IDEA League

Master of Science in Applied Geophysics

Research Thesis

An enhanced Genetic Algorithm and its application on two non-linear geophysical problems

Yenni Paloma Villa Acuna

August 10, 2018

An enhanced Genetic Algorithm and its application on two non-linear geophysical problems

Master of Science Thesis

for the degree of Master of Science in Applied Geophysics at
Delft University of Technology
ETH Zürich
RWTH Aachen University
by

Yenni Paloma Villa Acuna

August 10, 2018

Department of Geoscience & Engineering · Delft University of Technology
Department of Earth Sciences · ETH Zürich
Faculty of Georesources and Material Engineering · RWTH Aachen University

Delft University of Technology

Printed in The Netherlands, Switzerland, Germany

IDEA LEAGUE
JOINT MASTER'S IN APPLIED GEOPHYSICS

Delft University of Technology, The Netherlands
ETH Zürich, Switzerland
RWTH Aachen, Germany

Dated: *August 10, 2018*

Committee Members:

Dr. Ir. G.G.Drijkoningen

Dr. Yimin Sun

Dr. Ir. Florian Wellman

Supervisor(s):

Dr. Ir. G.G.Drijkoningen

Dr. Yimin Sun

Abstract

Since its inception in 1975, Genetics Algorithms (GAs) have been successfully used as a tool for global optimization on non-convex problems in a wide range of real world applications. Its creation was inspired by natural adaptation and selection mechanisms that evolve from one population of chromosomes to a fitter population by means of an artificial natural selection dictated by the operators of elitism, crossover and mutation. An advanced Genetic Algorithm (aGA) was proposed by Sun et al. [2017], and this algorithm seeks the global maximum of n-th dimensional non-convex functions. However, convergence speed is a key factor for the success of a global optimization algorithm when it comes to scalability; in production even a slight efficiency improvement matters as it can easily takes weeks or even months to process a huge data set. Therefore, the goal of this project is to improve the convergence speed of the currently available aGA by simultaneously enhancing both its global and its local search capabilities. To this end, two solutions were proposed. The first is a modified version of the well known *Island model GAs* and the second was named *Self Adaptive Differential Evolution (SADE) fine tuning* scheme.

After a successful demonstration of its improved performance on several multi-modal test functions, the enhanced Genetic Algorithm (eGA) is used to tackle two common non-linear geophysical problems: static correction and Common Reflection Surface (CRS) stacking. In the former, the near-surface related time-shifts are estimated without resorting to an explicit velocity-depth model; instead, the events of interest are aligned in a data-driven fashion by maximizing the stacking power. The latter is a novel alternative to the traditional Common Midpoint (CMP) stacking that has proven to yield higher quality images, specially when applied to low Signal to Noise Ratio (SNR) data or data with challenging structures like the anomalies encountered in the subsurface. This improvement in the quality of the stacked image is attributed to a non-local mean mentality, which enables using traces from different CMP gathers to the CMP gather being resolved. Owing to the high non-linearity of both problems, they are ideal test beds for global optimization algorithms. The effectiveness of the eGA is demonstrated on synthetic data sets with promising results in these problems.

Table of Contents

Acronyms

ACO Ant Colony Optimization

aGA advanced Genetic Algorithm

AI Artificial Intelligence

BLX Blending Crossover

BFCD Back and Forth Coordinate Descent

CI Computational Intelligence

CMP Common Midpoint

CRS Common Reflection Surface

eGA Enhanced Genetic Algorithm

EC Evolutionary Computation

GAs Genetic Algorithms

MSA Modified Simulated Annealing

PML Perfectly Matched Layer

PBC Periodic Boundary Condition

PP Primary to Primary or P-P wave

PS Primary to Secondary or P-S wave

PSO Particle Swarm Optimization

SA Simulated Annealing

SADE Self-Adaptive Differential Evolution

VFSA Very Fast Simulated Annealing

ZO Zero Offset

Chapter 1

Introduction

1-1 Seismic Exploration

Geophysical methods have allowed humans to study the Earth's interior in order to determine its structure and most likely composition. This need arises from the fact that most minerals and materials essential to the survival of humanity are extracted from the first several hundred meters of the subsurface.

Seismic exploration is one of many geophysical surveys used for this purpose and aims at retrieving an image of the subsurface from active acoustic/elastic reflection measurements. Take the marine seismic survey as an example. Seismic sources (airguns) and receivers are located below the water surface. The source emits an acoustic wavefield that gets reflected back and is finally detected by the receiver channels. This method involves large volumes of data given that thousands of receiver channels are used per source and the experiment is repeated for many different source locations. Depending on the ocurring physical phenomena and the type of waves recorded, seismic exploration can be classified into *Primary* (or P-P) and *Converted* (or P-S) wave exploration.

Seismic exploration faces a major challenge of complex seismic wave propagation effects in the near-surface area, due to its unconsolidated character. As a result, layers within the near surface are often characterized by highly complex velocities and densities. During the seismic survey, seismic waves pass through the complex near-surface area twice, and this distorts the image quality of deep events of interest. This is the well-known near-surface problem that degrades the final seismic image quality severely. For more than 70 years different techniques have been proposed to handle the near-surface problem, to name a few: common focus point [Sun and Verschuur, 2012; Sun et al., 2014], full wavefield redatuming [Vrolijk et al., 2012], Green's function estimation [Berryhill, 1984], full waveform estimations (FWI) [Jones, 2012] and static correction [Cox et al., 1999]. In production, static correction so far is a commonly accepted technology for handling the near-surface challenge.

1-2 Literature Review

1-2-1 Static correction

Static correction is based on the premise that topography and smooth velocity variations induced anomalies in the near-surface area that can be approximated merely by time shifts, given that rays follow almost vertical trajectories while traveling through this area [Ronen and Claerbout, 1985]. The aforementioned assumption is valid as usually the velocity increases at deeper depths.

Statics can be estimated by either model-driven [Berryhill, 1984; Bevc, 1997; Reshef, 1991; Shtivelman and Canning, 1988] or data-driven methods [Ronen and Claerbout, 1985; Sun et al., 2017; Sun and Verschuur, 2012; Sun et al., 2014; Sun and Verschuur, 2014]. In the former, an explicit velocity model of the complex near-surface layer must be estimated first, whereas in the latter the statics are estimated by maximizing the stacking power of the image containing the complex zone of interest.

For model-driven approaches, the velocity model of the near-surface can be obtained by surface wave inversion [Douma et al., 2011], refracted wave inversion [Duret et al., 2016] or full-waveform inversion [Liu et al., 2013]. Based on this velocity model, it is then possible to calculate statics.

In data-driven methods, maximization of the stacking power can be "guided" by pilot traces. However, if these pilot traces are not available, static correction can also be done in a fully automated manner, by maximizing the power of stacked traces [Ronen and Claerbout, 1985; Sun et al., 2017]. Furthermore, Koglin et al. [2006] integrated a residual static correction method into a data-driven common-reflection-surface-stack-based imaging work-flow to eliminate residual statics with 2-way travel times described with less than 10 parameters. The above-mentioned methods have been, to some extent, successfully applied to 2D data, but the computational expense and calculation time has prevented their application to large 3D data.

1-2-2 Commom Reflection Surface Technology

The Common Reflection Surface (CRS) stacking method can be seen as a more sophisticated version of the Common Midpoint (CMP) stacking, thanks to the adoption of the non-local mean mentality, as it allows for the inclusion of traces from different CMP gathers, now arranged in a super-gather. As a result, a higher quality stack image with more continuous reflectors, improved spatial resolution and a higher SNR is produced. Unfortunately, this comes at the expense of higher computational costs to estimate additional parameters required for the moveout correction.

CRS technology seeks to determine some parameters (three in 2D and eight in 3D) of the travel-time approximation that maximize the coherence for each same seismic event in all traces belonging to the same super-gather. In other words, the key is determining the optimal CRS attributes that best describe the stacking surface fitting the reflections of the prestack data. Therefore, this problem narrows down to simultaneously search for the attributes of each seismic event at every midpoint, by means of a multidimensional global optimization

algorithm that maximizes the coherence of those seismic traces within a certain aperture of the stacking surface.

Owing to the high number of traces involved in this search process, the computational cost can become very high. In order to decrease this burden, two strategies have been proposed. The first resorts to any previous velocity model at hand to restrict the search space and, this way, to reduce the number of possible parameter combinations to be tested. In the second, also known as the pragmatic search, the search is conducted in a sequence of single-parameter searches such that in 2D, the first parameter determined is the velocity to construct the CMP stack, then the emergence angle and finally the curvature. Nevertheless, the joint estimation of the CRS attributes have been claimed to yield better results than one-parameter sequential searches [Barros et al., 2015]. Müller [1998]; Jäger et al. [2001]; Mann et al. [1999] also proposed hybrid schemes where both, velocity and emergence angle, are searched for simultaneously and then the curvature can be obtained with a one-parameter semblance search. Either way, the result of this sequential search procedure can be used as the initial seed for a global optimization that further improves the final result.

Due to unacceptably high computational cost, the pursuit of these attributes cannot be done in a deterministic brute-force way, where all possible parameter-combinations in the discrete search space are tested. Heuristic algorithms like Simulated Annealing [Müller, 1998; Jäger, 1999] and Differential Evolution [Barros et al., 2015] have been applied onto this problem. Barros et al. [2015] designed a hybridized algorithm using the modified Simulated Annealing that is further refined by a Newton local search, in order to keep global and local searches at a good balance. Müller [1998] compared the performance obtained using Simulated Annealing (SA), Modified Simulated Annealing (MSA) and Very Fast Simulated Annealing (VFSA) algorithms, where he concluded that VFSA and MSA are more suitable for determining the CRS attributes and generate high SNR Zero Offset (ZO) sections, as the first two algorithms being more efficient than the last one. Minato et al. [2012] concluded that both Differential Evolution (DE) and SA are effective at locating the global optimum attributes, but SA converges faster. Walda and Gajewski [2015] proposed using GAs to tackle the CRS optimization problem, which resulted in promising results particularly at dealing with more complex structures like salt bodies.

1-2-3 Genetic Algorithms

Over the course of history, humans have struggled to understand and, to varying extents, to predict a wide range of natural phenomena. The human brain, for example, can accomplish a wide variety of activities, many of them effortlessly, such as identifying colors or smells, that thanks to its connection with our senses seem trivial to us. To artificially mimic these activities, pioneer computer scientists, such as Turing [2009], Von Neumann [2012] and others, envisioned the power of combining their understanding about natural systems with computer programs. Since then, these biologically motivated computational activities have aimed at 1) modeling the human brain, 2) mimicking learning and 3) simulating biological evolutions. The first developed into the field of neural networks, the second into machine learning, and the last one into the so called "Evolutionary computation" (EC), from which Genetic Algorithms (GAs) are the outstanding example [Mitchell, 1998].

Evolutionary Computation emerged from the idea that processes and mechanisms of biological

evolutions could be used as a search and optimization tool in a wide range of applications. The goal was to evolve an initial population of candidate solutions by using operators inspired by the neo-Darwinian theory of evolution: natural selection. In 1975, this motivated Holland [1975] to publish his work, where, inspired by the mechanisms of natural adaptation, presented GAs as an abstraction of biological evolutions. Based on this concept, he proposed the machinery for evolving from one population of "chromosomes" to a fitter population by means of a "natural selection" dictated by selection, crossover and mutation. At the first stage, the chromosomes allowed for offspring generation are selected based on their reproduction probability such that, on average, the fitter chromosomes are more likely to produce more offsprings than the less fit ones. Subsequently, in crossover, the strings of parent chromosomes are exchanged, roughly mimicking the biological recombination between parents, whereas in mutation, the values of some alleles of the chromosome are randomly changed, to promote diversity within the population that otherwise would be quickly exhausted by the former operators [Gallagher et al., 1991].

EC is currently a sub-discipline of Computational Intelligence (CI), which in turn also covers other subdisciplines focused on adaptive and intelligence systems such as Swarm Intelligence, Fuzzy Systems and Artificial Neural Networks. Other examples of CI include Particle Swarm Optimization (PSO) [Kennedy and Eberhart, 1995; Brownlee, 2011], inspired by bird flocking, Ant Colony Optimization (ACO) [Maniezzo, 1992; Mitchell, 1998], probabilistic algorithms inspired by the foraging behavior of ants, and Simulated Annealing (SA) [Kirkpatrick et al., 1983], among others.

1-3 Thesis Outline

The content of this thesis is outlined below.

Chapter 2: *An advanced Genetic Algorithm (aGA)*
In this chapter, a novel Genetic Algorithm, aGA, suitable for non-convex optimization problems is introduced. Additionally, the reasons for the selection of each operator, as well as its advantages and limitations are further discussed in this chapter.

Chapter 3: *An enhanced Genetic Algorithm (eGA)*
This chapter begins with the needs for improvement of the already available aGA. Next the ideas implemented are explained further in detail as well as their triggering mechanisms to wisely achieve a good balance between the local search and the global search.

Chapter 4: *Application onto multi-modal test functions*
In this chapter the eGA will be used to optimize three challenging benchmark functions commonly used to test the performance of non-convex optimization algorithms and compare its performance against that of the aGA and the well known PSO.

Chapter 5: *Application onto a synthetic 2D receiver-side static correction demo*
The proposed eGA will be applied to correct for receiver-side statics on ideally normal moveout corrected CMP gathers.

Chapter 6: *Application onto a synthetic 2D CRS demo*
The proposed eGA will be used as the optimization tool for CRS stacking of two 2D synthetic data sets designed to accommodate both low SNR data and one velocity anomaly.

Chapter 7: *Discussion and Conclusions*
In this section, we first make a further deep and insightful discussion on the content of this thesis and then summarize the main conclusions.

Chapter 2

An advanced Genetic Algorithm (aGA)

Optimization can be defined as the process of finding the combination of input parameters, or decision variables, that result in the optimum (minimum or maximum) output, dictated by a fitness or cost function, under a set of constraints. GAs are heuristic optimization methods that belong to the category of global optimization used for this purpose. Since their birth in 1975, GAs have been successfully used as a tool for non-convex global optimization in a wide range of realistic applications. Since then, further improvements in this field have focused on avoiding limitations of the global methods and improving its convergence speed while keeping the original idea of mimicking natural evolution to ensure the survival of the fittest in mind.

In Geophysics many problems are highly non-linear by nature and GAs have been proven to be a very suitable tool for efficiently addressing these challenges. However, as any other global optimization method, GAs also face the practical challenge of high computational costs. This section describes a novel GA developed by AOC GRC Delft, suitable for global optimization problems. The algorithm is called *"an advanced Genetic Algorithm"* (aGA) [Sun et al., 2017] and seeks the global maximum of n-dimension non-convex functions. The remainder of this chapter is dedicated to provide a clear explanation of its working principle, components and how altogether they successfully lead to the pursuit of the fittest.

Last but not least, it is known that the two main disadvantages of GAs are premature convergence and the difficulty at fine-tuning a located optimum. For this reason, the last two sections of this chapter are dedicated to explain in detail the solutions used in the aGA.

2-1 Evolution scheme

2-1-1 Initialization

The aGA starts with an initial population of N chromosomes, or row vectors of dimension $[1, n]$, selected at random, with n equal to the number of dimensions to be optimized.

2-1-2 Elitism

In aGA elitism is explicitly performed by passing the best N_c members of the current generation to the next generation. Furthermore, before mutation is applied, the best N_m members among these N_c members ($N_m \leq N_c$) are also protected from mutation.

2-1-3 Parent selection

At the current iteration t, the first step towards offspring generation is called parent selection. From a population of N chromosomes, each parent is selected by randomly picking two individuals from the population and running a tournament, based upon their fitness comparison, to determine which one gets selected. The winner of the tournament (the one with the higher fitness ϕ) is selected for crossover and this process is repeated for both parents ($X^{1,t}$ and $X^{2,t}$).

2-1-4 Crossover Operator

An offspring is generated via crossover of two parents $X^{1,t}$ and $X^{2,t}$ and a multipoint Blend Crossover (BLX-α) combination [Eshelman and Schaffer, 1993] is used as the crossover operator. In this operator, the k^{th} gene of the i^{th} offspring, for $i = 1, ..., N$, selected for crossover is a value randomly selected from the interval $[C_{min} - I\alpha, C_{max} + I\alpha]$, as follows:

$$X_{k,cross}^{i,t+1} = \begin{cases} L_k - \alpha I_k \rho * (2\alpha I_k + U_k - L_k) & \text{if } R < P_c \\ X_k^{i,t} & \text{if } R \geq P_c \end{cases}, \tag{2-1}$$

where R and ρ are uniform random numbers generated on the fly, P_c is the crossover probability, L_k and U_k define the lower and upper boundaries along the dimension k, $I = C_k^{max} - C_k^{min}$ with $C_k^{min} = min(X_k^{1,t}, X_k^{2,t})$ and $C_k^{max} = max(X_k^{2,t}, X_k^{2,t})$, $\alpha = 0.5$, $X_k^{1,t}$ and $X_k^{2,t}$ are the k^{th} genes of the first and second parents at the current iteration t, respectively. Sun and Verschuur [2014] claims this operator was selected for having reported the best results when dealing with highly non-linear functions.

2-1-5 Mutation Operator

Before mutation, once again, elitism takes place and the best N_m chromosomes, of those N_c chromosomes passed from the previous generation, are protected from mutation, while the $N - N_m$ remaining offspring chromosomes are subject to it. The genes of the chromosome selected for mutation are modified as follows:

$$X_{k,mut}^{i,t+1} = \begin{cases} X_{k,cross}^{i,t+1} + \Delta(t,k) & \text{if } R_1 < P_m \\ X_{k,cross}^{i,t+1} & \text{if } R_1 \geq P_m \end{cases}, \tag{2-2}$$

$$\Delta(k) = \begin{cases} \alpha r(U_k - L_k) & \text{if } R_2 \geq 0.5 \\ \alpha r(L_k - U_k) & \text{if } R_2 < 0.5 \end{cases}, \tag{2-3}$$

where R_1, R_2 and r are random numbers generated on the fly, $X_{k,cross}^{i,t+1}$ is the k^{th} gene of the i^{th} offspring obtained in the previous step, P_m is the probability of mutation, and α is a factor controlling the degree of exploitation.

2-1-6 Periodic Boundary Condition (PBC)

In aGA, every gene of each chromosome must always be within a valid range, bounded by the search space defined by $[\mathbf{L}, \mathbf{U}]$. However, as noted in the previous two steps, mutation and crossover are operators that might cause some genes to be out of the valid range or, in other words, out of the search space. As a consequence, new chromosomes may leave the boundary of the search space. Therefore, every time the mutation or/and crossover operators are applied, a continuous boundary condition is used to map those genes back to the valid range. The mathematical expression of this boundary correction is given in Equation 2-4.

$$X_{k,PBC}^{i,t+1} = \begin{cases} L_k + (X_k^{i,t+1} - U_k) & \text{if } X_k^{i,t+1} > U_k \\ U_k - (L_k - X_k^{i,t+1}) & \text{if } X_k^{i,t+1} < L_k \\ X_k^{i,t+1} & \text{if } L_k \leq X_k^{i,t+1} \leq U_k \end{cases}, \tag{2-4}$$

where $X_k^{i,t+1}$ is the k^{th} gene of the i^{th} offspring, obtained after either crossover $X_{k,cross}^{i,t+1}$ or mutation $X_{k,mut}^{i,t+1}$.

2-2 Beating the Premature Condition

One disadvantage of the GAs occurs when, at an early stage, a relatively good (but not the globally best) solution starts dominating the evolution and the whole population is dragged towards this local zone. As a result, the population diversity starts exhausting quickly and the global search capability weakens, eventually causing the search getting trapped in a local minimum. This effect is also known as the premature convergence situation and it is a major problem because, if not detected in time to prevent the population diversity from devastation, the GAs will waste time exploring local areas very likely without the possibility to jump out of it. One potential solution is increasing the size of the population such that the probability of finding better solutions, from the very beginning, is higher. A second potential solution is to increase the mutation probability to promote a certain amount of diversity in the population. However, increasing the population size has the negative impact of requiring more computational time whereas increasing the mutation probability results in an increase of the randomness, turning it into a Monte Carlo search.

The mechanism aGA proposes to skip the premature convergence first makes sure that the current best fitness ϕ_{Best} has increased at least $R_p\%$, compared to the best fitness $\phi_{best'}$ checked t_{es} generations ago, such that $\phi_{Best} \geqslant \phi_{best'} * (1 + R_p)$. If so, a good balance between the global and the local searches is being achieved (so far) and the evolution continues. Otherwise, an exhaustive search is used to help the algorithm jump out of this premature situation.

Exhaustive search is the building block designed to beat the premature situation. This exhaustive search scans possible parameter values along a certain dimension of the solution. To

do so, the gene selected for scanning is increased by a fixed scan step Δ_{es}, each time for a total number of n_{steps} times. Both parameters, n_{steps} and Δ_{es}, are carefully selected to cover the entire search space such that $n_{steps} = (U - L)/\Delta_{es}$. Unfortunately, following this procedure with every single gene of the chromosome would increase the number of fitness function evaluations, prohibiting its application onto all genes at once. Then, this exhaustive search is applied only on certain genes selected at random with a probability given by $\mu = x/n$, where x is a user pre-defined input indicating the total number of dimensions to be exhaustively analyzed every time the search is in a premature situation, and n is the problem dimension. In other words, every time the exhaustive search is triggered, the current number of function evaluations η is increased by $n_{steps} * n * \mu$ on average.

However, the cost of this brute force search is very high, and hence an improvement of this aGA should be aimed at promoting diversity and exploring the search space in a smarter way.

2-3 Stopping criteria

The aGA stops when either the best solution found converges or the algorithm reaches a maximum number of iterations t_{max}. For those experiments where the true solution is known and a convergence condition is accordingly defined, the former applies, otherwise only the latter condition is valid.

2-4 Workflow of the aGA

The step-by-step of the enhanced Genetic Algorithm, illustrated in Figure 2-1, is summarized in the following, and the parameters there used are defined in subsection 2-4-1.

Step 1: Initialize the population with N random candidates of size $[1, n]$, the loop iteration counter $t = 0$ and the number of fitness evaluations $\eta = 0$.

Step 2: Evaluate the fitness function ϕ_i of each i^{th} new member X^i and sort them in descending order of fitness ϕ.

Step 3: Increase the number of function evaluations η by N and assign the fitness of the first chromosome (as sorted in **Step 2**) to $\phi_{best'}$.

Step 4: Every t_{es} generations check premature convergence. If the current best fitness ϕ_{Best} is at least $(1 + R_p)$ times bigger than the best fitness achieved t_{es} generations ago $\phi_{best'}$, go directly to **Step 5**. If not, do an exhaustive search to replace weak chromosomes with new ones, update the number of function evaluations η and then go to **Step 5**.

Step 5: Create a new population of chromosomes, as explained in section 2-1, and continue.

Step 6: Evaluate the fitness function ϕ_i of each i^{th} new offspring $X^{i,t+1}$ and sort them in descending order of fitness ϕ. Next, increase the counter of function evaluations η by N and the iteration counter t by 1.

Step 7: Test whether one of the stopping criteria is met (see section 2-3). If so, the output is the current best solution ϕ_{best} and the algorithm is over, otherwise go to **Step 4**.

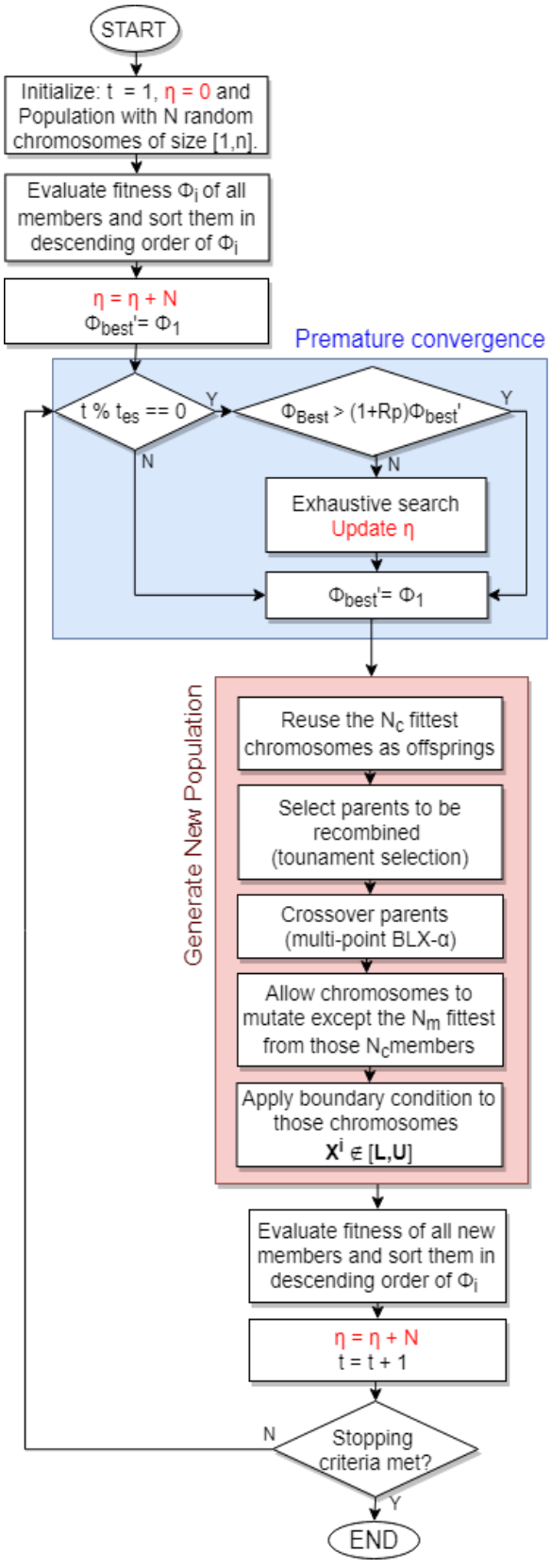


Figure 2-1: Flowchart of the aGA [Sun et al., 2017].

2-4-1 Definition of parameters

For completeness, the parameters used in this chapter are restated as follows:

[L,U]	Upper and lower boundary arrays delimiting the search space.
μ	Ratio of number of genes allowed to be exhaustively searched over n.
$\mathbf{X^{i,t}}$	i^{th} chromosome at current iteration t^{th}.
$\mathbf{X^{i,t+1}}$	i^{th} offspring.
η	Number of fitness function evaluations.
n	Number of independent parameters (dimensions) to be optimized.
N	Population size.
N_c	Number of members inherited from the previous population.
N_m	Number of members in those inherited members that are exempted from mutation.
P_c	Crossover probability.
P_m	Mutation probability.
$\phi_{best'}$	Best fitness when premature convergence was last checked.
ϕ_{Best}	Best fitness so far.
ϕ_i	Fitness of i^{th} chromosome.
Δ_{es}	Scanning step for the exhaustive search.
R_p	Improvement factor defining the premature condition gain.
n_{steps}	Number of steps exhaustively searched along a certain direction.
t	Number of loop iterations so far.
t_{max}	Maximum number of loop iterations allowed.
t_{es}	Number of iterations after premature convergence was last checked.
μ	x over n ratio.
x	Number of dimensions to be analyzed every the exhaustive search is applied.

Chapter 3

An enhanced Genetic Algorithm (eGA)

The key to success of GAs strongly depends on the operators used [Chinnasri et al., 2012; Selvi and Rajaram, 2007], the evolutionary machinery designed [Gomes and Selman, 2001; Gong and Fukunaga, 2011; Whitley et al., 1999], the parameters set [Harik and Lobo, 1999; Biazzini et al., 2009; Wolpert and Macready, 1997; Valenzano et al., 2010] and, more importantly, a good trade-off between its global and local searches. Achieving the last implies coping with two complementary searches able to fill in the gaps of each other. The degree of success of GAs is usually measured in terms of the final fitness value and the total amount of function evaluations.

The aGA has proven to yield outstanding results given enough time and fine-tuned parameters [Sun and Verschuur, 2014; Sun et al., 2016, 2017]. However, convergence speed is a key factor for the success of a global optimization algorithm when it comes to scalability; in production even slight efficiency improvements matter as it can easily take weeks or even months to process a huge data set. Therefore, the goal of this project is to improve the convergence speed of the currently available aGA by simultaneously enhancing both its global and its local search capabilities.

In order to improve both capabilities, two solutions were proposed. The first was a modified version of the well known *Island model GAs* and the second was named *Self Adaptive Differential Evolution (SADE) fine tuning* scheme. The former consists of multiple sub-populations (or islands) running simultaneously with different parameter sets selected at random. This idea was proposed by Gong and Fukunaga [2011], where it was claimed that it is not very difficult to quickly achieve *average performed* results if sufficient parameters are tested on unique islands. For the second idea, the effectiveness of the Differential Evolution (DE) mutation operator [Barros et al., 2015] at correcting outlier genes inspired us to design a novel SADE fine tuning search, capable of navigating around the best solution found, in directions different to the orthogonal. At the end of every subsection I will explain the potential pitfalls each solution may face and how a wise triggering of its functionality can help to prevent these problems.

3-1 Features of the eGA

3-1-1 Randomized Islands Model

This idea consists of defining multiple sub-populations (or islands) with different parameter sets and sequentially evolve them such that the update of each island is interleaved with an unidirectional communication system. The logic behind this communication is to transfer the best member from one island to the next if, and only if, its best current solution is fitter than the worst offspring last generated by the next island, as illustrated in Figure 3-1. Additionally, in order not to devastate the achieved extra-diversity, the frequency of communication among islands is controlled such that each island is given some time to evolve and find its currently best solution before it is forced to follow a better trajectory or eventually forces its neighbor island to do it.

As stated in the *No Free Lunch* theorem (NFL), on average, the outstanding performance of any algorithm over a certain class of problems is offset by its poor performance in other problems [Wolpert and Macready, 1997]. In a similar way, finding the parameters that result in the best performance of an algorithm can not guarantee these parameters will also perform well in all other applications. Therefore, a prior problem-specific tuning of parameters could dramatically improve its convergence speed. Unfortunately, this offline parameter tuning is a very expensive process and in practice, where time is a precious asset, users end up finding the *best averaged* parameters only once and expect them to be effective in any other problem and thus skip this extra-step in the future. Conversely, in this work, we completely avoid any on/offline parameter tuning by selecting a different parameter set (P_{mj}, P_{cj}, N_j) for each island at random from a valid range where extreme cases are neglected, i.e. $Pc_j \in [0.3, 0.8]$ and $Pm_j \in [1/n, 3/n]$. The upper bounds of the population size range depend on the number of island selected but, based on our tests, we concluded that the overall population should be smaller than ten times the number of parameters to be optimized.

The improvements achieved using this idea can be attributed to the enlargement of the sampling space and the degree of independence each island is given to freely explore the search space and follow potentially better trajectories. This freedom introduces a certain amount of diversity to the overall system by allowing the evolutionary behavior of each island to be different from each other, so that each sub-population explores the space in a different manner and at a different pace. As discussed in section 2-2, another strategy to promote diversity is to increase the population size. However, increasing the population size not only results in an enlargement of the sampling space but also in deceleration of the cross-fertilization of information among members. It is there where the most significant advantage of the island model relies on: even though by using multiple islands the sampling space is implicitly enlarged, the crossover is done between members of the same sub-population only. On top of that, the fact that after every successful communication, each sub-population is drastically imposed a "role model" to follow, the speed of information sharing between chromosomes is even faster.

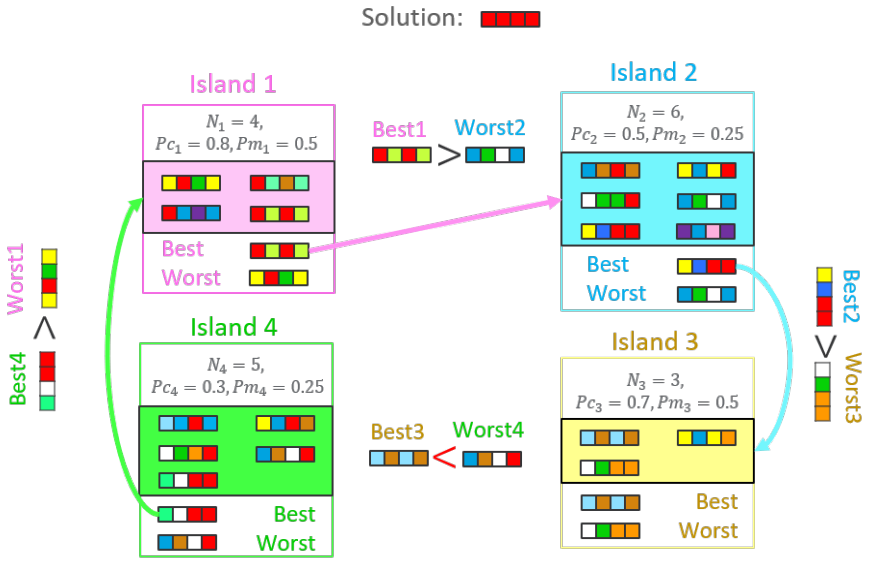


Figure 3-1: Smart communication system among islands for $n_i = 4$, $Pc_j \in [0.3, 0.8]$ and $Pm_j \in [1/n, 2/n]$. Observe there is no communication from island 3 to 4.

3-1-2 Self Adaptive Differential Evolution (SADE) fine tuning search

The DE mutation is a scheme first introduced by Storn and Price [1997], where the i^{th} chromosome at the current t iteration $X^{i,t}$ is mutated by adding the weighted difference between two members of the population selected at random, as expressed in Equation 3-1 and illustrated in Figure 3-2(a):

$$X_{DE}^{i,t} = X^{i,t} + \alpha(X^{r_2,t} - X^{r_1,t}), \tag{3-1}$$

where $X_{DE}^{i,t}$ is the DE-mutated chromosome, r_1 and r_2 are the indices of the parents selected at random (different from index i), and α is known as the DE scaling factor.

In general, GAs are known for their limited capacity at fine tuning their best chromosomes to reach the exact solution, and the aGA was not an exception. In our tests, we have also confirmed that the aGA was able to quickly achieve solutions close to the ground truth. The effectiveness of DE-mutation inspired us to use it as basis to redesign a local search capable of quickly fine tuning good chromosomes. Our SADE fine tuning scheme seeks to improve the best chromosome of all populations $X^{c,t+1}$, by iteratively allowing it to mutate, as follows:

$$X_{SADE}^{Best,t} = X^{Best,t} + \alpha(m)(X^{r_2,t} - X^{r_1,t}), \tag{3-2}$$

where $X^{Best,t}$ is the fine-tuned version of the current best chromosome and $\alpha(m)$ will be explained in the following paragraph.

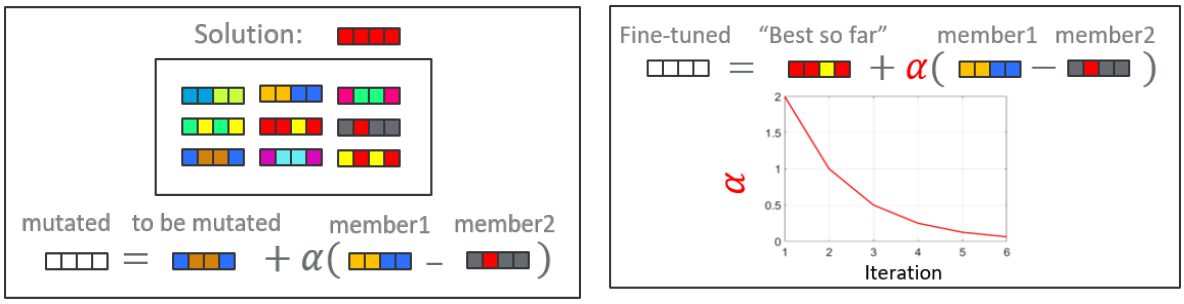
The DE-mutation scaling factor α is a real value controlling the step length given towards $X_j^{r_2,t} - X_j^{r_2,t}$ and, based on our tests, the effectiveness of the DE mutation operator is very sensitive to its selection. Higher values of α increases the exploration freedom of the algorithm whereas smaller ones are rather used to exploit the information contained in the current

population. Therefore, to wisely avoid manual tuning, in eGA α is automatically adjusted every iteration until a maximum number of n_{max} attempts is reached for that trial. In short, a line-search of α is performed, as expressed in Equation 3-3,

$$\alpha(m) = (\alpha_0)^{-m}, \tag{3-3}$$

where $\alpha(m)$ is the SADE scaling factor at the m^{th} attempt, α_0 its initial value and m the attempt counter (for $m = 1, ..., n_{max}$), as illustrated in Figure 3-2(b).

The SADE fine tuning process starts removing the 25% of the furthest members (cyan dots) to the best chromosome (red dot) of a *mature enough* population, as shown in Figure 3-3(a). Then, for every iteration, e.g. Figure 3-3(b), a couple (green dots) is randomly selected from this sub-population and α is automatically decreased for a maximum of n_α attempts or until the tuned chromosome is fitter than the initial best chromosome. Then, in the following iterations (Figure 3-3(c) to 3-3(e)), another couple is selected as parents and the process is repeated until the convergence condition is met (see Figure 3-3(f)) or the maximum number of repetitions n_{max} is reached. It is important to clarify that every attempt to self-adaptively improve α also includes $-\alpha$ so that both directions are equally scanned. In our example, α was decreased less than 8 times (observe the 8 gray asterisks to the left and 8 yellow crosses to the right of the red filled circle in Figure 3-3) until a fitter solution was obtained and, finally, the process met the convergence condition after 4 iterations. As evidenced, this mechanism was capable of correcting deviations in just a few iterations if the population is at an advanced stage of the evolution.

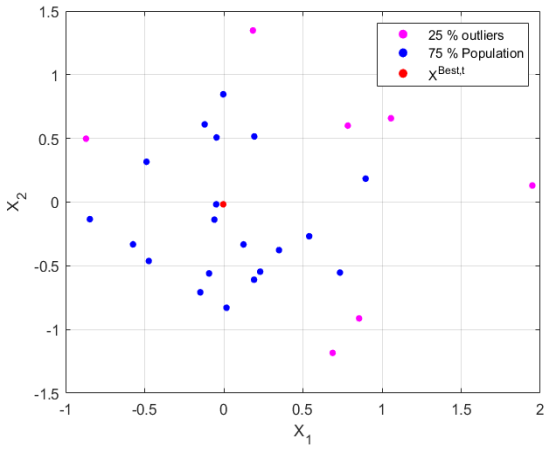


(a) DE mutation scheme with fixed α

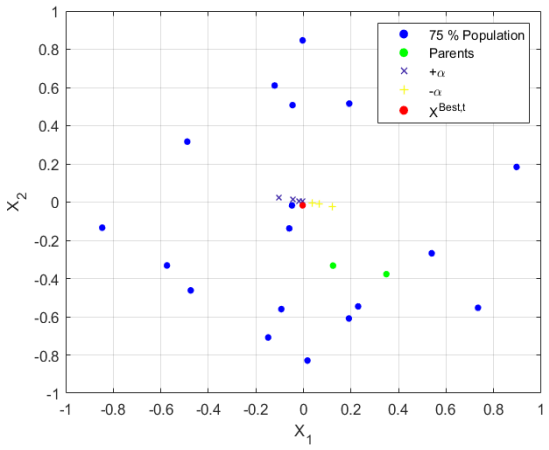
(b) SADE fine tuning with $\alpha_0 = 2$ and $n_\alpha = 6$

Figure 3-2: Comparison between a) DE mutation and b) SADE fine tuning.

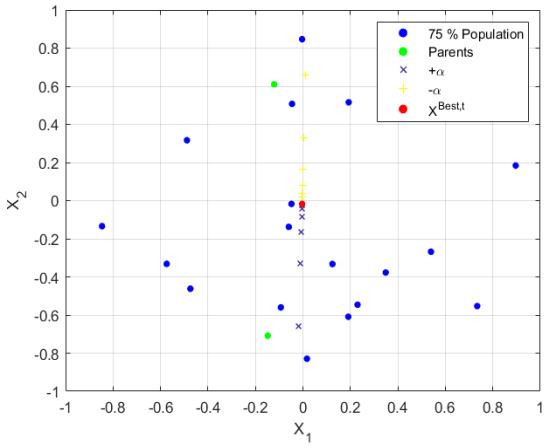
As a final remark, it is worth mentioning that this extra component added for fine tuning might become very expensive if applied at a premature stage and it would end up causing the opposite effect. Besides, as the true solution is usually unknown in practice, defining the limit when the population has become mature enough to be fine tuned is not an easy task, specially when the only measure at hand is fitness and its rate of growth. Keeping in mind that this mechanism was designed to correct those genes out of the GA acuity and stop it from wandering, a wise triggering criterion to bring this component into functionality is checking whether after a fixed number of iterations f_{DE}, the growth rate of the fitness has been zero more than twice. If so, the fine tuning starts and is expected to meet the convergence condition; otherwise, the population is not ready for tuning and the eGA continues.



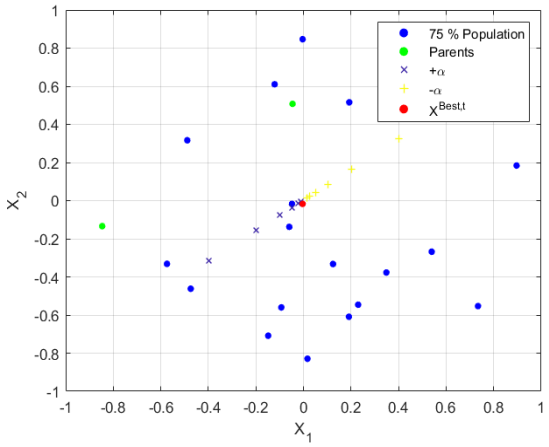
(a) i=0. Detect and remove the 25% furthest members (cyan dots) to the best member $X^{Best,t}$ (red dot).



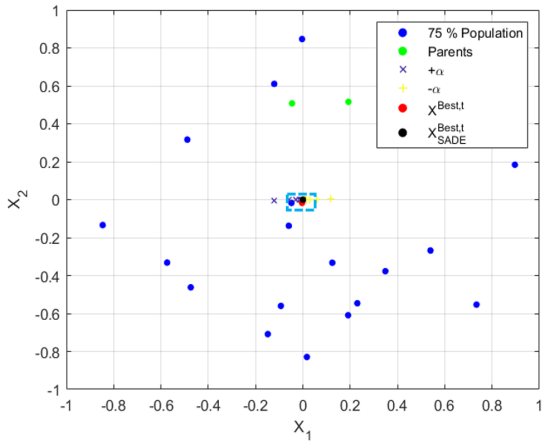
(b) i=1



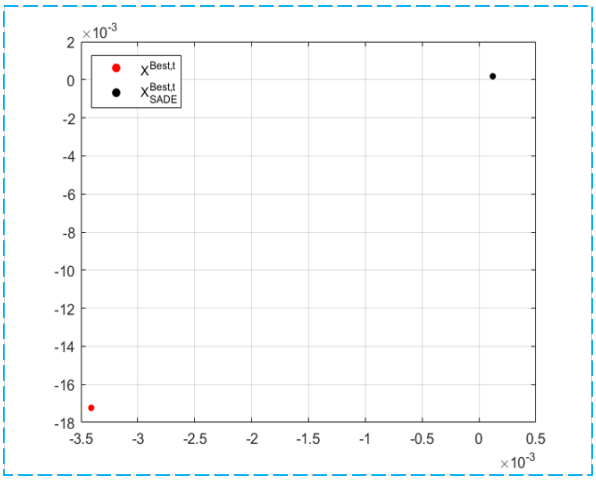
(c) i=2



(d) i=3



(e) i=4



(f) Zoomed version of i=4

Figure 3-3: Step by step illustration of the SADE fine tuning mechanism applied onto a 2D Ackley function (which will be further discussed in chapter 4) with minimum located at $[x_1, x_2] = [0, 0]$. a) Preliminary step aimed at removing 25% furthest outliers. For every iteration from b) to e), each couple is automatically decreased by 2 up to $n_\alpha = 8$ times or until a fitter solution was found. h) Zoomed version of the final result shown in Figure 3-3(d).

3-2 Workflow of the eGA

After adding the above mentioned ideas to the original aGA, we end up with a more complex but robust algorithm aimed at improving both local and global search capabilities, when possible. The step-by-step of the enhanced Genetic Algorithm, illustrated in Figure 3-4, is summarized in the following, and the parameters there used are defined in subsection 3-2-1.

Step 1: Initialize the generations counter $t = 0$, number of fitness evaluations $\eta = 0$ and island's number iterator j.

Step 2: Randomly select the GA parameters of each island from the user-defined ranges.

Step 3: Initialize every j^{th} population ($j = 1, ..., n_i$) with N_j random row vectors of size n.

Step 4: Evaluate the fitness function ϕ_i of all members within each island and sort them in descending order of fitness ϕ.

Step 5: Increase the number of function evaluations η by $\sum_{j=1}^{n_i}(N_j - N_m)$ and assign the fitness of the overall best chromosome to $\phi_{best'}$.

Step 6: Check whether the island's number iterator j is larger than the number of islands defined n_i. If so, go directly to **Step 7**, otherwise jump to **Step 11**.

Step 7: Check whether the current iteration is a multiple of t_{SADE}, if so determine whether the growth of rate has been stucked in the same value as last time it was checked to apply the SADE fine tuning scheme and go to **Step 8**. If not, go directly to **Step 9**

Step 8: Increase p by one.

Step 9: Restart the island's number iterator to $j = 1$, increase the number of iterations counter by one and go to **Step 10**.

Step 10: Test whether one of the stopping criteria (section 2-3) is met. If so, the output is the best current solution and the algorithm is over, otherwise go to **Step 6**.

Step 11: Every t_{com} generations transfer chromosomes between islands, if it is possible, as explained in subsection 3-1-1 and then go to **Step 12**.

Step 12: Every t_{es} generations check premature convergence. If the current best fitness of the j^{th} island $\phi_{Best,j}$ is at least $(1 + R_p)$ times bigger than the fitness of the overall best chromosome $\phi_{best'}$, achieved t_{es} generations ago, go directly to **Step 13**. If not, do an exhaustive search to replace weak chromosomes with new ones and then go to **Step 13**.

Step 13: Replace the fitness of the overall best chromosome $\phi_{best'}$, updated t_{es} generations ago, with $\phi_{Best,j}$, only if $\phi_{Best,j} > \phi_{best'}$.

Step 14: Create a new population of chromosomes for the j^{th} island, as explained in section 2-1, and then go to **Step 15**

Step 15: Evaluate the fitness function ϕ_i of every i^{th} ($i = 1, ..., N_j$) member belonging to the j^{th} island and subsequently sort them in descending order of fitness ϕ.

Step 16: Increase the number of function evaluations η by N_j, the island's number iterator by 1, store the best fitness of the j^{th} island $\phi_{Best,j}$ in $R_{Best,j}$ and then go to **Step 10**.

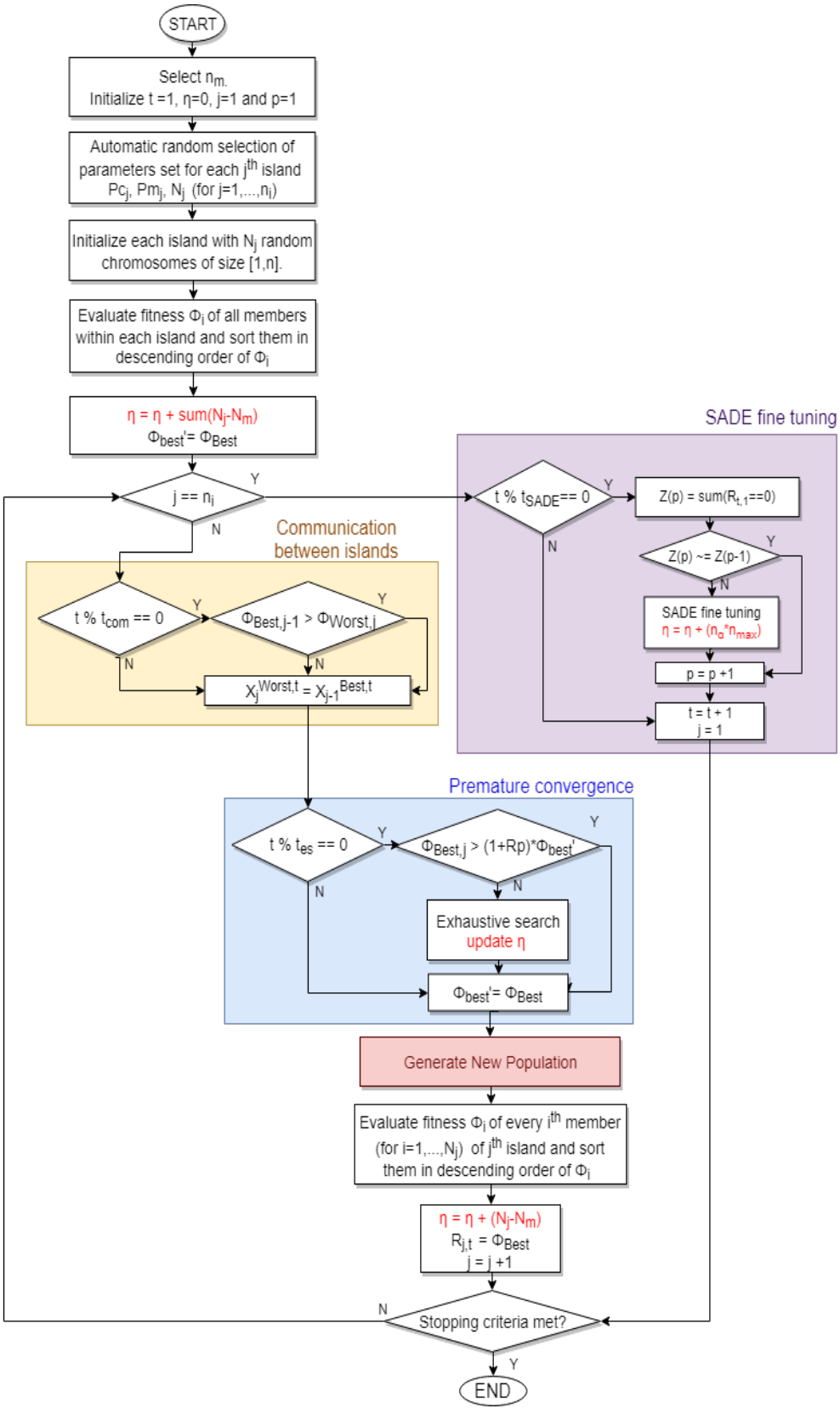


Figure 3-4: Flowchart of enhanced Genetic Algorithm.

3-2-1 Definition of parameters

For completeness, the new parameters introduced in this chapter are restated in the following.

α	DE scaling factor.
$\alpha(m)$	SADE scaling factor.
α_0	Initial DE scaling factor.
$\phi_{i,j}$	Fitness of i^{th} chromosome of j^{th} island.
$\phi_{best'}$	Fitness of the overall best chromosome t_{es} generations ago.
ϕ_{Best}	Fitness of the current overall best chromosome.
$\phi_{Best,j}$	Fitness of the best chromosome of j^{th} island.
$\phi_{Best,j-1}$	Fitness of the best chromosome of $(j-1)^{th}$ island.
$\phi_{Worst,j}$	Fitness of the worst chromosome of $(j)^{th}$ island.
n_α	Maximum number of times α is allowed to decrease for.
n_{max}	Maximum number of attempts to fine-tune the fittest chromosome.
n_i	Number of islands
N_j	Population size of j^{th} island.
Pc_j	Crossover probability of j^{th} island.
Pm_j	Mutation probability of j^{th} island.
p	Counter increasing every time the growth rate is stucked after t_{SADE} iterations.
$R_{j,t}$	Record containing the best fitness of every island in each iteration.
R_p	Improvement factor defining the premature condition gain.
t	Number of iterations so far.
t_{com}	Frequency of communication among islands.
t_{SADE}	Number of iterations after SADE fine tuning is checked.
t_{es}	Number of iterations after premature convergence was last checked.
t_{max}	Maximum number of iterations.
$\mathbf{X_j^{i,t}}$	i^{th} chromosome of j^{th} island at t^{th} iteration.
$\mathbf{X^{Best,t}}$	The best of all chromosomes at t^{th} iteration.
$\mathbf{X_{SADE}^{Best,t}}$	SADE fine-tuned version of $\mathbf{X^{best,t}}$.
$\mathbf{X_{j-1}^{Best,t}}$	The best chromosome of $(j-1)^{th}$ island at t^{th} iteration.
$\mathbf{X_j^{Worst,t}}$	The worst chromosome of j^{th} island at t^{th} iteration.
Z	Record containing the fitness growth rate measured every t_{SADE} iterations.

Chapter 4

Application onto multi-modal test functions

In order to test the performance of the proposed eGA, it will be used to optimize three benchmark functions for increasing number of dimensions and the results were compared against the aGA proposed by Sun et al. [2017] and the PSO by Brownlee [2011]. These functions are test beds for minimization problems widely used for testing the performance of non-convex optimization algorithms given that they are non-linear multi-modal functions and finding the global minimum is challenging due to the search space containing a high number of local minima. Our eGA wills search for the global solution by maximizing the negated functions. In this chapter, as the global maxima ϕ_{true} of the three test functions are known, the convergence condition for ϕ_{BEST} is defined as $\mid \phi_{BEST} - \phi_{true} \mid \leq \Delta\phi_{min}$, such that the algorithm finishes when the difference between the current best ϕ_{BEST} and the true fitness ϕ_{true} reach a minimum threshold dictated by $\Delta\phi_{min}$.

4-1 Benchmark Functions

The three functions used to test the performance of the eGA are the well-known Ackley, Rastrigin and Griewangk functions illustrated in Figure 4-1 for $n = 2$, i.e. the two-dimension version of these functions. The mathematical expression for the n-dimension Ackley function, expressed in Equation 4-1, presents a marked decreasing behavior towards the global minimum, located at the origin of the coordinate system.

$$\begin{aligned} A(x_1, ..., x_n) &= \left[20 - 20exp\left(-0.2\sqrt{0.5\sum_{i=1}^{n} x_i}\right)\right] + \left[1 - exp\left(0.5\sum_{i=1}^{n}(cos\left(2\pi x_i\right)\right)\right], \\ Range &: -5.12 \leqslant x_i \leqslant 5.12, \\ Solution &: x_i = 0, \\ \phi_{true} &: A(0, ..., 0) = 0. \end{aligned} \tag{4-1}$$

For the Rastrigin function, the global minimum is also located at the origin but hardly distinguishable from its surrounding locals. The expression for the n-dimension Rastrigin function is given as:

$$\begin{aligned} R(x_1, ..., x_n) &= 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i)), \\ Range &: -5.12 \leqslant x_i \leqslant 5.12, \\ Solution &: x_i = 0, \\ \phi_{true} &: R(0, ..., 0) = 0. \end{aligned} \tag{4-2}$$

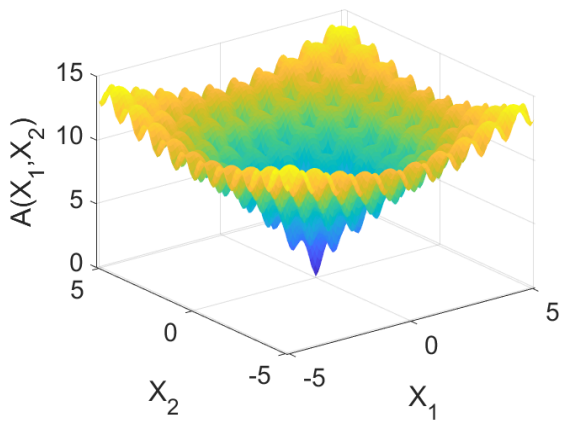
Finally, the Schwefel function is a deceptive function where the global minimum is not located at the origin of the coordinate system and far from the next local minimum. The expression for the n-dimension Schwefel function is given as:

$$\begin{aligned} S(x_1, ..., x_n) &= 418.982887n - \sum_{i=1}^{n}(x_i sin\sqrt{x_i}), \\ Range &: -512 \leqslant x_i \leqslant 512, \\ Solution &: x_i = 420.9687, \\ \phi_{true} &: S(420.9687, ..., 420.9687) = 0. \end{aligned} \tag{4-3}$$

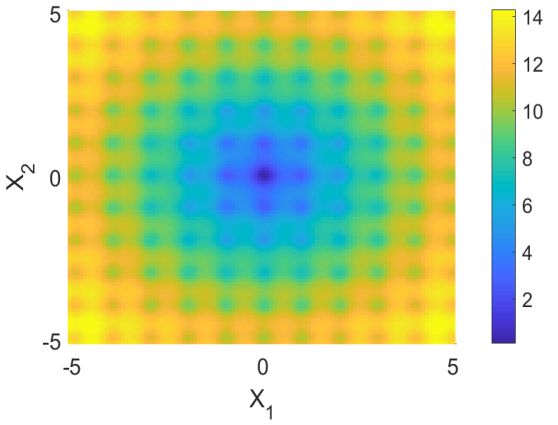
4-2 Particle Swarm Optimization (PSO)

In order to compare the performance of our eGA with other evolutionary algorithms, PSO was also used to maximize the aforementioned functions. PSO was proposed by Kennedy and Eberhart [1995] as a stochastic global optimization algorithm and, since then, it became the inspiration for future Swarm Intelligence algorithms in the sub-field of CI. Similar to GAs, PSO starts with a random population of candidates solutions that are updated iteration after iteration, influenced by their personal best past location and the best past location of the swarm, until the global solution is reached. However, unlike GAs, PSO have no evolutionary operators, such as mutation and crossover, but instead the potential solutions, or particles, move towards the member whose value is closest to the target at any given moment. The most significant advantages of PSO over GAs are its easier implementation as well as the smaller number of parameters to be tuned. Another important difference between both is their information-sharing mechanism: whereas in GAs the genetic material spreads omnidirectionally among all the members of the population, in PSO it is more like a one-way information-sharing mechanism from the best particle to the others.

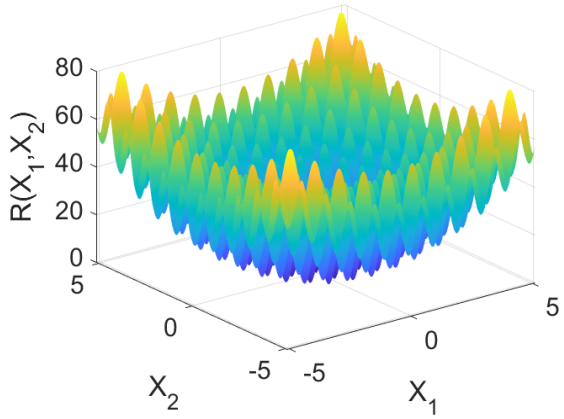
PSO is inspired by the schooling behavior of fish or the flocking behavior of birds working together to find a hidden source of food. They do this by circling around an area and chirping the loudest when any of them gets closer than the previous to the target and this way it attracts the others in its direction, until all others cluster or converge around an optimum, or several optima. To this end, each particle must carry the following information: data representing the plausible solution, the personal best value, indicating the closest the particle has been from the target, and the velocity value regulating how much the data can be changed.



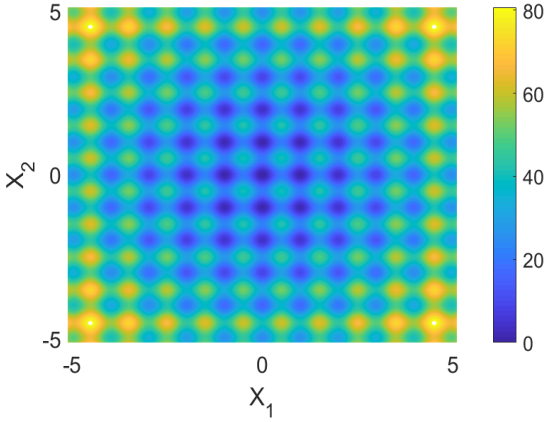
(a) Isometric view of 2D Ackley function



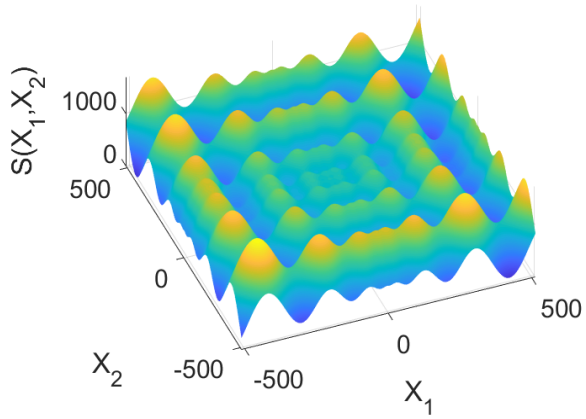
(b) Transversal view of $A(X_1, X_2)$



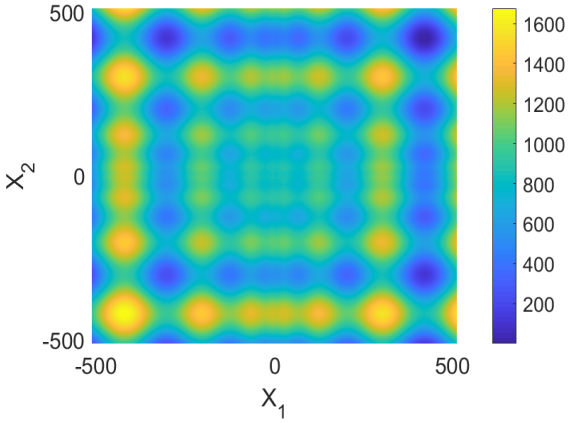
(c) Isometric view of 2D Rastrigin function



(d) Transversal view of $R(X_1, X_2)$



(e) Isometric view of 2D Schwefel function



(f) Transversal view of $S(X_1, X_2)$

Figure 4-1: Three 2D multi-modal test functions used to benchmark the performance of the proposed Genetic Algorithm: Ackley (a,b), Rastrigin (c,d) and Schwefel (e,f). Observe the search space was selected to include as many local minima as possible.

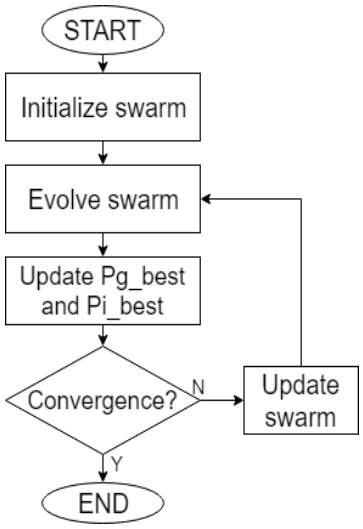


Figure 4-2: Flowchart of Particle Swarm Optimization.

The work-flow of the PSO algorithm is as follows: it starts with a group of random particles that move through the search space following two "best" values: the best solution (P^i_{best}) achieved by the i^{th} particle so far and the best value obtained by any particle in the population (P^g_{best}) [Kennedy and Eberhart, 1995]. Based on these two best values, the particle updates its velocity, as in Equation 4-4:

$$v^i(t+1) = v^i(t) + (c_1 * r * (p^i_{best} - p^i(t))) + (c_2 * r * (p^g_{best} - p^i(t))), \tag{4-4}$$

where r is a number randomly selected from a Gaussian distribution and c_1 and c_2 are the cognitive and social parameters, respectively. The particle position is also updated using:

$$p_i(t+1) = p_i(t) + v_i(t). \tag{4-5}$$

The flowchart of the PSO algorithm is illustrated in Figure 4-2

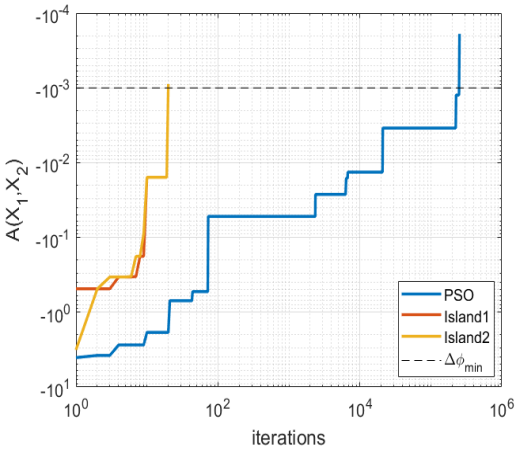
4-3 Experimental Results

4-3-1 Comparison against PSO

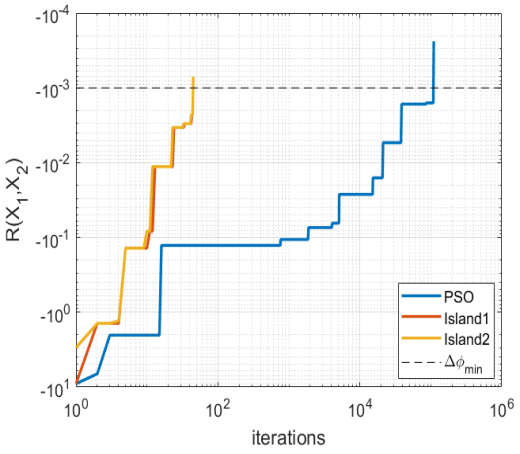
In this section, as a preliminary test, the performance of the eGA will be compared against a PSO implementation [Brownlee, 2011]. For the eGA, the number of islands is set to $N_i = 2$, its communication frequency to $t_{commu} = 1$ and, for the SADE fine tuning component, its frequency is set to $t_{SADE} = 10$, α is decreased for a maximum of $n_\alpha = 6$ attempts (starting from $\alpha_0 = 2$) and every time this component is triggered, the best chromosome is fine-tuned up to $n_{max} = 100$ times. For the PSO, cognitive and social parameters are set to 2 and swarm size equal to 30. Both algorithms were allowed to run for a maximum of $t_{max} = 10^6$ iterations and the convergence threshold was set to $\Delta\phi_{min} = 10^{-3}$.

First, PSO and eGA are applied to optimize the three above mentioned benchmark functions with a dimension equal to 2. As can be seen in Figure 4-3, eGA reaches the global minimum in less than 80 iterations whereas PSO almost takes the maximum number of iterations to converge. So far, it can be concluded that eGA has outperformed PSO. However, to be able to draw a more solid conclusion, the experiment had to be repeated for ten times with different random seeds. The average results obtained for increasing number of dimensions, starting from 2 up to 10, are shown in Figure 4-4. It can be seen in this figure that while eGA always reaches the goal point, for this maximum number of iterations PSO is incapable of converging for dimensions higher than two.

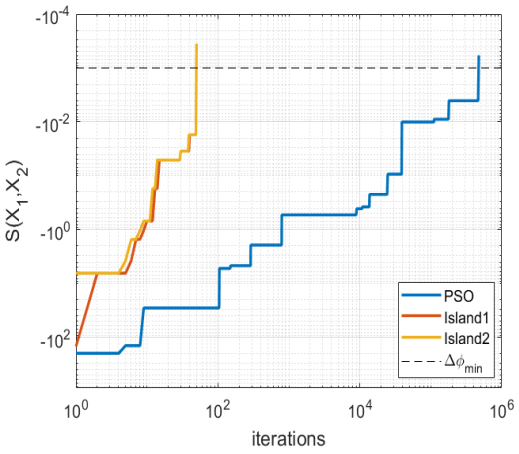
4-3-2 Comparison against aGA



(a) Ackley function



(b) Rastrigin function



(c) Schwefel function

Figure 4-3: Optimization of 2D a) Ackley, b) Rastrigin and c) Schwefel functions with PSO (solid blue curve) and eGA (red and yellow lines).

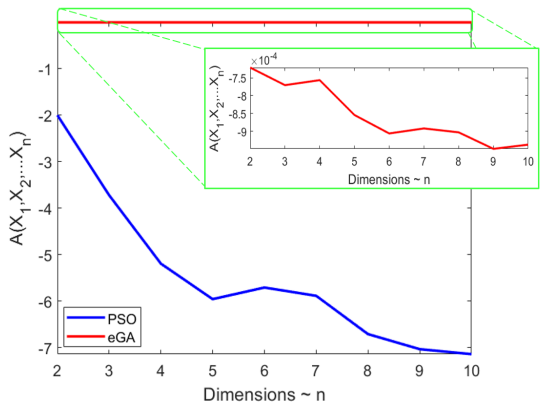
Finally, the performance of the eGA and aGA will be compared. To do so, the population size of the aGA was set to 100 and the crossover and mutation probabilities to $P_c = 0.5$ and $P_m = 2/n$, respectively. For the eGA, a total of $N_i = 2$ islands (communicated after every iteration) were used and the parameters of the SADE fine tuning were set to $t_{SADE} = 10$, $n_{max} = 100$, $n_\alpha = 6$, $\alpha_0 = 2$. Both algorithms were allowed to run for a maximum of $t_{max} = 10^6$ iterations and the convergence threshold was set to $\Delta\phi_{min} = 10^{-3}$.

The results of applying the aGA and eGA to optimize the three multi-modal test functions, for increasing number of dimensions, are shown in Figure 4-5; these are the results of averaging the performance obtained with ten different random seeds for each dimension. The right-hand side panels show, on average, the number of function evaluations required to meet one of the stopping criteria whereas the panels on the left display the average fitness achieved when the algorithm finished. As can be seen, both algorithms successfully met the convergence condition ($abs(\phi_b - \phi_{true}) \leq 10^{-3}$) before exceeding the maximum number of iterations $t_{max} = 10^6$, but the eGA is much faster; in some situations up to 2 orders faster.

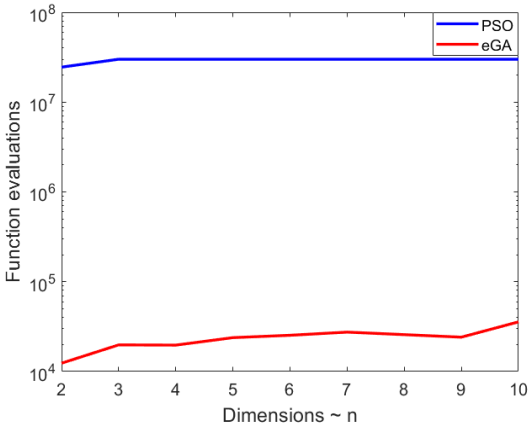
4-4 Summary

The success of the eGA proposed in this thesis was demonstrated in this chapter where, first, it was proven to be far superior than a second evolutionary algorithm known as *Particle Swarm Optimization*, as described in Brownlee [2011].

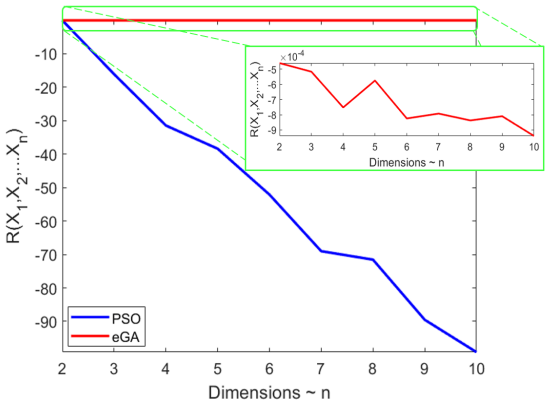
Furthermore, compared with the aGA, the eGA achieved a significant improvement in computation efficiency when used to optimize three multi-modal test functions commonly used to test the performance of non-convex optimization problems.



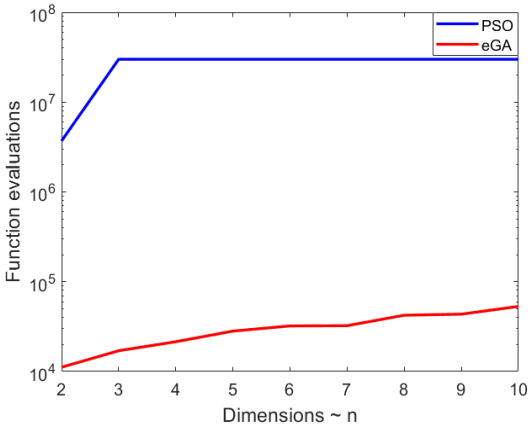
(a) Fitness function for increasing dimensions



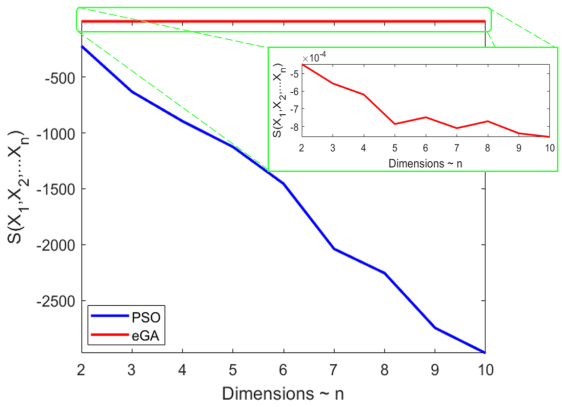
(b) Function Evaluations for increasing dimension



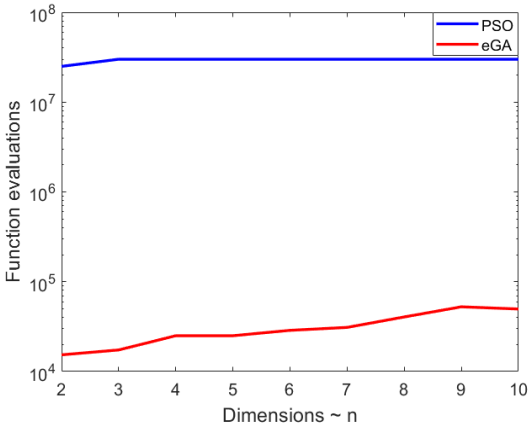
(c) Fitness function for increasing dimensions



(d) Function Evaluations for increasing dimensions

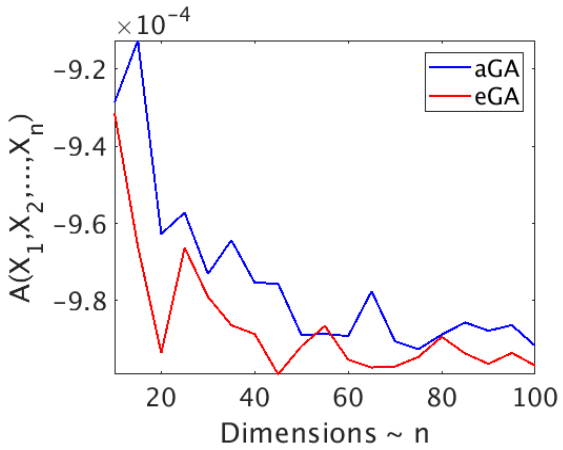


(e) Fitness function for increasing dimensions

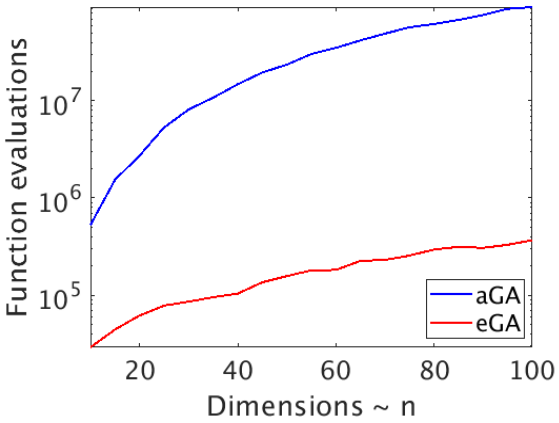


(f) Function Evaluations for increasing dimensions

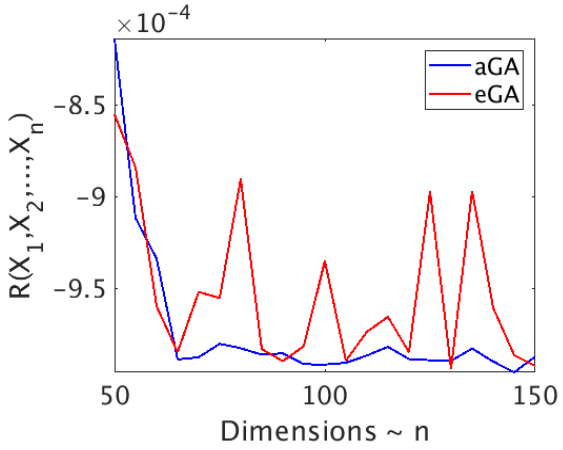
Figure 4-4: Comparative results showing the average (left) fitness and (right) total number of function evaluations of PSO (blue) against the aGA (red), as a function of number of dimensions, for (a,b) Ackley, (c,d) Rastrigin and (e,f) Schwefel.



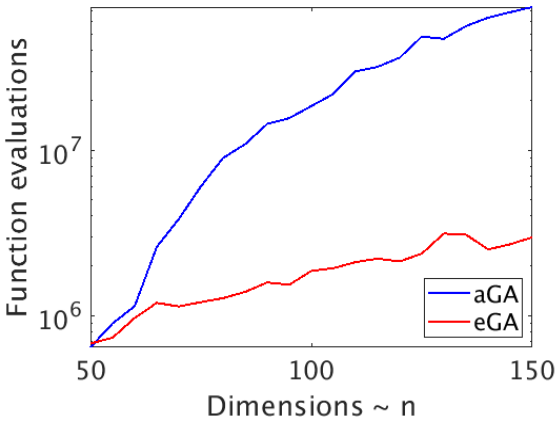
(a) Fitness function for increasing dimensions



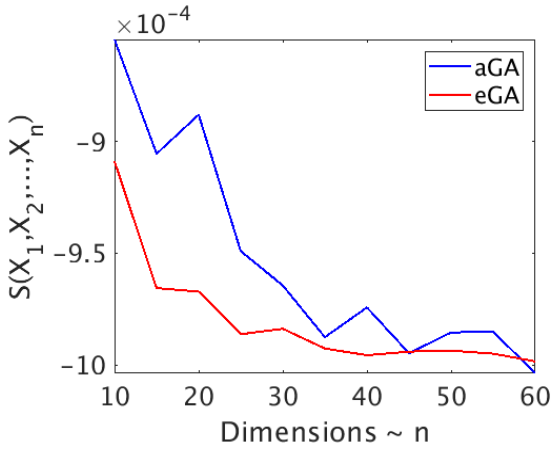
(b) Function Evaluations for increasing dimension



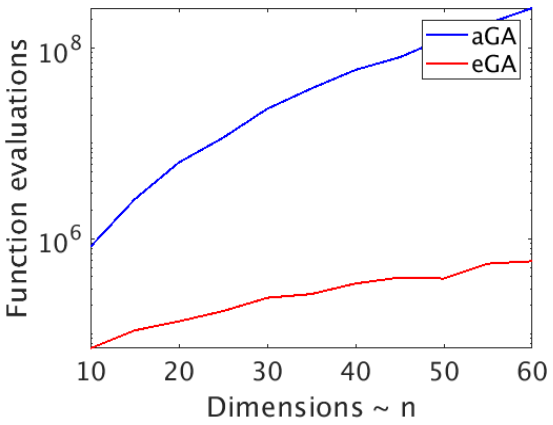
(c) Fitness function for increasing of dimensions



(d) Function Evaluations for increasing dimensions



(e) Fitness function for increasing dimensions



(f) Function Evaluations for increasing dimensions

Figure 4-5: Comparative results showing the average (left) fitness and (right) total number of function evaluations of the aGA (blue) against the eGA (red), as a function of number of dimensions, for (a,b) Ackley, (c,d) Rastrigin and (e,f) Schwefel.

Chapter 5

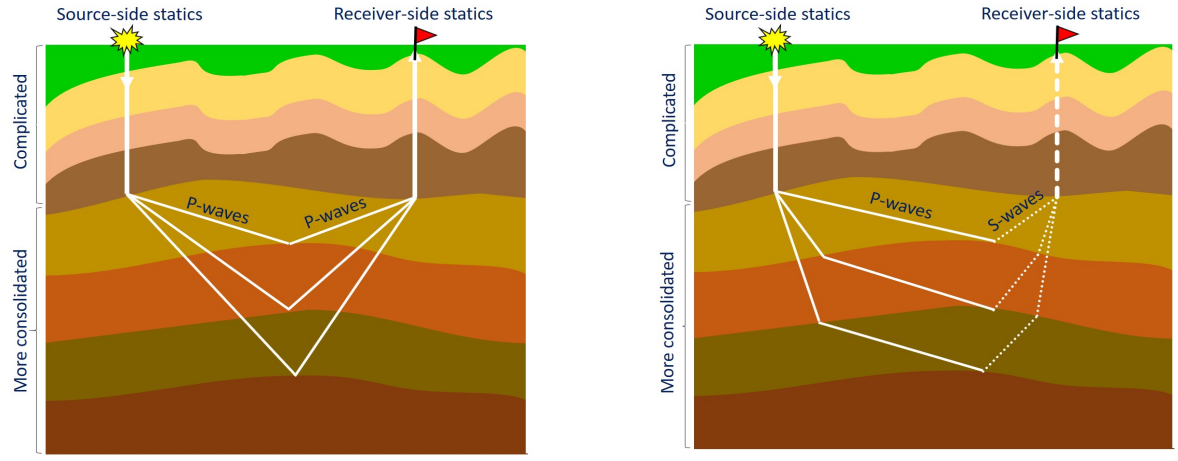
Application onto a synthetic 2D static correction demo

Although for many years compressional wave (or P-wave) seismic technology has featured the most technological developments, the use of shear-waves has recently proven to improve both imaging quality and reservoir characterization. In other words, the additional information provided by S-waves has allowed modelling reservoirs more adequately, redefining their quantitative properties and eventually finding new reserves in reservoirs that would otherwise be considered unsuccessful with P-waves alone.

We treat on-shore wave propagation of P-P and P-S waves with the models shown in Figure 5-1. In the former, the source excites P waves that travel vertically through the near-surface layer, after which deflecting beyond until they get reflected up somewhere in the subsurface. Then, while traveling on their way back, the reflected P waves experience another vertical propagation in the near-surface area before they reach the surface. For P-S waves the scenario is different, when the downgoing P-waves reach their reflection point, they get converted to S-waves and, before reaching the surface, S-waves travel vertically within the near-surface. For this reason, in P-S exploration, P-wave (or source-side statics) and S-wave (or receiver-side) statics are normally processed separately.

5-1 Definition of the fitness function

In data-driven static correction methods, the power of the stacked CMP section is considered a good quality indicator of the static correction because if all the traces in a CMP were applied with the correct statics, then all the traces would be perfectly aligned and its stacking power $\hat{E}$ should be maximized. Therefore, the idea is to align the seismic events by globally maximizing the corresponding stacking power as defined in Equation 5-1 for the 2D case and Equation 5-2 for the 3D case [Sun et al., 2017].



(a) P-wave seismic exploration

(b) S-wave seismic exploration

Figure 5-1: (a) Primary and (b) converted-wave seismic exploration. Source: adapted from [Sun et al., 2017]

$$\phi = \hat{E} = \sum_{i=1,step=M/2}^{N} \sum_{t=T_1}^{T_2} \left\{ \sum_{j=i}^{i+M} f_j(t) \right\}^2, \tag{5-1}$$

$$\phi = \hat{E} = \sum_{i=1,step=M/2}^{N} \sum_{k=1,step=V/2}^{Q} \sum_{t=T_1}^{T_2} \left\{ \sum_{j=i}^{i+M} \sum_{u=k}^{k+V} f_{ju}(t) \right\}^2, \tag{5-2}$$

where $f_{ju}(t)$ is the stacked CMP trace for $x = j$ and $y = u$, with iterators i and k scanning along the x and y dimensions, starting from 1 up to N and Q, respectively. Besides, M and V define the 2D local coherence window size that is covered by j and u, starting from (i, k). The increment steps are equal to $M/2$ and $V/2$, implying that there is a half-window overlap between two consecutive coherence windows. Finally, $[T_1, T_2]$ delimits the target zone.

Observe that, for the 2D case, k, Q, u and V can be neglected and then, the j^{th} stacked CDP trace $f_j(t)$ can also be expressed as:

$$f_j(t) = \sum_{i=1}^{F} P_i[t - C_R * stat(X_R) - C_S * stat(X_S)], \tag{5-3}$$

where F is the fold of the j^{th} CMP gather, $P_i(t)$ is the i^{th} trace in the current gather and $stat(X_R)$ and $stat(X_S)$ are the statics at source and receiver locations, respectively. Therefore, as the purpose of this demo is to handle receiver-side statics only, $C_S = 0$ and $C_R = 1$ and thus Equation 5-3 becomes:

$$f_j(t) = \sum_{i=1}^{N} P_i[t - stat(X_R)]. \tag{5-4}$$

To this end, real-valued chromosomes, describing $stat(X_R)$, are defined as candidate solutions where each gene represents the static time shift corrections of each receiver and are represented with float numbers.

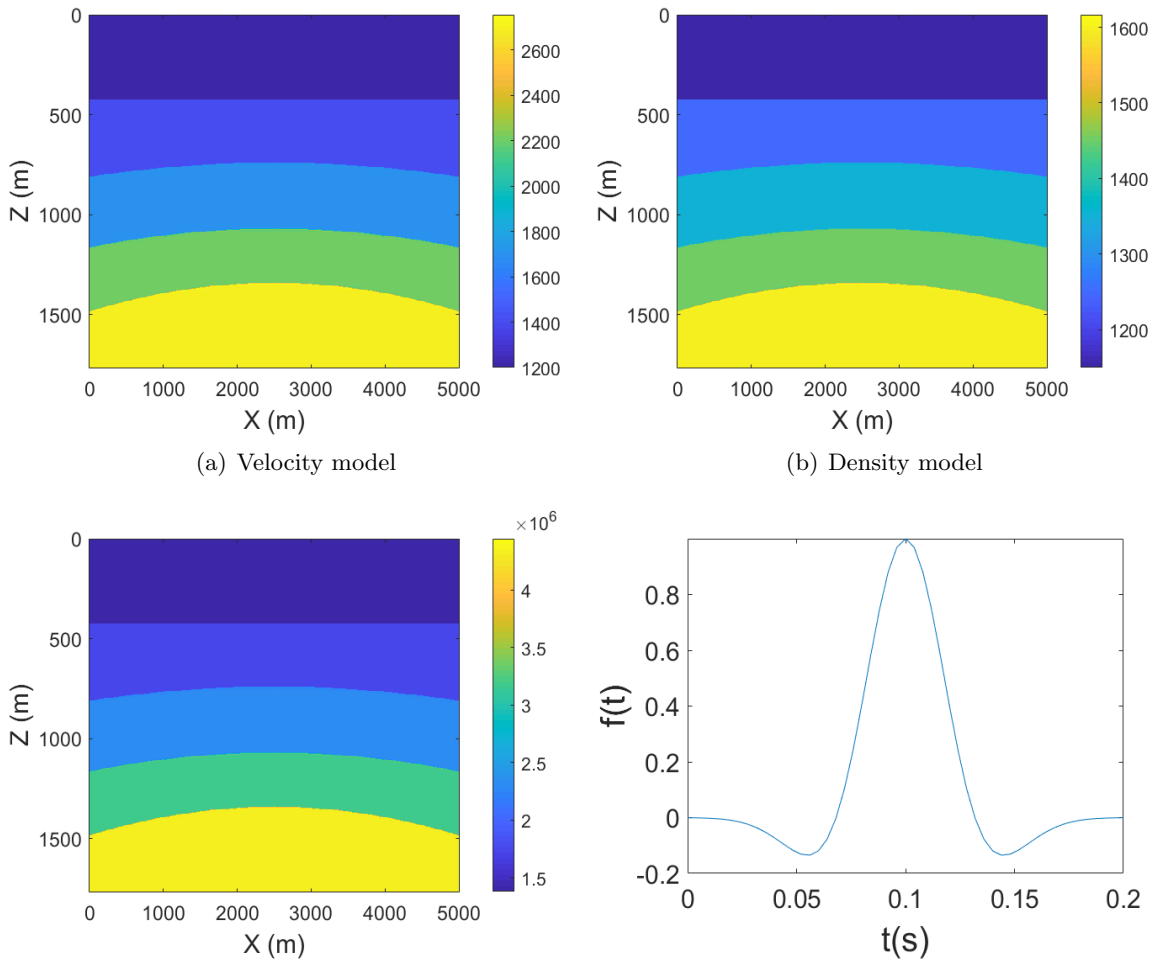
5-2 Synthetic data set

In this section we apply the eGA to three synthetic 2D data sets designed to correct for receiver-side statics, using Equation 5-1 as fitness function, with a local coherence window size equal to one ($M = 1$). The data sets were obtained using a layered model, consisting of four dome-like reflecting interfaces, as shown in Figure 5-2. The corresponding depth velocity and density models are shown in Figures 5-2(a) and 5-2(b), respectively, where we can see that with increasing depth, the velocity rises from 1200 up to 2700(m/s) and the density from 1150 to 1600 (kg/m^3). The acoustic impedance computed is shown in Figure 5-2(c) and, in all experiments, the source used was a Ricker wavelet (Figure 5-2(d)) with a central frequency of $F_c = 10(Hz)$, recorded at a sampling rate equal to 4(ms) during 2(s). Ray tracing was used as the modeling engine where only primaries are considered and perfect NMO correction, ignoring stretching and other side effects, is assumed.

To correct for receiver-side statics using our eGA, three experiments were designed. For the first and second experiments the synthetic data was generated using 101 receiver channels per shot in a moving spread-geometry, with an increasing offset from 25 up to 2475(m) on each side of the CMP. The receivers, located at the surface every 50(m) from a starting point $x = 0$, were contaminated with random static time-shifts varying from -200 to $+200(ms)$. On the source side, 100 locations were defined every 50(m), starting from $x = 25(m)$. Consequently, 200 CMP gathers with varying folds were obtained but, in order to imitate the reality, only the closest 40 traces of each gather were used to build the first data set.

The ideal noiseless CMP stack image obtained for the first experiment is illustrated in Figure 5-3(a). However, to make the demo even more realistic, in the second experiment, Gaussian noise was added to the data set so that the SNR becomes 5; the resulting CMP stack image is illustrated in Figure 5-3(b). For the third experiment, a more challenging data set was built: a total of 201 receiver channels, contaminated with statics ranging from -300 to $+300(ms)$, were used to more densely sample the subsurface with a spacing of 25(m). As a result, 400 CMP gathers with a $SNR = 5$ were sorted and processed to contain the 80 nearest offset traces only. Table 5-1 summarizes the design parameters of all experiments.

The random statics introduced per channel in each experiment are shown in Figure 5-4. In order to see the effect these time shifts have on the data set, the first CMP gather of the noisy data sets with statics introduced are shown in Figure 5-5. As it can be seen in Figure 5-6, noise and statics have severely degraded the quality of the CMP stack image, compromising its resolution and thus resulting in blurred and intertwined seismic events, compared to the original noiseless and statics-free CMP stack image in Figure 5-3.



(a) Velocity model

(b) Density model

(c) Acoustic impedance

(d) Ricker wavelet

Figure 5-2: (a) Velocity and (b) Density models, (c) Acoustic Impedance computed and (d) wavelet type used as source.

5-2-1 Convergence condition

Given the discrete character of this data set and the round-off problems associated with it, in this demo we found convenient to use a local search as the convergence condition for the eGA. This local search had to be computationally lighter than the eGA and provide faster convergence towards the actual result in order to speed up the search process. Therefore, to fulfill these requirements, a greedier version of the Back and Forth Coordinate Descent (BFCD) search proposed by [Nocedal and Wright, 1999] was designed and used as the stopping criterion for the execution of our eGA. In other words, the eGA was run for a considerable number of fixed iterations t_{BFCD}, after which the local search was applied; when the local search found the true solution the search finished, otherwise the eGA just continued. The pseudo-code of our Greedy BFCD is given in Appendix A.

The original BFCD local search [Nocedal and Wright, 1999] is one of the many coordinate descent methods and searches cyclically along all coordinate directions. Even-though its rate of convergence is often slower than that of steepest descent methods, e.g. quasi-newton method, they are commonly preferred because no gradient information is required, making it more versatile, specially when handling non-continuous functions.

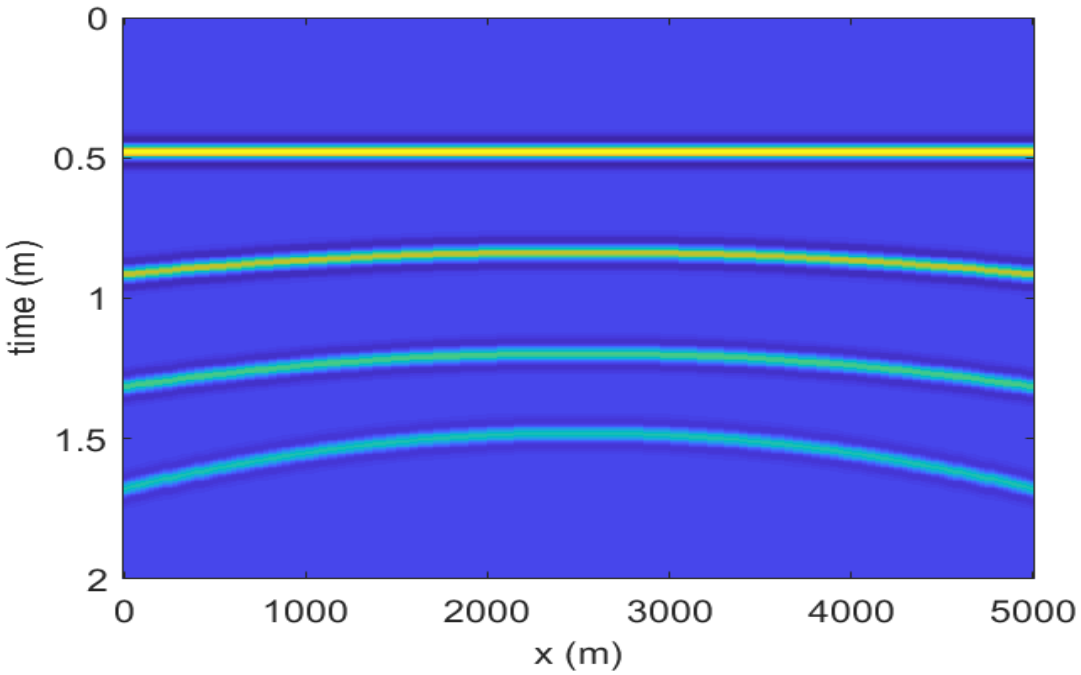
Item	Experiment 1	Experiment 2	Experiment 3
X dimension	5 (km)	5 (km)	5 (km)
Z dimension	1.77 (km)	1.77 (km)	1.77 (km)
Sampling interval	4 (ms)	4 (ms)	4 (ms)
Record length	2 (s)	2 (s)	2 (s)
Number of receivers	101 (m)	101 (m)	201 (m)
Receivers Interval	50 (m)	50 (m)	25 (m)
Number of sources	100	100	200
Sources Interval	50 (m)	50 (m)	25 (m)
Source Wavelet	Ricker	Ricker	Ricker
Central Frequency	10 (Hz)	10 (Hz)	10 (Hz)
Total number of CMP gathers	200	200	400
CMP interval	25 (m)	25 (m)	12.5 (m)
fold	40	40	80
SNR	∞	5	5
Min. statics	- 200 (ms)	- 200 (ms)	- 300 (ms)
Max. statics	+200 (ms)	+200 (ms)	+300 (ms)

Table 5-1: Design parameters of the three experiments built to generate the synthetic datasets used for the receiver-side statics-correction demo.

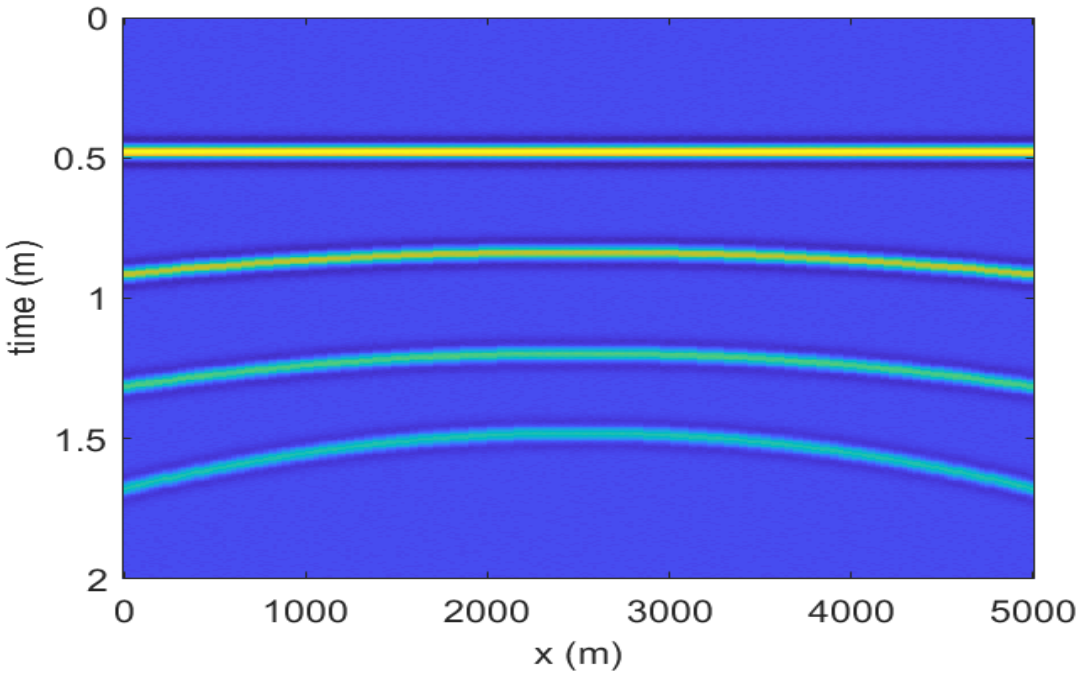
The parameters for the eGA in each of the experiments, as defined in subsection 3-2-1, are summarized in Table 5-2.

Item	Experiment 1	Experiment 2	Experiment 3
n	101	101	201
P_m range	(1/101)*[1,3]	(1/101)*[1,3]	(1/201)*[1,3]
N_j range	[50,100]	[50,100]	[50,100]
P_c range	[0.3,0.8]	[0.3,0.8]	[0.3,0.8]
t_{es}	50	50	50
t_{com}	20	5	5
N_i	4	4	5
n_α	100	100	100
t_{SADE}	40	40	30
α_0	2	2	2
m_{max}	6	6	6
t_{BFCD}	500	500	500

Table 5-2: eGA parameters used by the eGA in each experiment.

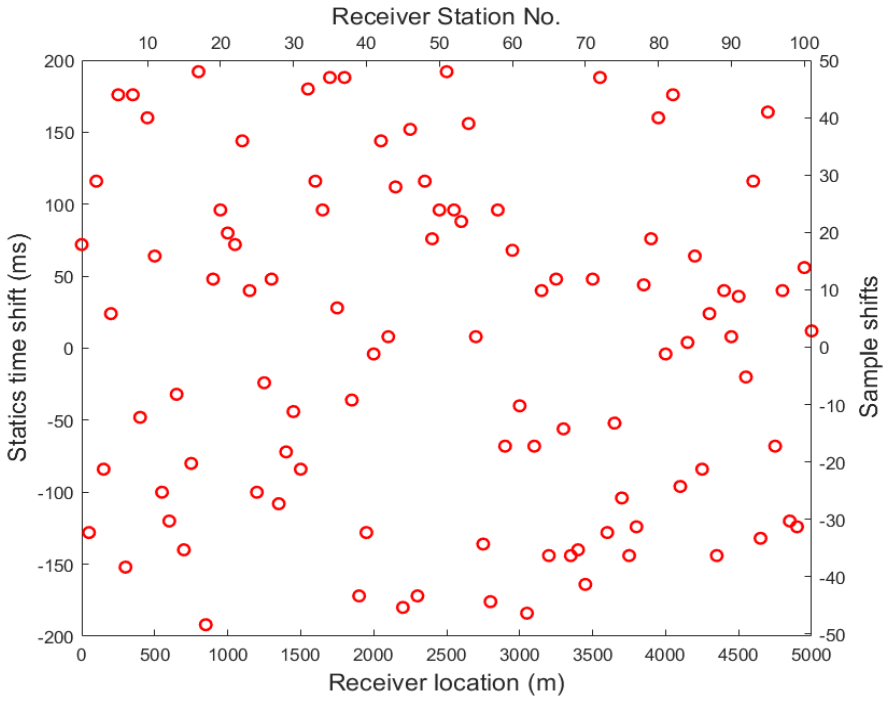


(a) Noisless CMP stack image

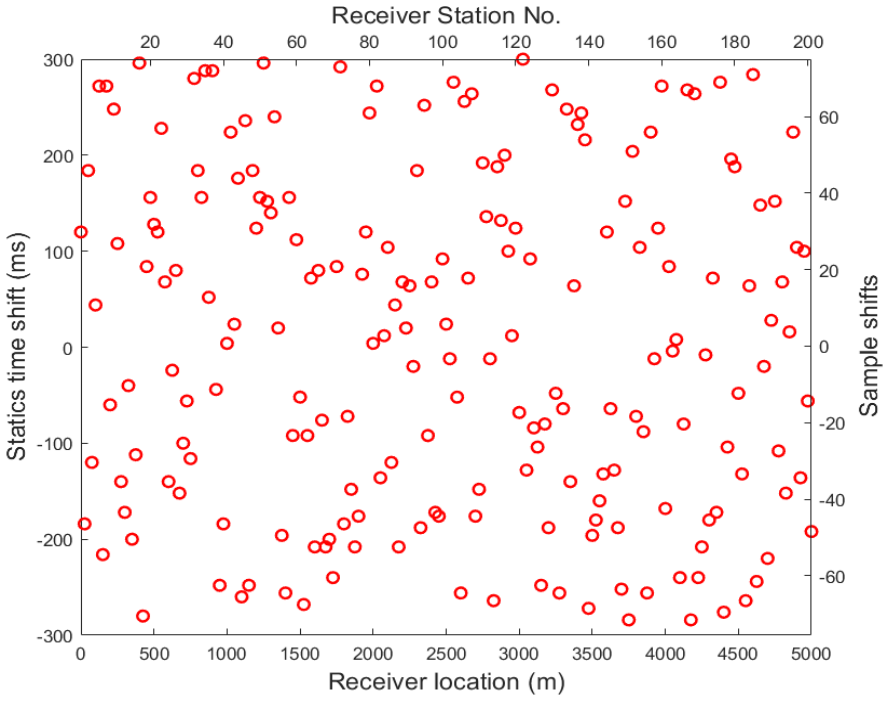


(b) Noisy CMP stack image

Figure 5-3: CMP stack image of the subsurface (a) without and (b) with Gaussian random noise.

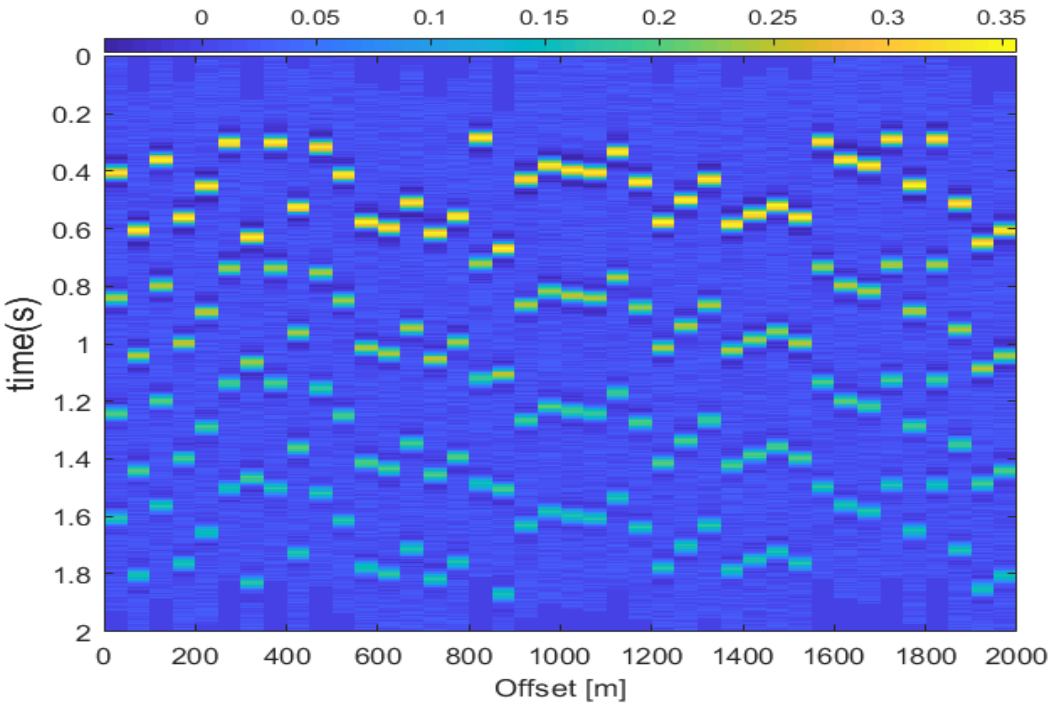


(a) Experiments 1 and 2: Random statics ranging from -200 to $+200(ms)$ introduced in 101 receivers.

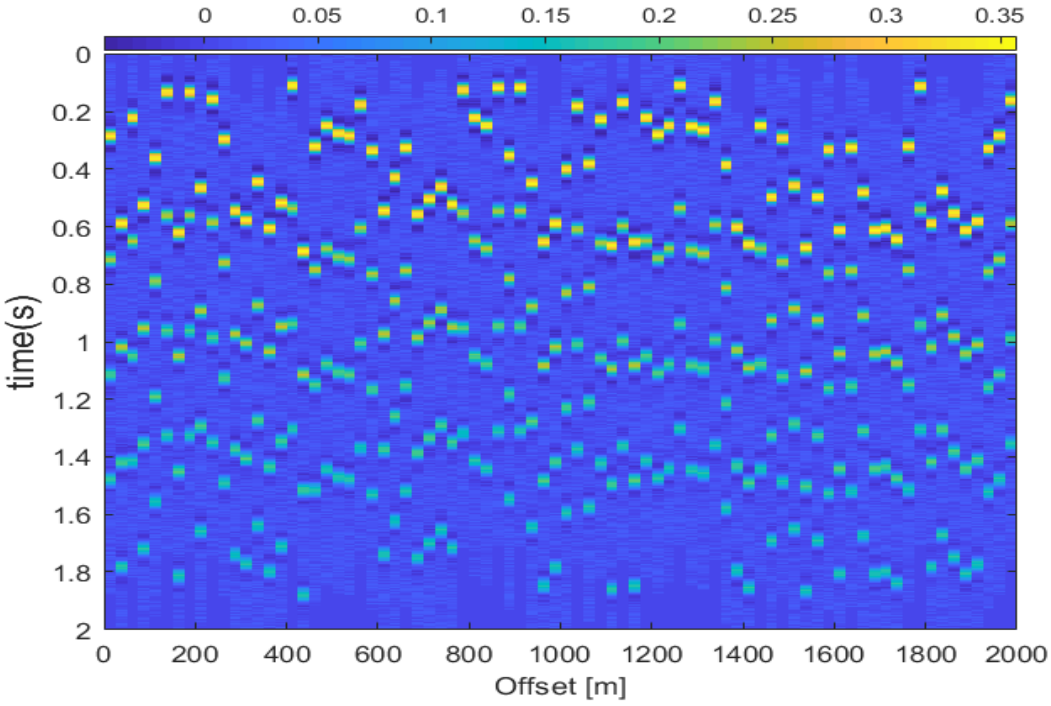


(b) Experiment 1: Random statics ranging from -300 to $+300(ms)$ introduced in 201 receivers.

Figure 5-4: Random statics introduced in the experiments (a) 1 and 2 and (b)3.

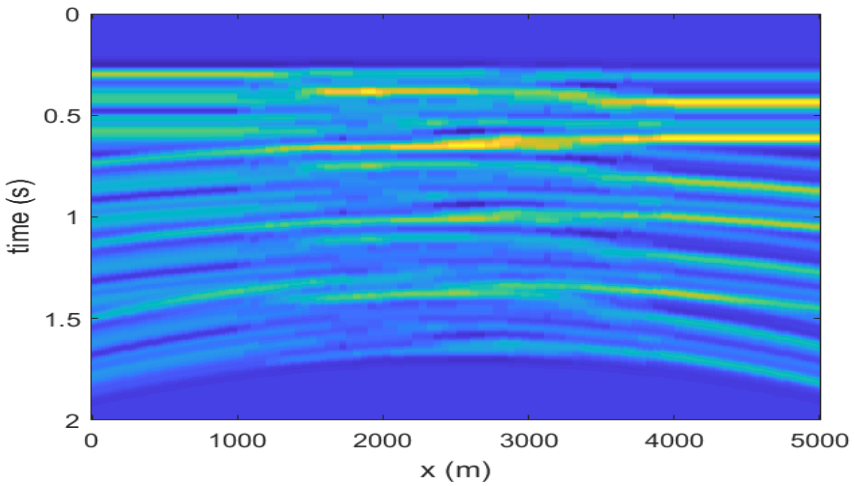


(a) Experiment No. 2

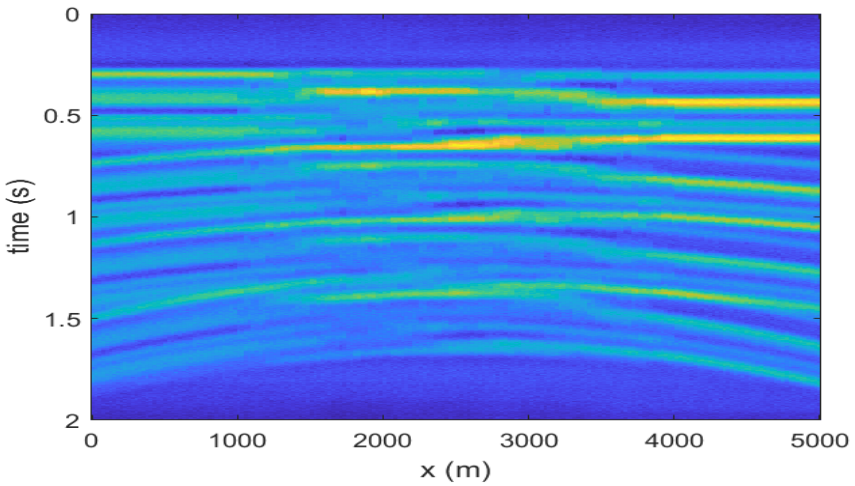


(b) Experiment No. 3

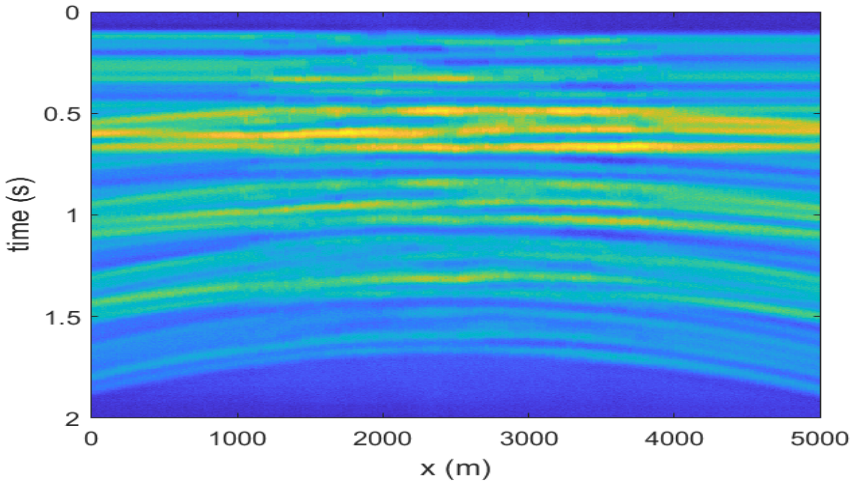
Figure 5-5: First CMP gather of the experiments a) 2 and b) 3 with random statics introduced.



(a) Experiment No. 1



(b) Experiment No. 2



(c) Experiment No. 3

Figure 5-6: CMP stacked image before static correction for the experiments a) 1, b) 2 and c) 3.

5-3 Results

For all of the experiments, the statics obtained after the eGA (blue crosses) and the Greedy BFCD (yellow filled circles) optimization are shown in Figure 5-7. It can be seen in this figure that, compared to the noiseless data set (Figure 5-8(a)), the number of outliers obtained using the eGA increases with the difficulty added by the noise (Figure 5-8(b)) and the wider statics range (Figure 5-8(c)). Although the statics obtained using the eGA are already close to the true solution, the application of the Greedy BFCD search lead to the global maximum (yellow filled circles) in all the experiments.

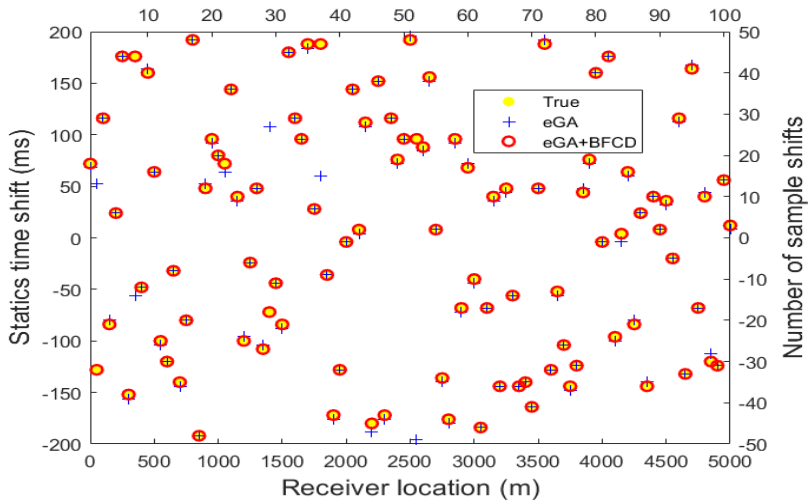
The CMP stack images after optimization by the eGA are shown in Figure 5-8. As it can be seen in these figures, the seismic events are already very well aligned and far better resolved compared to the statics-contaminated CMP stack images in Figure 5-6. However, due to a few outlier statics obtained with the eGA (blue crosses) in Figure 5-7, some artifacts (red dashed boxes) are present, resulting even in one false seismic event (green dashed box) in the Experiment 3, as illustrated in Figure 5-8(c).

Finally, the results obtained after application of the Greedy BFCD are shown in Figure 5-9, where it can be seen that the stacking power within the the target zone is further maximized and resulted in the same perfectly aligned CMP stack image for all of the three experiments, as shown in the noisy statics-free CMP stack image of Figure 5-3(b).

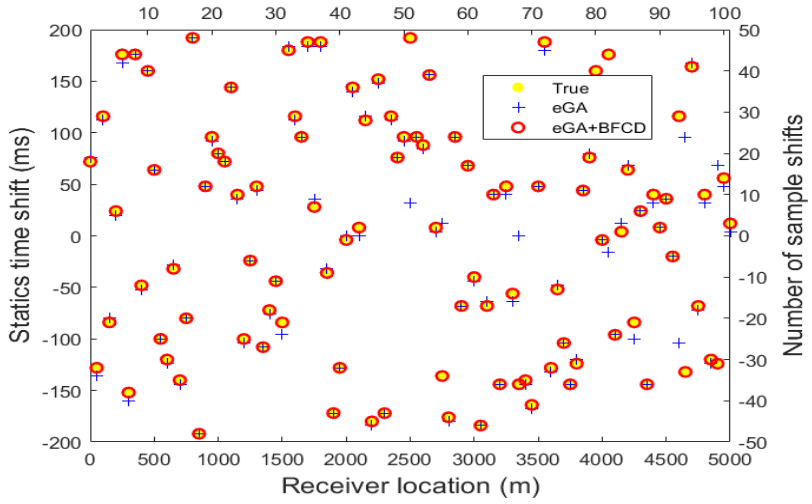
5-4 Summary

The eGA was successfully applied to correct for receiver-side statics in a $5(km)$ length 2D seismic section containing 5 dome-like reflecting interfaces, whose CMP gathers were simulated using ray tracing. To this end, three different scenarios were proposed: an ideal noiseless data set and two more realistic noise-contaminated setups.

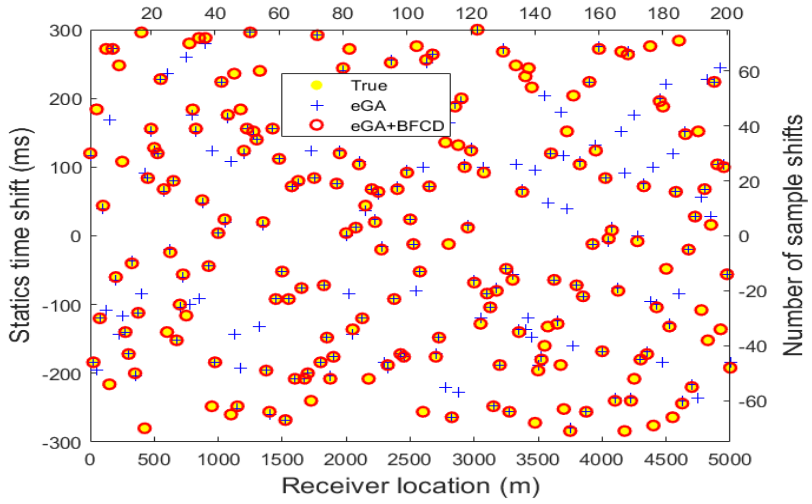
The statics were estimated by maximizing the stacking power of the CMP gathers via the eGA whose convergence condition was dictated by a Greedy BFCD local search.



(a) Experiment No. 1

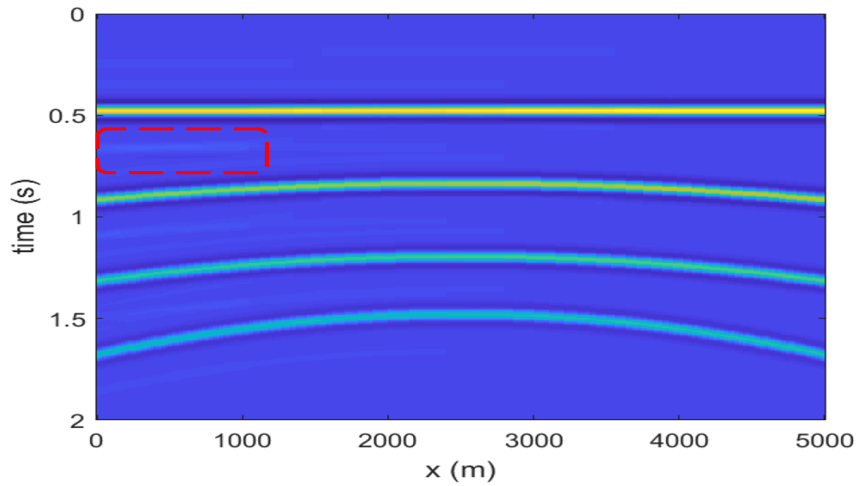


(b) Experiment No. 2

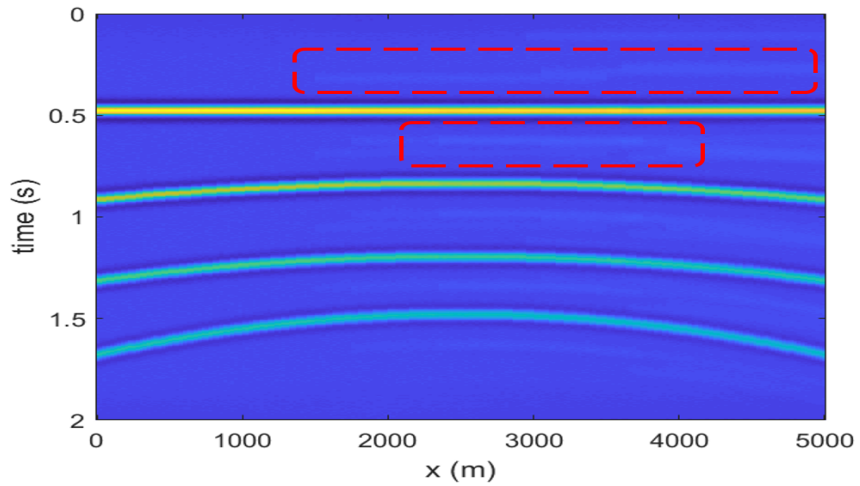


(c) Experiment No. 3

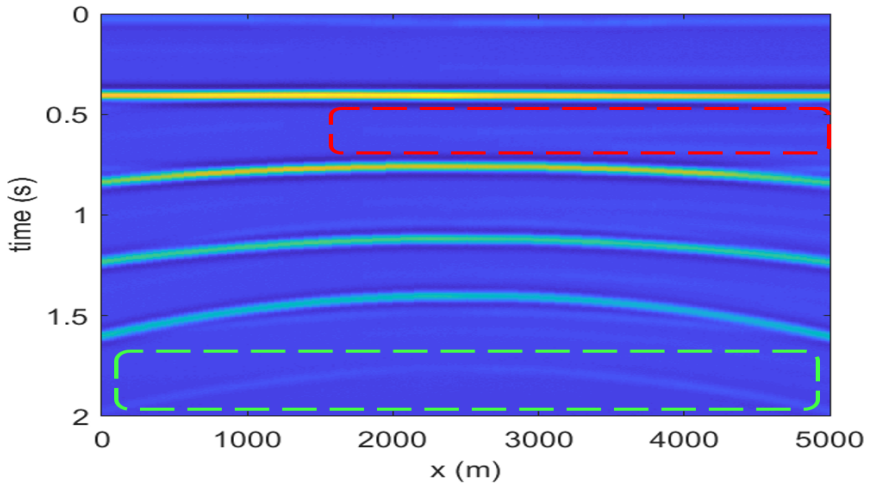
Figure 5-7: Random statics introduced (filled yellow circles) and calculated after eGA (red circles) and Greedy BFCD (blue crosses) for the experiments a) 1, b) 2 and c) 3.



(a) Experiment No. 1

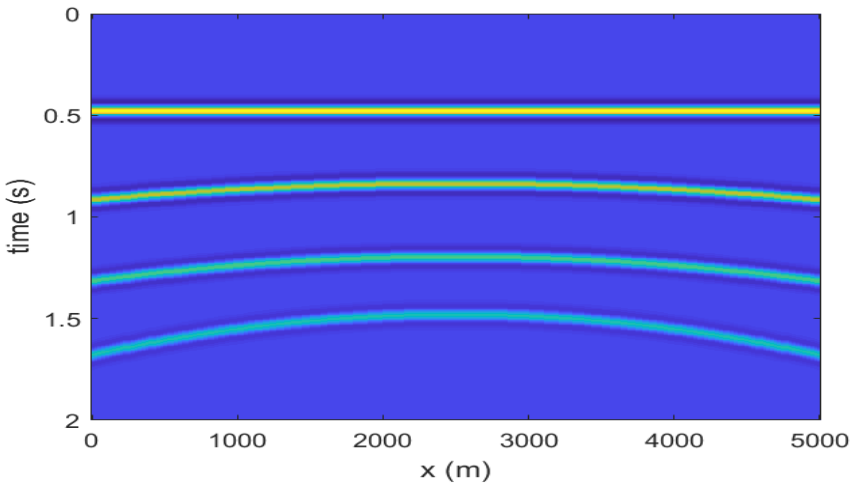


(b) Experiment No. 2

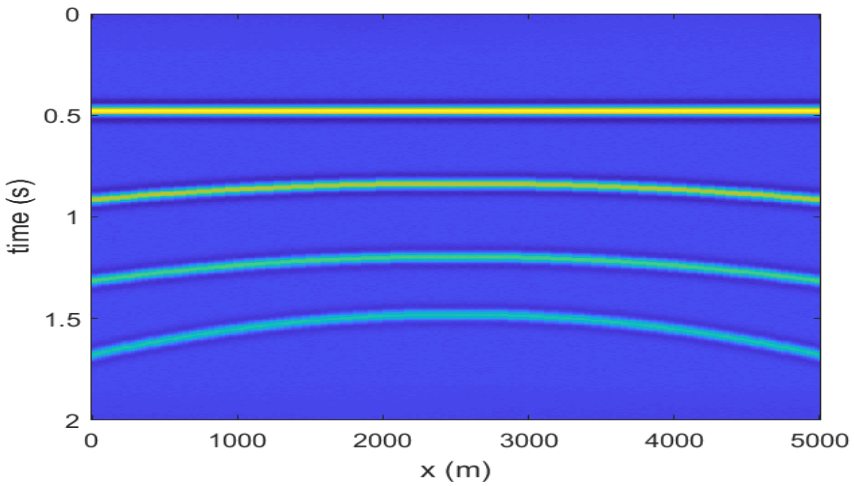


(c) Experiment No. 3

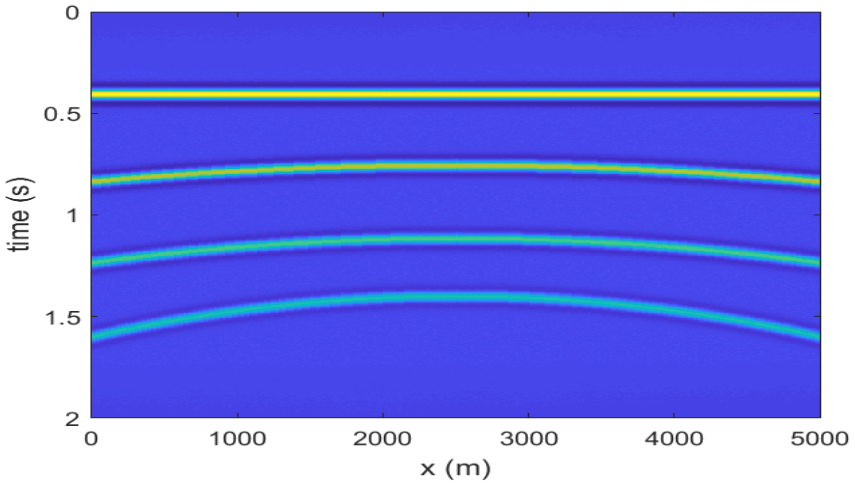
Figure 5-8: CMP stack image after optimization by eGA for experiments a) 1, b) 2 and c) 3.



(a) Experiment No. 1



(b) Experiment No. 2



(c) Experiment No. 3

Figure 5-9: CMP stack image after Greedy BFCD search for the experiments a) 1 b) 2 and c) 3.

Chapter 6

Application onto a synthetic 2D CRS demo

The purpose of CMP stacking is to enhance the quality of an ideal ZO section by using traces sharing a CMP but generated by source-receiver pairs separated a distance larger than zero. To do so, before stacking, the wave travel-time has to be corrected to account for this offset. This correction, also known as Normal Moveout time correction (t_{NMO}), is given by the following equation [Waldeland et al., 2017]:

$$t_{CMP}^2 = t_0^2 + \left(\frac{2h}{V_{NMO}}\right)^2, \tag{6-1}$$

where h is the half-offset of each source-receiver pair, t_0 the two-way traveltime and v_{NMO} is the NMO velocity. In this equation it is straightforward to see that the stacking velocity has to be determined first, in a model or data-driven fashion.

As mentioned in subsection 1-2-2, the purpose of CRS is also to transform multicoverage seismic reflection data into a higher quality ZO section and, in turn, provide more attributes useful for a broader variety of inversion problems than the standard stacking velocity alone. As the NMO travel-time approximation assumes all traces have the same midpoint, the number of traces used for its computation is dictated by the number of traces sharing the same midpoint. Conversely, when computing the CRS stack, not only traces belonging to the same CMP but also traces from nearby CMP gathers are used, which leads to a stacked image with an even higher SNR. Therefore, one super-gather is arranged at each central midpoint m_0 and contains those traces whose midpoints m are within the maximum aperture $m_{d,max} = m_{max} - m_0$ defined for that stacking surface.

In CRS, the final ZO section is the result of summing the amplitudes of moveout corrected prestack data. This travel-time correction is given by a second-order hyperbolic approximation (stacking surface) described by 8 kinematic attributes in 3D or 3 in 2D and defined as a function of midpoint and half-offset coordinates, as illustrated in Figure 6-1. According

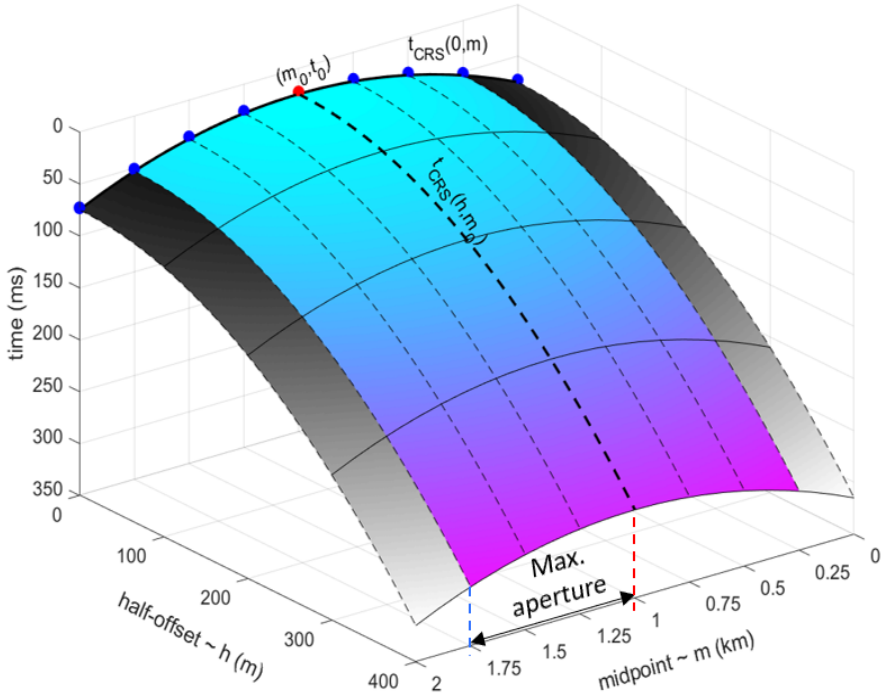
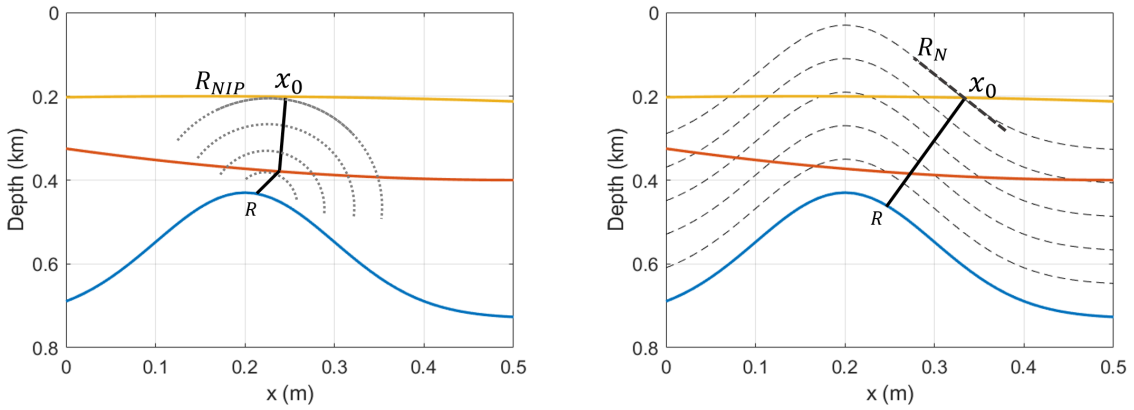


Figure 6-1: CRS stacking surface with $m_{d,max} = 0.75(km)$ at a fixed point $P_0 = (m_0, t_0)$ of the ZO section, as a function of half-offset h and midpoint m coordinates. The solid black curves highlights common-offset traces and dashed curves traces with a CMP.

to the model-space formulation of the travel-time approximation in 2D (Equation 6-2), the triple of parameters are related to the *emergence angle of the normal central ray*, β_0, and the radii of wavefront curvatures of the Normal Incidence Point R_{NIP} (Figure 6-2(a)) and Normal Wave R_N (Figure 6-2(b)), respectively. Furthermore, based on its data-space definition (Equation 6-3), these three parameters are the first- and second-order derivatives of the travel-time approximation with respect to its midpoint m and half-offset h coordinates, as expressed in Equation 6-4. The last parameter is of special interest because it is related to V_{NMO} and then carries information that can be used in subsequent stages of processing as a preliminary macrovelocity model.

These three parameters have to be defined for each sample in the simulated ZO section, which is why their simultaneous estimation can become a computational challenge. Then, the search strategy is to find the best combination of parameters for the super-gather that result in the ZO section with the maximum semblance. To lighten this burden and facilitate the convergence towards the global maximum, the lower and upper boundaries of the search space can be defined based on a priori knowledge of the field. After analyzing the geological complexity, the boundaries of β_0 can be set to define a dip filter of the zone whereas the limits for R_{NIP} range around the maximum and minimum values of an a priori stacking velocity model. Finally, as the radius of curvature of the Normal Wave approaches infinite for planar wavefronts, then to define its boundaries large values are used in practice. Nevertheless, when no a priori information is available, the search space can also be defined in a fully automated manner by assuming the extreme cases: $-\infty < R_N, R_{NIP} < +\infty$ and $\frac{-\pi}{2} < \beta < \frac{+\pi}{2}$.



(a) Normal Incidence Point wave

(b) Normal wave for the exploding reflector at R

Figure 6-2: Hypothetical experiments to define (a)R_{NIP} and (b) R_N. The former is the result of inducing an up-going normal incidence-point (NIP) wave at R whereas the latter results from a simultaneous excitation of up-going Normal waves along reflector R.

6-1 The Common Reflection Surface operator

The CRS stacking operator is a hyperbolic second-order Taylor approximation that relates the two-way travel-time of a reflection caused by one source located at x_s and recorded by one receiver at x_r, as described in the Equation 6-2 illustrated in Figure 6-1:

$$t^2_{CRS}(h,m) = (t_0 + \frac{2sin\alpha}{v_0}(m-m_0))^2 + (\frac{2t_c cos^2\alpha}{v_0})(\frac{(m-m_0)}{R_N} + \frac{h^2}{R_{NIP}}), \tag{6-2}$$

where t_0 is the two-way travel time of normal incidence from m_0 down to the seismic event of interest, $h = \frac{x_R - x_S}{2}$ the half-offset, $m = \frac{(x_R + x_S)}{2}$ the midpoint between source and receiver positions and m_0 its central midpoint. Note that Equation 6-2 is a valid approximation as long as neither h nor $(m - m_d)$ are too big.

Although this model-space definition is more intuitive and descriptive, when estimating the parameters it is more convenient to use the data-space definition:

$$t^2_{CRS}(m,h) = (t_0 + A(m-m_0))^2 + B(m-m_0)^2 + Ch^2, \tag{6-3}$$

where the parameters A, B and C are its first and second order derivatives [Waldeland et al., 2017], as follows:

$$\begin{aligned} A &= \left.\frac{\partial t}{\partial m}\right|_{\substack{m=m_0 \\ h=0}}, \\ B &= t_0 \left.\frac{\partial^2 t}{\partial m^2}\right|_{\substack{m=m_0 \\ h=0}}, \\ C &= t_0 \left.\frac{\partial^2 t}{\partial^2 h}\right|_{\substack{m=m_0 \\ h=0}}. \end{aligned} \tag{6-4}$$

To ease notation, Equation 6-3 can also be written as:

$$t^2_{CRS}(h, m_d) = (t_0 + Am_d)^2 + Bm_d^2 + Ch^2, \tag{6-5}$$

where $m_d = m - m_0$ is the aperture of the stacking surface, i.e. difference between the trace midpoint m and the central midpoint m_0.

From Equation 6-5 it is straightforward to see that t_{NMO} is an special case when only traces whose midpoint is the central midpoint itself ($m = m_0$) are used, i.e. when the maximum aperture of the super-gather is zero (observe the thickest dashed line of Figure 6-1):

$$t^2_{CRS}(h, 0) = t_{NMO} = (t_0)^2 + Ch^2. \tag{6-6}$$

After the move-out correction is performed, the process to create the final ZO section is the same as CMP stacking.

6-2 Common Reflection Surface Demo

6-2-1 Synthetic dataset

The model parameters used to simulate the synthetic CMP gathers are summarized in Table 6-1 and the velocity and density models are shown in Figures 6-3(a) and 6-3(b), respectively. The subsurface model accommodated one anomaly to make this demo more challenging. Furthermore, to illustrate the power of CRS, Gaussian noise was added to our simulated data. The first noise-contaminated data set has a SNR equal to 3 and the second one equal to 1.

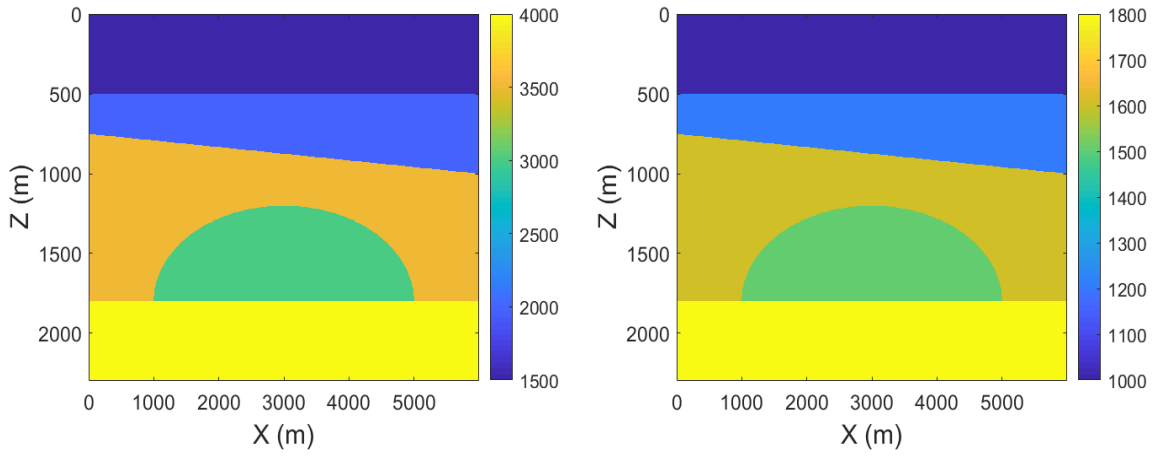
Data generation

Acoustic wave propagation is simulated via the finite difference method ($(2, 4)$ scheme, i.e. 2^{nd} order in time, 4^{th} order in space) using first-order equations. Furthermore, Perfectly Matched Layer (PML) absorbing boundary condition is applied at all sides of the model in order to neglect edge reflections. After the data were generated, a heavy pre-processing stage was done prior to optimization. This pre-processing was aimed at reducing the computational burden of CRS alignment and stacking. As a result, for the data set we work on, only 373 CMP gathers with a constant fold equal to 51 were selected.

In our demo the maximum aperture was set to $m_{d,max} = 50(m)$. Therefore, to build each super-gather at every CMP, four additional CMPs were used on either side. To illustrate this, the super-gathers built at the CMP No. 187, located right in the middle of the section, for the data sets with $SNR = 3$ and $SNR = 1$ are shown in Figures 6-4(a) and 6-4(b), respectively. Furthermore, to better understand the fitting of the seismic data with the stacking surface shown in Figure 6-1, the same super-gather (for the $SNR = 3$ data set) is arranged accordingly and shown in Figure 6-5. Last but not least, it is worthwhile mentioning that, as these extra CMP gathers are not available at the boundaries, and we strive to keep the symmetry and consistency of the problem in the whole data set, no super-gathers were built at the first and last four CMPs, being these locations also excluded from the final CRS stack image. Therefore, a total of 365 super-gathers were built from CMP number 5 to 369.

Item	Data set 1	Data set 2
Sampling rate	4 (ms)	4 (ms)
Record length	2 (s)	2 (s)
X dimension	6 (km)	6 (km)
Z dimension	2.5 (km)	2.5 (km)
Receivers Interval	12.5 (m)	12.5 (m)
Sources Interval	12.5 (m)	12.5 (m)
Source Wavelet type	Ricker	Ricker
Dominant Frequency	17.5 (Hz)	17.5 (Hz)
Maximum Frequency	60 (Hz)	60 (Hz)
Total number of CMP	475	475
Number of CMP gathers used	373	373
Total number of super-gathers	365	365
CMP interval	12.5 (m)	12.5 (m)
Fold	51	51
SNR	**3**	**1**

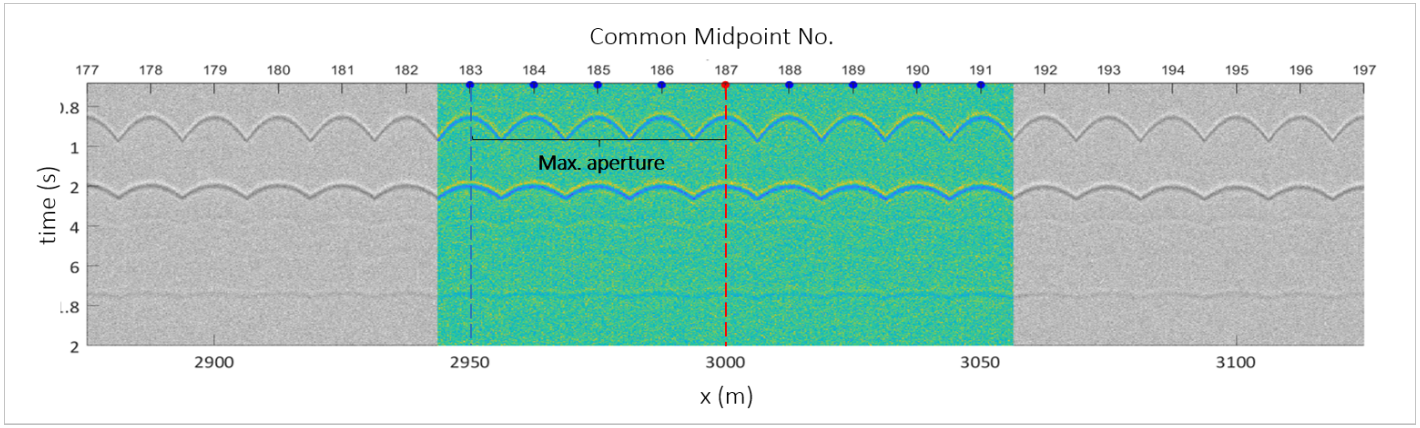
Table 6-1: Parameters of the synthetic data sets built for the CRS demo.



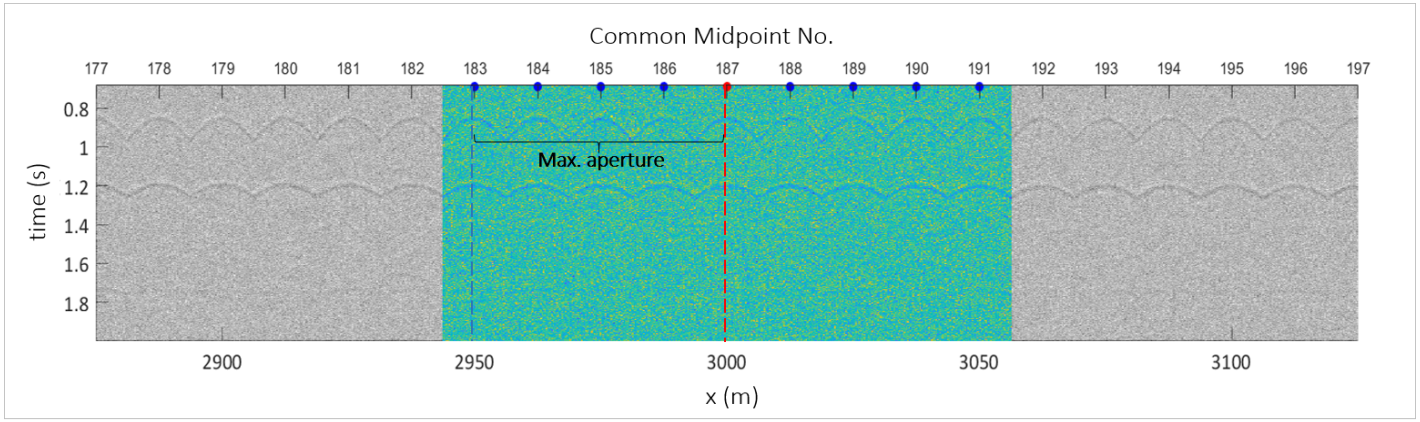
(a) Velocity model

(b) Density model

Figure 6-3: (a) Velocity and (b) Density models of the subsurface. Notice the low velocity and density anomaly in the subsurface.



(a) Data set 1: $SNR = 3$.



(b) Data set 2: $SNR = 1$.

Figure 6-4: Re-sorted data set: individual CMP gathers are plotted next to each other in 2D for the two synthetic data sets with a) $SNR = 3$ and b) $SNR = 1$. Observe that the maximum aperture is 50 (m) and thus, to construct the super-gather No. 187 (in full color), 4 more CMP gathers were used at each side of the central midpoint of interest ($CMP_x = 3000$(m)).

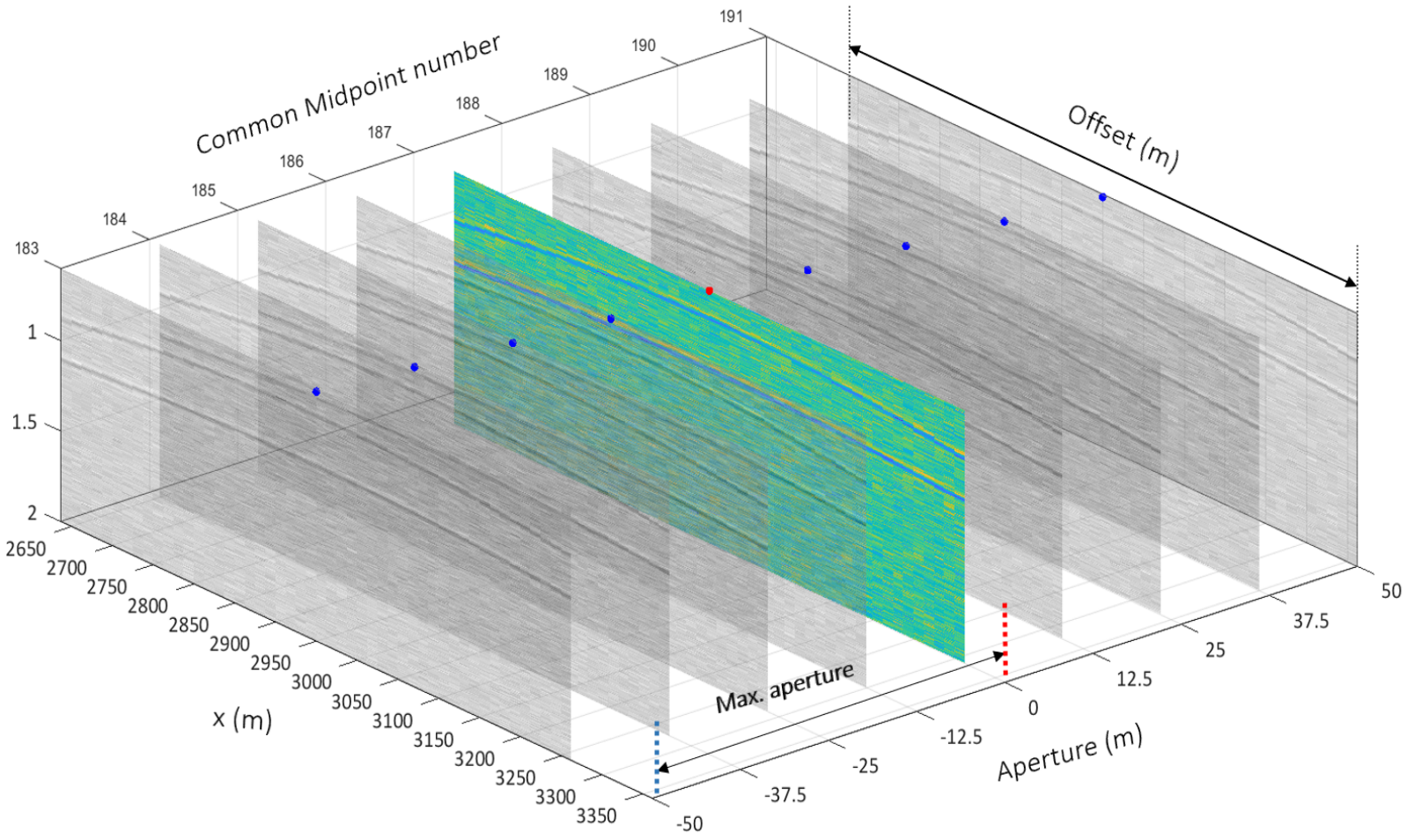


Figure 6-5: Super-gather built for the CMP No 187, located right in the middle of the section, of the noisy data set with $SNR = 3$.

6-2-2 CRS parameters estimation

Even though the computational complexity of a global search is significantly increased compared to the pragmatic search proposed by Mann et al. [1999]; Jäger et al. [2001] and Waldeland et al. [2017], where the three parameters are searched for sequentially, and the propagation of the errors obtained after every step compromises the accuracy of the CRS attributes and subsequently the quality of the final ZO section. In a pragmatic search, the attributes are determined individually in a three-steps single-parameter search procedure. For this reason, in this work we propose using our eGA to do a global optimization on the coherence-based objective function of Equation 6-5, by simultaneously determining the three CRS parameters in a single search procedure.

As aforementioned, the CRS parameters have to be estimated for every t_0 and m_0 such as those for which the amplitude of all seismic traces fitting the stacking surface are the most coherent. If so, all of the traces within $t_{CRS}(h, m)$ should ideally refer to the same seismic event. However, considering the dimensions of our original data set, accomplishing this optimization for every (t_0, m_0) point of the seismic section can become computationally very expensive. For this reason, based on a-prior knowledge, the first $700(ms)$ (or 170 samples) containing the water layer were neglected from the $2(s)$ (or 500 samples) of recording, as there are no seismic events. Then, in the 330 samples left, only 8 representative samples or *nodes* (separated every 50 samples along the time dimension) were anchored, which after the optimization procedure, were linearly interpolated to have dense CRS parameters at each (t_0, m_0) point.

In summary, our optimization problem narrows down to estimating 8 CRS triples for each of the 365 super-gathers cautiously built. To this end, real-valued chromosomes, containing the 24 CRS attributes, were defined as candidate solutions, with each gene represented with float numbers; the optimization process was repeated 365 times for each super-gather. On the other hand, as no a-prior information was available, there was no guide to restrict the search space. Then, a trial-and-error preliminary search was conducted, where a wide range of possible boundaries were tried until the most plausible solution ranges were found within a reasonable time framework. Finally, it is worth mentioning that, considering the physical meaning of C in Equation 6-2, i.e. proportional to inverse of the squared V_{NMO}, it was constrained to be a decreasing function of time.

6-3 Results

In this section we present some results that further validate the improved convergence speed of our eGA when applied to the CRS global optimization problem on true synthetic noisy data sets. To do so, first, the aGA is allowed to run a maximum number of $t_{max} = 100$ iterations and, as the true solution is unknown, its result is assumed to be our target solution. Then, our eGA is run for 30 iterations or until the difference between our and the target fitness is less than 3%. As a result our eGA finished its job approximately in half the time used by the aGA, with the workstation running in parallel at full capacity.

The final CRS stack image after the optimization is done on the noisy data with $SNR = 3$, using both aGA and eGA are shown in Figures 6-6(a) and 6-6(b), respectively. It can be

seen in these figures that both algorithms produce high quality ZO sections with very good continuity of the seismic events. However, it deserves pointing out that due to the way our synthetic data was pre-processed, i.e. only near-offset traces are available, after the CRS optimization, one internal multiple event (red dashed box) was also aligned and became visible. We believe that if further offset traces were included in the data set, this event should be stacked out.

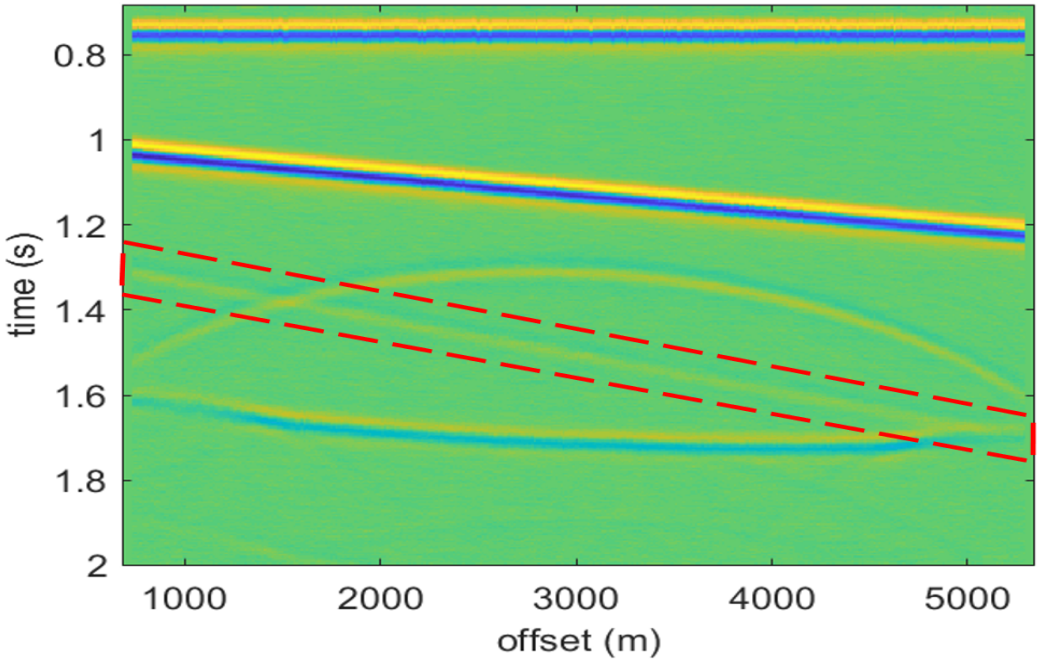
Additionally, the CMP gathers located at the extremes of the section (CMP No. 1 and 373) are illustrated in Figure 6-7. In these figures it is straightforward to see that the original CMP gathers, shown in Figures 6-7(a) and 6-7(b), were almost perfectly aligned after the optimization by aGA (Figures 6-7(c)) and 6-7(d)) and eGA (Figures 6-7(e)) and 6-7(f))).

The same process was repeated for the noisy data set with $SNR = 1$, shown in 6-4(b), and the ZO sections obtained using aGA and eGA, are shown in Figures 6-8(a) and 6-8(b), respectively. As observed, due to the complexity added by the noise, the continuity of the seismic events is not as good as in the previous data set. In particular, this undesired effect is more notorious in curved reflectors, like the ones encircled by the white dashed lines. However, keeping in mind the low SNR of this data set, the overall quality of the final CMP stack image proves the real power of CRS stacking. Furthermore, once again, the original CMP gathers shown in Figures 6-9(a) and 6-9(b), were also almost perfectly aligned after the optimization by aGA (Figures 6-9(c)) and 6-9(d)) and eGA (Figures 6-9(e)) and 6-9(f))).

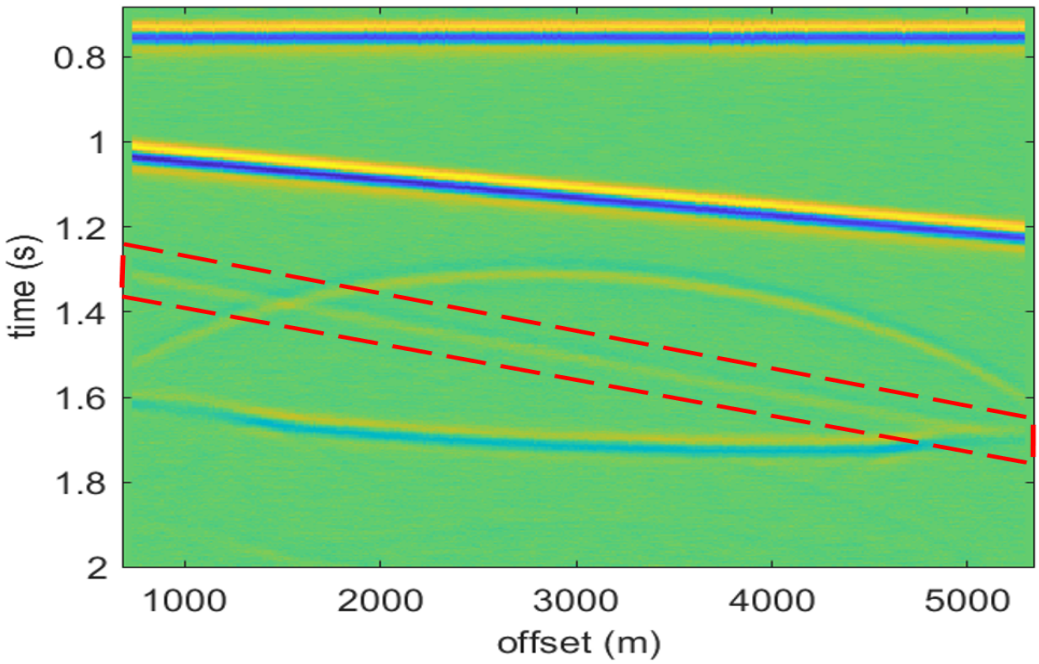
Finally, the CRS attributes A, B and C, in Equation 6-5, retrieved with both GAs, for the CMP No. 163, are shown in Figure 6-10. As it can be observed, the results obtained by our eGA (red circles) closely follow those obtained by the aGA (blue crosses), except for the attribute B that is slightly deviated in both data sets. However, in spite of the different noise levels, the C attribute, which is of most interest, is consistent in both cases for every data set.

6-4 Summary

The aGA was applied to solve the 2D CRS stacking optimization problem in a single-search procedure, i.e. simultaneously determining the three CRS attributes and its solution was used as our target solution. Later the eGA was challenged to reach that target solution by a difference of up to 3% in less number of iterations and the results are promising.

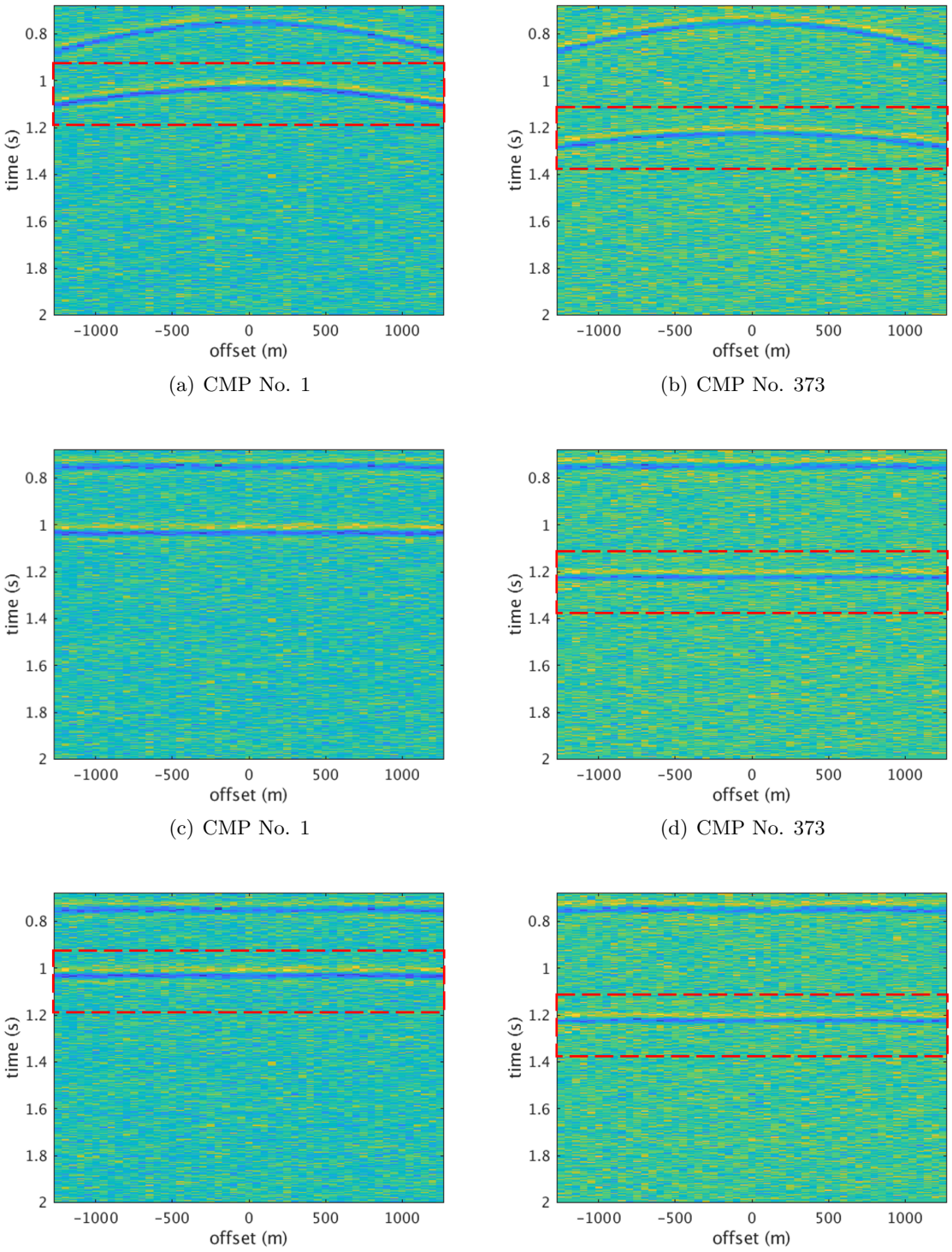


(a) optimization by aGA



(b) optimization by eGA

Figure 6-6: CRS stack images after optimization by a) aGA and b) eGA of the noisy CMP gathers with $SNR = 3$.



(a) CMP No. 1

(b) CMP No. 373

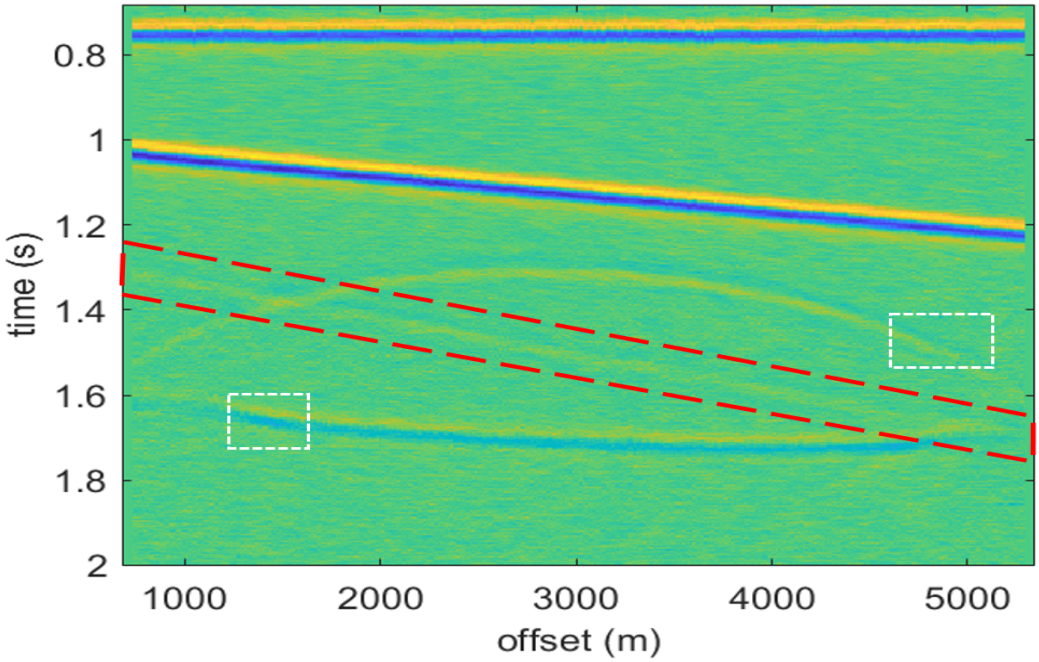
(c) CMP No. 1

(d) CMP No. 373

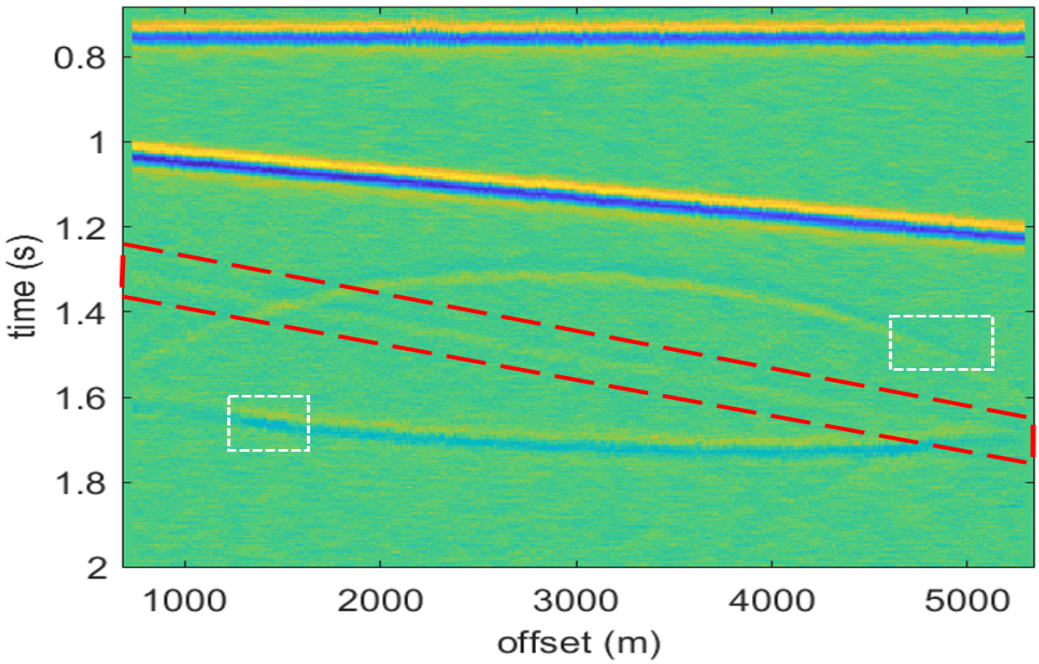
(e) CMP No. 1

(f) CMP No. 373

Figure 6-7: First (left) and Last (right) CMP gathers of the noisy data set with $SNR = 3$, (a,b) before and after optimization using (c,d) aGA and (e,f) eGA .

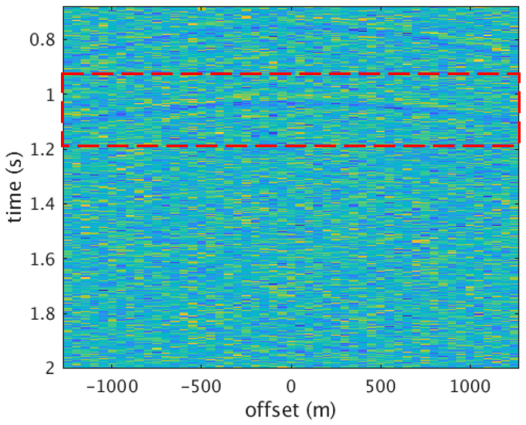


(a) optimization by aGA

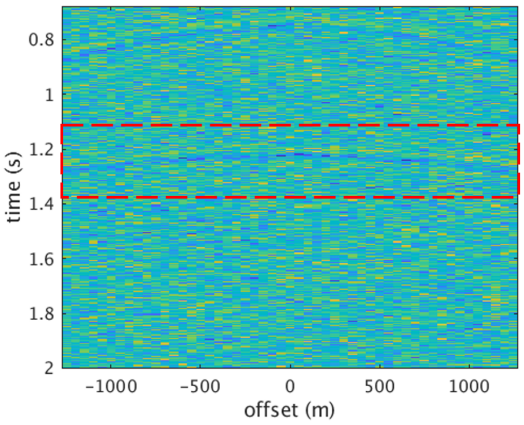


(b) optimization by eGA

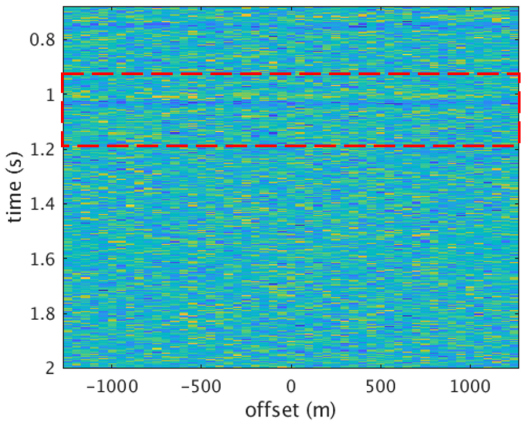
Figure 6-8: CRS stack images after optimization by a) aGA and b) eGA of the noisy CMP gathers with $SNR = 1$.



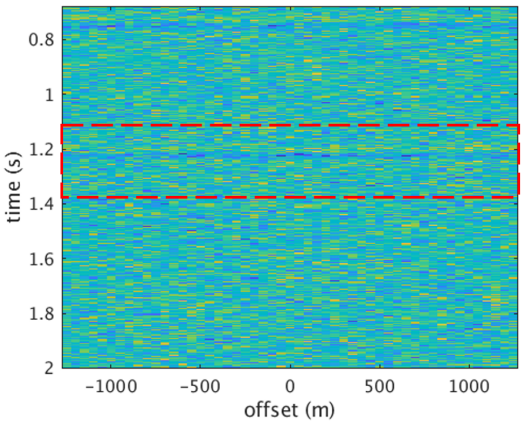
(a) CMP No. 1



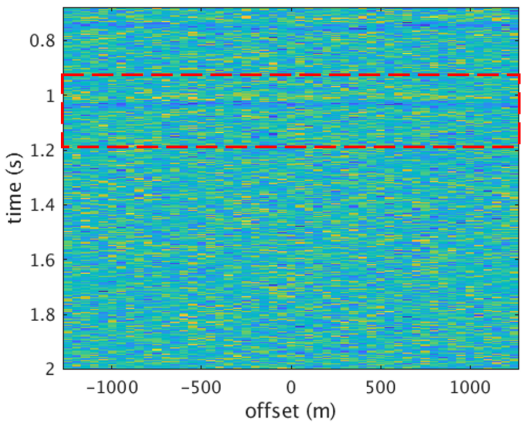
(b) CMP No. 373



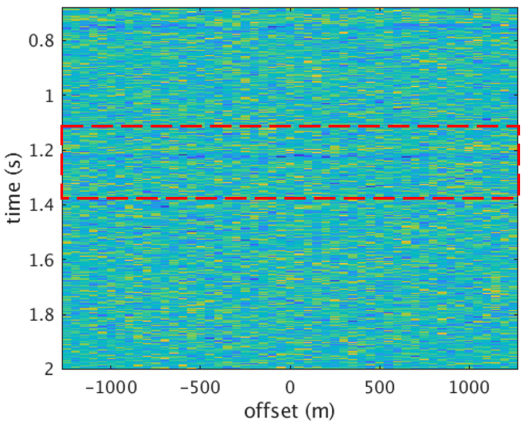
(c) CMP No. 1



(d) CMP No. 373

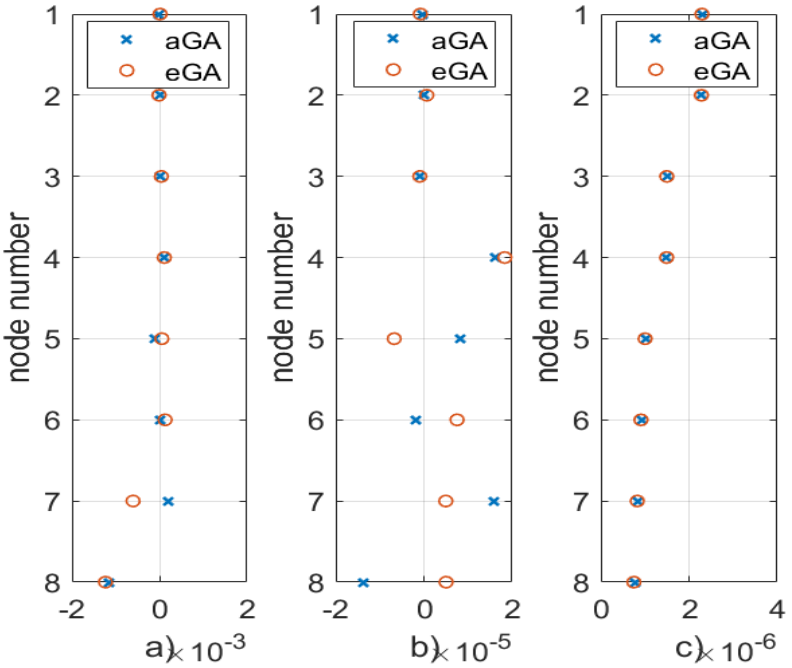


(e) CMP No. 1

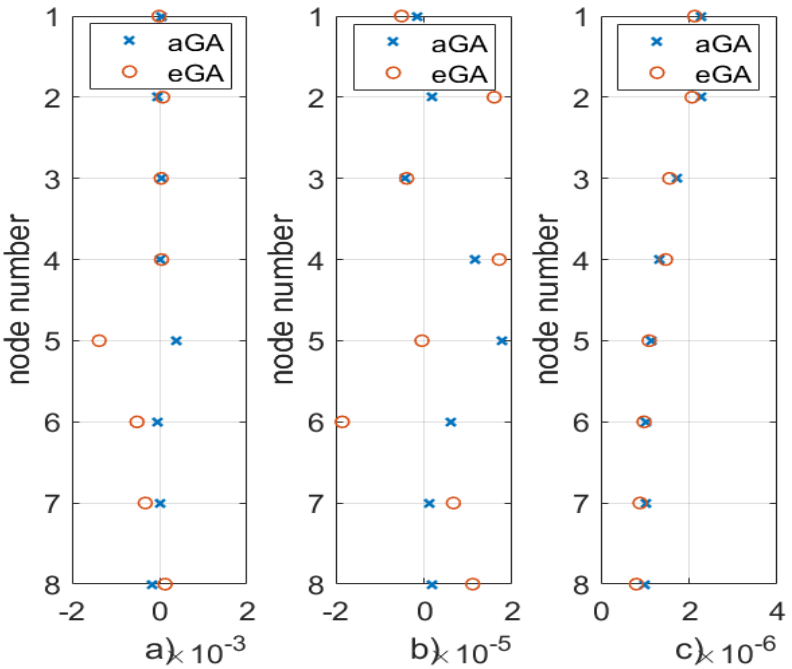


(f) CMP No. 373

Figure 6-9: First (left) and Last (right) CMP gathers of the noisy data set with $SNR = 1$, (a,b) before and after optimization using (c,d) aGA and (e,f) eGA.



(a) CRS attributes 1) A, 2) B and 3) C in Equation 6-5, optimized with the aGA (blue crosses) and the eGA (red circles).



(b) CRS attributes 1) A, 2) B and 3) C in Equation 6-5, optimized with the aGA (blue crosses) and the eGA (red circles).

Figure 6-10: Comparison of the CRS attributes obtained at CMP No. 163 for the noisy data sets with a) $SNR = 3$ and b) $SNR = 1$.

Chapter 7

Discussion and Conclusions

7-1 Discussion

It is imperative to conclude this research work reminding the reader that, based on the No Free Lunch (NFL) theorem, the generalisability of the performance of the eGA, with the parameters here used, on any class of problems is statistically not possible. However, once again, we reaffirm that the improvements achieved with these parameters are instead the result of a timely application of the components proposed in this thesis that simultaneously improved both local and global searches when it was possible.

For the receiver-side statics correction demo, the synthetic data set was constructed from a known set of parameters and statics and the eGA was used to retrieve those time-shifts back, in a data-driven fashion. In doing so, it was key to add appropriate levels of random noise to the data in order to challenge its performance. As a result, the eGA successfully found the statics introduced in the three scenarios presented. However, given the limited time range of this thesis, no formal tests were performed on field data and, as future wok, we propose its application onto real complex structures where, e.g. the statics variation range prevents using conventional cross-correlation based methods due to cycle skipping.

For the CRS stacking demo, only a small subset was actually used for the optimization procedure for two reasons. First of all, to overcome the hurdle of excessive computation time due to CRS stacking calculation being computationally heavy. The second reason was its easiness of implementation. Using CMP gathers containing only short offset traces helped us to avoid handling stretching effects caused during move-out correction; it also allowed us to better meet the CRS approximation adopted for the moveout correction. However, the disadvantage of this choice is the internal multiple information that is wrongly handled and results in a false dipping event in the final CMP stack image.

7-2 Conclusions

The main conclusions of this thesis are:

- This thesis presents an enhanced Genetic Algorithm, which takes advantage of the collaborative evolution of multiple populations communicating from time to time between them to transfer fitter solutions to their neighbors in order to accelerate their evolution process by increasing the overall diversity. Besides, conversely to the aGA, our eGA is also equipped with an advanced local search scheme, the *Self Adaptive Differential Evolution- based fine tuning*, which further improves its local search capability compared to the aGA.
- The improved performance of the eGA has been demonstrated after using it to optimize three challenging multi-modal test functions, where the convergence speed was improved greatly compared to the eGA. Furthermore, it has also been successfully applied onto two non-linear geophysical problems: static correction and CRS stacking optimization, where promising results are obtained.
- The success of the eGA is sensitive to the frequency of application of the SADE fine tuning scheme, the number of islands used and the frequency of communication between them. The fine tuning mechanism can quickly lead to the global optimum only if it is applied at a mature stage of the evolution, meaning that it has to be applied cautiously keeping a good trade-off between both local and global search capabilities. Furthermore, the number of islands used is a selection of the fitness function complexity and the number of dimensions to be optimized for, and its frequency of communication is a way to control the exhaustion of the extra diversity achieved.

Appendix A

In this Appendix, the pseudocode of the Greedy Back and Forth Coordinate Descent local search is provided, as mentioned in chapter 5.

Algorithm 1 Pseudocode for Greedy BFCD

Input: $chromosome_{in}, fitness_{in}, LOW, HIGH$
Output: $chromosome_{out}, fitness_{out}$

$n \leftarrow length(chromosome_{in})$
$Sentinel_{init} \leftarrow 1$
$chromosome_{tmp} \leftarrow chromosome_{in}$
$n \leftarrow length(chromosome_{in})$
while StopConditionNotMet() **do**
    $sentinel_{curr} \leftarrow sentinel_{init}$
    for $i \leftarrow sentinel_{curr}$ **to** n **do**
        for $j = LOW(i)$ **to** $HIGH(i)$ **do**
            $chromosome_{tmp}[i] \leftarrow j$
            $fitness_{tmp} \leftarrow$ Fitness($chromosome_{tmp}$)
            if $fitness_{tmp} > fitness_{out}$ **then**
                $sentinel_{init} \leftarrow i$
                $fitness_{out} \leftarrow fitness_{tmp}$
                $chromosome_{out} \leftarrow chromosome_{tmp}$
            end if
        end for
    end for
    if $sentinel_{curr} == sentinel_{init}$ **then**
        break
    end if
    $sentinel_{curr} \leftarrow sentinel_{init}$
    for $i \leftarrow sentinel_{curr}$ **to** 1 **do**
        for $j = LOW(i)$ **to** $HIGH(i)$ **do**
            $chromosome_{tmp}[i] \leftarrow j$
            $fitness_{tmp} \leftarrow$ Fitness($chromosome_{tmp}$)
            if $fitness_{tmp} > fitness_{out}$ **then**
                $sentinel_{init} \leftarrow i$
                $fitness_{out} \leftarrow fitness_{tmp}$
                $chromosome_{out} \leftarrow chromosome_{tmp}$
            end if
        end for
    end for
    if $sentinel_{curr} == sentinel_{init}$ **then**
        break
    end if
end while

Acknowledgements

. To the only ones deserving all my achievements: ¡papá y mamá!

First of all, I would like to start thanking Shell for fully sponsoring this dream come true! I would also like to thank all the people who have contributed to my thesis. My deepest gratitude to Yimin Sun for his guidance, constructive comments, and many hours of inspirational discussions and to Guy Drijkoningen for his very useful suggestions.

A big thanks to my colleagues in Aramco Overseas Company: Hannes Kutscha, Janny Prins, Rolf Baardman, Rob Hegge, Apostolos Kontakis, Yoo Jewoo and Roald van Borselen, for their support, motivational talks, laughs and, in general, the great working environment. Finally, I could not finish without stressing my gratitude to Roald van Borselen and Yimin Sun for allowing me the opportunity to conduct this research in cooperation with AOC.

Delft University of Technology
Swiss Federal Institute of Technology in Zurich
RWTH Aachen University

Yenni Paloma Villa Acuna
August 10, 2018

Bibliography

Barros, T., Ferrari, R., Krummenauer, R., and Lopes, R. (2015). Differential evolution-based optimization procedure for automatic estimation of the common-reflection surface traveltime parametersevolutionary-based CRS method. *Geophysics*, 80(6):WD189–WD200.

Berryhill, J. R. (1984). Wave-equation datuming before stack. *Geophysics*, 49(11):2064–2066.

Bevc, D. (1997). Flooding the topography: Wave-equation datuming of land data with rugged acquisition topography. *Geophysics*, 62(5):1558–1569.

Biazzini, M., Bánhelyi, B., Montresor, A., and Jelasity, M. (2009). Distributed hyper-heuristics for real parameter optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 1339–1346. ACM.

Brownlee, J. (2011). *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee.

Chinnasri, W., Krootjohn, S., and Sureerattanan, N. (2012). Performance comparison of genetic algorithm's crossover operators on university course timetabling problem. In *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, volume 2, pages 781–786. IEEE.

Cox, M. J., Scherrer, E. F., and Chen, R. (1999). *Static corrections for seismic reflection surveys*, volume 9. Society of Exploration Geophysicists Tulsa.

Douma, H., Haney, M., et al. (2011). Surface-wave inversion for near-surface shear-wave velocity estimation at coronation field. In *2011 SEG Annual Meeting*. Society of Exploration Geophysicists.

Duret, F., Bertin, F., Garceran, K., Sternfels, R., Bardainne, T., Deladerriere, N., and Le Meur, D. (2016). Near-surface velocity modeling using a combined inversion of surface and refracted p-waves. *The Leading Edge*, 35(11):946–951.

Eshelman, L. J. and Schaffer, J. D. (1993). Real-coded genetic algorithms and interval-schemata. In *Foundations of genetic algorithms*, volume 2, pages 187–202. Elsevier.

Gallagher, K., Sambridge, M., and Drijkoningen, G. (1991). Genetic algorithms: An evolution from monte carlo methods for strongly non-linear geophysical optimization problems. *Geophysical Research Letters*, 18(12):2177–2180.

Gomes, C. P. and Selman, B. (2001). Algorithm portfolios. *Artificial Intelligence*, 126(1-2):43–62.

Gong, Y. and Fukunaga, A. (2011). Distributed island-model genetic algorithms using heterogeneous parameter settings. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 820–827. IEEE.

Harik, G. R. and Lobo, F. G. (1999). A parameter-less genetic algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 258–265. Morgan Kaufmann Publishers Inc.

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

Jäger, R. (1999). The common reflection surface stack: theory and application. Master's thesis, Hamburg, University Karlsruhe.

Jäger, R., Mann, J., Höcht, G., and Hubral, P. (2001). Common-reflection-surface stack: Image and attributes. *Geophysics*, 66(1):97–109.

Jones, I. F. (2012). Tutorial: Incorporating near-surface velocity anomalies in pre-stack depth migration models. *First Break*, 30(3):47–58.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks. Proceedings.*

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Koglin, I., Mann, J., and Heilmann, Z. (2006). CRS-stack-based residual static correction. *Geophysical Prospecting*, 54(6):697–707.

Liu, Z., Zhang, J., et al. (2013). Elastic full waveform inversion for near surface imaging in cmp domain. In *2013 SEG Annual Meeting*. Society of Exploration Geophysicists.

Maniezzo, A. (1992). Distributed optimization by ant colonies. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, page 134. MIT Press.

Mann, J., Jäger, R., Müller, T., Höcht, G., and Hubral, P. (1999). Common-reflection-surface stacka real data example. *Journal of Applied Geophysics*, 42(3-4):301–318.

Minato, S., Tsuji, T., Matsuoka, T., Nishizaka, N., and Ikeda, M. (2012). Global optimisation by simulated annealing for common reflection surface stacking and its application to low-fold marine data in southwest japan. *Exploration Geophysics*, 43(2):59–69.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Müller, T. (1998). Common reflection surface stack versus nmo/stack and nmo/dmo stack. In *60th EAGE Conference and Exhibition*.

Nocedal, J. and Wright, S. J. (1999). Springer series in operations research. numerical optimization.

Reshef, M. (1991). Depth migration from irregular surfaces with depth extrapolation methods. *Geophysics*, 56(1):119–122.

Ronen, J. and Claerbout, J. F. (1985). Surface-consistent residual statics estimation by stack-power maximization. *Geophysics*, 50(12):2759–2767.

Selvi, R. M. and Rajaram, R. (2007). Performance study of mutation operator in genetic algorithms on anticipatory scheduling. In *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*, volume 1, pages 511–518. IEEE.

Shtivelman, V. and Canning, A. (1988). Datum correction by wave-equation extrapolation. *Geophysics*, 53(10):1311–1322.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Sun, Y., Tonellot, T., Kamel, B., and Bakulin, A. (2016). A 2D automatic converted-wave static-correction algorithm. *The Leading Edge*, 35(3):280–284.

Sun, Y., Tonellot, T., Kamel, B., and Bakulin, A. (2017). A two-phase automatic static correction method. *Geophysical Prospecting*, 65(3):711–723.

Sun, Y. and Verschuur, D. (2012). Solution to the 3D complex near surface problem by estimation of propagation operators. In *74th EAGE Conference and Exhibition incorporating EUROPEC 2012*.

Sun, Y. and Verschuur, D. J. E. (2014). A self-adjustable input genetic algorithm for the near-surface problem in geophysics. *IEEE Transactions on Evolutionary Computation*, 18(3):309–325.

Sun, Y., Verschuur, E., and Vrolijk, J. W. (2014). Solving the complex near-surface problem using 3d data-driven near-surface layer replacement. *Geophysical Prospecting*, 62(3):491–506.

Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer.

Valenzano, R. A., Sturtevant, N., Schaeffer, J., Buro, K., and Kishimoto, A. (2010). Simultaneously searching with multiple settings: An alternative to parameter tuning for suboptimal single-agent search algorithms. In *Third Annual Symposium on Combinatorial Search*.

Von Neumann, J. (2012). *The computer and the brain*. Yale University Press.

Vrolijk, J.-W., Haffinger, P., and Verschuur, E. (2012). Multi-datum based estimation of near-surface full-waveform redatuming operators. *Journal of Applied Geophysics*, 82:30–45.

Walda, J. and Gajewski, D. (2015). Global optimization of the CRS operator using a genetic algorithm. In *77th EAGE Conference and Exhibition 2015*.

Waldeland, A. U., Zhao, H., Faccipieri, J. H., Schistad Solberg, A. H., and Gelius, L.-J. (2017). Fast and robust common-reflection-surface parameter estimation. *Geophysics*, 83(1):O1–O13.

Whitley, D., Rana, S., and Heckendorn, R. B. (1999). The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7(1):33–47.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on Evolutionary Computation*, 1(1):67–82.