A MULTI-STAGE GENETIC ALGORITHM FOR TRAVEL TIME TOMOGRAPHY OF FLAT-LAYERED MEDIA

SEBASTIAN PADINA

A Multi-Stage Genetic Algorithm for Travel Time Tomography of Flat-Layered Media

by

A report submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Science

Computational Science Program

Memorial University of Newfoundland

July 2008

St. John's Newfoundland

Abstract

Genetic algorithms have long been employed in seismic tomographic inversion to obtain subsurface models from seismic traces, despite their relative lack of accuracy. While most such algorithms are basic in their design, I propose a multi-stage genetic algorithm for flat layer cellular seismic models which exploits the velocity similarities within individual layers. The algorithm starts coarse, with only one velocity value per layer, and gradually increases its granularity to 16 values, accordingly changing the algorithm parameters to reflect the different stages. By reducing the number of model parameters in early stages, the dimension of the search space is also made smaller leading to faster convergence. Although only approximations, the results of these early stages can then be used as improved initial guesses for the later phases of the algorithm. For a similar computational effort, this implementation yields more accurate models than the classic genetic approaches, thus rendering this type of inversion more practical.

Acknowledgements

I would like to thank my advisor, Dr. R. Phillip Bording, for his support and advice while working on this project. I would also like to thank PPSC, ACOA-AIF, IBM, Husky Energy and Memorial University for their funding. Dr. Tina Yu has also been of tremendous help in the early stages of developing the algorithm. Big thanks go to Dr. Andreas Atle, for being an excellent sparring partner in discussing ideas, and Dr. George Miminis, for his guidance. Finally I would like to thank my family for their moral support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Tomographic inversion is a highly non-linear error minimization problem and as such can benefit a lot from an evolutionary computation approach. In this report I look at the actual challenges of tomographic inversion as well as the work that has already been done on it, and describe a type of genetic algorithm that could be used to obtain relatively accurate models from field data. Given the interdisciplinary nature of the subject, the first two chapters introduce tomographic inversion and give a brief overview of the theory behind genetic algorithms. Chapter 4 presents a classic genetic algorithm that has been arrived at through testing of various parameters on a smaller scale problem. A discussion of the experimental results of the algorithm follows, leading to the observation of a shortcoming and proposing a way to address it. Chapter 5 is dedicated to the multi-stage algorithm with a focus on its approach of varying the number of parameters used to represent a model. The performance of the multi-stage GA is then investigated and its improvement over the classic algorithm accuracy is analysed through a direct comparison of the best model produced by each

method. Finally, Chapter 6 presents some final thoughts along with a few ideas for future improvement of the work presented here.

Chapter 2

Travel Time Inversion in Seismic Tomography

2.1 Introduction

The word *tomography* finds its origins in the two Greek words *tomos*, meaning section, and *graphy*, which translates as drawing. Many different types of tomography are used in many sciences such as medicine, biology, materials science and geophysics. Tomography uses surface measurements of the parameters pertaining to an object to infer various facts regarding the particular object's internal structure.

When processing seismic data, the Earth's internal parameters of velocity and density play a critical role [1]. Seismic imaging methods all require an accurate parameter estimation process [4]. For imaging algorithms based on depth rather than time for the output, it is essential that the interval velocities be determined in a cellular type of model rather than the traditional simple layered model. This allows

the velocity model to account for both lateral and vertical variations. The approach used in tomography, is a natural candidate for the cellular approach to seismic model building [1, 13].

In the following sections I review the notion of seismic tomography, how and why travel times are computed, the forward and inverse modeling problems, how a cellular model is developed, how ray tracing works, the linear algebra of the travel time equation and some of the challenges that arise from it.

2.2 Seismic Tomography

In the way that radiation and magnetic fields are used in medical tomography to produce an image of the interior of the body, the same principles can be applied to image the interior of the Earth. Because imaging the subsurface is the ultimate goal of seismic surveying, it makes sense for tomography to be discussed in this context.

However, due to the different scopes of investigations regarding said structure, most applications of seismic tomography find themselves divided between two types:

Global In global tomography scientists apply tomographic methods to surface data obtained naturally from earthquakes (about 2 million of them) to understand the deep structure of the entire Earth.

Near-surface This type of seismic tomography is concerned with the shallower subsurface, not going further than a few tens of kilometers. Its major application is in exploration geophysics where the subsurface is investigated in a search for specific substances such as oil or natural gas. As a major difference from global

tomography, this method does not rely on naturally-occuring earthquakes, but rather artificial seismic-wave sources, such as explosive detonations or air-guns. Artificial sources are better suited for this purpose because they can be placed at specific locations and detonated at a time of choice, unlike earthquakes.

This project concerns the latter type of tomography. Throughout the next sections I will explain the basic principles of tomography along with their associated computational issues.

2.3 Travel Time Tomography and Inversion

Travel time tomography refers to seismic tomography that puts the emphasis on the key aspect used to investigate the subsurface — *travel time*. The term refers to the time it takes a seismic wave to travel into the subsurface, reach a reflecting boundary and return to the surface. As mentioned previously, the seismic waves we are dealing with here are caused during systematic experiments involving the detonation of certain explosives or other energy sources.

To retrieve the travel time, receivers are placed on the surface away from the source of the explosion/vibration at predefined distances, typically in a straight line. These receivers will record any sounds that reach them along with the time of the occurrence and by using a method called *time picking* one can obtain approximate travel times for the waves from these records. Also known as *seismic reflection times*, these travel times can be used to estimate the velocities at which the energy waves traverse the subsurface. Knowing that seismic waves are elastic waves — the travel velocity of which is known for a certain medium — we can use this information to determine

the substances present in the subsurface and thus gain a better understanding of its structure. It should be at this point noted that for such a method to be successful, an elementary idea of the location of the reflecting boundaries is necessary.

The general procedure used in travel time tomography to determine velocities can be summarized in the three steps outlined below [3].

- In a primary step the field experiment is performed and the times for rays that reach each receiver are picked, thus obtaining the seismic travel times. As a remark, it should be mentioned that between the source and any receiver there is usually more than one ray. The actual number is usually equal to the number of reflecting boundaries.

- After retrieving the travel times one needs to gain an understanding of the distances travelled to be able make any assumptions on velocities. This is where ray tracing comes to our aid. Ray tracing is a method that applies the theories of how seismic waves travel within various mediums, the result of which is a schematic that will allow us to approximate the distances travelled between reflecting boundaries.

- In the concluding step, the data produced by the previous two steps is taken and so-called *travel time equations* are constructed, which can then be solved for velocity. This procedure is also called travel time *inversion*, because we start with experiment data and we infer a geophysical model that could have produced it.

In the sections to come I provide a more detailed explanation of the procedure de-

scribed above, with a focus on the last step and how it can benefit from an evolutionary computation approach.

2.4 The Cellular Model

In the initial stages of seismic tomography, it was generally assumed that the subsurface is divided into different layers [1] — each with a constant velocity and density throughout the layer — and that these layers were separated by flat interfaces. While this scenario makes for very easy computations it is not in fact at all accurate. Later experiments have confirmed that velocity varies a lot within the subsurface and while there are severe differences present in some areas causing an apparent layer structure, the interfaces present between layers are not flat and one layer cannot be approximated by having constant parameters throughout. This means that the velocity with which a seismic wave would travel within a layer does not stay the same throughout.

In this context there is a need to both divide the subsurface into constant parameter units that provide for a higher resolution and to find a mathematical way to better approximate the non-flat shape of the interfaces which become our reflection boundaries. It is clear that if an interface is not flat, its shape — regardless of how oscillating it may be — can be approximated to a curve or a set of curves that connect. Mathematics provides us with methods that can be used to take a set of connected curves and infer a function that will produce it. However, given the fact that we are dealing with real world structures here, their high degree of randomness renders this approach useless due to the complexity of the resulting functions which would make any computational effort unfeasible. Consequently, another approach is to divide the

subsurface into square cells for the 2-D problem [1]. The higher the number of cells, the higher the resolution and the better the accuracy. Interfaces can be represented by a jump in velocity between adjacent cells.

For the scope of this project, a compromise subsurface model was tackled. The model used here does indeed present a cellular decomposition, but there are horizontal flat layers between the cells. Figure 2.1 depicts the model used when developing the algorithm presented in later sections. As a remark it should be pointed out that although the flat-layer assumption might seem like an over-simplifying one, flat layers do a great job approximating real geology in some situations, such as is the case in central Kansas [14] or the shallow Cretaceous in Alberta [10]. Also, it will become clear when discussing genetic algorithms that there are only a few restrictions placed on the type of tomographic model that can be used with the proposed algorithm, flat interfaces not being among them.

Throughout this section, I described the subsurface layers and interfaces in a two-dimensional manner. In reality of course the interfaces are planes and the cells are cubes given the three dimensions of our space. Nonetheless, since tomography is by definition concerned with cross-sections, it is safe to discuss the problem in two dimensions instead of three.

2.5 Ray Tracing

The next aspect that needs to be addressed in order to understand the seismic tomography problem is ray tracing. Looking back at the travel time procedure explained in Section 2.3, we notice that, even though the experiment will provide the travel times,

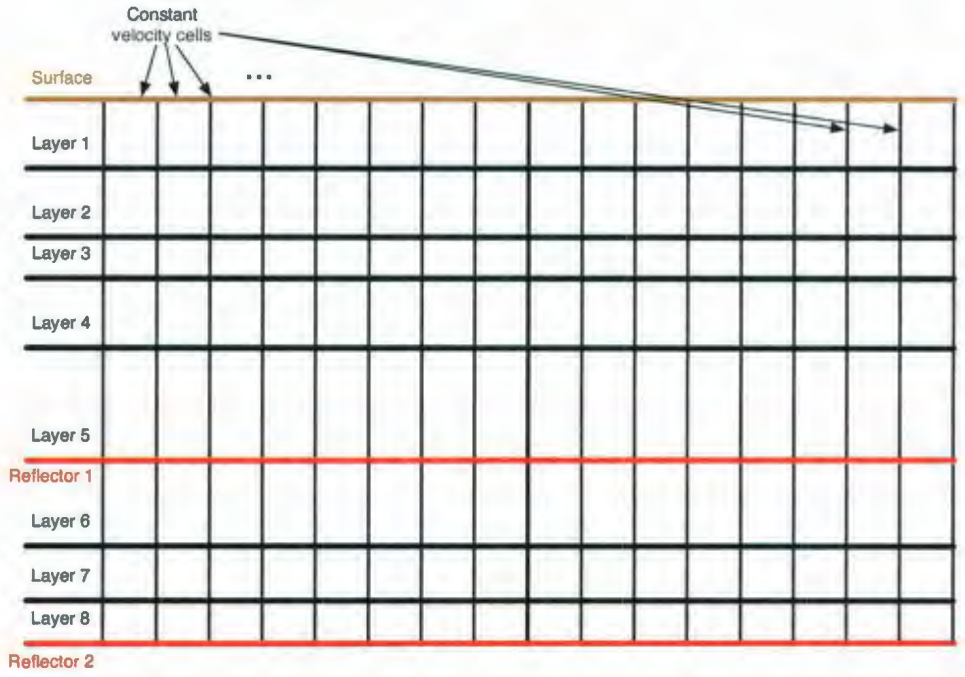

Figure 2.1: The two-dimensional flat-layer cellular model

we need an estimate of the distances travelled before we can calculate velocities. Ray tracing handles this part of the process and by superimposing our cell structure on the ray traced we can measure the distance each ray travels inside each cell.

Using the general approximation that energy travels from source to receiver along a ray path, ray tracing provides a set of rules that govern the path a seismic wave takes to reach a source. This includes methods of calculating the angle of reflection and also how the angle changes for the part of the ray that travels through the interface [12].

For this method to work properly we need to know where the reflection boundaries are located. Otherwise we have no way to say for sure how long the rays have travelled. This is to say we cannot use the method described here to deduce the structure of

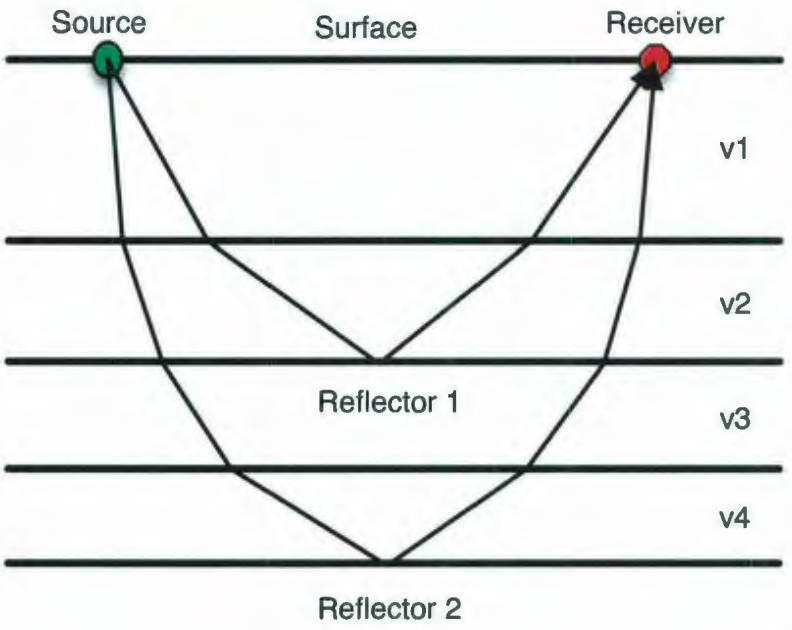

Figure 2.2: General example of ray tracing in a flat-layered model

the earth without any a priori knowledge, but rather to confirm and increase the accuracy of previous less exact findings. This is a constraint placed on the technique described in this report, but in the case of the flat layer model discussed here, all we need to know is the depth of the various layers which can be calculated by drilling a well anywhere along the surface and measuring them [14]. The normal moveout stack (nmo) and/or time migrations can also provide useful starting models. Because of the flat interfaces, the depth information will be the same regardless of the choice of drilling location. However, drilling one well tells us nothing about the velocity structure of the cells, and too many wells would be too costly.

In Figure 2.2 a typical source/receiver pair is pictured to illustrate the ray path bending and its return to the surface using Snell's law. Snell's law describes the

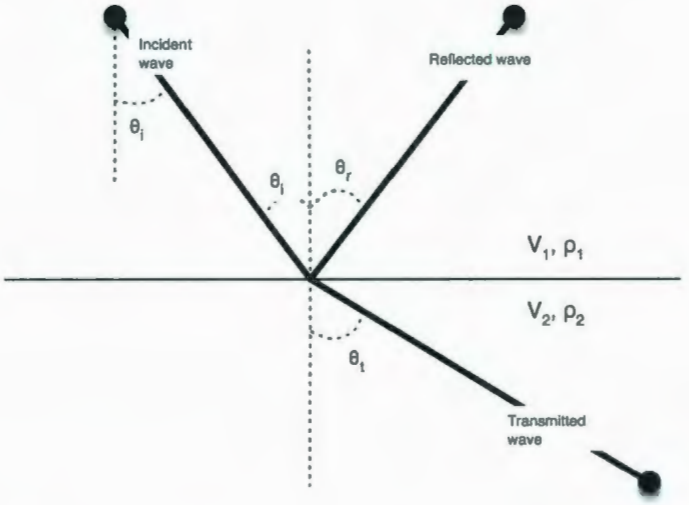

Figure 2.3: Illustration of Snell's Law

relationship between the refraction and reflection angles, and the two velocities of the media on both sides of the interface. A simple example is illustrated in Figure 2.3[1] and the relationship between the various parameters is given by Equation 2.1 [16]:

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{V_1}{V_2}, \theta_i = \theta_r \tag{2.1}$$

Figure 2.2 presents a simple flat layer model with four layers[2], two of which were chosen as reflectors because those represent the main reflections in the real data that provided the travel time picks. The real earth causes reflections at all impedence changes in the earth [14]. Some of the reflections have significant energy, more than others, and are dominating in the seismogram. By using a few layers as reflectors for the travel time picks we reduce that effort. These layers must encompass the breadth

[1]The is a simple representation. Usually, there is a differentiation between p-waves and s-waves when discussing Snell's Law [16], but that is beyond the scope of this project.

[2]The layers in the figure have constant velocity throughout, and the velocities increase the deeper the ray travels. This explains why the bending angles get larger in accordance with Snell's law.

and depth of the model [3].

2.6 Travel Time Equations

In order to reach the final goal of the inversion problem, to calculate the velocities of the rays within the cells based on the ray traces and the travel times, we need to describe the relationships present between the different values in a mathematical way. After the ray tracing, for every ray we will be able to write the following equation. Based on basic Newtonian mechanics we have

$$t_i = \frac{d_{i,1}}{v_1} + \frac{d_{i,2}}{v_2} + \frac{d_{i,3}}{v_3} + ... + \frac{d_{i,m}}{v_m} \tag{2.2}$$

where:

t_i is the time it took ray i to travel from source to receiver

$d_{i,j}$ is the distance travelled by ray i in cell j

v_j is the velocity corresponding to cell j (and constant throughout the cell)

To simplify the notation, we introduce a new unit S defining the slowness of a cell. S is defined to be the inverse of the velocity v in each cell, so we have

$$S_j = \frac{1}{v_j}$$

By replacing velocity with slowness equation 2.2 now becomes

$$t_i = d_{i,1}S_1 + d_{i,2}S_2 + d_{i,3}S_3 + ... + d_{i,m}S_m \tag{2.3}$$

Throughout the course of an experiment we deal with a very high number of rays, thus we have a multitude of equations of the form of equation 2.3. To present the

information in a clearer fashion we can rewrite all travel time equations in the form of a matrix-vector product which will give us

$$T_n = D_{n \times m} \times S_m \tag{2.4}$$

where:

n is the number of rays

m is the number of velocity/slowness cells

T is the vector of picked travel times

D is the distance matrix

S is the slowness vector for the cells.

Looking at equation 2.4 it might seem that knowing T — from the experiment — and D — from the model — is enough to solve for S. This is not the case as T is known to be accurate while D is an estimate obtained from ray tracing which was done based mainly on overly simplistic assumptions. Given an accurate set of travel times for the rays and estimated distance travelled within each cell — which has been calculated based on assumed slowness values — the goal becomes to calculate an accurate slowness vector.

This can be done by applying an algorithm that starts by formulating educated guesses about the slowness values and using those together with the ray tracing method to arrive at model travel time values [3]. These can be compared to the real travel times from that have been picked from the actual experiment data resulting in an error vector

$$\Delta T = T_{real} - T_{model} \tag{2.5}$$

where T_{real} stands for the picked travel times and T_{model} for the values obtained through ray tracing. The classical approach at this point would use the time travel errors to calculate the error in the slowness values, designated by ΔS and given by Equation 2.6.

$$\Delta S = S_{real} - S_{model}. \tag{2.6}$$

ΔS can be computed by solving the over-determined system[3] given by equation 2.7 for ΔS [3].

$$\Delta T = D \times \Delta S \tag{2.7}$$

Instead of the typical least squares approach usually employed in such situations, I suggest using a stochastic method — namely genetic algorithms — to continually iterate through slowness values until the error is minimized to an acceptable level. Even though this approach might not give a fully accurate set of values for the cell parameters, using its results as an initial guess for the linear algebra approach could dramatically enhance performance. This is because the least squares approach performs a very efficient local search, but it risks getting stuck in a local minimum, whereas genetic algorithms — though not entirely accurate — do a better job at finding the global minimum.

[3]This type of system is usually over-determined because in practice the number of rays (order of 10^4) far outnumber the number of cells (order of 10^2).

Chapter 3

Genetic Algorithms

3.1 Brief Overview

Genetic Algorithms (GAs) are a general term used to describe a series of stochastic search methods first developed in the 1960s and 1970s by J.H. Holland [9]. As part of the larger field of Evolutionary Computation, GAs take their inspiration from the process of biological evolution and are based on the principles guiding natural selection and genetics [5]. Unlike most theory in Evolutionary Computation, GAs first came about not as a method for algorithm design and optimization, but rather as a means to simulate and study natural adaptation [6]. The appeal of natural adaptation when investigating search algorithms stems from the similarity in purpose of the two. Many computational problems boil down to finding a solution in a very large search space G_n, but that is precisely the goal of adaptation in nature — searching through a large set of genetic combinations that would make an individual most likely to survive and thrive in its environment.

In their simplest form, genetic algorithms start with a *population* made up of a fixed number of candidate solutions. These individual candidates are represented as sets of parameters that make up the genetic material or the *chromosome*. Chromosome representation is chosen based on the dimensions of the search space G_n. Using an objective fitness evaluation function, all individuals in the population are tested to see how well they perform in the context of the problem to which an optimal solution is being sought. Ideally a fitness evaluation function should be chosen in such a way as to give a measure of how well the solution behaves, so that the members of a population can be ranked. The fittest candidate solutions — the ones whose parameters have performed best — are selected and the biological process of genetic recombination (or *crossover*) is applied to them resulting in *offspring* incorporating genetic material from more than one previously fit individual. The fitter an individual is, the higher its probability to be chosen as a parent for genetic recombination. The offspring are then mutated by applying random changes to their parameters and the results are stored in a new population. This consists of an iteration, also referred to as a *generation*, and the algorithm runs in cyclic fashion until either an arbitrarily fixed number of generations has elapsed or a solution has been found to be acceptably accurate, thus completing a *run*.

Probability and random numbers play a very important role in the way a genetic algorithm runs. The initial population is seeded with the help of random numbers, parent selection is done with a certain probability based on fitness, crossover and mutation apply with individual probabilities, and this is not meant as an exhaustive list. Consequently, two separate runs will yield slightly different results for a best overall individual and performing several runs that can be averaged is ideal [9].

J.H. Holland provides a more detailed description of the theory behind genetic algorithms in [6].

3.2 Relevance

Performing tomographic inversion as discussed in the previous chapter can easily be construed as a search problem. Representing a seismic model as a set of parameters means that the entire set of possible values for each parameter forms a search space. It seems natural then that GAs are a likely candidate for solving seismic inversion problems.

Genetic algorithms have three very useful properties [15]. GAs

- are tolerant of noise — because only the genetic material of the fittest individuals make it through to the next generation, particularly bad models in a population will have no impact on the search.
- work well in problem spaces where there are large numbers of local optima — a quick look at the travel time equation will reveal that they do not have unique solutions, resulting in many local optima that a typical algorithm could easily become stuck i.
- work well for problems where gradient information cannot be calculated or is difficult to obtain — this is indeed the case with tomographic inversion.

Stochastic methods are not a new development in seismic inversion. Simulated Annealing (SA) has been found to be very successful at producing useful models and has been used extensively in commercial applications [8]. The main difference

between Simulated Annealing and GAs lies in the way they deal with the models they produce. While the SA algorithm tests individual solutions one at a time and proceeds based on the results of the evaluation [8], GAs proceed in parallel with a set of models that are evaluated with the best being pursued further. One could argue that this would provide for more diversity and less risk of losing a promising solution, thus warranting the use of genetic algorithms in the context of seismic inversion.

3.3 Genetic Algorithms in Seismic Inversion

Most literature on genetic algorithms applied to seismic inversion has been written by either geophysicists making their first forays into GAs or by computer scientists with little domain knowledge, but who are given the necessary tools to be able to run a genetic algorithm on seismic problems. Because of this, most of the research done limits itself to rather simple and general types of genetic algorithms.

Most applications of genetic algorithms to inversion problems use a cellular model very similar to the one discussed in the previous chapter. Louis et al. [8] present some very encouraging results of genetic algorithms used to invert tomographic data with the use of ray tracing. The work is done with a ray tracing routine that the authors are given and their domain knowledge is limited. Nonetheless, their results prove that a typical genetic algorithm can easily outperform simulated annealing.

They also suggest interesting new crossover and mutation operators. The crossover operators used in their tests interchange columns or rows of cells between the two parents. While a row-oriented crossover operator would put the geological properties of flat layer models to good use, testing the column approach on my own data did

not produce statistically better models than the uniform crossover operator which I discuss in detail in the next chapter. The mutation operator in use by the authors works by selecting a rectangular block in the model and assigning all cells that fall within it a random velocity value. This changes the velocity values completely and in my own work I have opted to use less extreme mutation operators.

An even more compelling case for the use of genetic algorithms can be found in the work of Boschetti et al. [2]. Although approaching the problem with the same type of flat layer cellular model, velocity values are not assigned to cells, but rather to grid points around the cells. The velocity at a certain point is then computed through linear interpolation between immediately adjacent grid points. What is really interesting in this approach is the pseudo-subspace method described by the authors. At first the algorithm is run on a model with very few velocity grid points. Once a sufficiently accurate solution is found, the number of grid points is doubled, with each new point taking on the a velocity value equal to the arithmetic average of its neighbouring points. The algorithm is then run again and once a new accurate model is reached, parameters are doubled again [2]. This is very similar to my own work, with the main difference lying in my choice of expanding the model across layers as opposed to both horizontally and in depth.

Finally the work of Sadeghi et al. [11] describe a different type of genetic algorithm with remarkable properties. Although their research centres around the ray tracing part of seismic inversion — by using a GA to find the actual path a seismic wave travels between a source and receiver — they employ a particular type of genetic algorithm called a micro-GA. The micro-GA has a population size of five[1] and uses

[1]Typical GAs can have a population size anywhere between 50 and 1000 [9].

no mutation. In order to allow for random walks in the search space, the algorithm resets itself once a generation count has been reached by keeping the best individual found and replacing the other four with random seeds. A very interesting property of algorithms of this type is that they converge towards a correct solution relatively fast making them computationally efficient. However, their applicability are restricted to problems with small a numbers of parameters due to the small population size.

Chapter 4

The Classic Algorithm

4.1 Model Representation

The synthetic[1] tomographic model structure in Figure 2.1 presented in Section 2.4, which will form the basis for the experiments in this project, is eight layers deep with each layer further divided into 16 constant velocity cells. Although the cells are equally spaced in the horizontal direction, their height is equal to the thickness of the layer they find themselves in. The details regarding the depths and the range of velocity values expected for any given layer are outlined in Table 4.1[2]. The model has a depth of 2000 meters and a width of 1000 meters. On the surface, 50 sources and 50 receivers are placed at 20 meter intervals. The positions of the sources and the receivers coincide.

Please note that the velocities of the bottom two layers are slightly higher than

[1]This model does not correspond to real geology, but rather to a model specifically designed for this experiment. The velocity values and the layer thickness values were chosen at random but with care to allow for variation along a layer and a reasonable depth distribution.

[2](r) denotes that the interface at the bottom of the designated layer has been chosen as a reflector.

Table 4.1: Tomographic model details

Layer	Thickness (m)	Velocity Range (m/s)
1	500	1800-2500
2	500	2500-3500
3	300	2500-3500
4	500	3500-4500
5 (r)	800	4500-5500
6	600	5500-6500
7	400	6500-8000
8 (r)	300	8000-9000

typical geophysical models; they were chosen this way to emphasize and showcase an interesting effect of using the type of ray tracing outlined here as an evaluation function for a genetic algorithm. This will be addressed in detail in later sections.

A colour mapping of the velocity values for individual cells in the synthetic model which will serve as the reference for the experiments is presented in Figure 4.1. This is not a true scale representation of the individual cell sizes, but it should serve to give the reader an idea of the variation of velocities along individual layers.

While the reference model is restricted to flat layers, if a cell has a significantly higher velocity than its neighbours along the same layer, this could be seen as a sign that seismic waves travel faster through that region and perhaps the flat layer assumption could be wrong, giving the flat layer scheme a degree of usefulness even when the underlying geology does not present flat boundaries.

The structure of the model translates to $16 \times 8 = 128$ variables the algorithm

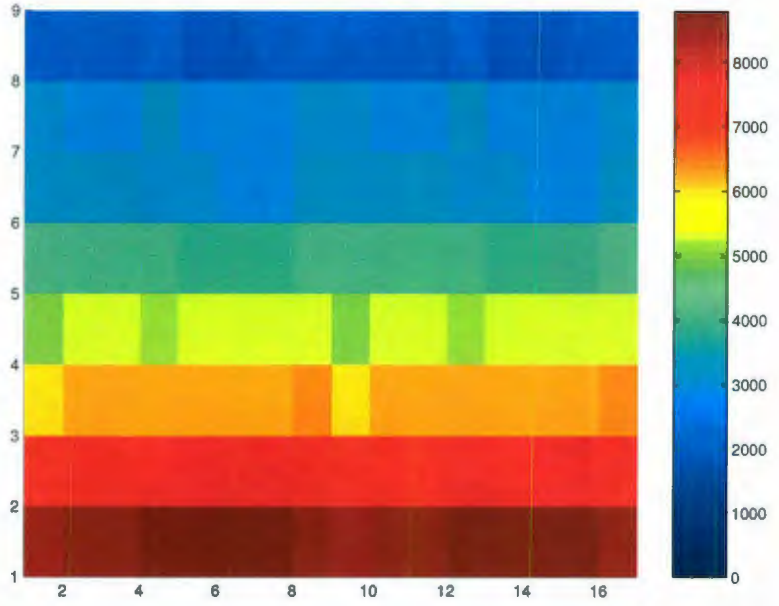

Figure 4.1: Colour mapping of the reference model

will be inverting for, given that depth information is considered known. Although typical general purpose genetic algorithms use a series of bits as the representation of a chromosome — with single or groups of bits used to encode individual traits — research has shown that real-coded GAs perform far better [2]. Consequently, the chromosome representing the tomographic model was chosen be a real-valued vector of length 128. Although it might seem like a matrix would be a more appropriate way to represent the structure of the model, the PGAPack GA libraries (which became the foundation on top of which my experiments were run) only support vectors natively [7]. This is not an inconvenience, as mapping vectors to matrices of given dimensions is an elementary operation.

In an earlier section of this report I made reference to the subsurface being three-dimensional, even though this work focuses mainly on a two-dimensional approach.

The main challenge when extrapolating the ideas presented here to a 3D environment comes in form of an increased search space G_m. If G_n is of the order n^2, the addition of a new dimension makes G_m a space of the order n^3. Additionally, even though the basic principles will work the same, sources and receivers would need to be spaced out over more than just one line when performing ray tracing in order to ensure proper coverage of all cubic cells. The number of sources and receivers is also required to increase by the order of n, leading to a higher computational effort when evaluating individual models.

4.2 Assigning Fitness

After establishing the chromosome representation for the model, the next step in describing a genetic algorithm is to define a fitness evaluation function. For this purpose, I wrote a ray tracing application that will calculate the travel times for seismic waves along the path from a source, to a reflecting boundary and back to a receiver on the surface. In homogeneous media, where the velocity is constant, seismic waves propagate in a straight line. Upon encountering a change in velocity, part of the wave's energy will be reflected back to the surface while the other part will continue through, albeit at a changed angle of incidence. Snell's Law governs the relationship between the two velocities and the angles of reflection and refraction [16]. For the refraction case, if the velocity is higher in the new medium, the refraction angle will also be higher than the incidence angle.

The theory is very easy to implement for flat layers with constant velocities, but in the models discussed here, the layers are further divided into cells of constant velocity,

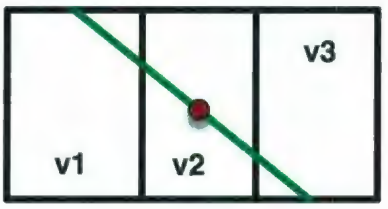

Figure 4.2: A ray crossing several cells within a layer

leading to the possibility of a ray passing through several cells within the same layer. Because the velocity differences within a layer are assumed to not be very significant, the ray tracer makes simple approximations to speed up the calculation of a travel time: the refraction angle will be calculated based on the velocity of the cell at which the ray enters the new layer, but the velocity used to calculate the travel time is the one at the midpoint of the ray segment enclosed within the layer's boundaries. Figure 4.2 describes a situation where a ray crosses three cells. The green ray travels at an angle based on the velocity v1, but the midpoint of the segment (the red dot in the figure) lies in the cell with velocity v2, which is what will be used to calculate the travel time.

After the ray tracing procedure is performed on a model, there will be a travel time associated with every source-reflector-receiver triplet. There are 50 sources/receivers and two reflectors, resulting in a total of $50 \times 50 \times 2 = 5000$ travel times. Initially, ray tracing is performed on both the reference model and the model under investigation resulting in two arrays of travel times of the same dimensions. Corresponding travel time values in the two models are then subtracted from each other, and the absolute values of the individual differences are added together to result in a positive error measure across the entire model. The smaller the error when compared to the

reference, the fitter a particular model will be. This will lead to a minimization GA, with the goal being to minimize the error.

As a remark, there is a possibility that no path can be found for a given triplet. This can happen when, due to the angle of refraction, rays leaving a source have no way to reach a certain receiver. To account for this, whenever a path cannot be found for a triplet, that triplet is assigned the travel time 0. This means that if there is a discrepancy between the reference and the candidate models in the number of actual travel times that could be computed, large differences in travel times will arise adding a large penalty to the fitness value.

4.3 Genetic Operators and Parameters

A typical genetic algorithm involves three genetic operators, corresponding to their biological equivalents: selection, crossover and mutation. In addition to these, other parameters describe the behaviour of the GA. Many types of operators and parameter settings are discussed in the literature and, to find the ones yielding the best results, I started by performing experiments on a similar tomographic model as the one presented here, but at a fourth of the scale so that the GAs produce results much quicker. Table 4.2 summarizes the operators and parameter values that were found to produce the most accurate models by comparing the results of these smaller experiments. They became a basis for the larger experiments. Following is a more detailed description of some of the table entries.

Elitism This determines the percentage of the population that will be carried across generations. In this case, the fittest 10% of the models will be copied, as they

Table 4.2: Parameters for Classic GA

Population size	100
Number of generations	500
Elitism	10
Crossover type	Uniform
Crossover probability	0.6
Uniform crossover probability	0.5
Fitness type	Linear ranking
Selection	Random binary tournament
Mutation type	Gaussian
Mutation probability	0.3125 (inverse of length of individual)
Intialization	Range

are, from one generation to the next. Elitism is useful in making sure crossover and mutation do not have a destructive effect.

Crossover type Uniform crossover proved to be the most accurate and it works by switching corresponding velocity values between the two parents.

Crossover probability The probability with which two parents will undergo crossover upon being selected.

Uniform crossover probability This value represents the probability that an individual cell in one parent will be exchanged for the corresponding one in the other parent. In essence, a value of 0.5 means that on average half of the values are exchanged between parents.

Fitness type Linear ranking simply means that once fitness is assigned, the individuals in the population are simply ordered from fittest to least fit.

Selection Random binary tournament selection works by picking two individuals at random and selecting the one that is fittest. The process is repeated to obtain a second parent. This type of selection ensures a relatively low selection pressure, thus preserving diversity.

Mutation type Gaussian mutation is typical for real-coded chromosomes. In this case, it works by adding a random number obtained from a Gaussian distribution with mean 0 and standard deviation 0.2.

Mutation probability This is the probability that a cell will undergo mutation. It is relatively high compared to typical GAs meaning that a lot of diversity

is introduced into the system. To make sure this does not have a destructive effect on good genetic material, the standard deviation of the Gaussian mutation number generator is comparatively low.

Initialization Setting this parameter to range means that individuals are first seeded from the range of possible velocities for any given layer. These ranges were set out in Table 4.1.

4.4 Experimental Results and Analysis

Performing the experiment described above over five runs resulted in an average best error of 57.37 seconds. Although the absolute error is used by the algorithm to assign fitness, it would be a good idea to put it in perspective by comparing it to the total sum of travel times in the reference model, which is 7622.68 seconds. The relative error is defined by the absolute error as a percentage of the total travel times, giving an average best for this experiment of 0.752%.

Figure 4.3 presents a colour mapping of the errors in velocity for the best model found by the GA. The values resulted from taking the absolute values of the differences in velocities of corresponding cells between the reference and GA models. As can be seen, the algorithm does a particularly good job with the top layers, but not so much with the bottom ones. In particular some cells have high enough velocities that they fall outside the allowed range. This can occur because although the algorithm seeds the velocities to be within acceptable ranges, it does not check at any later point whether they still conform. It is assumed that if these values become to high or too low, the resulting contribution to the fitness function will be high enough as to

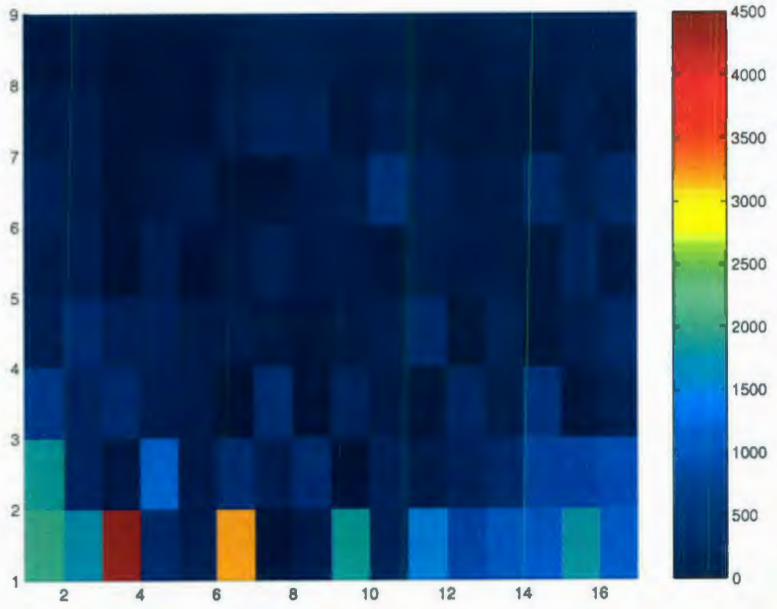

Figure 4.3: Colour mapping of velocity errors for classic GA

penalize the respective individuals. One reason why bottom layers are not as accurate as the top ones could be because the cells in these layers are less travelled by rays, thus resulting in a smaller contribution to the error. This is true because half of the travel times are the results of reflections from the first reflector, with rays only travelling as far as the fifth layer. Unlike the other half, these rays will never reach the lower layers. In particular, this effect is further emphasized in the bottom corners of the model, with even less ray paths crossing those cells.

Figure 4.4 illustrates a plot of the average best errors across the five runs, focused around generations 211-224. Analysing this plot reveals an interesting stair-like pattern of the curve, particularly around generations 214-215 and 219-221, where the plot becomes flat before suddenly dropping only to flatten out again. This behaviour starts around generation 180 and propagates throughout the rest of the GA. Such

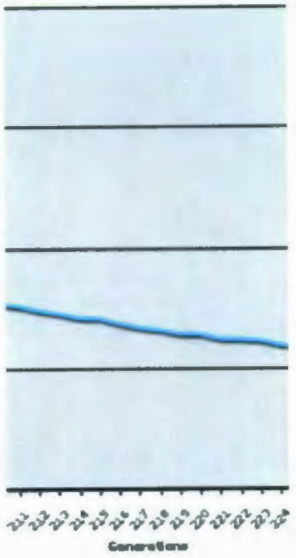

Figure 4.4: Focused plot of average best error

an occurrence would seem to suggest that there are certain generation transitions where the best error is not reduced and the GA gets stuck in local minima. Despite finding better models in later generations, this cycle repeats itself. This is most likely caused by the underlying error calculation, which can lead to situations where different models map to the same error value, a problem inherent in the summation used to calculate the travel time error. The generations where *neutral walks* in the search space are performed are a waste of computational effort. To eliminate this shortcoming, it might prove wise for the algorithm to identify areas in the search space that hold the most promise and investigate those in an attempt to avoid becoming stuck in local minima.

Chapter 5

The Multi-Stage Algorithm

5.1 Exploration vs Exploitation

Two key concepts involved in choosing the ideal parameters for a genetic algorithm are *exploration* and *exploitation*. The former refers to the degree to which the GA covers the entire search space and it is an indicator of diversity. Like the term's name suggests, this boils down to making sure the entire breadth of the solution space G_n is *explored*. The latter concept relates to a genetic algorithm's ability to *exploit* the solutions that have already been found to be quite fit, both by recombining such solutions to form new — perhaps better — ones and by performing a local search around fit solutions. In the ideal case, a strong GA will strike a perfect balance between those to aspects.

To this end, the multi-stage algorithm proposed here will start by focusing on exploration in an attempt to quickly scan the search space and identify the areas with the highest potential. Gradually, as these areas are identified, exploration will

give way to the classic, more balanced approach. To achieve efficient coverage of the solution space in the early stages without incurring a very high computational cost requires a type of GA that converges quickly. The main aspect of a genetic algorithm that determines the speed at which it will converge is the size of its chromosomes — which in turn leads to a smaller population size required to maintain diversity — with smaller size individuals preferred for fast convergence [11]. In keeping with this approach, the multi-stage algorithm will start by running the GA on a coarse version of the model and, as the error is reduced to a satisfactory degree, more refinements can be introduced in subsequent stages until the model reaches full complexity. The size of the problem given by the dimension of the search space G_n changes every stage, with $dim(G_n^{coarse}) < dim(G_n^{fine})$. Essentially, the multi-stage algorithm is a collection of GAs working on models with varying granularities and with accordingly adjusted parameters.

5.2 Multi-Staging and Relevant Parameters

The best way to reduce granularity, and thus the size of the chromosomes, is to take advantage of the layer structure present in the model. Although there are differences in the cell values across any given layer, these differences are much smaller when compared to the velocity jumps between layers. Hence, the ideal approach is to reduce the number of cells per layer while keeping the layer structure intact, with the new larger cells holding values that would in actuality be averages of their higher resolution counterparts. The number of cells can then increase from stage to stage.

The algorithm has four stages, with the models starting out with one velocity

value per layer in stage 1 and eventually reaching 16 cells by stage 4, as illustrated in Figure 5.1 on a four-layer model. The blue solid lines identify the division that occurs after the first stage is complete, the red dotted lines represent the transition to stage 3 and, finally, stage 4 is represented with purple dashed lines. Given the approximations and averaging that need to occur for the initial stage models to approximate the travel times for the reference model, it would be unrealistic to expect accurate solutions to be found so early in the algorithm. Fortunately, the purpose of the early stages is simply to provide subsequent phases with a better starting guesses than random seeding alone would be able to produce.

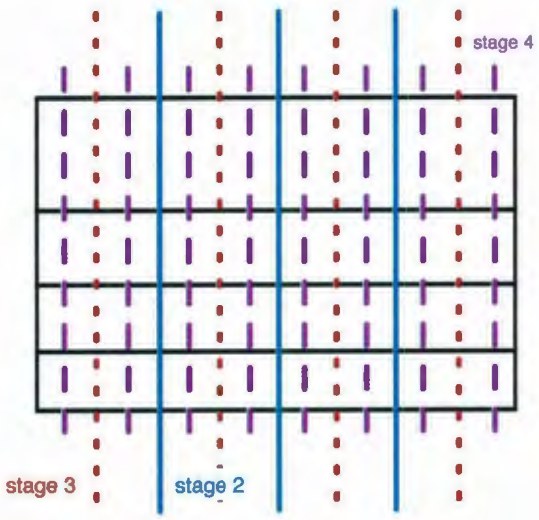

Figure 5.1: Dividing layers into cells across the four stages

Table 5.1 outlines the stages and their differing parameters. In the cases where the chromosomes are of small size, there is no need to maintain the same type of diversity as in later stages, so the population size can be reduced. The number of generations the low granularity GAs need to run for can also be made significantly smaller due

to their most useful property: fast convergence. The actual generation counts were chosen in such a way as to result in an overall algorithm with similar computational requirements to the classic GA outlined in the previous section, in order to allow for a direct comparison of the two. The most computationally intensive section of both algorithms is represented by the ray tracing routine used as the fitness evaluation function. The classic GA has a constant population size of 100 and will iterate over 500 generations in one run, yielding up to 55,000 calls to the ray tracing routine. With its varying population and generation counts, the multi-stage algorithm will need to perform a maximum $10 \times 50 + 20 \times 100 + 40 \times 200 + 100 \times 300 = 44,000$ model evaluations, which is lower than the classic GA in order to make up for the extra work that needs to be performed at each stage transition described later in this section.[1] All other parameters of the individual GAs represented by each algorithm phase are kept the same as in the classic algorithm.

Table 5.1: GA and model parameters across stages

Stage	Cells per layer	Chromosome size	Population size	Generations
1	1	8	10	50
2	4	32	20	100
3	8	64	40	200
4	16	128	100	300

Generally speaking, the transitions between phases of the algorithm boil down to

[1]The number of calls to the ray tracing routines given here are upper bounds. As a result of the 10% elitism used in both algorithms, some of the individuals will not need to be re-evaluated for each generation once their fitness has been computed.

dividing each existent cell of the model into either two or four new ones that are close in magnitude. To find the values for the new cells, the best individual at the end of each stage is saved and Gaussian perturbations are applied to each of its velocities. A Gaussian perturbation is merely another way to refer to the addition of a random number from a narrow Gaussian distribution to the already existing value — this is very similar to the mutation process employed by the algorithm. The results of the perturbations will then become the new values for cells resulting from the division. A simple illustration of this concept for a cell being divided in two is presented in Figure 5.2. This process is repeated until enough individuals have been created to populate the first generation of the new algorithm phase. One could argue that taking only the one best individual from each stage to populate the next could be narrow-minded and damaging to the diversity of the models, but analysing the velocities of the individuals in the last generations of the early phases shows that there is a tendency for all models to converge to similar values. Picking more than one fit individual as a seed for the next step would thus not offer any major advantage.

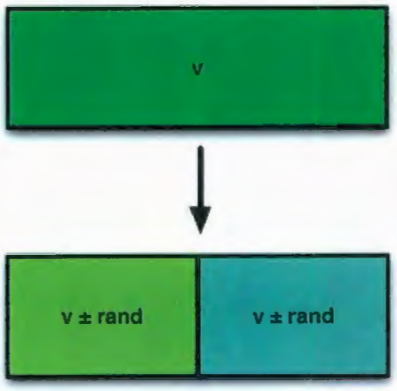

Figure 5.2: Cell division between stages

5.3 Experimental Results and Analysis

Like the classic GA, the multi-stage algorithm experiment was performed using the same reference model over five runs. The best average error came in at 26.32 seconds, or 0.345% of the reference model travel times, which is less than half the best error of the classic GA of 0.752%. Table 5.2 outlines the evolution of the best errors as the algorithm progressed through its stages by listing both the starting and the ending best errors for each stage. The reader will notice that there are slight discrepancies in the ending error for one phase and the starting error for the next. This occurs because when a new stage is being seeded, the Gaussian perturbations applied to the previous best model may result in fitter chromosomes. This is not necessarily always the case, and whether or not the new stage will start with a lower error depends exclusively on the chance that the stochastic seeding process does indeed result in positive walk through the search space. Nevertheless, with an increase in the number of individuals inherent in the transition to a new stage in the algorithm, it is very likely that a better error will be identified within the first generation, such as was the case in my experiments.

Table 5.2: Starting and ending best errors

Stage	Starting average best error (seconds)	Ending average best error (seconds)
1	201.135	125.21
2	124.76	86.97
3	85.98	60.79
4	59.97	26.32

If one thinks of each individual stage in the algorithm as a separate GA, the reader will no doubt notice that stage 4 of the algorithm is in many ways identical to the classic GA, bar a lower number of generations. A very interesting observation in this context is that the starting error for this stage is actually only slightly higher than the final error of the classic GA, which stands as a testament to the efficiency of the preceding algorithm phases. In fact it can be said that the best error of the classic algorithm was nearly matched by a three-stage algorithm with less than a third of its ray tracer calls $(44,000 - (300 \times 100) = 14,000)$.

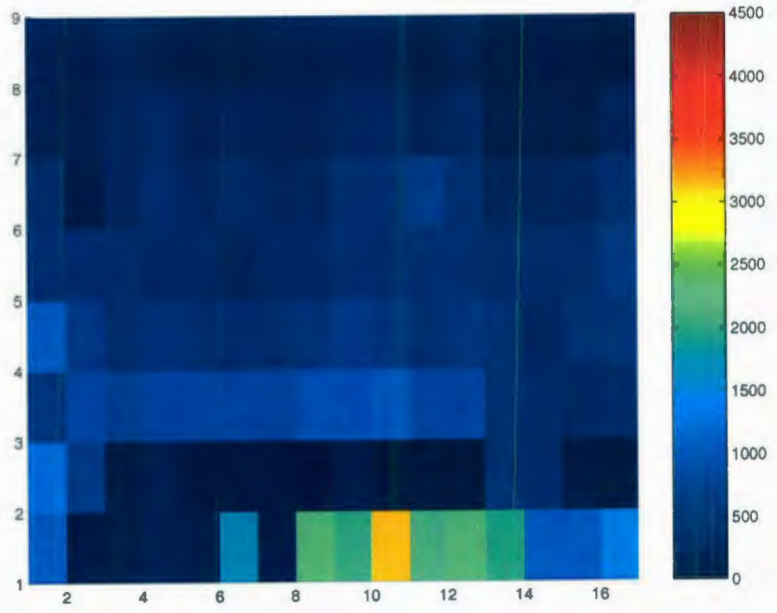

Figure 5.3: Colour mapping of velocity errors for multi-stage GA

Just like in the case of the classic GA, Figure 5.3 presents a colour mapping of the differences between the velocities in the best model and the reference model. When compared to the same mapping for the classic algorithm given in Figure 4.3, the most immediate observation is that the multi-stage algorithm does a much better job in

matching the velocity variations across individual layers. Whereas the colour map of the classic GA errors presented many changes in shading across the same layer (this effect increases with depth), the multi-stage GA is much smoother. The best model produced by the multi-stage GA also does a very good job with the top two layers and the bottom two layers. Despite the warmer colours of the bottom layer, this is in fact fairly accurate, the warmer shades being the result of the fact that these are high velocities and they will result in higher magnitude errors. Having said that, one cannot help noticing that the colours seem to indicate that this model has higher errors, especially across the middle sections. For a better look, Figure 5.4 displays the colour mappings of the reference model along with the best results form the classic and multi-staged algorithms. Although confirming the initial observations of the multi-stage model doing a better job in matching the velocity structure across individual layers and the low errors in the top and bottom layers, the middle of the model is visibly worse when compared to the classically obtained model. In particular, it seems like layers 5 and 6 have been exchanged for each other.

The explanation for this paradox comes from considering the way errors are calculated, in particular the penalty incurred by a model when no path can be found for a given source-reflector-receiver triplet, although that path exists in the reference model. According to Snell's Law, there is a direct proportionality relationship between the angle of refraction in a new layer and the layer's velocity [16]. This leads to the observation that given the very high values of the velocities in the bottom layers, inaccuracies here can lead to ray paths bending so far as to leave the model and not be able to reach their designated receivers. In turn this will result in many travel times equal to zero and, when calculating the error, these zero values will cause the

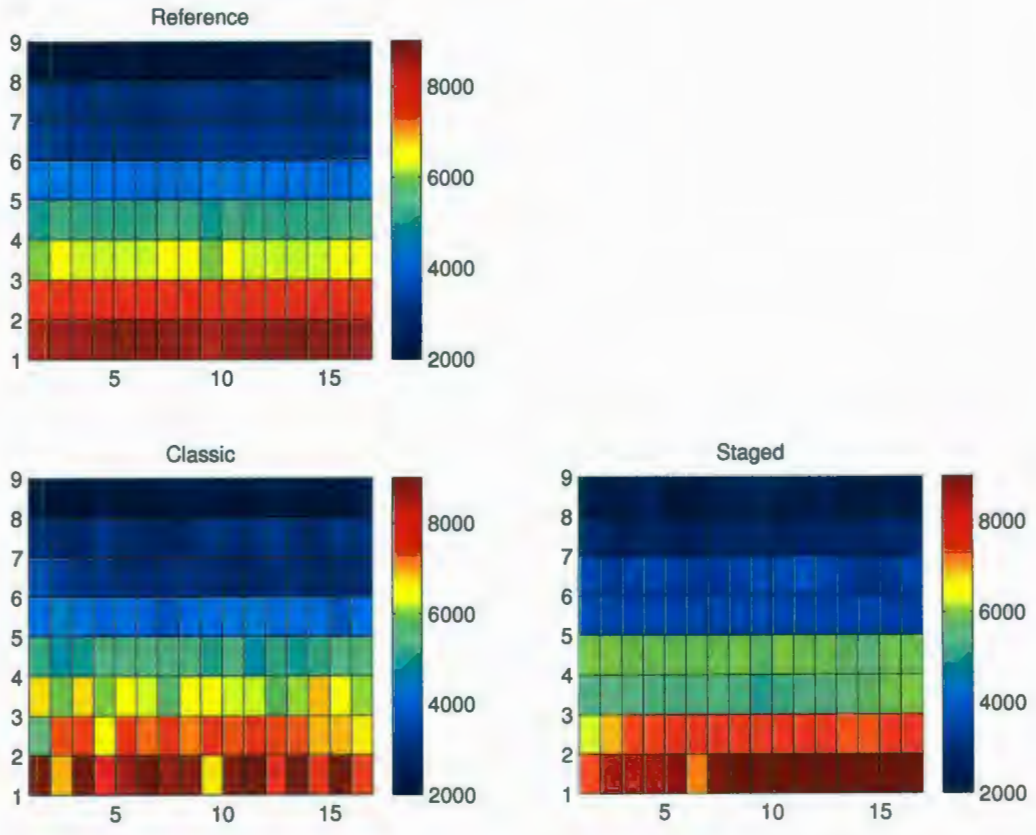

Figure 5.4: Comparison of the reference model to the classic and multi-stage best models

corresponding travel times from the reference model to be added directly to the total error value. A key point is that these travel times that feed straight into the error will come from the reference model and as such will be *constants*. This means that there will be ranges[2] for the velocities in the bottom layers that will always induce constants into the error function, forcing the algorithm to focus on improving the fit of the upper layers where the error function is more sensitive towards slight changes in values.

As such, the model produced by the classic algorithm does a fairly poor job on the bottom layers, and is thus pushed to focus more on the upper layers. Because the multi-stage algorithm takes advantage of the layer structure in its domain decomposition, it will result in much better estimates for the bottom layers fairly early on leading to less penalties and less constants added to the error. This significantly increases the sensitivity of the total error to changes in these bottom layer velocities, and the algorithm is able to more accurately evolve these velocities at the expense of the middle layers. Although this might seem disadvantageous, it is worth mentioning that this approach, if left to run over more iterations in the last stage, will eventually start refining the middle layers once no major improvements can be made to the deeper end of the model. The same cannot be said for the classic algorithm.

[2]These ranges are larger than the changes the algorithm is able to bring about in the bottom layer velocities through mutation, thus making it improbable that the GA will find better fits for those particular cells.

Chapter 6

Conclusions

Overall, genetic algorithms are a search method very well suited for travel time tomography, mainly because of their ability to tackle highly non-linear problems. They can also offer a lot in terms of computational efficiency when compared to other stochastic methods due to the highly parallel approach of assigning fitness to individuals — the fitness evaluation of each model is independent from the others, offering a situation where each call to the ray tracing routine can be processed in parallel. Also, there is a clear separation between the inner workings of the genetic algorithm itself and the method employed to actually compute the error. This makes GAs highly versatile plug-and-play techniques with no restrictions placed on the error function that could be used, with the exception that it be an objective function which would allow the ranking of results. In addition, as long as a suitable parameterisation can be found, there are endless possibilities for the choice of model representation. The ones shown here are merely an example.

The results of the two algorithms outlined in this report are very promising with

both producing an error significantly smaller than 1% of total travel times. The two GAs share a lot their parameters, and these were determined to indeed be the most suitable through trial and error on smaller scale models. Future work can be done to improve on the results of these experiments. Assigning weights to the high velocity cells, or any cells that would not be contributing to the error function as much as others, as well as attempting to evolve values for the layers one at time are particularly promising approaches. Although not directly related to the algorithm per se, changes to the actual ray tracing routine — where perhaps the situations where no paths can be found for a triplet can be ignored.

What sets the multi-stage algorithm apart is the way it uses domain knowledge, in particular its ability to exploit the layered structure of the model. By trying to find approximate models with fewer parameters during the initial stages, the dimensions of the search space are also reduced. This technique allows for much quicker initial convergence, resulting in models that serve as excellent guesses for the later stages that increase in complexity. Even though it appears that this tactic can be detrimental to the accuracy of some of its velocity values, this type of algorithm is in fact more robust than a classic algorithm given the type of ray tracing used for error calculation as the experiments have shown. Perhaps the best testament to the superiority of the multi-stage approach lies in the fact that its performance in terms of error reduction is only slightly higher than its equivalent classic algorithm but for 28% of the computational cost.

Genetic algorithms are not designed to be extremely accurate, but rather fast search methods. Ideally, their results would be fed into a local search method employing mathematically more accurate techniques, such as Newton's method. These

techniques need good starting guesses and GAs are very well suited for this purpose. Due to a more accurate approximation of the relative velocity structure across layers, the multi-stage algorithm's results recommend it as a better choice in this context.

Bibliography

[1] R. P. Bording, A. Gerszetenkorn, L. Lines, J. Scales, and S. Treitel. Applications of seismic travel time tomography. *Geophys. J., Roy. Astr. Soc.*, 90:285–303, 1987.

[2] F. Boschetti, M. C. Dentith, and R. D. List. Inversion of seismic refraction data using genetic algorithm. *Geophysics (Society of Exploration Geophysicists*, 61(6):1715–1727, 1996.

[3] D. Churchill, S. Padina, and R. P. Bording. Seismic tomography as a high performance application. In *Proceedings of the 20th Annual International Symposium on High Performance Computing Systems and Applications (HPCS 2006)*, page 32. IEEE Computer Society, 2006.

[4] M. Gadallah. *Reservoir Seismology: Geophysics in Nontechnical Language.* Tulsa, Oklahoma: PennWell Books, 1994.

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

[6] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press (Second Edition, MIT Press 1992), 1975.

[7] D. Levine. *Users Guide to the PGAPack Parallel Genetic Algorithm Library.* Argonne National Laboratory, 1996.

[8] S. J. Louis, Q. Chen, and S. Pullammanappallil. Seismic velocity inversion with genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation.* IEEE Computer Society, 1999.

[9] M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, sixth printing edition, 1999.

[10] B. Russell. The old and new in seismic inversion. The 2006-07 CSEG Distinguished Lecture, 2007.

[11] H. Sadeghi, S. Suzuki, and H. Takenaka. Inversion of seismic refraction data using genetic algorithm. *Physics of the Earth and Planetary Interiors*, 113:355–365, 1999.

[12] M. M. Slotnick. *Lessons in Seismic Computing.* Society of Exploration Geophysicists, fourth printing edition, 1986.

[13] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation.* Philadelphia, Pennsylvania: SIAM, 2005.

[14] S. Treitel, L. Lines, and G. Ruckgaber. Geophysical inversion and applications. Course notes book printed by Memorial University and published by SEG, 1993.

[15] L. D. Whitley. A genetic algorithm tutorial. Technical Report Nb. CS-93-103, Colorado State University, 1993.

[16] O. Yilmaz and S. M. Doherty. *Seismic Data Processing (Investigations in Geophysics, Vol 2)*. Society of Exploration Geophysicists, 1987.