Chapter 4

Description and Implementation of Genetic Algorithms

This chapter is dedicated to the description of the Genetic Algorithms employed in this research. A number of different operators have been implemented and compared in order to find the most suitable configuration for the different problems here addressed. Both the details about the different operators and their effects on the Genetic Algorithms search are described with graphic aids and references are made to the FORTRAN77 code reported in Appendix B. However, an exhaustive treatment of the Genetic Algorithms process and of the underlying mathematics is beyond the scope of this chapter. More detailed insights into such process may be found in the literature listed at the end of this document.

4.1 Introduction

Many geophysical optimisation problems are non-linear and result in irregular objective functions. Consequently local optimisation methods, e.g. matrix inversion, steepest descent, conjugate gradients, are prone to trapping in local minima and their success depends heavily on the choice of starting model. Furthermore, in many problems the calculation of the derivatives can be difficult and computationally costly. Thus, global optimisation methods that can avoid these limitations are particularly attractive for geophysical applications.

Genetic Algorithms are a search method suitable for the global optimisation of irregular, multimodal functions. Starting with a set of initial solutions, these algorithms progressively modify the solution set by mimicking the evolutionary behaviour of biological systems, until an acceptable result is achieved. Because of their initial random and progressively more deterministic sampling of the function domain, these algorithms offer the possibility of locating relatively efficiently the most promising areas of the solution space. They are able to solve non-linear, non-local optimisation problems, without the need of curvature information, and consequently without the need for derivative calculations. This can be particularly useful because it allows for the use of fast, approximate, forward modelling for which no exact derivative may be available, with consequent reduction of the computation effort (see for example the description of the ray-tracing routine employed in this research in Chapter 3). Also, because Genetic Algorithms are based only on direct space sampling any kind of linearisation of the problem to be solved is avoided, with the consequent elimination of the errors involved in such an approximation.

Thes above factors make Genetic Algorithms particularly attractive for addressing complex real-word problems and they are increasingly being used to address geophysical problems [1, 7, 10, 12]. However, the large dimensionality involved in most geophysical optimisation problems can reduce the efficiency of Genetic Algorithm search, both in terms of quality of the result and computational cost.

In this chapter the development of a method is described, referred to as the pseudo subspace method, which facilitates the search in high-dimensional spaces. It works by reducing the problem dimensionality in the first stages of the process and by progressively increasing it once promising areas in the solution space have been discovered.

The method has been included into the Genetic Algorithm process and tested against different geophysical problems. The results from such tests are illustrated in the following chapters, whilst here a description of the different Genetic Algorithm implementations is given.

4.2 Genetic Algorithms

A standard Genetic Algorithm involves three basic operators corresponding to the biological processes of selection, crossover and mutation. Selection involves the choice of the individuals for the generation of offspring. Crossover is the method of combining (mating) two individuals to produce an offspring. Mutation is the random changing of some individual within the population. However, other operators may also be implemented in a Genetic Algorithms. Detailed descriptions of Genetic Algorithms can be found in [2, 3, 4, 5, 6]. Each operator can be implemented in different ways. Choosing the correct combination is vital to the effectiveness of the algorithm.

In setting up a Genetic Algorithm choices relating to the genetic operators as weel as the parameter representation must be made. Such choices are generally problem dependent. This step is crucial to the success of the algorithm since it can seriously affect its performance. A general outline of Genetic Algorithms together with the choices made to develop an algorithm to successfully invert geophysical data is now given. Also, details about how the different genetic operators have been coded in FORTRAN77 is discussed. In particular, two programs have been implemented: program LinNorm and program Parent, listed in Appendix B. The rationale behind the implementation of two programs, together with a detailed description is given in the section 4.3.

4.2.1 Parameter representation

Most of the research and almost all the theoretical analysis of Genetic Algorithms has been applied to binary-coded algorithms, i.e. each individual member of the population is represented in binary form. However, recent experimental results show that real coded Genetic Algorithms outperform binary ones in most applications [5, 14]. For this reason a real-coded Genetic Algorithm was implemented, i.e., an individual is represented by an array of real values. These real values correspond to the parameters to be reconstructed in the inversion process. In Genetic Algorithm literature terminoly inherited from biology is often used: a parameter is often referred to as gene and array of real number as chromosome. Figure 4.1

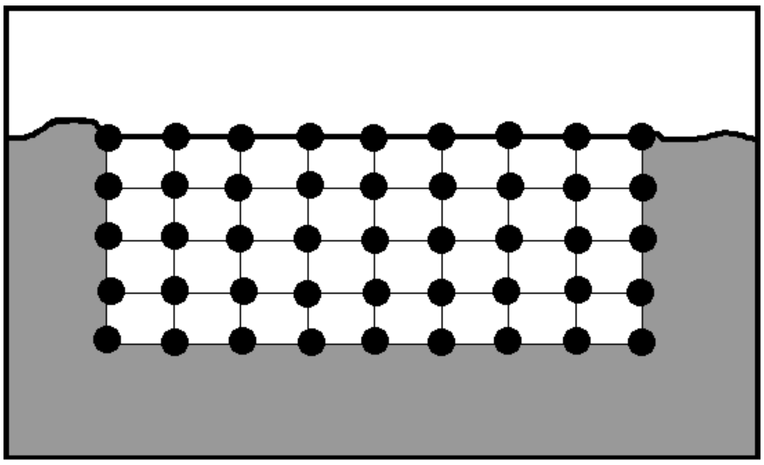

Figure 4.1: Genetic Algorithm parameterisation for the seismic refraction tomography problem. The parameters represent the seismic slowness at the nodes of a regular grid.

presents the Genetic Algorithm parameterisation for the seismic refraction tomography problem discussed in Chapter 5. The parameters correspond to the seismic slowness at the nodes of a regular grid. Figure 4.2 shows the Genetic Algorithm parameterisation for the 2-D inversion of gravity and magnetic data that is discussed in Chapter 6. In this case the parameters represent the depth of the contact between two geological bodies of different density and/or susceptibility. The parameters are then linked in arrays of real numbers to which the genetic processes of selection, crossover and mutation are applied.

A second aspect of parameter representation is the choice of the limits the parameters in the solution set are required to lie within. The first generation of individuals, i.e., the members of the starting model, are randomly generated between these values and during the evolutionary process they are constrained to remain within these limits. These limits depend on the nature of the problem and are discussed in the specific applications. The generation of the starting random population in the Genetic Algorithm process is performed by the subroutine CREATE in the programs listed in Appendix B.

When detailed 'a priori' information is available, this may be easily incorporated in the inversion process at this stage. Nodes for which the parameter value

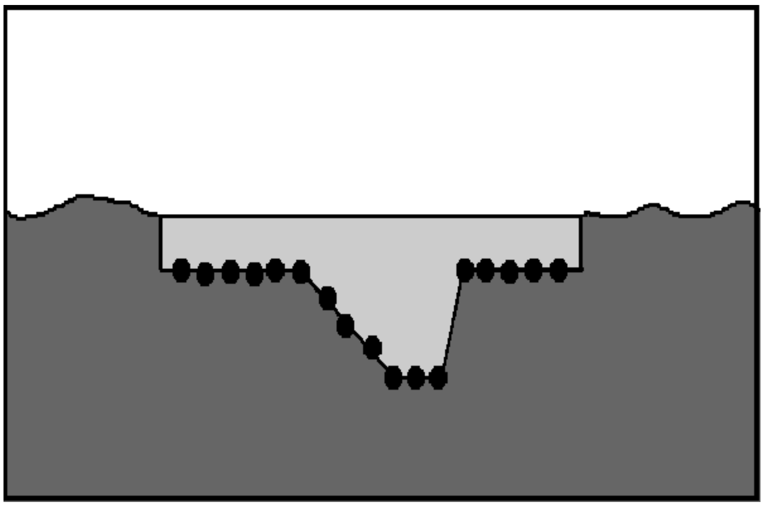

Figure 4.2: Genetic Algorithm parameterisation for the 2-D inversion of gravity and magnetic data. In this case the parameters represent the depth of the contact between two geological bodies of different density and/or susceptibility.

is known with accuracy may be kept fixed during the entire process, or allowed to vary between close constraints, thereby reducing the complexity of the search domain.

4.2.2 Genetic operators

Different genetic operators have been implemented and tested in order to select the most effective configuration. These implementations are described below.

Selection

Selection is the principal driving force that pushes the Genetic Algorithm population towards a final solution. It works by first assigning a measure of fitness to each individual, related to the value of the objective function at the corresponding point in the solution space. Then, using this assigned measure of fitness, a rule for selecting which individuals to use to create the next generation must be chosen. This operator can be implemented in many different ways and its influence can be varied. Implementations resulting in a strong selection pressure allow for fast convergence, but have the drawback of concentrating most of the population (and in few generations all of it) in a small part of the solution space. Due to this limited space sampling there is a greater risk of becoming trapped in a local minima (a phenomenon known as premature convergence). Implementations resulting in a low selection pressure will tend to decrease such risk, but at the cost of a much slower convergence. For the algorithm to be effective a balance between these two effects must be found.

A number of different selection implementations have been proposed in the literature [5, 6] and two of the most common choices have been implemented: linear normalisation selection, which has a high selection pressure [5, 6], and parent selection, which has a low selection pressure [3].

In linear normalisation selection, an individual is ranked according to its fitness and then it is allowed to generate a number of offspring proportional to its rank position. Using the rank position rather than the actual fitness values avoids problems which occur when fitness values are very close to each other (in which case no individual would be favoured) or when an extremely fit individual is present in the population (in such a case it would generate most of the offspring in the next generation). This selection technique pushes the population towards the solution in a reasonably fast manner, avoiding the risk of a single individual dominating the population in the space of one or two generations. The linear normalisation selection is performed by the subroutines EVALUATE and REPRODUCTION in the Genetic Algorithm program. EVALUATE ranks the solutions as a function of their fitness and gives them a weight proportional to their position in the rank, while REPRODUCTION determines the number of offspring to be produced by each solution and creates the offspring.

In parent selection an individual is allowed to generate only one offspring regardless of its fitness. All the individuals are mated randomly and through crossover each couple creates two offspring. If the offspring fitness are better than those of the parents they are substituted for them in the population. If not the parents keep on living. With this method the selection pressure is low and diversity is kept in the population because, as has been empirically demonstrated [9], the offspring are statistically the closest points currently present in the population to the parents they have been generated from. This means that an individual is substituted by the (statistically) closest point if its fitness is better, a process that suggests the idea of a 'stochastic parallel optimiser'. A good summary of the advantages of such operator can be found in [9]. The parent selection is performed by the subroutines REPRODUCTION and EVALUATE in the program Parent. REPRODUCTION randomly mates the individuals in the population and calls for the crossover process and for the fitness evaluation of the new individuals. EVALUATE determines to which of the parents one offspring is closer and substitute for it in the population if its fitness is better.

Crossover

Different kinds of crossover have been discussed in the literature [5, 6]. Four of them have been implemented and tested: two-point crossover, multi-point crossover, uniform crossover and uniform crossover with averaging of the parameters. These different methods basically describe how the two parents are combined. A graphic illustration of these different crossover processes is presented in Figure 4.3. In two-point crossover the chromosome is broken at two locations and the part contained between such locations is swapped between the parents (Figure 4.3b). This has been one of the first kinds of crossover proposed in the literature. Its main limitation is that it is strongly dependent on the relative location of the parameters inside the chromosome. Parameters far apart in the representation will be rarely involved in the same crossover process. Multi-point crossover is a generalisation of two-point crossover, in which the chromosome is broken at more than two locations (Figure 4.3c). A further generalisation is uniform crossover. In this case a number of parameters are randomly selected independently by their location inside the chromosome and swapped (Figure 4.3d). Uniform crossover with averaging of the parameters is equivalent to uniform crossover, but in this case the parameters are not simply swapped but averaged between the two individuals (Figure 4.3e).

After extensive trials uniform crossover was found to be the most successful, and it has been incorporated in all the implementations tested in this study. The reason for uniform crossover being more successful than two-point and multi-point crossover has already been stated and is due to the fact it is not dependent on the relative location of the parameters that undergo crossover inside the chromosome.

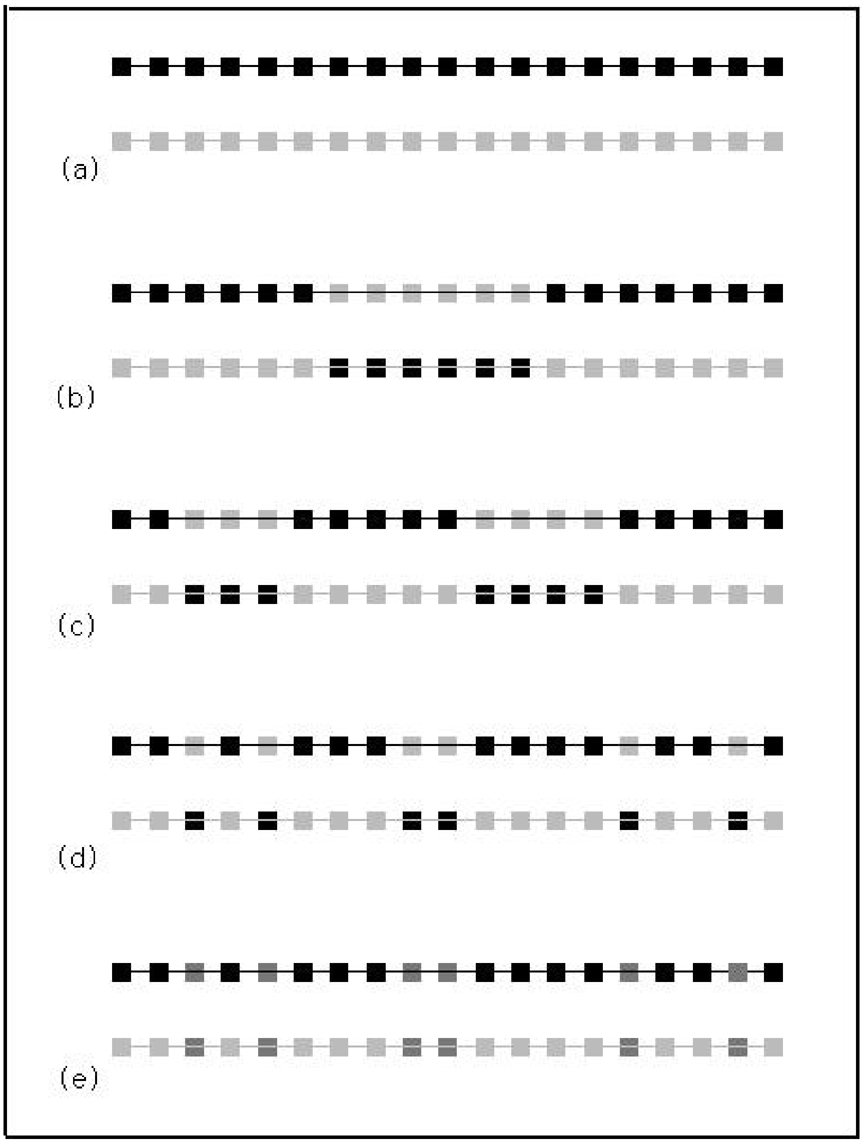

Figure 4.3: Graphic illustration of the different crossover processes tested in order to select the most effective operators configuration. The initial parents are shown in Figure a. Offspring are shown for two-point crossover (b), multi-point crossover (c), uniform crossover (d) and uniform crossover with average (e). In (e) the dark grey represents the average between the parameters in the two parents.

It is necessary to appreciate the geometrical implications of uniform crossover and uniform crossover with average in order to understand their different performances. When two individuals undergo uniform crossover they swap some parameters, i.e., some of the components in the solution space. This means that new individuals are created at the cross between lines parallel to the axis passing through the parents. This is graphically illustrated for a simple 2 dimensional case in Figure 4.4b. Two parents swap their coordinates and create two offspring at the opposite corners of a rectangle.

In uniform crossover with average the coordinates are not swapped but averaged according to some random weights. The result is that new individuals may be created anywhere inside the rectangle. This is shown in Figure 4.4c. It is reasonable to think that, due to the large amount of crossover performed in every Genetic Algorithm run, with the use of uniform crossover with average every parameter undergoes a large amount of averaging processes with the result that it tends to assume a value close to the average between the maximum and the minimum limit allowed in the representation. This is confirmed by my experiments in which the solutions obtained tended to show homogeneous values close to such an average. The better results obtained with uniform crossover without averaging seem to be due to the fact that the individuals are kept more spread out in the inversion process, and accordingly they can more effectively sample the solution space.

In this study uniform crossover has been implemented in the following way: two individuals are randomly chosen and a random number n between 1 and the number of parameters is selected; then n random gene locations are chosen and the floating point values of the parameters at such locations are swapped between the two individuals. This process is performed by the subroutine CROSSOVER in the program LinNorm.

The proportion of a population that is mated after each generation is defined by the cross-over rate. The experiments performed suggested an optimum tuning of the cross-over rate is 0.8.

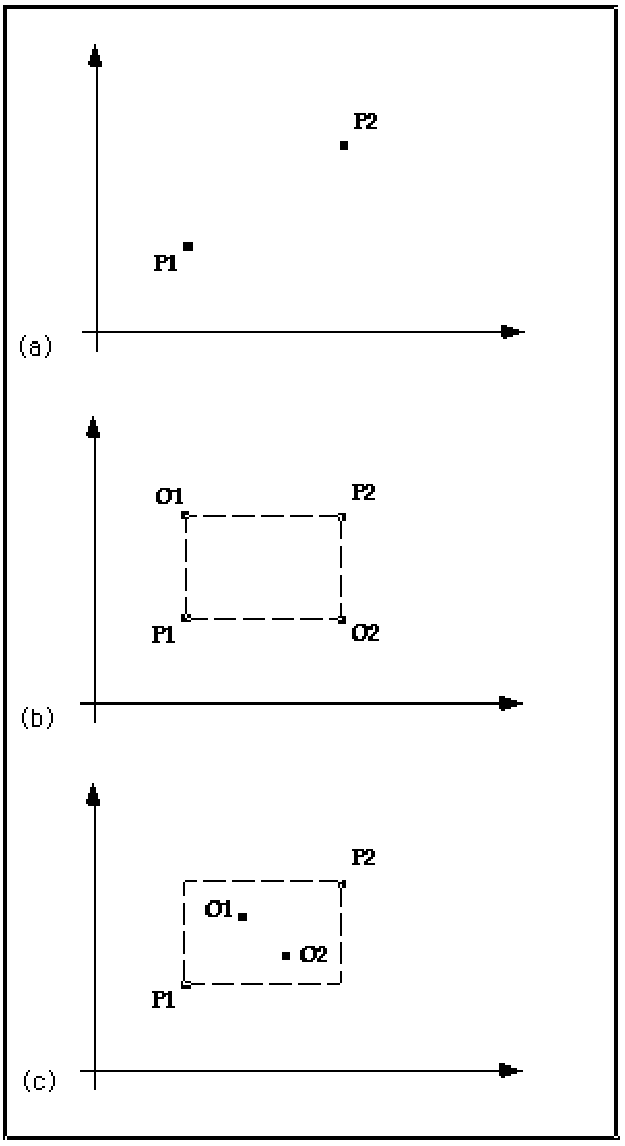

Figure 4.4: Geometrical representation of uniform crossover and uniform crossover with average in a small 2 dimensional space. The parents are shown in (a). Through uniform crossover they swap some components and the offspring are created at the corners of the rectangle generated by parallels to the axis passing through the two parents (b). With uniform crossover with average the offspring are created inside such a rectangle (c).

Mutation

This operator parallels the biological process of mutation. In this research Mutation has been implemented by randomly changing some parameter values in selected individuals. This is achieved by replacing the selected parameter values by a new value randomly chosen within the limits allowed in the parameter representation. The probability of this event must be kept very low in order to reduce the chance of removing good individuals currently present in the population. In the tests discussed below the mutation rate has been fixed at 0.01. In the programs in Appendix B mutation is contained in the subroutine REPRODUCTION. When the new generation is created each gene passed from one generation to the next is mutated with a probability of 0.01. This involves assigning a random number within the parameters limits.

Elitism

When crossover and mutation are applied it can happen that the best individual in the population is altered. If its new fitness is worse than the previous one the best solution found so far is lost. In this way convergence is not a monotonic process but proceeds through improvements and worsening. A simple way to avoid this phenomenon is to copy the best solution in the population and to pass it to the next generation without any alteration. This operator is not known to have any drawbacks and it has been implemented in the program in the subroutine KEEPBEST.

Creeping

This operator has been proposed by [5] who used it as an alternative to random mutation of parameters. It works by slightly perturbing some genes in the individuals it is applied to. In this way it performs a small scale mutation of some parameters in an individual in order to slightly modify its position in the search space.

The assumption behind the use of such operator is that after many generations most of the population is expected to have high fitness, and this operator aim

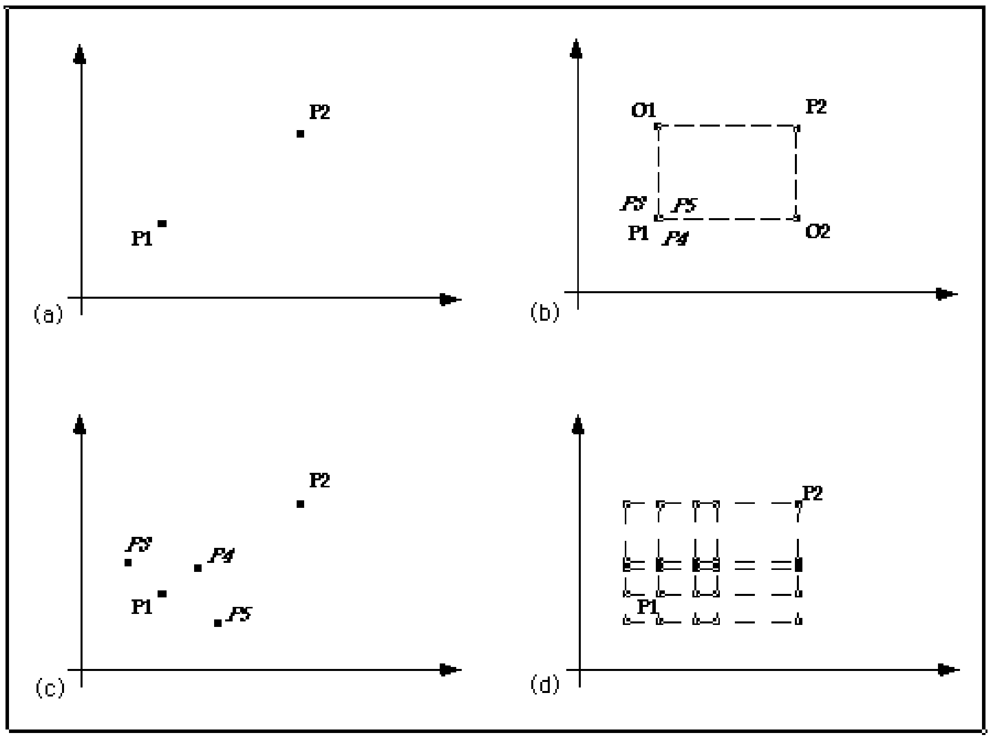

Figure 4.5: Geometrical representation of the combined effect of Creeping and Crossover operators

should be to sample the solution space in the immediate surrounding of good solutions, performing a kind of stochastic optimisation. However, the probability of improving the solutions through such a random event is quite low. Accordingly, this operator was implemented in a different way. Not all the population undergoes crossover in a Genetic Algorithm iteration. Since in the reproductive process more copies of the best solutions are produced it is quite likely that copies of the fittest solutions are passed unaltered to the next generation. In my implementation such unaltered copies undergo Creeping, i.e., the small random perturbation of some of their parameters. This process has been coded in the subroutine RNCREEP and sensibly enhanced the algorithm performance.

I propose that the reason for the success of this implementation is strictly connected to the crossover process and not to the actual space sampling performed at these perturbed locations. This is shown with the help of Figure 4.5. Suppose we have two parents, P1 and P2 in Figure 4.5a. By uniform crossover they can generate offspring O1 and O2 at the corners of the rectangle defined by their coordinates. If individual P1 is passed unaltered to the next generation it produces n copies, whose location in the 2 dimensional space coincides with P1. Accordingly, the only points that may be created by crossover with P2 are still only O1 and O2 (Figure b). However, if such copies undergo CREEPING, they are scattered in the surrounding of P1 (P3, P4 and P5 in Figure 4.5c). Now 20 different offspring may be generated by crossover with P2. This allows for a potential better sampling of the solution space in the surrounding of the best solutions in successive generations. Obviously, not all the new locations illustrated in Figure 4.5d will be sampled, due to the statistical behaviour of the crossover operator, but the potentiality involved in this process is much stronger than the one due only to the sampling of the new location P3, P4 and P5. A more in depth analysis of such phenomenon is on-going.

4.2.3 'Pseudo subspace method'

In a recent paper Williamson [13] describes the inversion of seismic reflection data using a multi-staged approach in which the resolution of the Earth model is progressively increased during the process. The method has been used in the context of a local optimisation scheme and a more detailed description of its theoretical basis may be found in [8, 11]. A similar approach has been used in the Genetic Algorithm implemented here. This operation works by projecting the parameter space onto a smaller subspace in the first stages of the inversion process, in order to allow the Genetic Algorithm search to rapidly discover the most promising area of the solution space. In the very first generations the total number of parameters required by the problem is 'compressed'. In geophysical applications of Genetic Algorithms it is common for the problem to involve the distribution of some physical property within the subsurface, usually expressed as values at nodes of a grid. In this case the 'compression' is achieved by connecting groups of adjacent nodes into a single node. This is illustrated in Figure 4.6a. The velocity field in the seismic experiment is described with only 6 nodes where 45 were used in the parameterisation shown in Figure 4.1. In the resulting smaller dimensional space the Genetic Algorithm defines approximately correct values of the parameters within a few generations. Because an exact solution very likely cannot be found with such a coarse grid spacing there is no need to run the Genetic Algorithm for many generations at this stage and consequently only a small sample population is also sufficient. In subsequent generations the spacing of the grid nodes is halved and hence the parameters are progressively 'decompressed' as new parameters are inserted in the domain. This is illustrated with the help of Figure 4.6. Each new parameter is given a value linearly interpolated between adjacent nodes. This guarantees that the solution is passed to the next stage without any alteration. Again the Genetic Algorithm is run, in what is now a higher dimensional space, for few generations with a relatively larger population. This process is repeated until the size of the grid reaches some pre-determined limit, whereupon the Genetic Algorithm is run for a larger number of generations, and with a larger population, until an acceptable convergence is reached. In the case of the seismic experiment this predetermined limit is represented by the parameterisation in Figure 4.1. It was found that an initial population of 20 individuals and a final one of 100 gave optimal results. The analogous process for the implementation employed in the potential field inversion is shown in Figure 4.7.

4.3 FORTRAN implementation

The Genetic Algorithms above described have been coded in FORTRAN77. Because of the different processes involved in linear normalisation selection and parent selection, two programs have been implemented: program LinNorm performs linear normalisation selection and program Parent performs parent selection. Comparisons of the two implementations on seismic refraction tomography is given in the next chapter. The implementation of two different programs is justified by the fact that the choice of the selection procedure affects also the way the solutions are reproduced, mated and evaluated. Also, in the case of parent selection the Creeping operator loses its meaning, because no multiple copies of any individual are created during reproduction. Eventually, as it is shown in the next two chapters, linear normalisation selection performed better in seismic applications, while the diversity imposed in the population by parent selection was beneficial in the description of ambiguity in potential field analysis. Accordingly, Appendix B contains the program LinNorm for the seismic experiments and Parent implementation for the potential field problem. It is important to notice that the

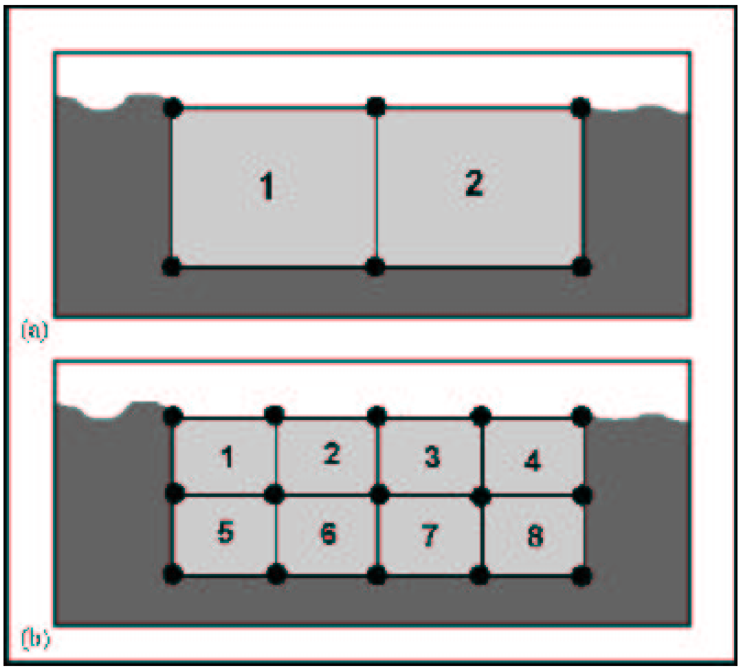

Figure 4.6: Pseudo subspace implementation for the seismic refraction problem. At each stage the search space is 'decompressed' by inserting new parameters into the domain.

different applications influenced only the Genetic Algorithm parameterisation and consequently the way the pseudo subspace method in carried out. No specific operator to address the mathematics underlying the two geophysical applications is included and consequently the Genetic Algorithm implementation itself should not be considered problem specific.

Details of how the different Genetic Algorithm operators have been coded in the different subroutines reported in Appendix B have been given in the above sections. Here the main processes are described by the use of the flow charts in Figure 4.8 and Figure 4.9.

Figure 4.8 shows the flow chart for the program LinNorm. In the first pseudo subspace method stage the population is initialised randomly. Each individual fitness is evaluated and depending on such values the population is ranked, selected and reproduced following the linear normalisation selection procedure. Mutation is applied during the reproduction process as described above. At this stage a fraction of the population equal to the crossover rate undergoes crossover and the fraction that is passed unaltered to the next stage undergoes creeping. These steps are repeated for a predetermined number of iterations after which the process enters the second stage of the pseudo subspace method. High fitness individuals are passed to the second stage and their parameters are decompressed in order to concentrate the search in the good areas in the solution space found so far, while new individuals are randomly generated and added to the population in order to allow the algorithm to keep on exploring different areas of the solution space. In this way the population is also increased in order to partly compensate for the larger solution space. The overall process continues until a converge criteria is satisfied and the program ends. The choice of a convergence criteria is a very subtle and crucial problem, and no satisfactory guideline has been found in the literature. The problem is addressed in the next chapter but its study is on-going. Therefore the convergence criterion had to be determined experimentally, as did the crossover rate, mutation rate, and the fraction of high fitness population to pass from one pseudo subspace stage to the next.

Figure 4.9 shows the flow chart for the program Parent. After the population random initialisation all the individuals are evaluated. Then, unlike in the linear normalisation case, all the population is mated. For each pair of parents, crossover is applied, two offspring are created, their fitness is evaluated and selection is performed by passing to the next generation the offspring or the parents depending on their fitness. The subspace process is implemented in the same way as for the linear normalisation case.

In the next two chapters the different Genetic Algorithm implementations are compared and the results of their application to the inversion of seismic and potential fields data presented.

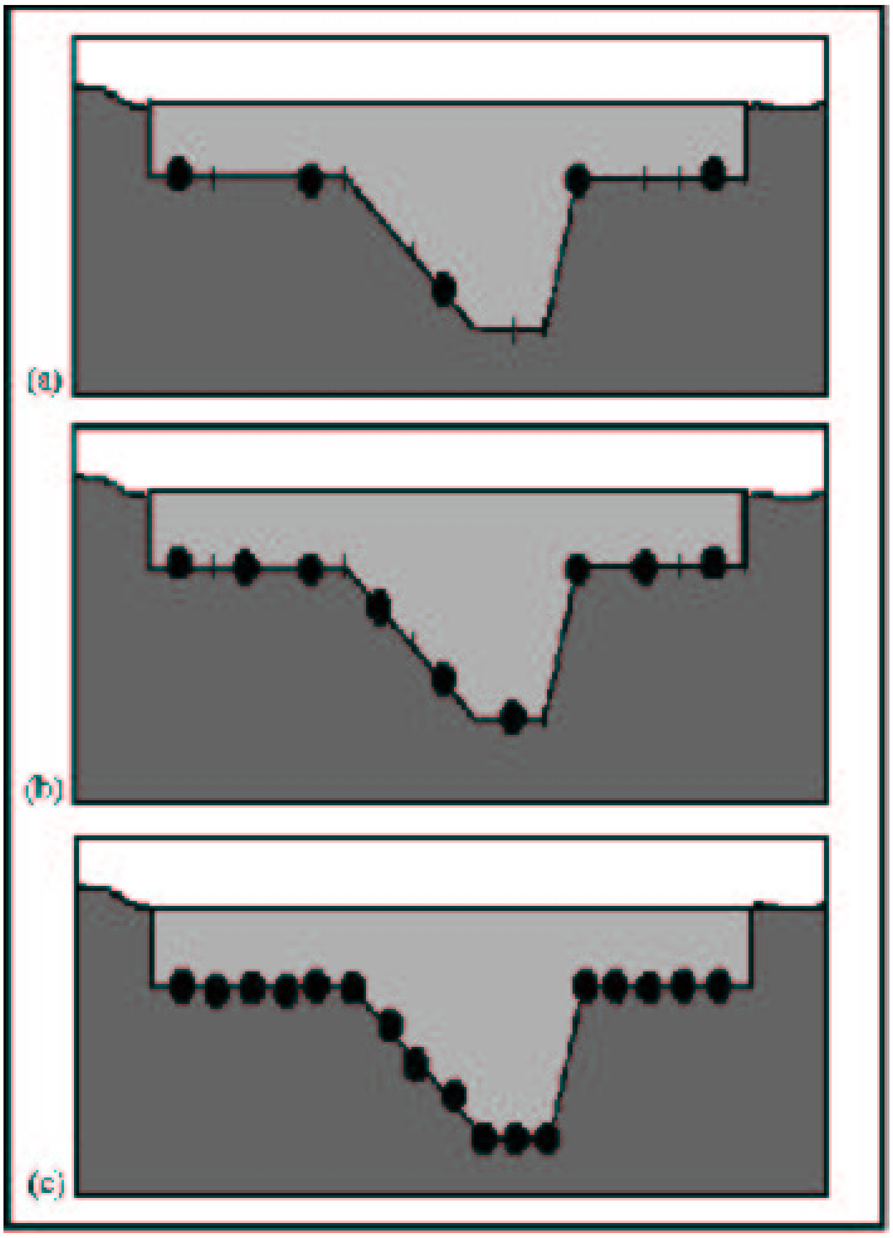

Figure 4.7: Pseudo subspace implementation for the potential field inversion problem. At each step the spacing between the parameters is halved and new parameters are inserted in the domain.

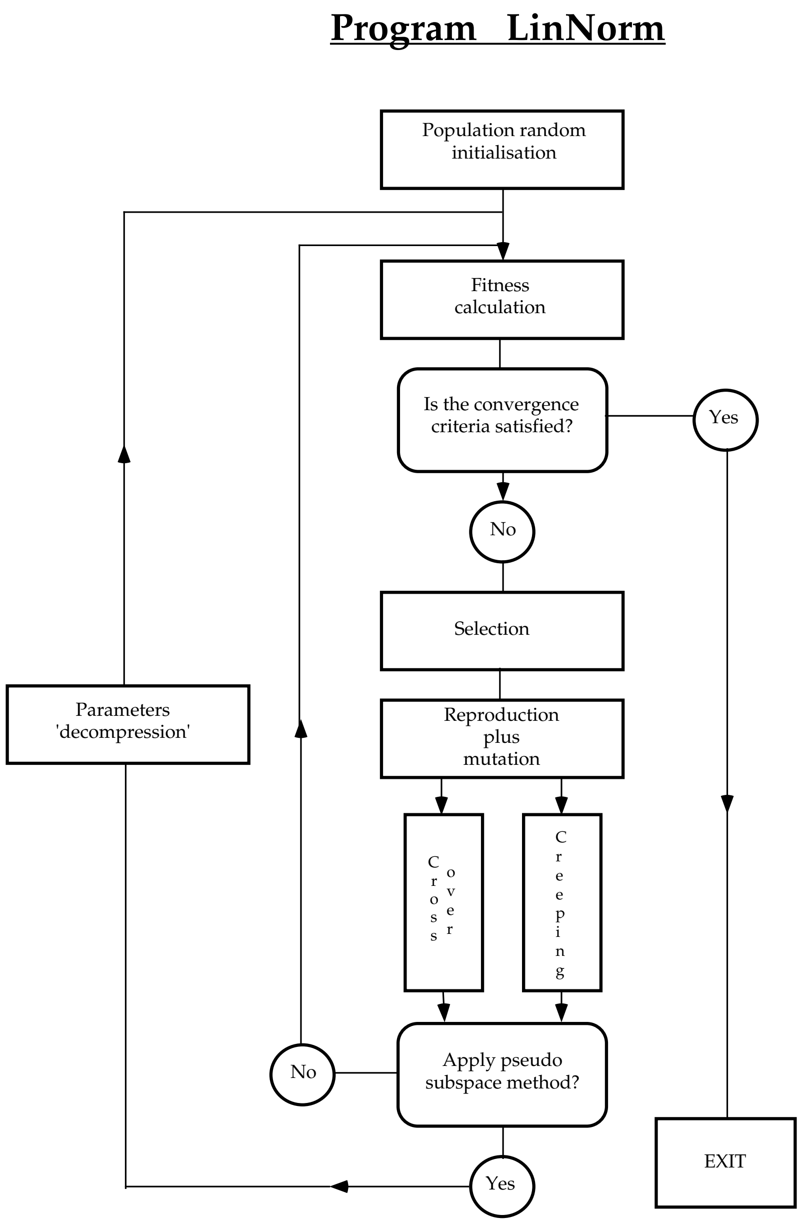
Program LinNorm
Population random initialisation
Fitness calculation
Is the convergence criteria satisfied?
Yes
No
Selection
Reproduction plus mutation
Cross over
Creeping
Apply pseudo subspace method?
No
Yes
Parameters 'decompression'
EXIT

Figure 4.8: Program LinNorm flow chart.

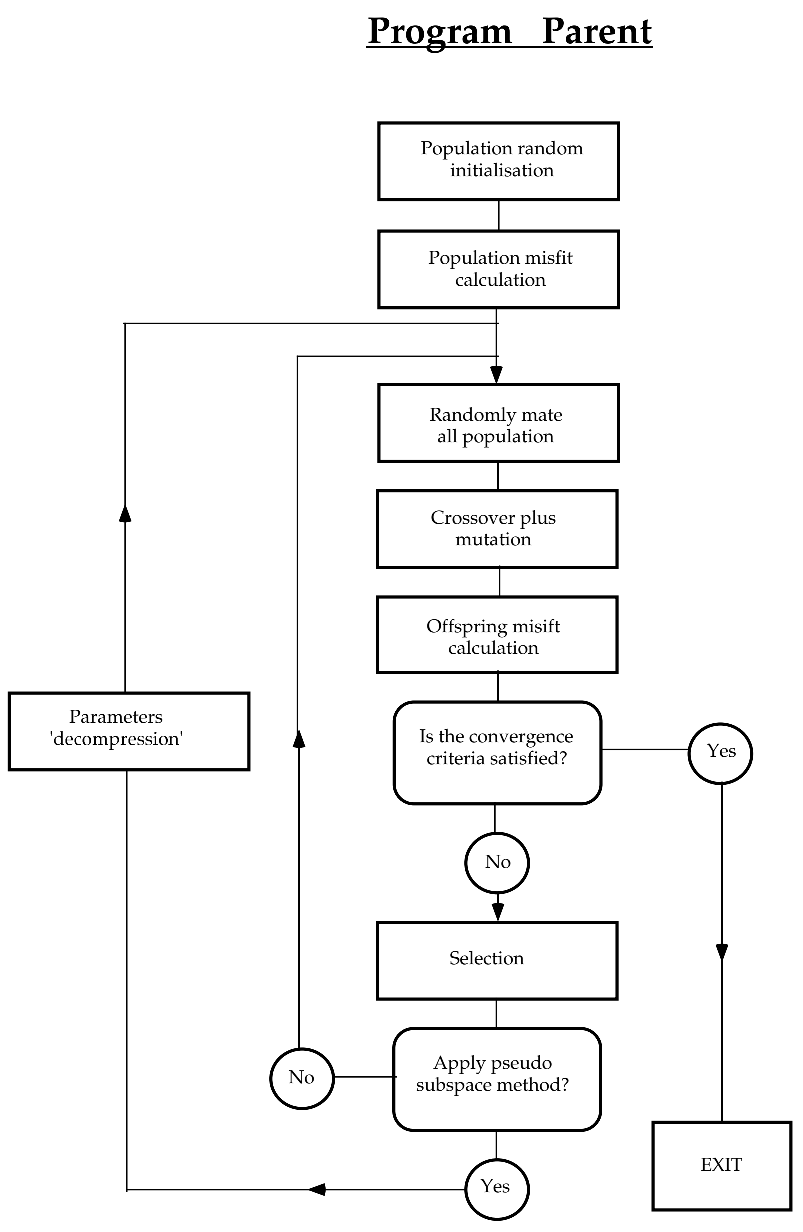
Program Parent
Population random initialisation
Population misfit calculation
Randomly mate all population
Crossover plus mutation
Offspring misift calculation
Is the convergence criteria satisfied?
Yes
No
Selection
Apply pseudo subspace method?
No
Yes
Parameters 'decompression'
EXIT

Figure 4.9: Program Parent flow chart.

Bibliography

[1] S. Billings, B. Kennett, and M. Sambridge. Hypocentre location: genetic algorithms incorporating problem specific information. *Geophysical Journal International*, 118:693–706, 1994.

[2] B. P. Buckles and F. E. Petry. *Genetic algorithms.* IEEE Computer Society Press Technology Series, 1992.

[3] D. J. Cavicchio. *Adaptive search using simulated evolution.* PhD thesis, University of Michigan., 1970.

[4] L. Davis. *Genetic algorithms and simulated annealing.* Morgan Kaufmann Publishers, Inc., 1987.

[5] L. Davis. *Handbook on genetic algorithms.* Van Nostrand Reinhold, 1991.

[6] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley Publishing Company, Inc., 1989.

[7] S. Jin and R. Madariaga. Background velocity inversion with a genetic algorithm. *Geophysical Research Letters*, 20:93–96, 1993.

[8] B. L. N. Kennett, M. S. Sambridge, and P. R. Williamson. Subspace methods for large inverse problems with multiple parameter classes. *Geophysical Journal International*, 94:237–247, 1988.

[9] S. W. Mahfoud. Crowding and preselection revisited. *Parallel Problem Solving from Nature*, 2:27–35, 1992.

[10] K. Mathias, D. Whitley, C. Stork, and T. Kusuma. Staged hybrid genetic algorithm search for seismic data imaging. In *ICEC 1994*, pages 356–361. IEEE, 1993.

[11] D. W. Oldenburg, P. R. McGillivray, and R. S. Ellis. Generalized subspace methods for large-scale inverse problems. *Geophysical Journal International*, 114:12–20, 1993.

[12] P. L. Stoffa and M. K. Sen. Nonlinear multiparameter optimization using genetic algorithms: inversion of plane-wave seismograms. *Geophysics*, 56:1794–1810, 1991.

[13] P. R. Williamson. Tomographic inversion in reflection seismology. *Geophysics*, 100:255–274, 1990.

[14] A. H. Wright. Genetic algorithms for real parameter optimization. In *Foundation of Genetic Algorithms*, pages 205–215. Morgan Kaufmann Publishers, 1991.