorithm, a number of preliminary trial runs are necessary in order to achieve an efficient implementation of the algorithm. In the comparison performed by Scales *et al.* two general features were evident. Firstly, the Simulated Annealing algorithm converged more uniformly towards the correct global solution than the genetic algorithm, and secondly that the genetic algorithm converged more 'quickly and completely'. The results also showed that the Simulated Annealing algorithm suffered from the problem of 'critical slowing down'. This is the term used to describe the situation when the computation time of the algorithm is dominated by a length-scale (step size or grid size) which is unrelated to the scale at which the misfit function varies in parameter space. Scales *et al.* suggest that, since the move class in conventional Simulated Annealing is restricted to a neighbourhood region, it can only accumulate information about the misfit function at this scale, and therefore information about the large-scale structure can only be gained after taking many steps. Genetic algorithms on the other hand do not have a comparable neighbourhood

restriction and appear to be able to assimilate information globally with each step.

The results of Scales *et al.* (1991) are very illuminating; however, other implementations of either technique may perform differentially, and so a sound theoretical basis is still required before any far-reaching conclusions can be drawn about comparisons of the two methods. Here we have tried only to highlight the different characteristics of the two approaches. We expect that in many geophysical applications a choice between the two will depend as much on their ease of implementation as on a potential return in computational efficiency.

5 DISCUSSION

The results of this paper suggest that genetic algorithms have the potential to solve global optimization problems much more efficiently than a Monte Carlo search, while making the same demands on the forward problem. The genetic algorithm requires only minimal information, i.e. misfit function evaluations, and is therefore independent of the details of the forward problem. This is a convenient feature since exactly the same algorithm may be applied to a wide range of optimization problems without difficulty. In fact the same set of computer subroutines was used in all three examples described in this paper without modification. The algorithm is independent of the form of the misfit function and avoids any use of derivative information but is able to efficiently exploit information and find near-optimal solutions rapidly.

At present, the study of genetic algorithms is still a young field, even though they have been used with success in problems of Artificial Intelligence for a decade or more. In addition to the theoretical shortcomings discussed above, there are two practical difficulties which must be highlighted. The first arises when a model is generated, which is not particularly close to the global solution, but relatively good compared to the rest of the population. It then makes multiple copies of itself and begins to dominate the population early on in the life of the algorithm. If the model manages to reproduce itself at a fast enough rate to overwhelm the rest of the population, then the algorithm is effectively stalled through a lack of diversity. This problem is known as 'premature convergence'. Usually it occurs if the population size Q is too small and is easily avoided by increasing Q. It actually occurred in the waveform example above for some trials with $Q = 26$, but not for larger population sizes. Other authors suggest some modifications to avoid it (see Goldberg 1989; Booker 1990). 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 the driving force of the algorithm is lost. This is essentially the opposite case to the first problem. The usual way to cure it is to adjust, or rescale, the cost function so that the reproduction probabilities, $P_r(\mathbf{m})$ of the models have a greater range. Other more sophisticated mechanisms for these problems are discussed in Davis (1990).

Another aspect of the genetic algorithm which is somewhat unsatisfactory is that they must be tuned for each particular problem. In the examples presented here it was possible to obtain significant improvements in efficiency by some preliminary exploration of alternative values for Q, P_c and P_m. This tuning is a rather *ad hoc* process. Indeed, most of the research into the subject seems to be centred on providing minor improvements to efficiency and developing strategies for adjusting the control parameters without much theoretical justification. However, as we discussed above, a clear theoretical framework is inhibited by the sampling error associated with the finite stochastic processes involved. In understanding and improving genetic algorithms it seems that one should concentrate on the underlying concepts of the method rather than the particular embodiment of those concepts which make up the mechanism described here. It seems reasonable to suppose that the basic ideas, i.e. the use of a stochastic process which uses the model fitness to control the likelihood of model survival, together with the decomposition and manipulation of the model in a binary form, could equally well be combined in a very different type of numerical algorithm; perhaps one which allowed a greater amount of control and predictability over its efficiency for a particular class of problems. Much of the work on genetic algorithms, however, has not taken this approach, preferring instead to concentrate on improving performance in particular problems by way of minor adjustments to the mechanism.

This paper has sought to demonstrate the feasibility of the genetic approach to geophysical optimization problems. Even though the two waveform fitting examples have dealt with parameter spaces consisting of up to 2.5×10^{25} models, and near-optimal solutions have been found with only 8000 samples, it is clear that we have considered only relatively small-scale problems, i.e. with up to 22 unknowns. We must expect that the computational cost of the procedure will increase with the number of unknowns (as the bit-string length, l, increases). For problems involving a much larger number of model parameters, say $\approx$1000's or 100 000's it is not yet clear whether a genetic algorithm, or a variation of it, can be applied efficiently. What is more certain is that genetic algorithms cannot, at present, be applied to problems for which the computational cost of solving the forward problem makes it prohibitive to sample more than at least a few hundred models. Indeed it is doubtful whether any algorithm would make much progress in a highly non-linear multidimensional problem where such a restriction is imposed. In the examples considered here the efficiency with which synthetics may be calculated with Chapman's method is crucial in making the problem viable for the stochastic approach.

We have outlined the main differences and similarities between genetic algorithms and Simulated Annealing methods, and have discussed some results of Scales *et al.* (1991) who have performed comparisons between the two. Both methods are developing rapidly and at present we expect that any problem feasible by one could also be tackled by the other. It seems likely that genetic algorithms will find applications in other geophysical optimization problems arising from both forward and inverse problems. One example would appear to be earthquake hypocentre location. An application of genetic algorithms to this area is currently under investigation by Sambridge & Gallagher (1992). Other optimization problems for which the method seems to have some potential are boundary value ray tracing through highly heterogeneous media and non-linear data fitting problems. In conclusion we feel that this intriguing

and powerful new methodology form a highly efficient and flexible class of algorithms for strongly non-linear optimization problems.

ACKNOWLEDGMENTS

The authors are indebted to Bruce Buffett, Chris Chapman, J. Huw Davies and Kerry Gallagher for many stimulating discussions and encouragement in this work. Special thanks to R. S. White for his support for GGD during the time this work was initiated.

REFERENCES

Andresen, B., Hoffman, K. H., Mosegaard, K., Nulton, J., Pederson, J. M. & Salamon, P., 1988. On lumped models for thermodynamic properties of simulated annealing problems, *Journal de Physique,* **49,** 1485–1492.

Bolt, B. A., 1991. The precision of density estimation deep in the earth, 1991 Harold Jeffreys Lecture of the Royal Astronomical Society, *Q. J. R. astr. Soc.*, in press.

Booker, L., 1990. Improving search in genetic algorithm, in *Genetic Algorithms and Simulated Annealing,* pp. 61–73, ed. Davis, L., Pitman, London.

Brower, R. C., Giles, R., Moriarty, K. J. M. & Tamayo, P., 1989. Combining renormalization group and multigrid methods, *J. Comp. Phys.*, **80,** 472–479.

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

Chapman, C. H., 1976. A first motion alternative to geometrical ray theory, *Geophys. Res. Lett.*, **3,** 153–156.

Chapman, C. H., 1978. A new method for computing body-wave seismograms, *Geophys. J. R. astr. Soc.*, **54,** 481–518.

Chapman, C. H. & Orcutt, J. A., 1985. Least squares fitting of marine seismic refraction data, *Geophys. J. R. astr. Soc.*, **82,** 339–374.

Davis, L. (ed), (1990. *Genetic Algorithms and Simulated Annealing,* Research Notes in Artificial Intelligence, Pitman, London.

Davis, L. & Steenstrup, M., 1990. Genetic algorithms and simulated annealing: An overview, in *Genetic Algorithms and Simulated Annealing,* pp. 1–11, ed. Davis, L., Pitman, London.

Dey-Sarkar, S. K. & Chapman, C. H., 1978. A simple method for the computation of body-wave seismograms, *Bull. seism. Soc. Am.*, **68,** 1577–1593.

Gallagher, K. L., Sambridge, M. S. & Drijkoningen, G. G., 1991. Genetic Algorithms: an evolution on Monte Carlo methods in strongly non-linear geophysical optimization problems, *Geophys. Res. Lett.*, in press.

Goldberg, D. E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, Reading, MA.

Grefenstette, J. J. (ed.), 1987. *Proceedings of the Second International Conference on Genetic Algorithms and their Applications,* Lawrence Erlbaum Associates. Hillsdale, NJ.

Grefenstette, J. J., 1990. Incorporating problem specific knowledge into genetic algorithms, in *Genetic Algorithms and Simulated Annealing,* pp. 42–60, ed. Davis, L., Pitman, London.

Holland, J. H., 1975. *Adatptation in Natural and Artificial Systems,* The University of Michigan Press, Ann Arbor.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P., 1983. Optimization by Simulated Annealing, *Science,* **220,** 671–680.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E., 1953. Equations of state calculations by fast computing machines, *J. Chem. Phys.* **21,** 1087–1092.

Nolet, G., van Trier, J. & Huisman, R., 1986. A formalism for nonlinear inversion of seismic surface waves, *Geophys. Res. Lett.*, **13,** 26–29.

Nulton, J. D. & Salamon, P., 1988. Statistical mechanics of combinatorial optimization, *Phys. Rev.*, **A37,** 1351–1356.

Parker, R. L., 1977. Understanding inverse theory, *Ann. Rev. Earth planet. Sci.*, **5,** 35–64.

Press, W. H., Flannery, B. P., Saul, A. T. & Vetterling, W. T., 1987. *Numerical Recipes,* Cambridge University Press, Cambridge, UK.

Rothman, D. H., 1985. Nonlinear inversion statistical mechanics, and residual statics corrections, *Geophysics,* **50,** 2784–2796.

Rothman, D. H., 1986. Automatic estimation of large residual statics corrections, *Geophysics,* **51,** 332–346.

Sambridge, M. S. & Kennett, B. L. N., 1986. A novel method hypocentre location, *Geophys. J. R. astr. Soc.*, **87,** 679–697.

Sambridge, M. S. & Gallagher, K. L., 1991. Earthquake hypocentre location using genetic algorithms, in preparation.

Scales, J. A., Smith, M. L. & Fischer, T. L., 1991. Global optimization methods for highly nonlinear inverse problems. *J. Comp. Phys.*, in press.

Shaw, P. R., & Orcutt, J. A., 1985. Waveform inversion of seismic refraction data and applications to yound Pacific crust, *Geophys. J. R. astr. Soc.*, **82,** 375–414.

Spudich, P. & Orcutt, J. A., 1980. Petrology and porosity of an oceanic crustal site: results from wave form modelling of seismic refraction data, *J. geophys. Res,* **85,** 1409–1433.

Szu, H. & Hartley, R., 1987. Fast simulated, annealing, *Phys. Lett., A,* **122,** 157–162.

Tarantola, A., 1987. *Inverse Problem Theory,* Elsevier, Amsterdam.

van Laarhoven, P. J. M. & Aarts, E. H. L., 1987. *Simulated Annealing: Theory and Practice,* Reidel, Dordrecht.