IMPROVING GENETIC ALGORITHMS FOR OPTIMUM WELL PLACEMENT

A REPORT SUBMITTED TO THE DEPARTMENT OF ENERGY RESOURCES ENGINEERING

STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

By
Mohammad Moravvej Farshi
June 08

I certify that I have read this report and that in my opinion it is fully adequate, in scope and in quality, as partial fulfillment of the degree of Master of Science in Petroleum Engineering.

_____

Prof. Khalid Aziz

(Principal Advisor)

Abstract

Optimum well placement can help reservoir management teams in developing a field development plan that could result in substantial increase in productivity and profitability of any new or existing field. The proposed location and configuration for new producers and injectors is usually nontrivial, due to the complexity of the fluid flow in highly heterogeneous reservoirs.

The objective of this work was to understand the steps involved in the optimum well placement by GAs, and to introduce enhancements to the algorithm that could increase the possibility of obtaining promising solutions. Based on the success of binary GAs in optimum well placement problems and motivated by the advantages observed in application of continuous GAs in other fields, here we attempt to use continuous GAs in for field development.

To meet our objectives, we have investigated the design of continuous GA that retain the important benefits characteristics of binary GAs, while solving some of the problems associated with binary GAs. The implementation of continuous GA was designed to avoid generating invalid wells during the reproduction process. Continuous GAs have shown considerable potential to achieve higher fitness values. The gradual progress of a continuous GA during the generations, compared to the stepwise evolution observed in a binary GA, has the possibility of achieving more desirable outcomes. However, the design of a powerful optimization tool with continuous GAs is harder because of the higher number of GA parameters involved.

The study also implemented dynamic mutation to take advantage of the exploring capacity of mutation in each period of the evolution. Furthermore, it has been shown that the efficiency of the GA search can be increased by imposing a minimum Euclidian distance between the individuals within the population, utilizing our engineering knowledge by requiring a minimum physical distance between all the wells in a reservoir.

Finally, a model was introduced to include curved wells during the search. Through this model the possibility of capturing straight wells still exists, while providing the opportunity of exploring more promising configurations.

Throughout this work we have demonstrated that various improvements introduced lead to finding higher objective values in shorter time.

Acknowledgments

I would like to acknowledge Professor Khalid Aziz, my research and academic adviser for his continuous support and guidance during my years at Stanford. I am thankful for his valuable suggestions and comments that were essential to the completion of this work. I would also like to thank Professor Louis J. Durlofsky, and the other faculty members of the Energy Resources Engineering Department for providing useful comments.

I would express my gratitude to Dr. Martin Mlacnik, my supervisor during my internship at Conoco Philips, for the valuable insight he gave me. My appreciation also goes to Marie Ann Giddins and Dr. Chris L. Farmer for their helpful comments during my internship at Schlumberger, Abingdon Technology Center.

Also, I would like to acknowledge Jerome Onwunalu's help in introducing me to the subject, and for facilitating my research through valuable discussion.

I am grateful for the financial support for my study provided by the SUPRI-HW research consortium on Advanced Wells.

My special appreciation goes to my wife, Hosniyeh Nekoofar, for her consistent support and patience through the course of my study. I also appreciate the support I have received from my family and friends.

Above all I would like to thank God for his guidance throughout my life.

Contents

List of Tables

List of Figures

Chapter 1

1. Introduction

1.1. General Background

"The main task of a reservoir engineer is to develop a scheme to produce as much hydrocarbon as possible within economic and physical limits" (Bittencourt, 1997). Optimum well placement can help reservoir management teams in the preparation of a field development plan that could bring substantial increase in productivity and profitability of an existing or new field.

Using conventional reservoir management methods, only about 10 percent of a reservoir's original oil in place is typically produced during primary recovery (i.e. through natural drive) and the average secondary recovery (i.e. injection of water or gas) reaches 20 to 40 percent of the original oil in place (DOE, 2008). With the oil prices booming, development of any new method that could help the management team to increase the productivity and profitability of the reservoir is highly desirable.

With advances in drilling technology, drilling of wells with arbitrary trajectories and multiple branches, knows as multi lateral wells (MLWs), is becoming routine. Although, drilling of a MLW is more expensive in the first place, it can be more cost effective after some production period. Since, petroleum reservoirs are complex heterogeneous environments with possible shale layers, faults, fractures, low permeable and even depleted regions; the branches of a MLW are capable of penetrating through the most productive part of the reservoir that could result in substantial increase in the field recovery factors. However, MLWs could only enhance the recovery if they have the right configuration and are located in the right place.

The current practice in the industry is to design several well configurations by intuitive judgment of experienced reservoir engineers, and then perform reservoir simulation and

economic analysis studies on pre-selected scenarios to find out the most efficient one. However, due to the nonlinearity of the problem and complexity of interactions between branches and reservoirs, there is a very low chance for an intuitive well design to be the most efficient scenario. This promotes utilization of optimization methods for the well placement problem.

An optimization problem defines, an "objective function" whose value should be optimized. The "input variables" for the objective function are selected to optimize its value. In the optimum well placement problem the objective function is usually the project's net present value (NPV), or cumulative field production during some specific period. The input variables consist of the parameters describing possible locations, configurations, and control criteria for the new wells to be drilled in the development plan. Numerical reservoir simulation has to be performed for predicting the production profile of the field for use in evaluating the objective function. However, as each numerical simulation is computationally expensive, surrogates could be used for providing an approximate value of the objective function.

1.2. Literature Review

Previous research in optimization of well placement and field development could be categorized in the following three areas:

1- Construction of well optimization frameworks and algorithms;

2- Designing proxies to accelerate the optimization process; and

3- Assessing the uncertainty and incorporating it in the optimization routine.

Some of the works have focused on one of the above categories, while others have attempted to come up with solutions to more than one of the above issues. Also, the parameters to be optimized in some earlier works were well rates or locations (assuming only vertical wells). However, in some more recent works well shapes and production

scenarios (with multiple producers and injectors) were optimized in addition to well locations.

Beckner and Song (1995) devised an optimization framework with simulated annealing to propose the placement of a sequence of production wells maximizing the net present value for full field development. Their program searched for 12 horizontal wells in the mid layer of a 36×3×3 model. The wells penetrated through all 3 grids in the *y* direction. So, the well was only modeled by its grid node in the *x*-direction, and the year it would be placed. They framed the well scheduling and placement problem as a "traveling salesman problem", and used numerical simulation to calculate the objective function. They concluded that uniform well spacing does not generally maximize NPV, particularly for phased development in heterogeneous reservoirs.

Montes et al. (2001) used standard genetic algorithms (GAs) to optimize the placement of vertical wells penetrating through the whole vertical domain, having total filed oil production as the objective. They introduced a pointer to map the well positions from 2D to 1D to further simplify the problem. Also, they used short term simulations to estimate long term runs to speed up the optimization.

Bittencourt and Horne (1997) developed a hybrid GA for optimizing placement of vertical or horizontal wells in a 2D reservoir. Each well was modeled by the three input parameters: well location, well direction (vertical or horizontal), and horizontal well orientation (N, NE, E …). They applied an indexing system for well locations that only included active cells. They combined GA with the Polytope method to benefit from the best features of each method and speed up the search. They also integrated economic analysis and some practical design consideration in their optimization algorithm.

Yeten et al. (2002) designed a well placement optimization framework with GAs to search for optimum multilateral wells in a 3D reservoir. They introduced a parameterization technique that models multilateral wells for GA optimization. Their optimization framework also handles variable number of producers and injectors. Since the number of variables increases in optimization of multilateral wells, they also

examined use of an artificial neural network, a hill climber, and a near-well upscaling technique to accelerate the search. They also included the effect of geological uncertainty, by running the numerical simulation with several realizations for each set of wells and averaging the NPV.

Rigot (2003) has introduced an iterative approach to improve the efficiency of multi well placement optimization by dividing the original problem into several single well optimizations.

Pan and Horne (1998) used Least Squares and Kriging as proxies to reservoir simulation. They selected a number of sample well locations for numerical simulation using "uniform design", which was developed by Fang (1980). Then Least Squares and Kriging were applied to generate NPV surface maps, which were used in estimating the objective function values at the new points. They observed that the objective functions estimated by Kriging are more accurate than those estimated by Least Squares interpolation.

Guyaguler et al. (2000) used a hybrid optimization technique based on the GAs and Polytope algorithm, to find optimal locations and rates for vertical wells in a water flooding project in the Gulf of Mexico. They used Kriging and neural networks as proxies. They proposed adding local mutation which perturbs the best solution in each generation within a given range.

Onwunalu (2006) applied a statistical proxy based on cluster analysis into the optimization process for nonconventional wells. He used the multilateral well model introduced by Yeten et al (2002). He also extended the proxy to perform optimization of multiple nonconventional wells opened at different times.

Guyaguler and Horne (2001) designed a framework to quantify risk attitude through utility functions and transform the uncertain well placement problem to a deterministic problem. They used a hybrid GA as the optimization engine.

Ozdogan and Horne (2006) suggested coupling well placement optimization with recursive history matching to address the value of time-dependent uncertainty. They showed that the utility of the optimized scenario could be improved by including time-dependent uncertainty during well placement, through the use of a "pseudohistory" concept.

1.3. Problem Statement

As we have seen in the previous section, many studies have been performed on the well placement optimization problem with various optimization methods. Among those, utilizing GAs as the main search engine has been shown to be more promising than greedy optimization methods.

Well designed GAs have shown the capability of handling highly multimodal functions that are hard to attack by other optimization methods. However, because of the high dimensionality of optimization space, caused by the number of parameters needed to describe each nonconventional well and the number of injector and producers, the well placement optimization problem is still challenging and computationally expensive. The GAs used in previous optimum well placement works were all binary.

There have been many studies recommending the use of continuous (or real-valued) GAs, instead of binary GAs, for optimizing variables with inherently continuous domains (Deb and Agrawal 1995; Herrera et al. 1998; Lee et al. 1999; Chelouah and Siarry 2000; and Harikumar et al. 2004). Each work has proposed different operators for continuous GAs to achieve a robust tool for global optimization over complex multimodal function with continuous variables.

Use of continuous GAs for solving problems with continuous search spaces, could overcome issues involved in the coding and decoding of binary GAs, such as "deception", that results in premature convergence to a suboptimal solution (Michalewicz, 1994), and "Hamming cliffs", that makes gradual search over continuous space difficult (Deb and Agrawal, 1995). The other benefit that arises from the use of continuous GAs as function

optimizers is in achieving high precision for representing candidate solutions without increasing the computation burden. In binary GAs the string length chosen for binary coding of each variable results in certain precision of the variable representing the solution. To gain higher precision, a higher string length is needed resulting in a long chromosome. Longer chromosomes require larger population size resulting in high computational complexity (Goldberg et al. 1992).

As it can be observed in results of the previous works in well optimization such as those in Yeten (2003), the optimum well location and configuration proposed by well placement optimization programs could be somewhat nonintuitive. This provides motivation for the design of a general optimization framework that could be used to determine the optimal location and configuration for the new wells to be deployed. Based on the success of binary GAs in well placement optimization problems and motivated by the advantages observed in the implementation of continuous GAs in other fields, here we will attempt to use continuous GAs for field development.

The objective is to design a continuous GA that would benefit from the positive characteristics of binary GAs, while solving some of the problems related to them. The implementation of continuous GA was designed to avoid generating invalid wells during reproduction.

The outline of this report is as follows. In Chapter 2 we first discuss different optimization methods and their characteristics. Then we focus on GAs and compare binary and continuous GAs. In Chapter 3 we investigated all the steps in well optimization by GAs and look for possible improvement. In Chapter 4 we introduce a modified model for representing multilateral wells in well placement optimization by GAs. In Chapter 5 we present the improvements we have proposed for well placement optimization framework with GAs. Finally, in Chapter 6 we bring our conclusions and make some recommendations for future works.

Chapter 2

2. Optimization Tools

2.1. Optimization

The objective of optimization is to come up with the most efficient solution to existing problems. In mathematical terminology, it refers to systematic methods of finding the values of the variables, within the problem limits, that minimize or maximize the value of a user defined the objective function. The minimization problem can be represented in the following way:

$$\begin{aligned} &\text{Finding } x^* \in \Omega, \qquad \text{Such that } F(x^*) \le F(x), \qquad \text{For all } x \in \Omega \\ &F : \Omega \subset R, \qquad x = (x_1, ..., x_n) \end{aligned} \tag{2-1}$$

Here, the vector x^* minimizes the value of function $F(x)$ for the constraint $x \in \Omega$.

Optimization methods can be categorized in the following groups (Nocedal and Wright, 1999):

- Continuous or Discrete
- Constrained or Unconstrained
- Global or Local
- Stochastic or Deterministic

2.1.1. Continuous or Discrete

The objective function that is subject to optimization could be inherently defined on a continuous or discrete domain. Continuous optimization methods search for the optimum input variable within a continuum of eligible variables. On the other hand, discrete optimization methods are designed to explore a limited domain of enumerable discrete variables. Naturally, continuous functions can also be discretized and approximated with

finite separate values, to be suitable for taking advantage of discrete optimization methods (Nocedal and Wright, 1999).

2.1.2. Constrained or Unconstrained

Optimization problems of the form (2-1) are categorized as unconstrained optimization problems when $\Omega = R^n$, and as constrained optimization problems when $\Omega \subset R^n$, but $\Omega \neq R^n$. Physical problems generally have constraints on their input variables. However, since unconstrained optimization problems are usually simpler to implement, there are some cases where unconstrained optimization methods are used on a problem with natural constraints. The first group, are problems where the constraints will not affect their optimum solution. In the second group, the constraints are implemented by adding penalty terms to the original objective function, for the inputs not honoring the constraints.

2.1.3. Global or Local

Local optimization techniques such as Steepest Descent, Quasi-Newton, and Conjugate Gradient, converge to a local minima or maxima in the function domain, depending on the starting point. These methods are fast in converging to local extrema as they take advantage of the solution space characteristics. However, their dependency on the solution space and the initial guess limits their application for non-smooth and multimodal objective functions (Abo-Hammour, 2002). A local minimum is a point for which the objective function is smaller than those of the neighboring points. The function F has a local minimum at x^* if there exist a neighborhood around x* that for all x in that neighborhood $F(x^*) \leq F(x)$:

$$\exists \varepsilon > 0, \text{ That } F(x^*) \leq F(x), \text{ For} \forall x, \text{ When } \left| x - x^* \right| < \varepsilon \qquad (2\text{-}2)$$

For a differentiable function F the value of gradient at local optimum at x^* is zero. In convex systems a local optimum would also be the global optimum of the system (Nocedal and Wright, 1999). However, most practical problems are multimodal. Hence, finding the global optimum is not easy. This is due to the large gap between the necessary

conditions for optimality and the known sufficient conditions for global optimality (Neumaier, 2004). Global optimization approaches include branching algorithms such as "branch and bound methods", Monte-Carlo-based algorithms such as simulated annealing, and stochastic tunneling, and heuristic methods such as Genetic Algorithms, and particle swarm and ant colony optimization (Wikipedia, 2008).

2.1.4. Stochastic or Deterministic

Stochastic optimization methods are algorithms in which random choices are made in the search direction as the algorithm iterates toward the optimum. The randomness added to the search process can help with convergence of the optimization algorithm by allowing surprise moves to unexplored areas of the search domain that could possibly contain an unexpected good solution (Spall, 2004). Even with the same initial guess, stochastic optimization methods will introduce a new path toward the optimum each time the search is performed. In contrast, in deterministic optimization methods all the steps are the same every time the algorithm is used with a similar starting point. Stochastic optimization algorithms have been growing rapidly over the past decade with a number of methods now becoming "industry standard" approaches for solving the challenging optimization problems (Spall, 2004).

2.2. Genetic Algorithms

Genetic algorithms (GAs), originally introduced by Holland (1975), are biologically motivated stochastic population-based search techniques, built on the principles of natural selection and genetic recombination. The appeal of GAs comes from their simplicity and robustness as well as their power to discover good solutions for complex high-dimensional global optimization problems that are very difficult to handle by more conventional techniques (Forrest and Mitchell, 1993).

2.2.1. Survival of the fittest

One of the main characteristics of GAs that differentiate them from other search techniques is their ability of handling a population of potential solutions, rather than

modifying a single point. Through performing optimization simultaneously at different regions in the problem domain, a GA offers higher chance of discovering very good results. GAs obtain their ability to recognize trends toward optimal solutions by combining the principles of survival of the fittest with randomized information exchange (Baluja, 1994).

The implementation of a genetic algorithm starts with generating a population of possible solutions. The proposed solutions are parameterized and the variables identifying each of them are put together to create a "chromosome," representing that solution in the evolution process in the GA. The next step is evaluating the "fitness" of each chromosome, and allocating selection probabilities in a way that chromosomes representing better solutions are given more chance to be selected, as "parents", for reproduction. Then the selected chromosomes are paired for recombination. The "recombination" operator combines the information of each pair of selected parents and generates new "offspring" chromosomes, which create the next generation. The chromosomes evolve over a number of generations, guiding the GA search toward an optimal solution.

2.2.2. Diversity

Although the chromosomes representing better solutions have higher chance to be selected for reproduction, other chromosomes would also have the likelihood of appearing in the parents' pool. Moreover, the offspring population generated by recombination would not necessarily have a higher fitness values than their parents, due to the stochastic nature of the selection and recombination operators.

This creates a mechanism for the GA to keep the diversity of its proposed solutions during the evolution of generations; that is a key to its extensive global search power. Selecting merely good solutions in each generation and combining them to form the next generation, results in the population of solutions to rapidly become trapped in a confined region of the domain. This would prevent the algorithm from an extensive search and converging to a possibly suboptimal solution. The other mechanism for maintaining the

diversity of the candidate solutions over the generations is the introduction of limited random alteration of the chromosomes throughout the evolution. Adding this stochastic random alteration, called "mutation", brings up the possibility of exploring new regions that could contain unpredicted exceptional solutions. It also helps the searching power of the GA by pushing the population away from to local optimuma.

2.3. Binary GAs

2.3.1. Representation

An important characteristic of all optimization methods is the way they represent possible solutions to the problem. In the standard form of GAs, Canonical Genetic Algorithms, each candidate solution is represented by a binary string of length L which is referred to as a "chromosome". The chromosome is created by putting together the binary encoded form of all parameters characterizing the possible solution. Figure 2-1 demonstrates the way that a chromosome corresponding to the solution vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ is created in binary GAs.

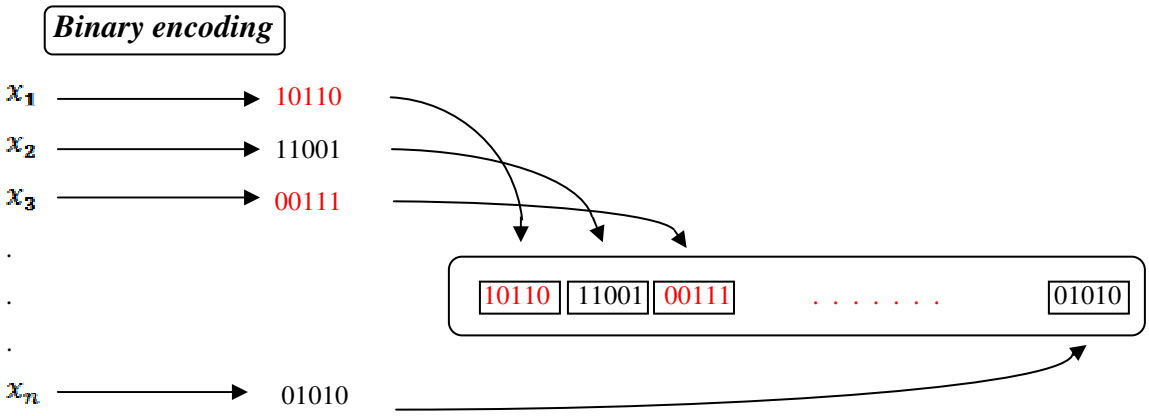

Figure 2-1: Creation of the chromosome in Binary GAs

Due to the binary structure of the chromosomes the algorithm searches within a finite parameter space. This attribute makes binary GAs ideal for optimizing functions whose parameters have limited number of states (Harikumar et al., 2004). However, most real world functions work with real value parameters in a continuous space. Therefore, before performing binary encoding each parameter has to be "quantized" to a limited number of

values. Three values are needed for each parameter. The first two, are the minimum and maximum allowed values of that parameter in the model. These values can represent the physical range in which the parameter can exist (actual limits) or the parameter range in which the objective function could have a good value (imposed limits). The third value is the desired length for the binary encoding of the parameter. Then each quantization level can be determined by:

$$\Delta x_i = \frac{(x_i^{\max} - x_i^{\min})}{2^n} \tag{2-3}$$

where "n" represents the binary encoding length. Figure 2-2 illustrates quantization of sin(x) function for the case n=4:

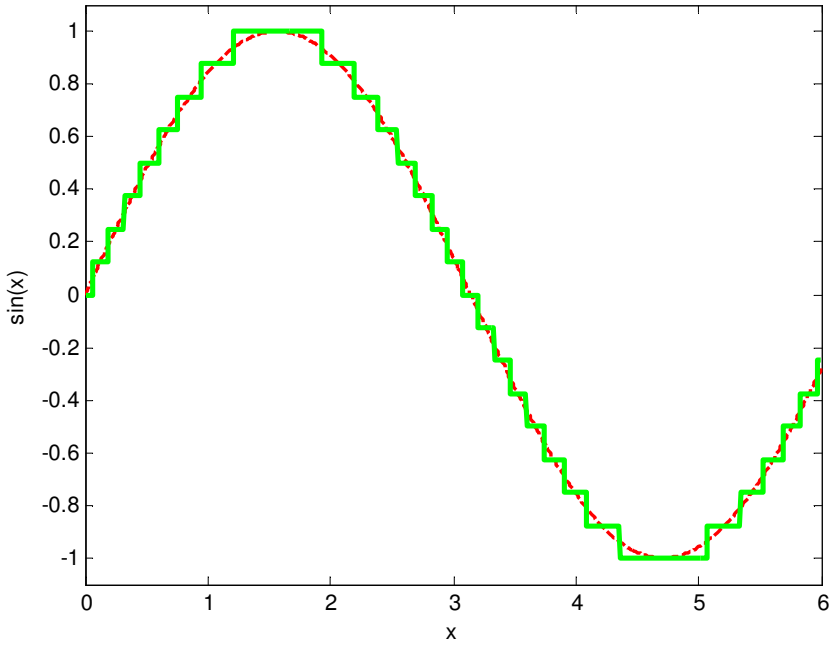

Figure 2-2: *sin(x)* function and its quantization to 4 bits

For having smaller quantization levels and thus more complete coverage of the parameter range, larger binary encoding lengths should be chosen. However, allotting larger binary strings to each parameter will result in longer chromosomes that will slow the convergence of binary GA search. So, in a binary GA after generating the initial

population and evaluating the fitness of the individuals, they need to be coded to binary chromosomes to be able to go through the binary GA reproduction process. Furthermore, after the production of the next generation the binary chromosomes need to be decoded. This step is essential as the actual value of parameters that define the solution is required for calculating their fitness. Figure 2-3 gives the flowchart for a binary GA:

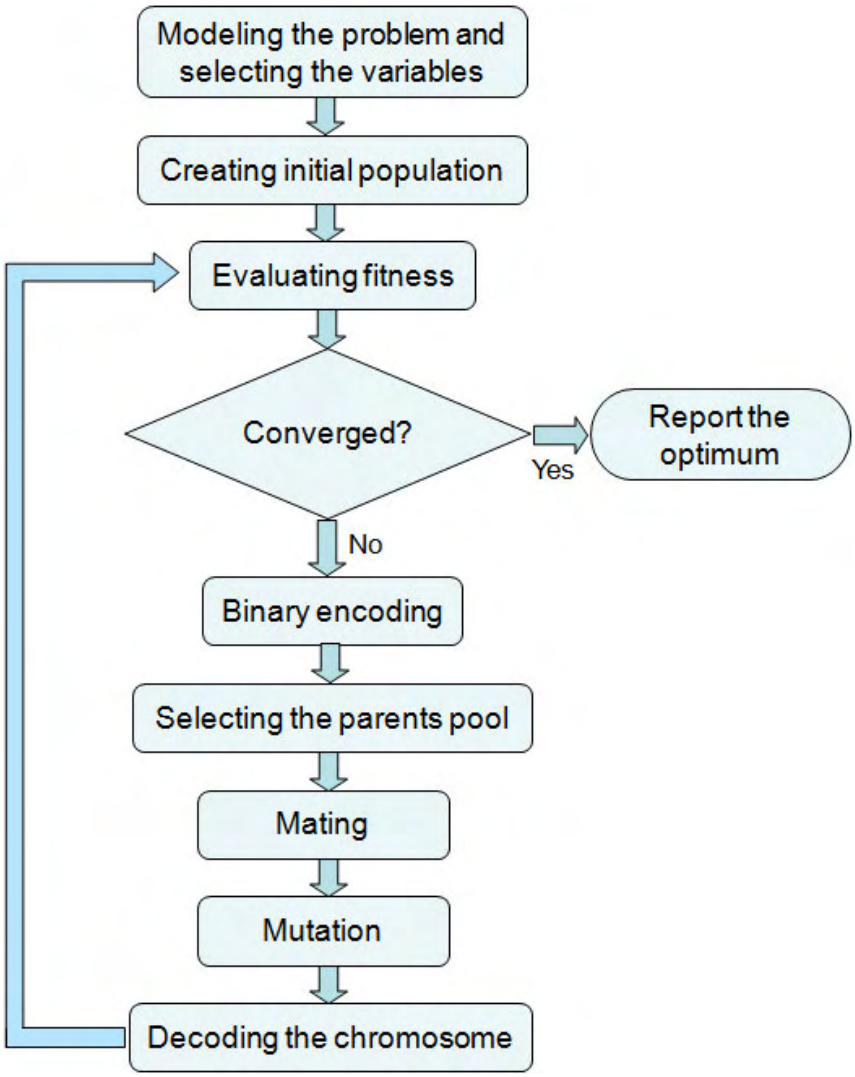

Figure 2-3: Flowchart for the use of binary GA

What is special about binary GAs is the simple and straightforward implementation of the bitwise mutation and crossover, as reproduction operators. In the next section we explain reproduction operators used in binary GAs.

2.3.2. Reproduction Operators

2.3.2.1. Crossover

Crossover or mating is the GA operator that attempts to mix each pair of chromosomes selected as parents, to create the likelihood of keeping the good properties of each parent chromosome in the offspring chromosome. The crossover operator in binary GAs is implemented by cutting some part of each parent chromosome and replacing it into the other parent chromosome. This operator is implemented in several works in the literature. The simplest crossover implementation that is used in Canonical GAs is single point crossover. In this method one random point in the chromosome is selected and with some predetermined probability, called crossover probability, the rest of the chromosome string after that point is swapped between two parents (Figure 2-4).

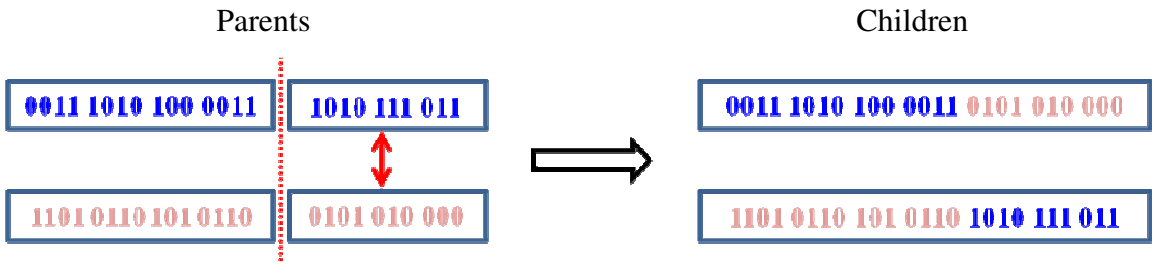

Figure 2-4: Single point crossover in binary GAs

Other crossover methods involve selecting two or more random crossover points within the chromosomes, and random swapping of the strings between those points. These methods are called two-point crossover or multi-point crossover considering the number of crossover point selected over the entire chromosome length. The idea for these crossover operators has been taken from the crossover in a standard GA. However, the chromosomes split up into more than two pieces in multi-point crossover. Uniform crossover is a special case of multi-point crossover in which the possible number of breaking points in the chromosomes is equal to $\boldsymbol{n-1}$ ($\boldsymbol{n}$ being the length of the chromosome). Therefore, in uniform crossover each bit can swap between the parent chromosomes. Both methods are illustrated in Figure 2-5 (Initially the top parent chromosome was in blue and the bottom parent chromosome was in pink).

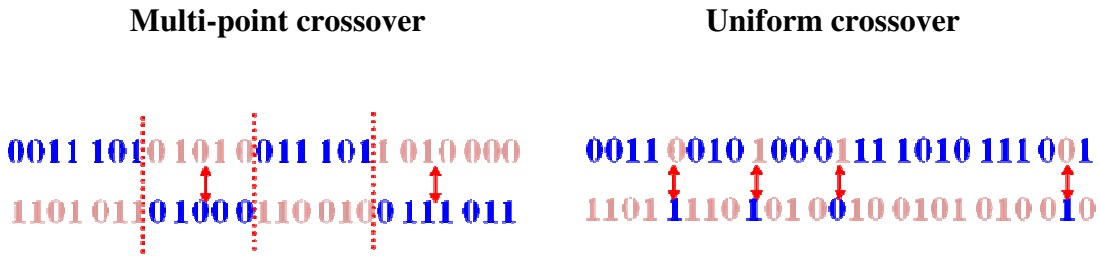

Figure 2-5: Multi-point crossover and uniform crossover in binary GAs

2.3.2.2. Mutation

The role of mutation operator is to keep the diversity of the population during the evolution process. In a binary GA it is modeled by simply altering the value of some bits in the chromosome. The value of any bit in the chromosome can be changed to its complement with probability of P_{mut}, assigned as the mutation probability of the GA. This probability is usually selected to be small ($P_{mut} \ll 1$). One way to implement this operator is by allocating a random variable to each bit of the chromosome. Each random variable determines if the value of that bit is going to be changed. Figure 2-6 shows the mutation in a binary GA:

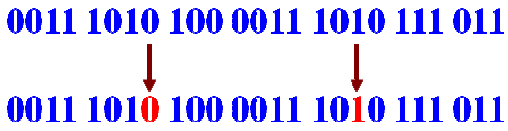

Figure 2-6: Mutation in binary GAs

2.4. Continuous GAs

2.4.1. Representation

The chromosomes in binary GAs were generated using binary representation of variables describing the possible solutions. However, binary is not the only method for encoding problem solutions with a continuous domain. Various representations using different alphabets for encoding have been tested in the literature (Liepins & Vose, 1990). Many studies recommend the use of real numbers, instead of binary coded values to represent

the possible solution chromosomes for optimizing functions with a inherently continuous domain (Deb and Agrawal 1995; Herrera et al. 1998; Lee et al. 1999; Chelouah and Siarry 2000; Harikumar et al. 2004). This family of GAs is called continuous GAs or real-valued GAs. In continuous GAs a chromosome representing a candidate solution is created by putting together the parameters defining the solution in their actual form. Evidently they are in fact being represented by the computer-based floating-point representation of real numbers. A comparison of binary and real-valued chromosomes representing a multilateral well for well the placement optimization problem is given by shown in Figure 2-7 (a chromosome in continuous GAs) and Figure 2-8 (a chromosome in binary GAs).

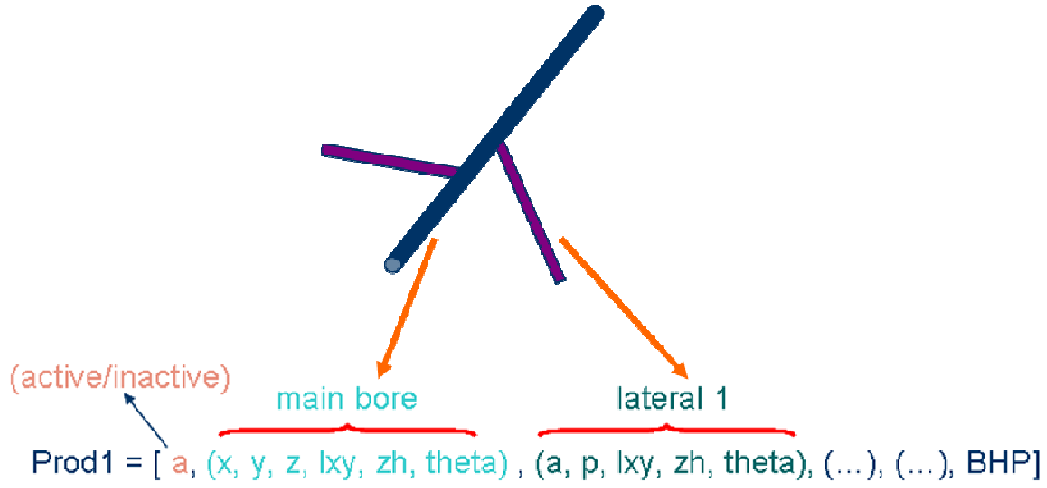

Figure 2-7: A chromosome in continuous GAs for a multilateral well placement optimization

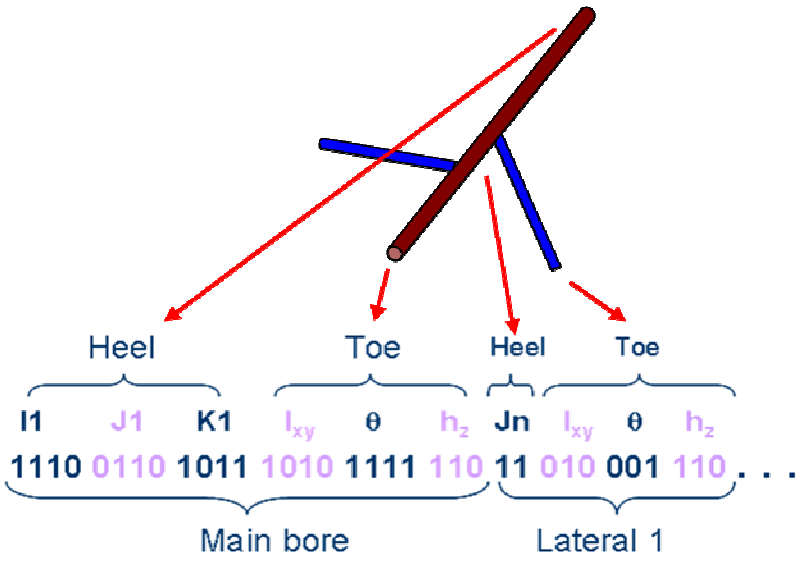

Figure 2-8: A chromosome in binary GAs for multilateral well placement optimization

2.4.2. Reproduction Operators

2.4.2.1. Crossover

Crossover operator is believed to be the main operator that creates the search power of GAs in optimization problems (Goldberg, 1989). During the evolution, the crossover operator has two roles. At first, it searches the initial random strings containing problem variables to come up with a good solution. Then, in the later stages its role is to combine good portions of these strings to form even better solutions. Since the use of real-value representation was proposed for GAs, many crossover operators have been introduced. Crossover could be defined similar to its form in binary GAs, by only swapping some values between two parents, but they generally would not lead to satisfactory results. The reason is that using this type of crossover in continuous GAs will result in propagating values generated for each parameter in the initial population to next generations but only in different combinations. However, there would be no new value introduced through crossover for any parameter. This would decrease the search power of the GA. The rationale that makes this type of crossover work well for binary GAs is that by only swapping some parts of the strings in binary chromosomes there remains the possibility that the coded form of a parameter is broken and new values introduced (Figure 2-9).

Figure 2-9: Crossover with only swapping in continuous and binary GAs

Blending methods are introduced to overcome this problem. They present a way to combine values of each variable in the parent chromosomes and to introduce new values for that variable in the next generations. Radcliff (1991) used the following blending method to generate a new offspring variable value, P_i^{new}, from a combination of corresponding variable values in parent chromosomes.

$$P_i^{new} = \beta\ P_{Mi} + (1-\beta) P_{Fi} \tag{2-4}$$

where P_{Mi} is i^{th} variable of mother's chromosome, P_{Fi} is i^{th} variable of father's chromosome, and $0 \leq \beta \leq 1$ is a random variable.

Using this blending method the offspring variables inherit that property from their parent's variables, and their value always fall between the values in parent's variables. As an example, consider the chromosome representing solution scenarios for well placement optimization problem in which the mainbore length of the second producer is 200ft in the mother chromosome and 180ft in the father chromosome. Through this blending method mainbore lengths of second producer for the offspring chromosomes could turn out to be 185ft and 195ft. Michalewicz (1994) has showed this method to work well on several problems.

However, it remains to be shown how to choose variables for blending. We have chosen to mix blending with uniform crossover. In this way each variable in the chromosome could be combined with its counterpart with some predefined probability, $\boldsymbol{P}_{xo}$. The blending random variable, β, could be generated once for all blendings in each crossover operation, or different β's for each variable blending within the crossover. We decided to

generate different random variables for all variable blending within a crossover to combine the information of the parents effectively.

Since, applying these blending methods does not result in introducing values beyond the extreme values of that variable in the initial population, some extrapolating blendings have been suggested in the literature. However, these methods could generate value outside the acceptable range for a parameter. Then the offspring must be discarded and another β selected. Eshelman and Shaffer (1993), presented (BLX-α) method that limits how far each offspring variable could fall outside the range of its parent variables. The problem with extrapolating blending methods is that the second role of crossover, that is combing good properties of the parent chromosomes, is undermined. This problem could be solved by using extrapolating methods only in the early stages of evolution or using small α's in the (BLX-α) method. Adewuya (1996) designed a crossover method performing quadratic fit to the fitness function over variable values from three parent chromosomes. In this work, we will use simple blending and gave the extrapolating role only to mutation.

2.4.2.2. Mutation

Mutation operator has two important roles during the evolution process in a GA. Its first role is to introduce unexplored genetic material to the population. Its second role is to maintain the diversity of the candidate solutions in a population over the generations, preventing premature convergence of the GA to suboptimal solutions.

Mutation in binary GAs is rare compared to crossover, resulting in low mutation probability. Also, since the mutation probability for all the bits in a binary string is the same, it will lead to higher possibility of small changes and lower possibility of large changes in the variables value. For example one can consider a binary string of length 10 representing value of a variable in the chromosome. For mutation probability of 0.1, there would be 50% likelihood of alteration in the lower half of the string which could result in a change of less than 1/32 (about 3%). However, there is only a 10% chance of alteration of the last bit resulting in chances greater than 50%. This can be observed in Figure 2-10.

One way to implement mutation in continuous GAs is to make it work similar to binary GAs. In this way if variable, P_i, in the chromosome has been chosen for mutation, a random number P is selected between 0 and 1. Then using a graph similar to the one in Figure 2-10, the parameter F_{mut} is selected and the variables new value would be, P_{im}:

$$P_{im} = P_i^{new} + F_{mut}\,(P_i^{\max} - P_i^{\min}) \tag{2-5}$$

where P_i^{new} is i^{th} variable of offspring chromosome chosen for mutation, P_i^{max} is upper limit for variable P_i, P_i^{min} is lower limit for variable P_i, and F_{mut} is mutation coefficient selected from the graph using random variable $0 \le P \le 1$.

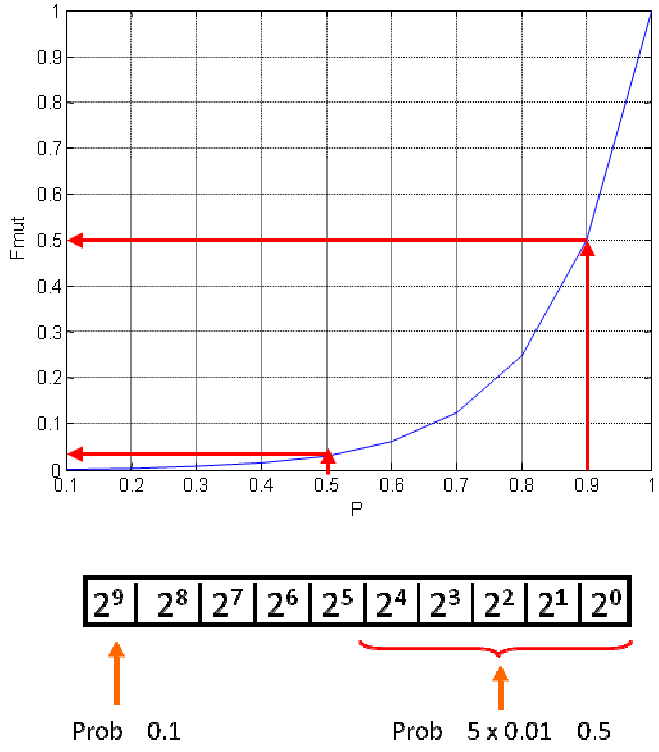

Figure 2-10: Building continuous mutation similar to binary mutation

Another way of implementing mutation in continuous GAs, is to add a normally distributed random number to the variable selected for mutation (Haupt and Haupt, 2004), (see Figure 2-11).

$$P_{im} = P_i^{new} + \sigma\, N(0,1) \tag{2-6}$$

where σ is the standard deviation of the normal distribution, and $N(0,\ 1)$ is a standard normal distribution with mean of 0 and variance of 1.

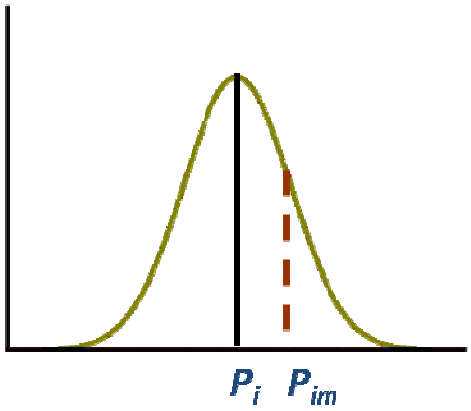

Figure 2-11: Implementing mutation in continuous GAs using normal distribution

With this implementation of mutation, σ is the parameter that is used to control the effect of mutation, beside the mutation probability. This parameter should be determined carefully for any specific problem. Larger σ's will allow larger change in variable value due to mutation.

2.4.2.3. Advantages of Continuous GAs

Following a review of binary GAs and continuous GAs and their operators, we will now briefly mention some of the deficiencies of binary GAs that are eliminated using continuous GAs. Also we will discuss the benefits of the using of continuous GAs in well placement optimization problems.

One of the advantages of the continuous GAs over the binary GAs is their high precision in representing possible solutions without requiring the use of extra long chromosomes, which increase computational complexity. For comparison we look at the representation of the property "angle", θ_x, in well placement optimization problem, in binary and continuous GAs. As it can be seen in Figure 2-12, using binary GAs with 4 bits to

represent an angle, the values of θ_x in an optimal case could fall at specific locations. However, by using continuous GAs θ_x can fall anywhere within the valid range. While precision in binary GAs could be achieved by assigning higher string length representing each variable value this results in a long chromosome and hence a slower convergence. Moreover, continuous GAs do not suffer from under sampling in the search towards the optimum. In binary representation the optimum value of a parameter (that in not known initially) may lie between the pre-specified coded values. In this case, binary GAs would miss the optimum value due to their under sampling of the parameter space.

0010	0°
0011	30°
0001	60°
0000	90°
1000	120°

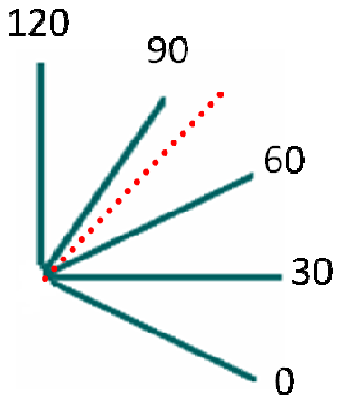

Figure 2-12: Comparing the result of binary and continuous coding for θ_x

There are also some difficulties that arise from the coding, when binary coding is used to represent values in continuous search space. One of these difficulties is "Hamming cliffs". For variables with certain properties a transition between two neighboring values may require alteration of many bits. This would create an obstacle in the path of gradual search in the coded variable space (Deb and Agrawal, 1995).

Another difficulty in using binary GAs is the arranging of coded variables within the chromosome. Unless related building-blocks are coded "tightly", the crossover operator cannot combine the building-blocks efficiently (Goldberg et al. 1989; Deb and Agrawal, 1995). In tight coding, the intension is to put the variables having closer relation in determining the value of the objective function, as close as possible in the chromosome. However, for complicated problems the relationship between different parameters is not

understood easily. The problem of tight or loose coding is known as “linkage” problem (Deb and Agrawal, 1995).

Another advantage of continuous GAs as the search engine in well placement optimization problem is that invalid wells can be prevented during the reproduction stage. This can be done because in continuous GAs, reproduction operators deal with actual value of parameters instead of their coded form, empowering them to honor the ranges in which they are valid.

Finally, in continuous GAs there is no more need to code solutions from decimals to binary for reproduction and decode the chromosomes from binary to decimal to evaluate the objective function. This also leads in increased efficiency of the code.

2.5. GAs Convergence

Since we are using GAs as the search engine for our problem, it is important to know something about their convergence. We would like to know if the algorithm will finally converge to the global optimum of the problem given infinite time, and the conditions required for convergence. This is important to know as many optimization algorithms will miss the optimum. Rudolph (1994) has modeled the behavior of Canonical GAs and showed that they do not guaranty convergence to the global optimum in their standard from. However, by adding “elitism” Canonical GAs will converge to the global optimum given infinite time. Agapie (2001) used random systems with complete connections instead of Markov chains, to account for a complete, rather than recent, history of the algorithm's evolution. He modeled binary and continuous GAs with adaptive mutation probability, and has come up with global convergence conditions for the problem he studied.

However, for an optimization algorithm to be used in practical problems it should also have high convergence rate. From a practical point of view it means good enough solutions could be obtained within a reasonable time frame. The information on

convergence rates given by the mathematical models are still general, and could not be used for comparison.

2.6. Continuous GAs Convergence Test

Here, we have designed a simple optimization test to compare convergence rate of a continuous GA (with elitism) and exhaustive search, both of which, given infinite time, would finally converge to the global optimum. For this purpose we have designed an optimization problem with a known global optimum.

The problem is to find a vertical well with highest production in a 2-D homogenous square reservoir with closed boundaries. The dimensions of the reservoir model used are 100×100. The best solution is known to be in the center of the reservoir, which falls in grid block (51, 51). A continuous GA with parameters given in Table 2-1 was tested for comparison. Evolution path of the GA over generations could be observed in Figure 2-13. It shows all members of the GA population at each generation until convergence.

Table 2-1: Continuous GA parameters used in the convergence test

Continuous GA Parameter	Value
Initial population	20
Maximum generation	100
Crossover probability	0.5
Crossover factor	1
Ranking scale	3
Mutation probability	0.1
Mutation factor	0.06
Mutation power	1
Kept fraction	0.3
Rejected fraction	0.3

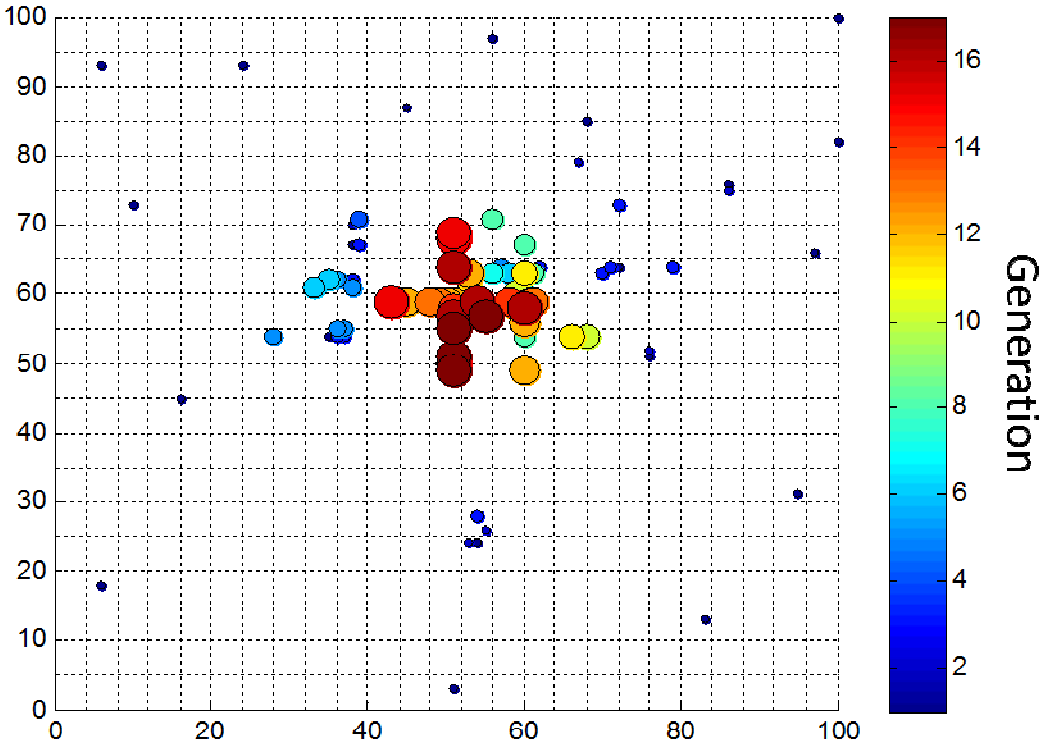

Figure 2-13: Evolution path of GA over generations

To diminish the stochastic effects in measuring GA convergence rate, 100 different initial populations were used. Then, the continuous GA code was run 200 times for each initial population. Using the given parameters, all the 20000 cases converged to the known optimum before 100 generations. Using the continuous GA we converged after 103.5 simulations on the average (see Table 2-2). However, on average exhaustive search converges after 5000 simulations.

Table 2-2: Results of continuous GA convergence test

When Converged	Value
Average number of generations	10.7
Average number of simulations	103.5
Median number of simulations	91.5

Chapter 3

3. Steps in Well Placement Optimization with GAs

In the early stages of this research the purpose was to understand issues involved in well placement optimization with GAs. This is an essential step towards introducing improvements to current models. The necessary understanding was achieved though the investigation of all the steps in well optimization by GAs. The idea was to look at things that could be done to make improvements in specific steps. The main steps in well optimization by GAs could be listed as:

1. Modeling the wells and coding the solution scenarios for GAs
2. Generating initial population of possible solutions
3. Evaluating the fitness of individuals in each generation and ranking them
4. Selecting the parents and paring them for reproduction
5. Generating next generation and designing reproduction operators

The well placement optimization results, provided in this section, are generated using a binary GA code written by Artus (2005) and updated by Onwunalu (2006). This code was used in the early stages of the work, since we had not developed our continuous GA code at that time.

3.1. Modeling and coding

Looking into the first step, we came up with the idea of implementing continuous GAs to take advantage of real-valued coding. Also, there were two types of improvements to well models to look at. The first is to model the well with parameters that have the greatest correlation to the objective function outcomes. This was expected to benefit the evolution process in a GA, which is trying to find and combine better building blocks, to generate

better solutions. The concept of better building blocks used by GAs in the evolution is valid when the building blocks used in the chromosome have high correlation to outcomes of the objective function. This aspect is covered in the Modeling Section. Some minor changes have been introduced in choosing the parameters to be used in the optimization process. The second type of well model enhancements, are those that make the well model more physical. To move in this direction, we introduce a well model that allows curved mainbores for the potential solutions of the optimization problem. Moreover, practical issues in drilling advanced wells and producing from them should be incorporated in the model to make the optimization code more useful. For example in multilateral wells mainbore is usually only perforated before the first junction to prevent crossflow. Also the laterals are not perforated from the junction, and their perforation usually start about 3 meter away from the junction. Furthermore, the risks associated with drilling different well designs are not similar. Therefore, it is important to have a way to incorporate this type of risk into our model.

3.2. Initial Population

Since GAs are stochastic search processes, even by using the same initial population the evolution path and the optimum solution may not be the same. We were interested in seeing the effect of different initial populations on search outcomes. A test was set up to study this issue. In this test we are searching for well with highest production in a $40 \times 40 \times 7$ channeled reservoir. The binary GA parameters used are given in Table 3-1. The optimization was performed with 4 different initial populations, and it was repeated 3 times with each initial population.

Table 3-1: Binary GA parameters used in investigating the effect of initial population

Binary GA Parameter	Value
Initial population	30
Maximum generation	30
Crossover probability	0.8
Mutation probability	0.07
Ranking scale	2

Figure 3-1 shows the fitness of best solution of each case at all generations. Optimizations using the same initial population are plotted in the same color. In this figure, we observe strong effect of initial population on the best solution. This is due to the fact that in half of the cases for the same initial population the best solution has evolved exactly in the same way. We can also see that final optimum solutions depend on the initial population to some extent. The difference of about 10% is seen between fitness values of all final optimum solutions. To take advantage of the effect of different initial populations, we designed a parallelized GA search using multiple initial populations.

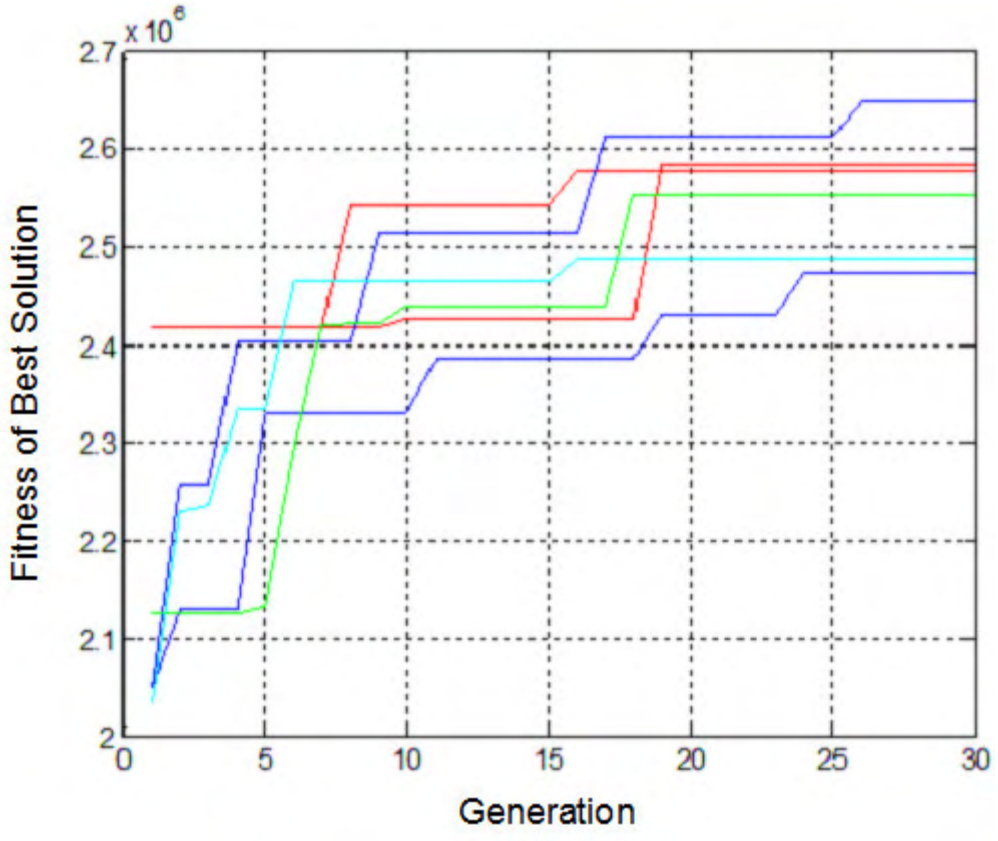

Figure 3-1: Effect of initial populations (4 different initial populations, 3 runs each)

3.3. Fitness Evaluation

Reproduction in GAs is a two stage process. First an intermediate population is constructed from current population. Chromosomes in this intermediate population are parents of the next generation. Next they are paired and then each pair goes through crossover and mutation operators to create members of the new generation. The selection process takes care of constructing the intermediate population. The way this process works in GAs is to select chromosomes randomly from current generation giving more weight to better solutions. Therefore, we need a measure of goodness to use in assigning weights for the selection process. This measure could come from calculating the objecting function value for all members of the current population, or by using various methods that estimate the objecting function value, called proxies. Different proxies have been used in the literature to estimate to fitness, such as Kriging, Artificial Neural Networks, and

Statistical proxies. However, in this work we put our emphasis on improving the optimization framework, rather than improving the estimating functions. We tried short time simulations as proxies to full simulation. But they did not work well because of the following reasons:

- Cutting the simulation time to half or even one third will not lead to considerable decrease in the computational time of the numerical simulation. This is because usually time steps are much smaller in the earlier stages of the simulation and larger at later stages. Hence, a large portion of the computational time is used at earlier stages.

- By cutting the simulation time to much smaller values there would be low correlation between the outcomes of short time simulations and full simulation. This happens because water breakthroughs and interference between depletion zones all happen at latter stages.

3.4. Selecting Parents

As it was discussed in the previous section the selection process requires a measure that compares goodness of individuals in a population. This measure of performance could be achieved by calculating the objective function value or estimating it using proxies. Then fitness function is used to transform that measure of performance into selection opportunities assigned to each individual for the selection process. In Canonical GAs fitness or selection probability is defined by (Whitley, 1994):

$$p_n = \frac{f_n}{\sum_{i=1}^{N} f_i} \tag{3-1}$$

where is f_n objective function value for nth chromosome, and N is the population size.

This would give a chromosome with larger objective value the higher probability to be selected for mating. To increase the probability of selecting fitter individuals as parents, scaled fitness function can be used:

$$p_n = \frac{(f_n)^r}{\sum_{i=1}^{N}(f_i)^r} \tag{3-2}$$

In this equation r is the ranking scale. It means higher r would increase the chance of selection of fitter individuals. However, large r's would be very selective and will result in making the whole population uniform, leading to premature convergence. Fitness can also be assigned based on the ranking of the individual, instead of its objective function value. In this way first individuals should be sorted by their objective function values. Then their fitness could be defined by:

$$p_n = \frac{(N+1-n)^r}{\sum_{i=1}^{N}(i)^r} \tag{3-2}$$

Each of these fitness functions has its own advantages. Objective based fitness has the benefit of making a distinction between successive individuals with low or high objective function values. However, in the later stages of evolution the objective function values could become very close to each other, giving more distinction power to ranking based fitness.

Regardless of whether fitness is calculated using the objective function value or ranking, a cutoff value can been enforced by the selection process. This would mean that individuals with objective function value lower than a specific number, or individuals with a ranking more than N_{select} will receive zero probability of being selected as parents (see Figure 3-2). The formula to calculate ranking based fitness would change to:

$$p_n = \frac{(N_{select}+1-n)^r}{\sum_{i=1}^{N_{select}}(i)^r} \tag{3-3}$$

where N_{select} is the number of individuals selected as potential parents (size or intermediate population).

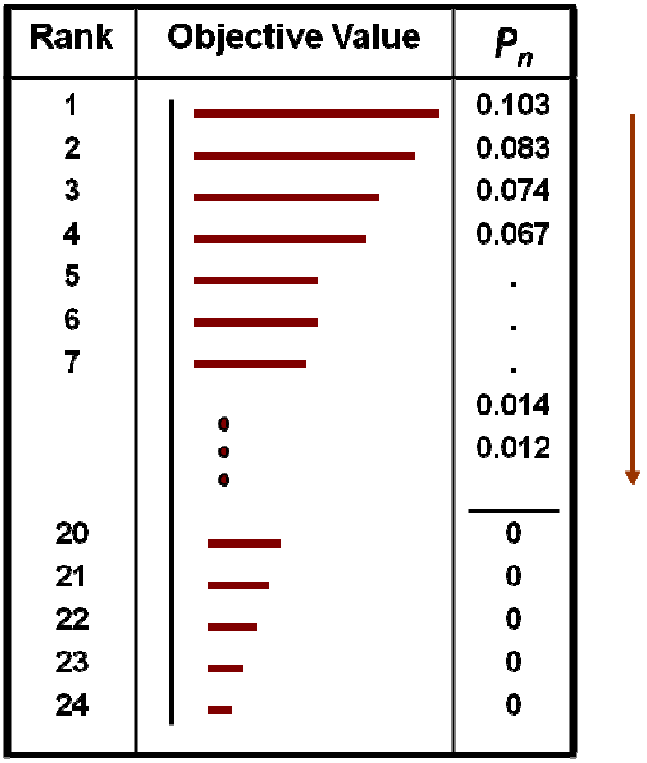

Figure 3-2: Fitness values with cutoff enforced

Here, we performed a test to examine the effect of enforcing a cutoff value in the selection process. We used the reservoir and same binary GA parameters from the previous example. Also, the objective function is the cumulative oil production. In this test we ran the optimization code 3 times without enforcing a cutoff value, and 3 times with enforcing a cutoff value of rank 15 (individuals ranked higher than 15 are assigned zero selection probability). All the runs were performed with the same initial population, to remove the effect of initial population. The evolution of the best solution of each case over the generations is shown in Figure 3-3. As shown in the figure enforcing cutoff value of 15 resulted in slower evolution process. This could happen because the diversity of the population in the early generations (resulting in more thorough search of the whole domain) is decreased a lot by using cutoff value of 15 for a population of size 30.

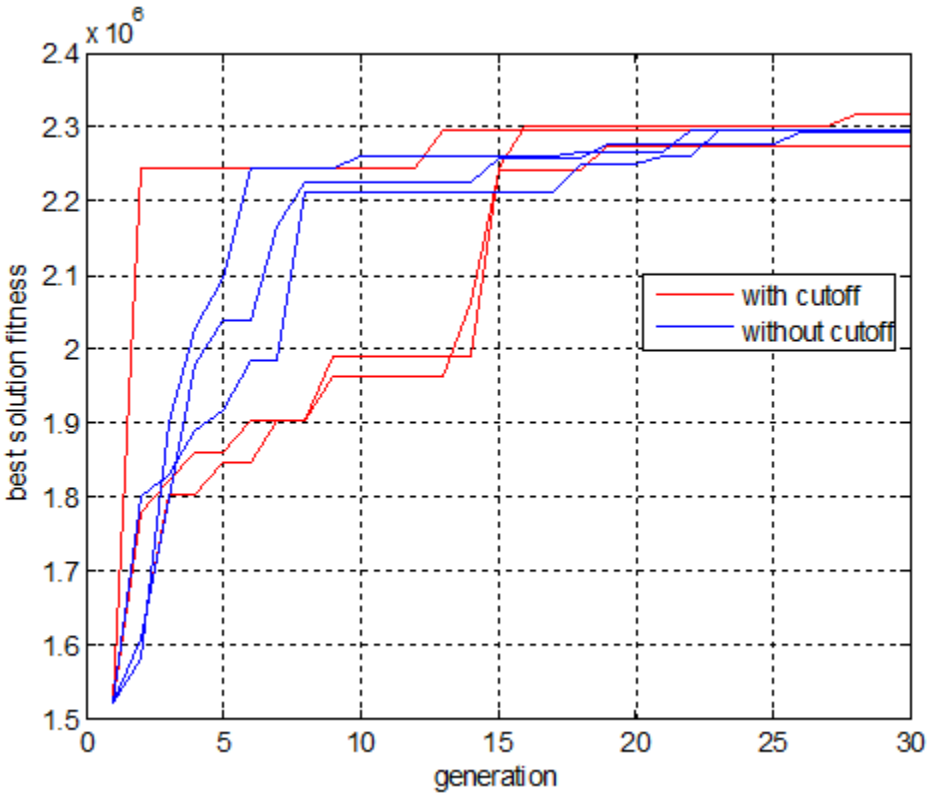

Figure 3-3: Comparing enforcing and not enforcing a cutoff value in the selection process

3.5. Reproduction

During the reproduction stage, first members of the intermediate population are paired. Then each pair goes through crossover and mutation, to produce new members of the next generation. The crossover operator in binary GAs is controlled by the probability of crossover, P_{xo}. Crossover probability, defines the chance of two parents to mate when they are paired. The extent of mutation is also controlled by mutation probability, P_m, which is the likelihood of any bit in the recombined chromosome to be changed from its state. We have also seen that in the selection procedure, the selectiveness of the process could be controlled by ranking scale, r.

Since the values chosen for binary GA parameters P_{xo} , P_m , and r, have strong effects on the way the evolution takes place, we are interested in understanding their effect of on the

search. This may help us in enhancing the search process by using improved search parameter values.

In this part we have designed a test that could help us to understand the effect of using various binary GA parameters on the search outcome and the evolution path that the population goes through during the search for the optimum. For performing this test we used the $40 \times 40 \times 7$ channelized reservoir used in the Example B by Onwunalu (2006). The properties for this reservoir are provided in Table 3-2. There are 10 realizations representing this reservoir. The optimization process is performed to find the best placement for of a monobore production. The criteria used is to maximizes the NPV of the filed after 1000 day of production, with a risk neutral attitude.

Table 3-2: Reservoir and fluid properties used in the "effect of each parameters test"

Property	**Value**
Grid dimensions	$40 \times 40 \times 7$
Field dimension	$6000 \times 6000 \times 210$ ft^3
Porosity	0.20
Average permeability of channels	90 mD
Average permeability of matrix	1 mD
Compressibility factor	3×10^{-5} psi^{-1}
B_o	1.3

The GA optimization was performed using a high and a low value for crossover probability and ranking scale, as well as a high, a medium and a low value for mutation probability. The values used as binary GA parameters are provided in Table 3-3. All combinations of the given GA parameters were used to perform optimizations (a total of 12 set of optimization parameters). All optimization cases were made using a similar initial population to diminish the effect of initial population. Also, each combination of

parameters was used four times, to reduce the effect of the stochastic nature of GA search.

Table 3-3: GA parameters used to test the effect of each parameter

Binary GA Parameter	**Value**
Initial population	30
Maximum generation	30
Crossover probability	[0.5, 0.8]
Mutation probability	[0.001, 0.05, 0.2]
Ranking scale	[2, 3]

The objective values for the optimum wells found in all optimization cases are presented in Table 3-4. It can be seen that higher crossover probability of 0.8 resulted in faster convergence in almost all cases. For mutation probability and ranking scale, none of the values was always the best choice. Lower ranking scale of 2 was a better choice when used with smaller mutation probabilities of 0.001 and 0.05 (that are more usual). But, higher ranking scale of 3 was a better choice when used with high mutation probability of 0.2. Also, medium mutation probability of 0.05 performed generally better than the high and low values of 0.2 and 0.001.

In Figure 3-4 we present the evolution map for all three $\boldsymbol{P}_m$ values. The evolution map is created by sketching objective function values of all the individuals in the population over all generations. For each $\boldsymbol{P}_m$ value, the map that is the most representative of all cases is presented.

Table 3-4: Results from the testing of the effect of binary GA parameters on search outcome

	P_m	P_c	R_s	NPV (Case1)	NPV (Case2)	NPV (Case3)	NPV (Case4)	Average NPV
1	0.2	0.8	2	1.01×10^8	1.01×10^8	8.98×10^7	1.00×10^8	9.81×10^7
2	0.2	0.8	3	9.73×10^7	9.73×10^7	9.82×10^7	1.04×10^8	9.93×10^7
3	0.2	0.5	2	9.67×10^7	9.79×10^7	9.74×10^7	9.67×10^7	9.72×10^7
4	0.2	0.5	3	1.01×10^8	1.01×10^8	1.01×10^8	1.01×10^8	1.01×10^8
5	0.05	0.8	2	1.04×10^8	1.04×10^8	1.04×10^8	1.04×10^8	1.04×10^8
6	0.05	0.8	3	9.84×10^7	9.84×10^7	1.06×10^8	9.84×10^7	1.00×10^8
7	0.05	0.5	2	1.06×10^8	1.06×10^8	1.06×10^8	1.06×10^8	1.06×10^8
8	0.05	0.5	3	1.02×10^8	9.96×10^7	9.98×10^7	9.96×10^7	1.00×10^8
9	0.001	0.8	2	1.02×10^8	9.75×10^7	1.00×10^8	9.75×10^7	9.93×10^7
10	0.001	0.8	3	9.60×10^7	9.60×10^7	9.60×10^7	9.60×10^7	9.60×10^7
11	0.001	0.5	2	1.02×10^8	1.02×10^8	1.02×10^8	1.02×10^8	1.02×10^8
12	0.001	0.5	3	9.43×10^7	9.43×10^7	9.43×10^7	9.43×10^7	9.43×10^7

These results demonstrate that higher mutation probability values lead to faster evolution in earlier generations. However, lower mutation probability values help in continuing evolution in later generations. This can be observed by comparing the evolution maps provided in Figure 3-4. This behavior could be explained by the two roles of mutation discussed in Chapter 2. We could take advantage of this behavior by designing an optimum dynamic mutation trend that starts from a higher mutation probability values in earlier generations and decreases to a lower mutation probability values in later generations. This provides the motivation for introducing dynamic mutation to our continuous GA code.

We also looked at the location and objective value (NPV) of the best solutions found in each of the 48 runs. The Maximum variation in optimal NPV was observed to be 10%. Also, the location of best wells found was observed to be considerably different between some cases.

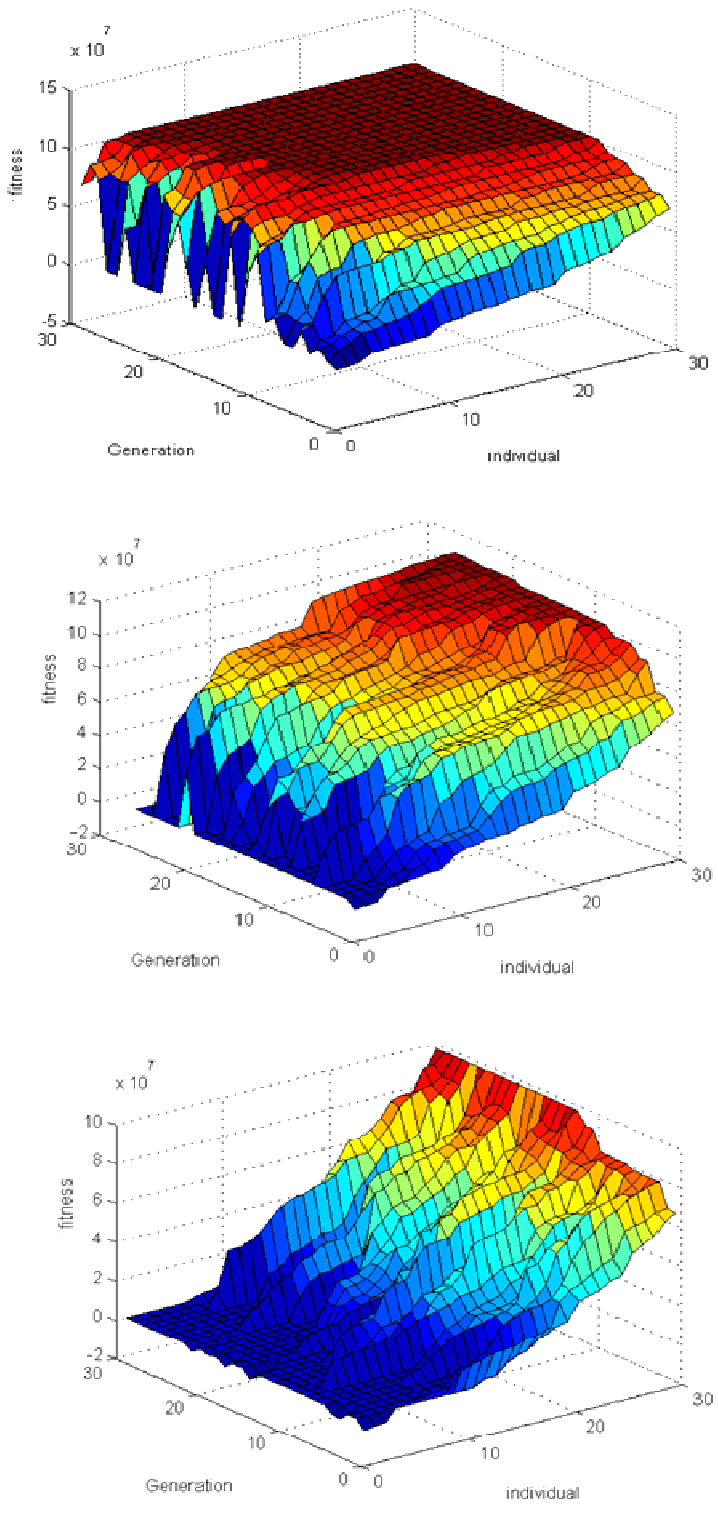

Figure 3-4: Effect of mutation probability (from top P_{mut} = 0.001, 0.05, and 0.2)

Chapter 4

4. Continuous GAs Implementation in Well Optimization Framework

For a well placement optimization framework, we first need to have a good model for representing possible solutions. This model should provide the parameters to be optimized, in the form of a chromosome, to the GA engine. It is also used to characterize the wells for numerical simulators to calculate the objective value. The numerical simulators used as objective function evaluator during this study were Schlumberger GeoQuest's ECLIPSE simulator (GeoQuest, 2006) and Stanford's General Purpose Research Simulator (GPRS, 2006). For each simulator the code should be able to use the parameters provided by the model to generate the input files required by that reservoir simulator. In the first section of this chapter we will discuss multilateral well modeling for GA optimization. In the second section we discuss some practical implementation issues. Finally, in the third section we will present and discuss some results.

4.1. Multilateral Well Modeling for GAs

Modeling is using a simplified representation of a complicated physical system that could help us in understanding the behavior of the system. Here, we are interested in finding the optimal scenario for maximizing the NPV of a petroleum field. Multilateral wells are complicated systems and many parameters are needed for modeling them.

To design a suitable model for GA search, two criteria should be considered. First, the number of parameters describing the model needs to be as small as possible. This requirement arises from the fact that the parameters that describe the model are the same variables that the GA optimizes. Higher number of variables means larger chromosomes and hence, harder optimization problems. The second criterion is that the parameters have to be selected in a way that their values have physical relations with the outcome of the

objective function. This comes from the nature of the evolution in GAs. During the evolution, genes associated with fitter solutions survive. The stronger the connection of the parameters with the outcome of the objective function, the more information is gathered by the GA, in the search to find their optimum values. This increases the pace by which the algorithm moves towards the optimum.

In this work we use the NCW model introduced by Yeten (2003), with some modifications. The model characterizes a NCW as an independent straight line for the main bore and a number of lines initiating from that line as the laterals. A line segment in 3D, could be represented by the coordinates of its end points (see Figure 4.1). We will call the point with the smallest z coordinate (top point), the Heel, and the point with larger z coordinate (bottom point), the Toe, as proposed by Yeten (2003). The optimization parameters he used for modeling a line segment in 3D were: the coordinates of the Heel (h_1, h_2, and h_3), vertical coordinate of the Toe (t_3), horizontal length of the well (l_{xy}) and the counterclockwise angle from the x-axis (θ_x). Then the horizontal coordinates of the Toe are calculated by:

$$\begin{aligned} t_1 &= h_1 + l_{xy} \sin \theta_x \\ t_2 &= h_2 + l_{xy} \cos \theta_x \end{aligned} \tag{4-1}$$

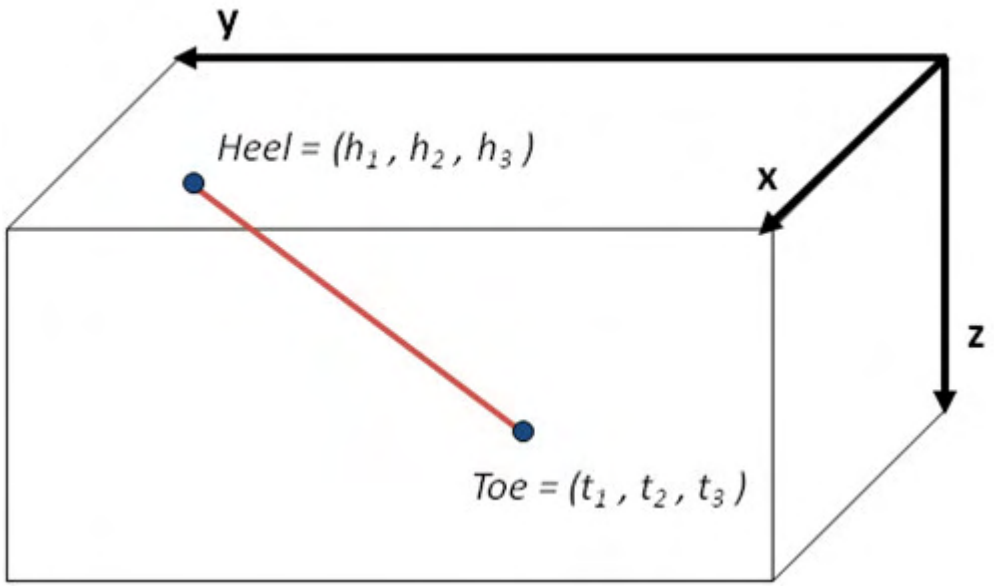

Figure 4-1: Heel and Toe coordinates for a line segment in 3D

Therefore, a line segment was modeled by four coordinate values, one angle and one length property.

We have chosen to use the coordinates of the midpoint (the point between Heel and Toe) instead of the coordinates of the Heel. It has been chosen since the midpoint of a well is more representative of the effect of the well location on production. We also use total well length (l_m) instead of the horizontal well length (l_{xy}), as total length has more correlation with production. Moreover, vertical well length (l_z) was used instead of vertical coordinate of the Toe (t_3), since the vertical coordinate value by itself does not give as much information about production as vertical length (see Figure 4.2). The reason that l_z was chosen over other parameters such as θ_z, is that by putting bounds on the value of l_z we can assure keeping the well within the valid vertical range (usually much narrower than the horizontal range). Using this representation coordinates of the Heel are given by:

$$\begin{aligned} h_1 &= m_1 - \frac{1}{2} l_{xy} \sin\theta \\ h_2 &= m_2 - \frac{1}{2} l_{xy} \cos\theta \\ h_3 &= m_3 - \frac{1}{2} l_2 \end{aligned} \tag{4-2}$$

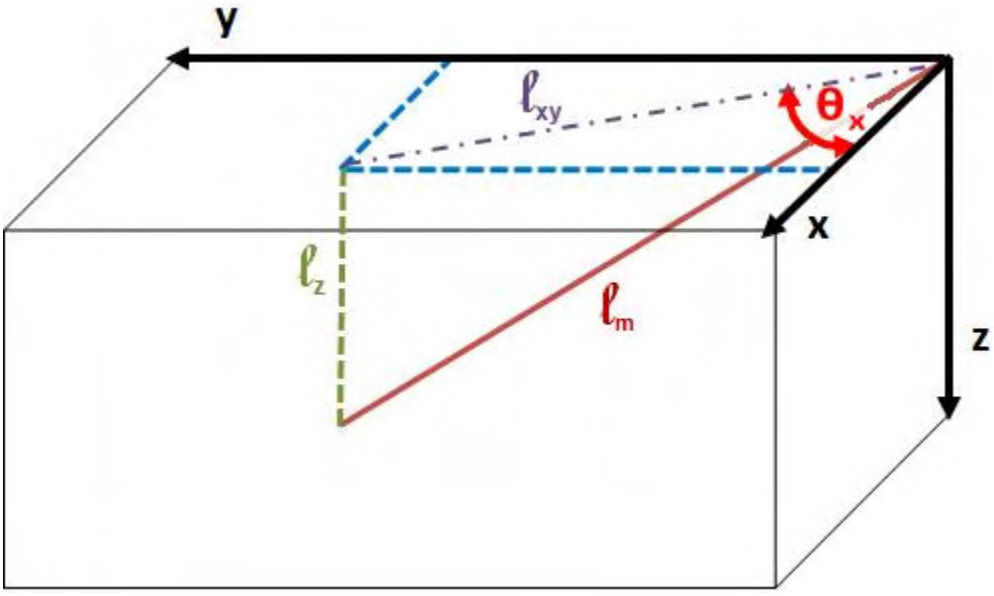

Figure 4-2: Well trajectory optimization parameters

Therefore, an independent line segment is modeled with three coordinate values (representing location), one angle and two length properties (representing trajectory). However, for a lateral the location is not an independent property, as its heel always lies on the mainbore. Therefore, the lateral location can be represented by a junction point, j_p, as a fraction of mainbore length. A multilateral well chromosome could be generated by putting together the variables defining mainbore and laterals. Also, a chromosome for multi-well production scenario can be generated by putting together the variables defining multilateral wells. To implement dynamic number of wells and laterals we need to add an activation parameter, $\boldsymbol{a}$, for each well and lateral.

4.2. Practical Implementation Issues

4.2.1. Input File

Since, optimization parameters and input data required for the optimization are subject to change with time, it is helpful to define all of them in an input file. This diminishes the necessity of going to the code for changing the values, each time a change is required Here the file is called *input.par*. This file includes: file names to be created or used, the objective function used, continuous GA parameters, economic variables, number of wells and laterals, the high and low limits of the variables describing the wells and the grid data. Then a function called *read_input.m* reads all the parameters needed to run the continuous GA optimization program from the *input.par* file and saves them in a *struct* called *params*.

4.2.2. Writing the Input File for Simulation

The simulation input file also contain well data and beside reservoir properties needed for numerical reservoir simulation. Since, the well data are the only part that changes for different individuals, it is more convenient to put the well data in an include file called *simewell.inc*. The program converting individual chromosomes to include files that could be read by simulators is called *ECL_well.m* for ECLIPSE, and *GPRS_well.m* for GPRS. This program uses a function called *crossCells* that takes Heel and Toe positions of well

or lateral and calculates the cells that are crossed by its trajectory and their crossing points. Then the connection factor for each cell that has been crossed is calculated using projection well index (Shu, 2005).

4.2.3. Performing Simulations and Reading the Results

Simulations for each individual field development scenario in the current population are performed and the simulation results for each individual are saved. This is done by *run_sim.m*. The point to be considered is the presence of multipliers such as **10**3* below the variable names in the *.RSM* file. So, the file should be searched for all such multipliers before reading the simulation results, otherwise we would be working with the wrong values. Cumulative oil and water productions and cumulative water injection after each year are saved for each individual. These values are then used in calculating the objective value of each individual by *rank_pop.m*. Then the individuals in the population are sorted in a descending order with respect to their evaluated fitness.

4.2.4. Reproduction with continuous GA operators

The role of *reproduce.m* function is to reproduces the next generation from the current generation. Since, a chromosome defined here consists of actual well properties (real values) and activation parameters (binary value), they are separated initially and each goes through separate reproduction operators. Dynamic mutation is implemented by reducing the size of the mutation factor as generations evolve. The function *checkV* is used to validate each new offspring before considering as new generation. We have used this function to impose minimum distance between individual chromosomes and minimum well distance.

4.3. Example

In this section we present an example to compare the performance of binary and continuous GAs. For this example the optimization engine uses GPRS as the reservoir simulator. The reservoir used for this example is a $40 \times 40 \times 7$ channeled reservoir shown in Figure 4.3.

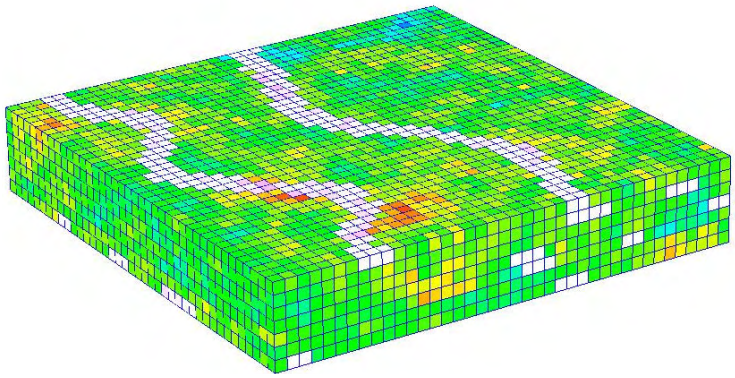

Figure 4-3: The reservoir used for comparing performance of binary and continuous GAs

The objective of the search is to find the best location and configuration for two producers and one injector, each with two laterals. The continuous GA parameters values used in this example are provided in Table 4-1. The chromosome for this example has 51 variables.

Table 4-1: Parameters used for comparing performance of binary and continuous GA

Continuous GA Parameter	**Value**
Mutation probability	0.12
Mutation factor	0.25
Mutation power	2
Crossover probability	0.4
Crossover factor	0.2
Ranking scale	2

The objective function used here is the NPV of the field over four years, using 10% discount rate. Two type of continuous GAs were compared with binary GAs. In the first type (continuous GA1) only the best individual in the population was directly kept for the next generation (This one is always kept as we are using a GA with Elitism). In the second type (continuous GA2) 30% of the population was kept for the next generation. Figure 4-4 compares the evolution of best individuals using binary and continuous GAs. It can be observed that by keeping some portion of the population in continuous GAs higher results are achieved. However, keeping some portion of the population in binary

GAs would have caused premature convergence due to lower space dimensions. The other advantage of keeping some portion of the population is that it leads to faster generation of and evaluation of new populations. This happens because part of it has already been evaluated.

By comparing the evolution path of optimizations with binary and continuous GAs, we observe more gradual evolution for continuous GAs, as opposed to typical jumps and flat periods observed in evolution with binary GAs. Also, a more steady evolution is observed for continuous GAs. This increases the chance of achieving fitter outcomes faster with continuous GAs. However, the higher dimensionality of continuous search space and absence of the jumps observed during the evolution of a binary GA, have resulted in slower convergence rates in some cases.

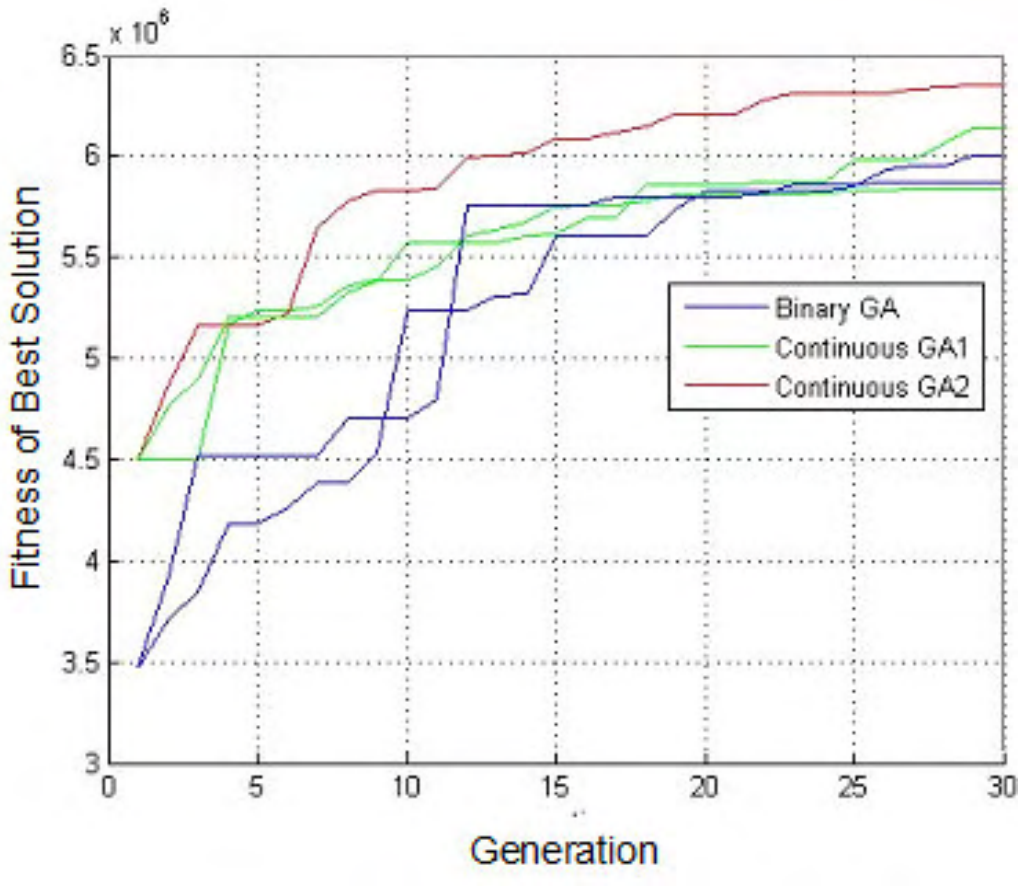

Figure 4-4: Evolution of best individuals with binary and continuous GAs

Chapter 5

5. Improvements and Results

In this chapter we will discuss the improvements that we have proposed for well placement optimization framework with GAs. In each section, we first explain the necessity or benefits of implementing that improvement. Then we describe some of the details involved in its implementation. Finally, we present some results from implementing each technique.

The improvement techniques presented in this chapter are a result of rethinking about each step of the well placement optimization by GAs that was discussed in Chapter 3. In the first section we consider use of engineering knowledge as input for the optimization process. In the second section we study control of the Euclidean distance between the individuals in the population. In the third section we suggested using multiple initial populations to decrease the effect of initial population on the optimum solution. In the forth section we introduced a well model that enable us to also consider curved mainbore configurations in the search for the optimum.

5.1. Utilizing Engineering Knowledge

5.1.1. Motives

In this section we discuss the importance of utilizing engineering knowledge as input for the optimization process. Although our goal is to make the whole well placement process automatic, this could not be easily achieved unless we provide the optimization program some prior engineering information. This information helps the optimization program to avoid running computationally expensive simulations for cases which could be easily recognized to be non-optimum through our engineering knowledge. This process could also be viewed as a proxy.

One example of the engineering knowledge that could be incorporated in the optimization framework is minimum plausible distance between same and different types of wells in a reservoir. For example we know if a producer is too close to an injector, early water breakthrough will occur, resulting to poor sweep of reservoir, higher water production and hence higher water treatment cost. Also, if two producers are at a short distance of each other, the early interference between their depletion regions will result in lower bottomhole pressure, hence lower production rates. Although, it is the role of the optimizer to reject all scenarios with bad characteristics, we would be much better off computationally to filter all undesirable candidate solutions.

However, we would need a well chosen value for the minimum distance between two wells to be enforced. Given the reservoir properties, the production and injection rates, and the simulation timeframe of interest, we could come up with reasonable values for minimum distance between a producer injector pair and between two producers. For the case of producer injector pair, the minimum distance that we are interested in is the distance that would cause early water breakthrough if two well are that distance.

5.1.2. Implementation

Since, we are using a multilateral well model in our optimization framework, we first need a method to calculate the distance between any two multilateral wells. To calculate the minimum distance between two multilaterals, we need to calculate and compare the distance between all segments of the two wells (see Figure 5-1).

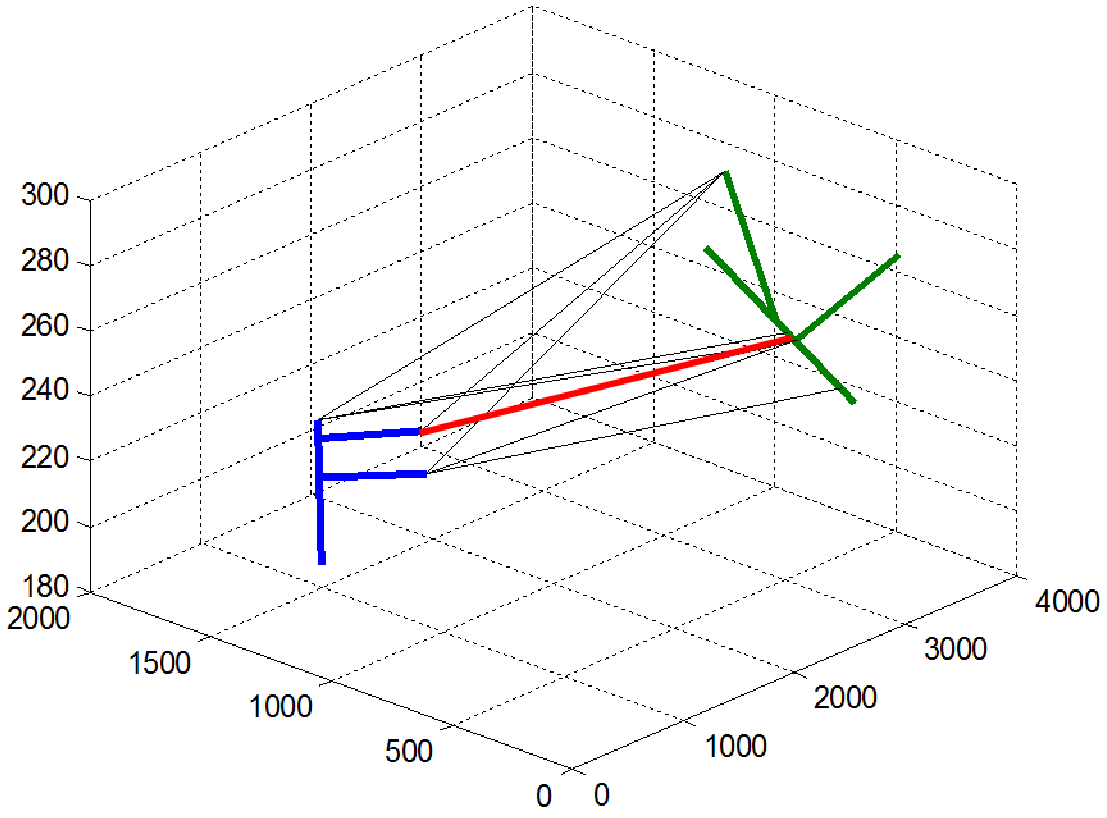

Figure 5-1: Minimum distance between two multilateral wells

To calculate the distance between two line segments, we first calculate the distance between their extended lines. In Figure 2 we can see the two lines P and Q. These lines can be represented by two line direction vectors, $\vec{u}$ and $\vec{v}$, and two coordinate points P_0 and Q_0:

$$P(s) = P_0 + s\vec{u} \qquad (s \in R) \tag{5-1}$$

$$Q(t) = Q_0 + t\vec{v} \qquad (t \in R) \tag{5-2}$$

Also, the distance between any two points on these two lines can be represented by:

$$\vec{w}(t,s) = P(s) - Q(t) \tag{5-3}$$

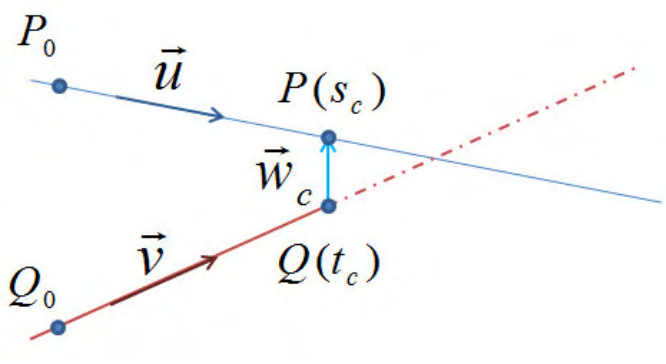

Figure 5-2: Calculating minimum distance between two lines

If the closets points on the two lines are $P(s_c)$ and $Q(t_c)$, the vector connecting them, $\vec{w}_c$, will be perpendicular to the line direction vectors, $\vec{u}$ and $\vec{v}$:

$$\vec{w}_c.\vec{u} = 0 \tag{5-4}$$

$$\vec{w}_c.\vec{v} = 0 \tag{5-5}$$

By substituting $w_c = P_c - Q_c = P_0 - Q_0 + s_c u - t_c v$, in equations (5-4) and (5-5), we can solve for s_c and t_c:

$$s_c = \frac{(\vec{v}.\vec{u})(\vec{v}.\vec{w}_0) - (\vec{v}.\vec{v})(\vec{u}.\vec{w}_0)}{(\vec{u}.\vec{u})(\vec{v}.\vec{v}) - (\vec{v}.\vec{u})^2} \tag{5-6}$$

$$t_c = \frac{(\vec{u}.\vec{u})(\vec{v}.\vec{w}_0) - (\vec{u}.\vec{v})(\vec{u}.\vec{w}_0)}{(\vec{u}.\vec{u})(\vec{v}.\vec{v}) - (\vec{v}.\vec{u})} \tag{5-7}$$

Now we can find the distance between two segments by checking if $P(s_c)$ and $Q(t_c)$ lie on each of the segments or on their left and right side. The code implementing this can be found in *segDistance.m*.

5.1.3. Results

In this section we present an example to see the effect of implementing minimum well distance enforcement on the optimization process and outcomes. For this example we use the upscaled version of the tenth SPE comparative solution project model 2 (Christie and Blunt, 2001) from Coats Engineering (2008). The properties for this reservoir are

provided in Table 5-1. The optimization process is performed to find the optimum placement for two producers and one injector. The optimization criteria is maximizing of the NPV after four years of production. Two optimization runs were made, one without enforcing minimum distance between wells, and the other one with enforcing minimum distance of 400 ft between producer-producer pairs, and 600 ft between producer-injector pairs. The optimum well placement scenario found is shown in Figure 5-3.

Table 5-1: Reservoir and fluid properties used in "minimum well distance" test

Property	**Value**
Grid dimensions	$10 \times 20 \times 10$
Field dimension	$120 \times 110 \times 68$ ft^3
Average porosity	0.17
Average vertical permeability	9.2 mD
Average horizontal permeability	206 mD
Compressibility factor	3×10^{-5} psi^{-1}
B_o	1.05

The comparison between the evolution of best individuals, in the two cases with and without the enforcing of minimum well distance is presented in Figure 5-4. As can be seen from Figure 5-4, the fitness of the optimum value with minimum well distance enforcement always stays on top of the optimum value from the other case. It can be observed that by not accepting the weak solutions in the population, and generating new solutions instead, the population space is used more efficiently resulting in considerable improvement in the final optimum fitness value.

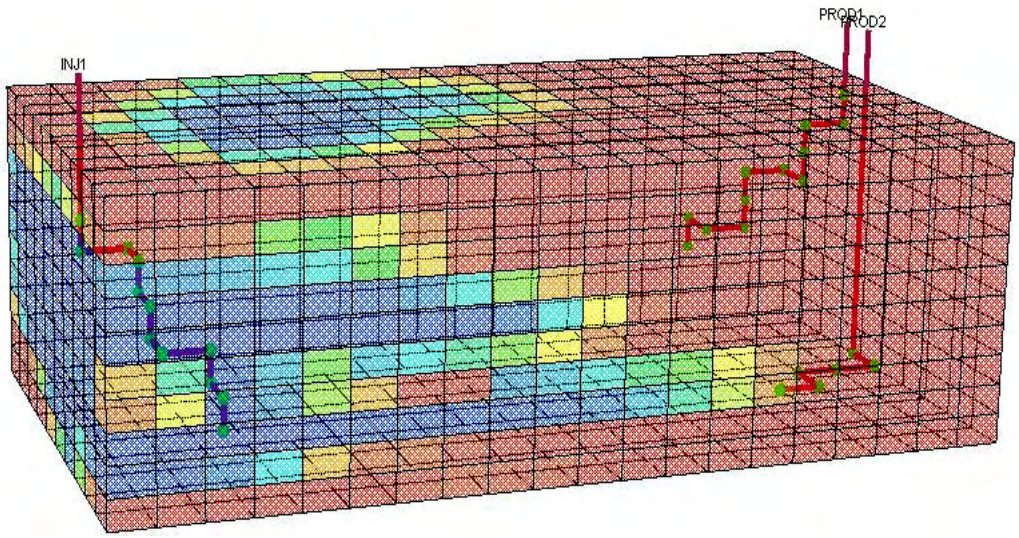

Figure 5-3: Optimum well placement scenario with minimum well distance

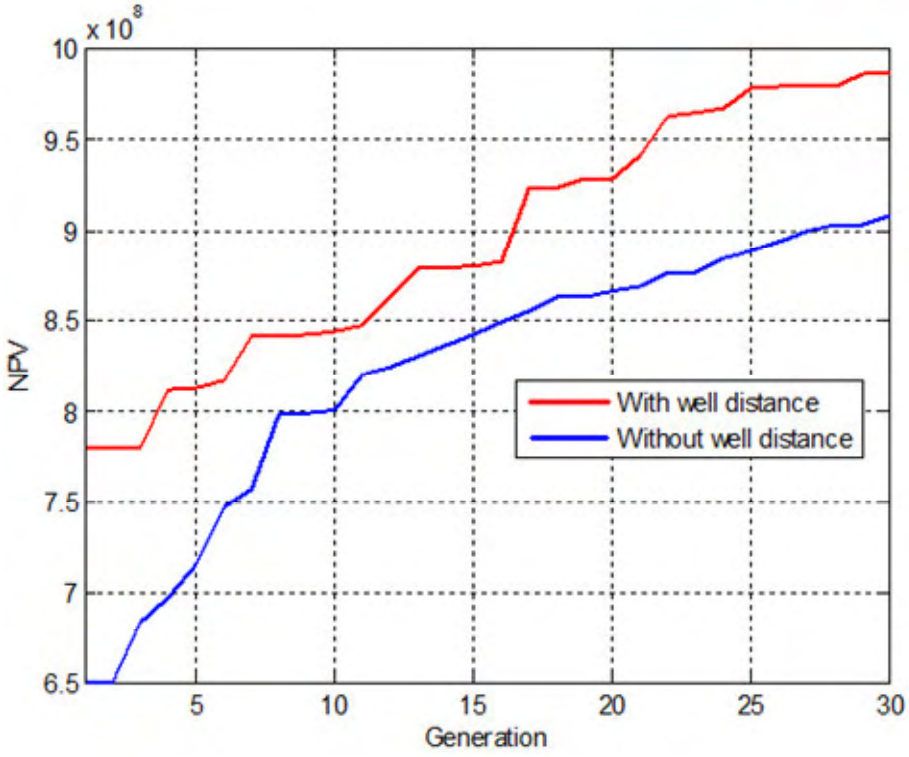

Figure 5-4: Comparing the evolution of the best individuals with and without minimum well distance enforcement

5.2. Controlling Population Diversity

5.2.1. Motives

Since, GAs are population based methods their search power arises from searching the problem space by efficient hyperplane sampling. This requires a diverse population at all stages. The initial population is desired to cover the whole solution space homogeneously to have maximum diversity (Chelouah and Siarry, 2000). Also, in the later generations as the population is intensified in more promising areas, we are interested in keeping it diverse enough so that it converges to the optimum located between many sub-optimal solutions.

5.2.2. Implementation

To calculate the population diversity and the distance between individual chromosomes, first the property vectors should be normalized. Then the Euclidian distance between the normalized property vectors of each pair of individual is calculated. When a new individual is generated, either in the initial population or during reproduction, we calculate its distance with all previously generated individuals in the population. If the calculated distance is more than the preselected minimum distance between individuals the new individual is accepted, otherwise it is rejected. If an individual is rejected, another individual is generated. This process continues until the desired population sized is reached.

However, minimum distance should be selected carefully. It should be large enough to maximize the diversity, but not too large to push the new individuals to specific locations. A minimum distance that results in rejecting half of the generated individuals on average was selected. Also, the minimum distance selected for each generation should be smaller than its value in previous generations. In our implementation we chose the minimum distance at each generation, g, by:

$$\min Dist = 0.2^{\left(1+\frac{g}{4}\right)} \tag{5-8}$$

The implementation of this idea can be found in the file *reproduce.m*.

5.2.3. Results

In this section we provide a simple example to illustrate the effectiveness and importance of calculating and enforcing minimum distance between individuals. We generate two initial populations of size 30. In the first case we impose a minimum distance with a rejection factor of ½ (half of the generated individuals are rejected). In the second case we do not impose a minimum distance. Here we have chosen individuals with only two parameters, for better visualization. The distribution of initial population over the solution space in both cases is shown in Figure 5-5.

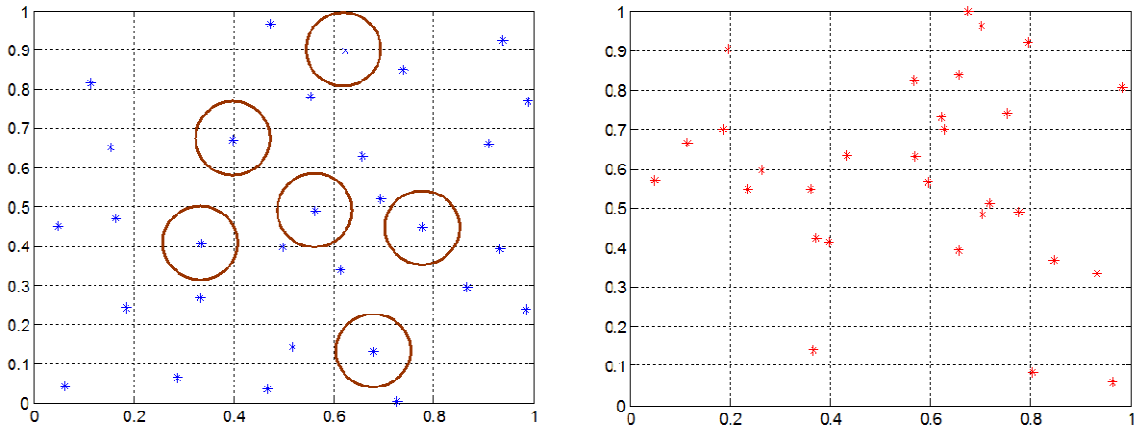

Figure 5-5: Distribution of initial population over the solution space with or without enforcing minimum distance between individuals

As can be seen in the first case (Figure 5-5, left picture) new points are not allowed to fall in the neighborhood of previous points, shown by circles around them. This has resulted in a more uniform, but still random coverage. However, in the second case (Figure 5-5, right picture) there are some individuals very close to each other. Existence of these cases results in redundant numerical simulation runs.

5.3. Multiple Initial Populations

5.3.1. Motives

In Chapter 3 we observed a strong effect of initial populations on the outcomes of the GA optimization. Not only that different initial populations resulted in different optimum fitness values (about 10% variation), but also in some cases it resulted in final optimum solutions at different locations. Here we are interested in designing a parallelized GA search framework using multiple initial populations, to take advantage of the effect of different initial populations.

5.3.2. Implementation

The framework designed here starts with four initial populations. Each of the initial populations is subjected to a separate GA optimization process. During the evolution of each population we save all the generated individuals. After all populations has gone through a certain number of generations, we select the best individuals generated in each of them that also have a minimum distance with each other (they are not very similar). Then we put all these selected individuals together in a pool of fittest generated individuals. Again, we choose the best individuals in the pool with a minimum distance constraint. This creates our new initial population that goes through another GA optimization process (see Figure 5-6). The role of the minimum distance constraint is to prevent similar individuals in the new initial population.

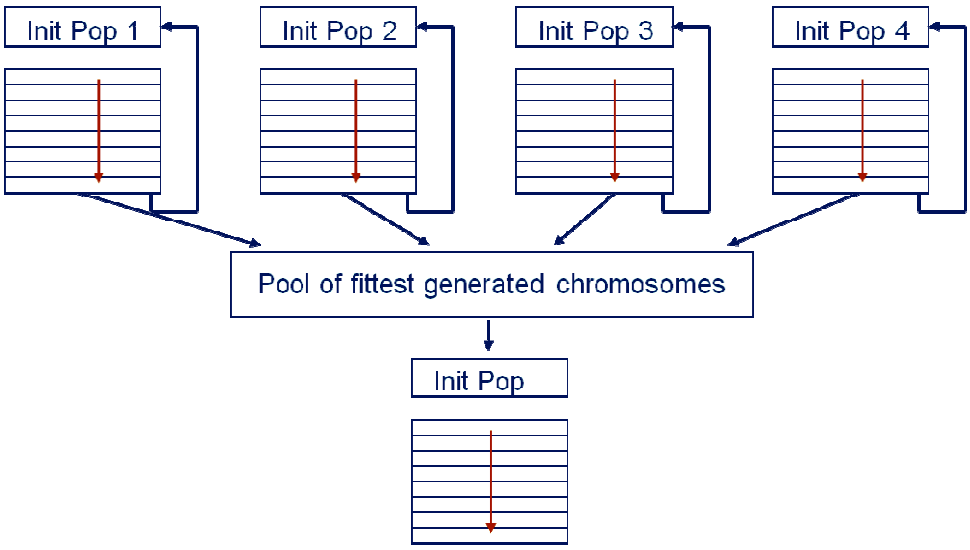

Figure 5-6: Suggested framework for optimization with multiple initial populations

5.3.3. Results

To test this framework we used the same reservoir properties and same optimization parameters as in the example of Section 5.1. The evolution of best individuals for all populations is shown in Figure 5-7. The dashed lines represent different initial populations, and the red line is the selected initial population. It can be observed that with this new initial population we can reach higher fitness value. Comparing this result with running each initial population has shown that using multiple initial populations is better than just running an optimization with one initial population much longer.

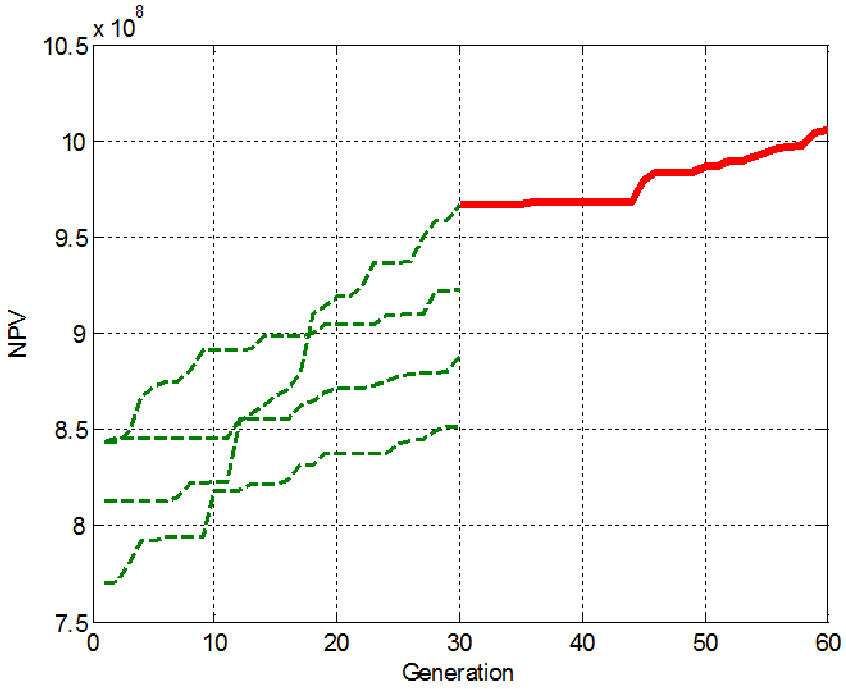

Figure 5-7: The evolution of best individuals for all populations for multiple initial populations

5.4. Curved Well Implementation

5.4.1. Motives

In Chapter 3 we discussed the importance of employing well models that could represent actual advanced wells in an improved way. This would improve our optimization framework, as the calculated objective values could become closer to actual field values. Since many advanced multilateral wells have curved mainbores one possible enhancement could be modifying the well model, in a way that would also consider curved mainbores in the potential solutions of the optimization problem.

5.4.2. Implementation

A simple proposal for a model handling wells with curved mainbores could be to use connected multi-segments for constructing a curved mainbore (see Figure 5-8a). However, this model has not been implemented in the previous works because of apparent inefficiencies it would trail. The first problem for representing a curved well with this model is the high length of the chromosome required to model the well. This would create the need for using larger populations and will result in very low convergence rates. The other problem with this model is the high chance of creating invalid (impractical) wells during GA reproduction stage (see Figure 5-8b).

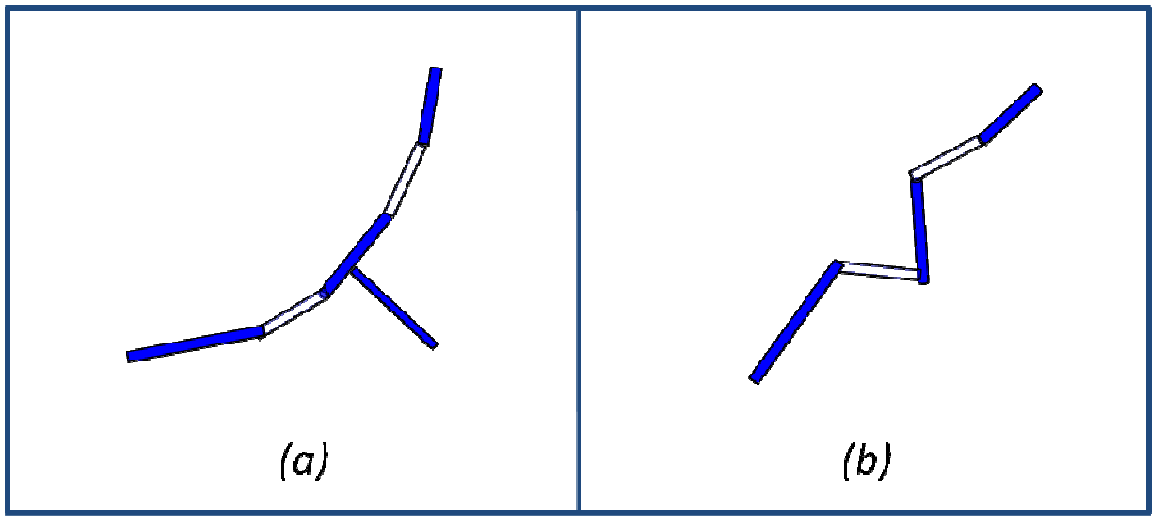

Figure 5-8: Using multi-segment representation for implementing curved mainbores

Our goal was to come up with a curved well model that does not result in a big change in the chromosome length, and guarantees the generation of practical wells during GA reproduction.

In our model we keep all the parameters that we have used for representing a 3D line segment in our multilateral well model (m_1, m_2, m_3, l_m, l_z, θ_{xy}). These parameters would specify the heel and toe points for the mainbore. We assume the well to be on the vertical

plane passing through heel and toe points (this assumption could be changed by adding another parameter, θ_p, representing the angle the well plane makes with the vertical plane). Then all valid curved wells on that vertical plain could be represented by introducing only one additional parameter, called *curvature*, using the following normalized exponential formula:

$$y = \left(\frac{e^{\left(\frac{x}{e^{curvature}}\right)} - 1}{e^{\left(\frac{1}{e^{curvature}}\right)} - 1} \right) \tag{5-9}$$

where x=[0: 0.001: 1], *curvature* = parameter describing the curvature of the well $\in [-4, 2]$

x and y will give the normalize horizontal and vertical coordinates of the curved well over the vertical plane passing through its heel and toe points. Then the normalized values of x and y should be mapped back to that vertical plane to provide the actual coordinated of all the points on the curved well. Figure 5-9 shows some curved well configurations using different curvature parameter with same heel and toe coordinates.

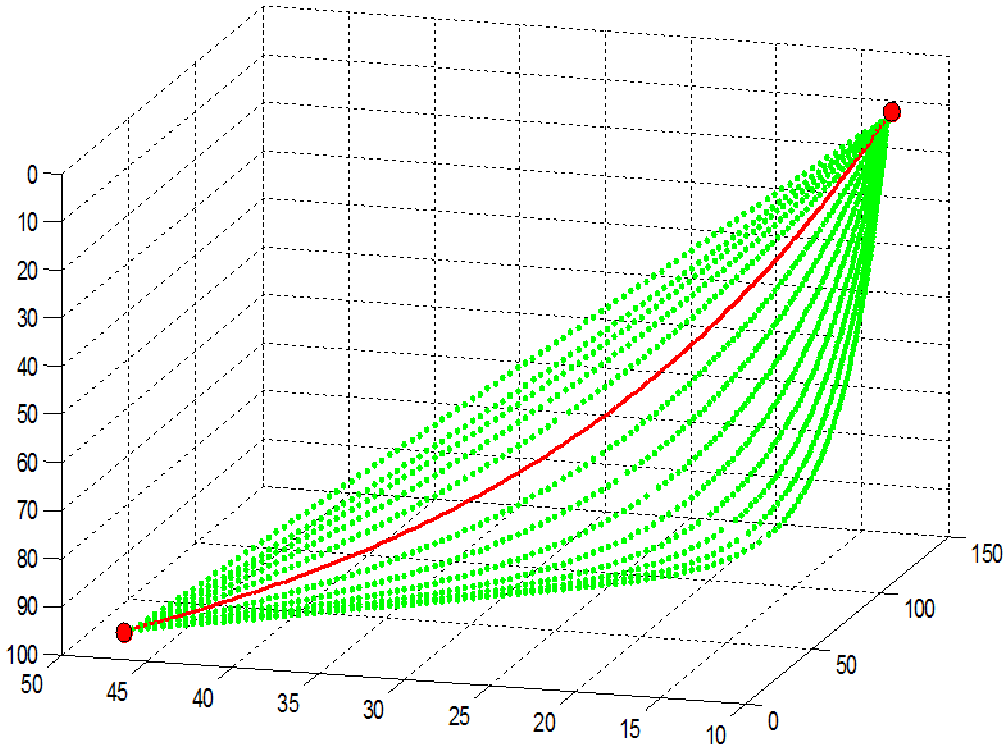

Figure 5-9: Curved well configurations with different curvature parameters

This model captures both curved and straight wells. The *curvature* value of -4 generates wells that look bent (very high curvature) at the curving point. However, the *curvature* values between 0 and 2 generate wells that are close to being straight. Hence, when the *curvature* value for a new well falls within this range it is considered to be a straight well in the numerical simulation.

5.4.3. Results

To test our curved well implementation we use the same reservoir model that was used in the example in Section 5.1. The goal of the optimization process is to find the optimum placement one monobore producer, maximizing the NPV of the field. The candidate solutions could be either straight or curved wells (depending on their *curvature* values). However, they all have the same high and low bounds for well length and vertical penetration. Figure 5-10 compares the evolution of the best individuals with straight and curved well models. It can be observed that using the curved well model the final objective value achieved is higher. However, in the first 30 generations the best fitness

using the curved well model is lower due to additional complexity added to the model. Finding higher final fitness value with curved well implementation could be interpreted to be due to the higher number of possible well configurations explored during the search for the best well.

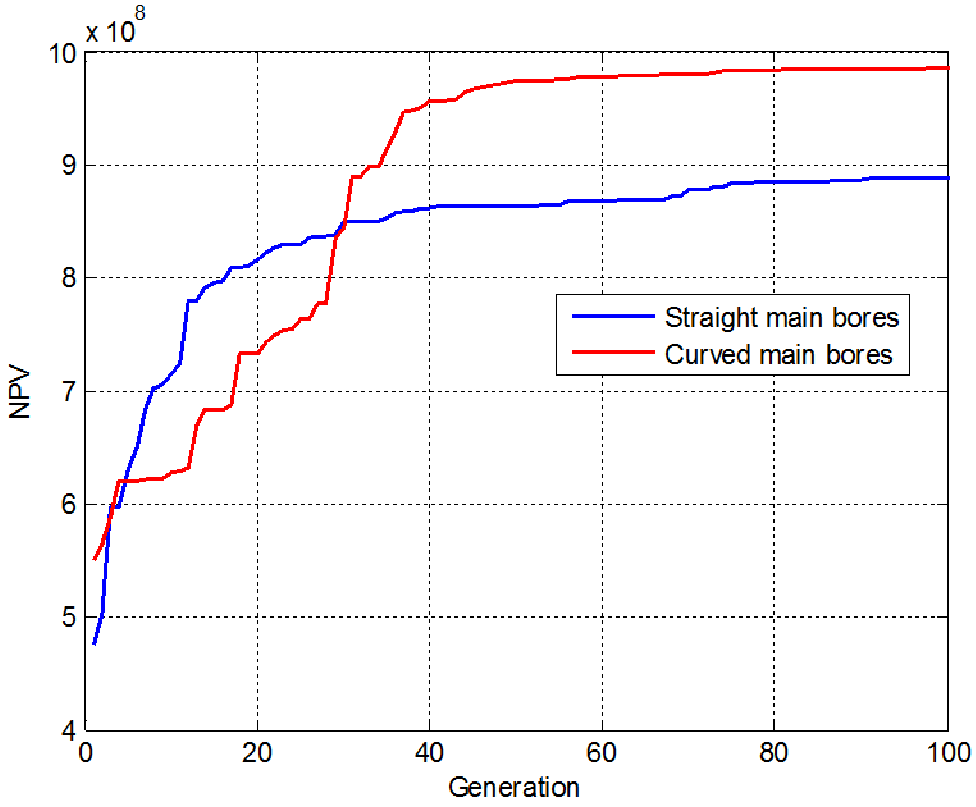

Figure 5-10: Comparing the evolution of the best individuals with straight and curved well models

Chapter 6

6. Conclusions and Future Work

6.1. Conclusions

The purpose of this study was to possible enhancements for well placement optimization with GAs. The main contributions of this work are:

- We modified the multilateral well model to be more efficient for GA optimization. In our model the total well length replaces the horizontal well length, the vertical well length replaces the vertical coordinate of the Toe, and the coordinate of midpoint replaces the coordinates of Heel. This change results in increasing the connection between the model parameter values and the objective function value (which could be cumulative oil production or NPV).
- A well placement optimization framework was been generated using continuous GAs as the search engine. The continuous GA implementation was designed to avoid generating invalid wells during the reproduction. The evolution process in continuous GAs was observed to be more gradual compared to stepwise evolution in binary GAs. However, it also showed to be steadier than in binary GAs. This would enable continuous GAs to achieve higher fitness at later stages of evolution.
- Dynamic mutation was also implemented to take better advantage of the exploring capacity of mutation in each stage of the evolution.
- We have imposed a minimum normalized Euclidean distance between the individuals in the population, controlling their distribution at each generation.
- As an example of incorporating our engineering knowledge into the optimization framework, we imposed minimum distance between multilateral wells in each individual. This resulted in improvements in both the final fitness achieved and the time required to reach good outcomes.

- A curved mainbore model was designed by adding only one extra parameter and using a normalized exponential formula. This introduced model can also capture straight wells. Higher final fitness value was achieved with the curved well implementation. This could be because of the higher number of possible well configurations explored during the search for the best well using curved well implementation.

6.2. Future Work

We propose the implementation of the following items to achieve a more powerful optimization framework:

- The curved well model can be represented by a circular formula. We have received suggestion in a meeting of industrial sponsors that drilling engineers may prefer circular representation.
- Dynamic mutation can be implemented in an adaptive manner to help the algorithm avoid premature convergence to a suboptimal solution. This could be done by calculating the divergence of the offspring population at each generation and increasing the mutation if it falls below a certain value.
- Dynamic population size can be used to provide as many individuals as is most efficient for evolution at each generation.
- Derivative information from Adjoint method could be incorporated into the GA optimization framework to make it more intelligent.

Nomenclature

$\boldsymbol{a}$	activation parameter
B_o	oil formation volume factor, volume/volume
f_n	objective function value for n^{th} chromosome
F_{mut}	mutation factor
h_i	i^{th} coordinate of the Heel
g	generation
l_m	total well length
l_{xy}	horizontal well length
l_z	vertical well length
m_i	i^{th} coordinates of the midpoint
n	binary encoding length
N	GA population size
N_{select}	number of individuals selected as parents
P_{Fi}	i^{th} variable of father's chromosome
P_i^{new}	new value of the i^{th} variable of offspring chromosome after crossover
P_{im}	mutated value of the i^{th} variable of offspring chromosome chosen for mutation
P_{Mi}	i^{th} variable of mother's chromosome
$\boldsymbol{P_{mut}}$	mutation probability
$\boldsymbol{P}_{xo}$	crossover probability
r	ranking scale
t_i	i^{th} coordinate of the Toe
x	vector of normalize horizontal coordinates of the curved well
$x_i^{\max}$	maximum allowed values of the i^{th} parameter in the chromosome
$x_i^{\min}$	minimum allowed values of the i^{th} parameter in the chromosome
x^*	optimum value of a function
y	vector of normalize vertical coordinates of the curved well
β	blending random variable

Δx_i	quantization level for the ith parameter in the chromosome
ε	a very small value
Ω	function domain
σ	standard deviation of the normal distribution
θ_x	horizontal counterclockwise angle from the x-axis
θ_z	inclination angle

References

Abo-Hammour, Z.S., 2002, "Advanced Continuous Genetic Algorithms and their applications in the motion planning of robot manipulators and the numerical solution of boundary value problems", Ph.D. thesis, Quaid-i-Azam University, Islamabad, Pakistan.

Adewuya, A.A., 1996, "New methods in Genetic Search with Real- Valued", MS report, Massachusetts Institute of Technology.

Agapie, A., 2001, "Theoretical Analysis of Mutation-Adaptive Evolutionary Algorithms", Evol. Comput. 9, 2 (Jun 2001), 127-146.

Artus, V., Durlofsky, L.J., Onwunalu, J. and Aziz, K., 2005, "Optimization of nonconventional wells under uncertainty using statistical proxies", submitted for publication.

Badru, O. and Kabir, C.S., 2003, "Well Placement Optimization in Field Development", paper SPE 84191 prepared for presentation at the SPE Annual Technical Conference and Exhibition held in Denver, Colorado, 5 – 8 Oct, 2003.

Badru, O., 2003, "Well-Placement Optimization using Quality Map approach", MS report, Stanford University.

Ballester, P.J., Carter, J.N., 2003, "Real-parameter genetic algorithms for finding multiple optimal solutions in multi-modal optimization", In Cantu-Paz, E., et al., eds.: Lecture Notes in Computer Science 2723, Springer 706-717.

Ballester, P.J., Carter, J.N., 2004, "An effective real-parameter genetic algorithm with parent centric normal crossover for multimodal optimization", In Deb, K., et al., eds.: Lecture Notes in Computer Science 3102, Springer 901-913.

Ballester, P.J. and Richards, W.G., 2006, "A Multiparent Version of the Parent-Centric Normal Crossover for Multimodal Optimization", Evolutionary Computation, IEEE Congress on, vol., no., pp. 2999-3006, 16-21 July 2006.

Baluja, S., 1992, "A Massively Distributed Parallel Genetic Algorithm", Technical Report CMU-CS-92-196R, Carnegie Mellon University.

Baluja, S., 1994, “Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning”, (Tech. Rep. No. CMU-CS-94-163). Pittsburgh, PA, Carnegie Mellon University.

Baluja, S. and Caruana, R., 1995, Removing the Genetics from the Standard Genetic Algorithm. Technical Report. UMI Order Number: CS-95-141., Carnegie Mellon University.

Barrios, D., Manrique, D., Porras, J. and Rios, J., 2000, “Real-Coded Genetic Algorithms Based on Mathematical Morphology”, In Proceedings of the Joint IAPR international Workshops on Advances in Pattern Recognition (August 30 - September 01, 2000). F. J. Ferri, J. M. Iñesta, A. Amin, and P. Pudil, Eds. Lecture Notes In Computer Science, vol. 1876. Springer-Verlag, London, 706-715.

Bittencourt, A.C., 1997, “Optimizing Hydrocarbon Field Development Using a Genetic Algorithm Based Approach”, Ph.D. thesis, Stanford University.

Bittencourt, A.C., and Horne, R.N., 1997, "Reservoir Development and Design Optimization," SPE 38895 presented at the 1997 SPE Annual Technical Conference and Exhibition, San Antonio, Texas, October 5-8.

Chelouah, R. and Siarry, P., 2000, “A Continuous Genetic Algorithm Designed for the Global Optimization of Multimodal Functions”, Journal of Heuristics 6, 2 (Jun. 2000), 191-213.

Chen X., Qian J., Ni G., Yang S. and Zhang M., 2001, “An improved genetic algorithm for global optimization of electromagnetic problems”, Magnetics, IEEE Transactions on, vol.37, no.5, pp.3579-3583, Sep 2001.

Chiang, C.L., Chai, C.W. and Wang, C.A., 2006, “Improved Genetic Algorithm for Economic Dispatch of Power Systems having Special Units”, Cybernetics and Intelligent Systems, IEEE Conference on, vol., no., pp.1-6, June 2006.

Christie, M.A., Blunt, M.J., 2001, "Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques", SPE Reservoir Engineering and Evaluation, 4, 308-317.

Coats Engineering, 2008, http://www.coatsengineering.com

Deb, K., Agrawal, S., 1995, "Simulated binary crossover for continuous search space", Complex Systems 9(2), pp. 115--148.

Deb, K. and Agrawal, S., 1998, "Understanding interactions among genetic algorithm parameters. Foundations of Genetic Algorithms", pages 265-286.

Deb, K., Anand, A., and Joshi, D., 2002, "A computationally efficient evolutionary algorithm for real-parameter optimization", Evol. Comput. 10, 4 (Dec. 2002).

Djurisic, A.B., Rakic, A.D., Li, E.H., Majewski, M.L., Bundaleski, N., and Stanic, B.V., 1999, "Continuous Optimization Using Elite Genetic Algorithms With Adaptive Mutations", In Selected Papers From the Second Asia-Pacific Conference on Simulated Evolution and Learning on Simulated Evolution and Learning (November 24 - 27, 1998). B. McKay, X. Yao, C. S. Newton, J. Kim, and T. Furuhashi, Eds. Lecture Notes In Computer Science, vol. 1585. Springer-Verlag, London, 365-372.

DOE, 2008, U.S. Department of Energy website: (Fossil Energy Program)
http://www.fossil.energy.gov/programs/oilgas/eor/index.html

Eshelman, L.J. and Shaffer D.J., 1993, "Real-coded genetic algorithms and interval schemata", In D. L.Whitley (ed.), Foundations of Genetic Algorithms 2. San Mateo, CA: Morgan Kaufman, pp. 187–202.

Fang, K, 1980, ?Uniform design: application of number theory in test design,? ATA Mathematicae Applicatae Sinica, Sin 3, pp363-372.

Forrest, S. and Mitchell, M., 1993, "What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation", Mach. Learn. 13, 2-3 (Nov. 1993), 285-319.

GeoQuest, 2006, ECLIPSE Reference Manual 2006A. Schlumberger.

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

Goldberg, D. and Bridges, C., 1990, "An analysis of a reordering operator on a GA-hard problem", Biological Cybernetics, 62, 397–405.

Goldberg, D.E., Deb, K. and Clark, J.H., 1992, “Genetic algorithms, noise, and the sizing of population”, Complex Systems, 6, 333-362.

Gong, D., Pan, F. and Sun, X., 2002, “Research on a novel adaptive genetic algorithm”, Industrial Electronics, Proceedings of the 2002 IEEE International Symposium, vol.1, no., pp. 357-360 vol.1.

GPRS, 2006, GPRS User documentation, Stanford University.

Guyaguler, B., Horne, R.N., Rogers, L. and Rosenzweig, J.J., 2000, “Optimization of Well Placement in a Gulf of Mexico Waterflooding Project,” paper SPE 63221 presented at the 2000 SPE Annual Technical Conference and Exhibition, Dallas, Texas, October 1-4.

Guyaguler, B., Horne, R., 2001, “Uncertainty assessment of well placement optimization”, paper SPE 71625 presented at the 2001 SPE Annual Technical

Guyaguler, B., 2002, “Optimization of well placement and assessment of uncertainty”, Ph.D. thesis, Stanford University.

Hao, P., Jingling, Y. and Luo, Z., 2003, “Multi-Modal Function Optimization Problem for Evolutionary Algorithm”, In Proceedings of the 15th IEEE international Conference on Tools with Artificial intelligence (November 03 - 05, 2003). ICTAI. IEEE Computer Society, Washington, DC, 157.

Harikumar, R., Sukanesh, R. and Bharathi, P.A., 2004, “Genetic algorithm optimization of fuzzy outputs for classification of epilepsy risk levels from EEG signals”, TENCON 2004. 2004 IEEE Region 10 Conference, vol.C, no. pp. 588-591 Vol. 3, 21-24 Nov. 2004.

Haupt, R.L. and Haupt, S.E., 2004, “Practical genetic algorithms”, 2nd edition, John Wiley & Sons, New York.

Herrera, F., Lozano, M. and Verdegay, J,. 1998, “Tackling real-coded genetic algorithms: operators and tools for the behavior analysis”, Artificial Intelligence Review 12, 265-319.

Holland, J.H., 1975, “Adaptation in Natural and Artificial Systems”, The University of Michigan Press, Ann Harbor.

Jefferys, E.R., 1993, “Design Applications of Genetic Algorithms”, Paper SPE 26367, presented at the 68th Annual Technical Conference and Exhibition of the Society of Petroleum Engineers, Houston, Texas, October 3–6.

Kubalik, J., Rothkrantz, L.J.M., Lazansky, J., 2005, “Genetic Algorithm with Limited Convergence”, In Grana, M., Duro, R., d'Anjou, A., and Wang, P.P., (Eds.), Information Processing with Evolutionary Algorithms: From Industrial Applications to Academic Speculations. Springer-Verlag, pp. 216-235.

Kubalik, J., Posik, P. and Herold, J., 2006, “A Selecto-recombinative Genetic Algorithm with Continuous Chromosome Reconfiguration”, In Parallel Problem Solving from Nature - PPSN-IX. Heidelberg: Springer, s. 959-968.

Kubalik, J., 2004, “Binary or Real? Real-coded Binary!”, In proceedings of the 5th International Conference on Recent Advances in Soft Computing 2004, Nottingham, United Kingdom, 16 -18 December 2004.

Law, N.L. and Szeto, K.Y., 2007, “Adaptive genetic algorithm with mutation and crossover matrices”, Twentieth International Joint Conference on Artificial Intelligence, 2007, IJCAI-07., pp. 2330-2333, Jan 6-12, 2007.

Lee, Y.H., Marvin, A.C. and Porter, S.J., 1999, “Genetic algorithm using real parameters for array antenna design optimization”, High Frequency Postgraduate Student Colloquium, 1999, vol., no., pp.8-13.

Lee L.H. and Fang Y.L., 2000, “Developing a self-learning adaptive genetic algorithm”, in: Proceedings of the Third IEEE World Congress on Intelligent Control and Automation, vol. 1, pp. 619-624.

Li, Y.N, Dai, Y. and Ma, X., 2005, “A Heuristic Immune-Genetic Algorithm for Multimodal Function Optimization”, In Proceedings of the international Conference on Computational intelligence For Modeling, Control and Automation and international Conference on intelligent Agents, Web Technologies and internet Commerce Vol-2 (Cimca-Iawtic'06) - Volume 02 (November 28 - 30, 2005). CIMCA. IEEE Computer Society, Washington, DC, 36-40.

Liepins, G.E. and Vose, M.D., 1990, “Representational Issues in Genetic Optimization”, Journal of Experimental and Theoretical Artificial Intelligence 2: 101-115.

Mathworld, 2008, http://mathworld.wolfram.com

Michalewicz, Z., 1994, “Genetic Algorithms + Data Structures = Evolution Programs”, 2nd ed. New York: Springer-Verlag.

Montes, G., Bartolome, P. and Udias, A. L., 2001, “The Use of Genetic Algorithms in Well Placement Optimization”, SPE 69439, SPE Latin America and Caribbean Petroleum Engineering Conference, Buenos Aires, Argentina, Mar25-28, 2001.

Neumaier, A., 2004, “Complete Search in Continuous Global Optimization and Constraint Satisfaction”, pp. 271-369 in: Acta Numerica 2004 (A. Iserles, ed.), Cambridge University Press.

Nocedal, J. and Wright, S.J., 1999, “Numerical Optimization”, Springer, New York.

Onwunalu, J., 2006, “Optimization of nonconventional well placement using Genetic Algorithms and statistical proxy”, MS report, Stanford University.

Ozdogan, U., 2004, “Optimization of well placement under time-dependent uncertainty”, MS report, Stanford University.

Ozdogan, U., Horne, R.N., 2006, “Optimization of Well Placement With a History Matching Approach,” SPEREE (April 2006), 9(2), 135-145.

Pan, Y., and Horne, R.N., 1998, "Improved Methods for Multivariate Optimization of Field Development Scheduling and Well Placement Design," SPE 49055 presented at the 1998 SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, September 27-30. also Journal of Petroleum Technology, December 1998.

Radcliff, N.J., 1991, “Formal Analysis and Random Respectful recombination”, In Proc. 4th Int. Conf. on Genetic Algorithms, San Mateo, CA: Morgan Kauffman.

Rigot, V., 2003, “New well optimization in mature fields”, MS report, Stanford University.

Rudolph, G., 1994, "Convergence Analysis of Canonical Genetic Algorithms", IEEE Trans. on Neural Networks, special issue on Evolutionary Computation, vol. 5, no. 1, pages 96-101, Jan 1994.

Shu, J., 2005, "Comparison of various techniques for computing Well Index", MS report, Stanford University.

Spall, J.C., 2004, "Stochastic Optimization", in Handbook of Computational Statistics (J. Gentle, W. Härdle, and Y. Mori, eds.), Springer–Verlag, New York, pp. 169–197.

Srinivas, M., Patnaik, L., 1994, "Adaptive probabilities of crossover and mutation in genetic algorithms", IEEE Trans. Syst., Man, and Cybernetics, vol. 24, no. 4, pp. 656--667, Apr 1994.

Varahram, A.A. and Rashed-Mohassel, J., 2002, "Side lobe level optimization using modified genetic algorithm", Antennas and Propagation Society International Symposium, 2002. IEEE, vol.1, no., pp. 742-745 vol.1.

Varahram, A., Rashed-Mohassel, J. and Mafinezhad, K., 2003, "Design of matching transformers for UHF band power splitters using modified genetic algorithms", Antennas and Propagation Society International Symposium 2003. IEEE, vol.2, no., pp. 96-99 vol.2, 22-27 June 2003.

Whitley, D., 1994 "Genetic Algorithm Tutorial", Statistics and Computing (4):65-85.

Wikipedia, 2008, http://en.wikipedia.org

Yeten, B., Durlofsky, L.J. and Aziz, K., 2002 "Optimization of nonconventional well type, location and trajectory", paper SPE 77565 presented at 2002 SPE Annual Technical Conference and Exhibition held in San Antonio, 29 September – 2 October 2002.

Yeten, B., 2003, "Optimum deployment of nonconventional wells", Ph.D. thesis, Stanford University.

Zhao, S., Zhao, J. and Jiao, L., 2005, "Adaptive Genetic Algorithm Based Approach for Evolutionary Design and Multi-objective Optimization of Logic Circuits", In Proceedings of the 2005 NASA/DoD Conference on Evolvable Hardware (June 29 - July 01, 2005). EH. IEEE Computer Society, Washington, DC, 67-72.

Zhu, K.Q., 2003, “A Diversity-Controlling Adaptive Genetic Algorithm for the Vehicle Routing Problem with Time Windows”, In Proceedings of the 15th IEEE international Conference on Tools with Artificial intelligence (November 03 - 05, 2003). ICTAI. IEEE Computer Society, Washington, DC, 176.

Appendix I

Contents of this Appendix are provided on a CD that is part of this report. Here we describe the files and folders inside the CD. We have also provided a sample input file for the well placement optimization program at the end of this Appendix. The CD contains two folders: ***CGA*** and ***Inputs***.

The ***CGA*** folder includes following Matlab files used in this work:

1. ***main.m***: This is the main program for performing continuous GA search for optimal multi-lateral wells in the reservoir.
2. ***read_input.m***: This function reads all the required input parameters from the input file.
3. ***init_pop.m***: This function generates an initial population for the GA search.
4. ***ECL_well.m***: This function reads each chromosome and creates the input file ECLIPSE.
5. ***run_sim.m***: This function performs ECLIPSE simulations for each individual chromosome in the population and reads the simulation results.
6. ***rank_pop.m***: This function evaluates the objective values for each individual in the current population and rank them in descending order.
7. ***reproduce.m***: This function reproduces the next generation from the current generation.
8. ***Vector2Indiv.m***: This function converts a population represented in a vector format to a *struct* format.
9. ***uniqueRand.m***: This function produces an array of unique random numbers. This is required for implementing reproduction.
10. ***showWell.m***: This function visualizes all the wells in a given individual
11. ***creatObjReport.m***: This function outputs a file containing parameterized vectors of each new individual generated and its objective.

12. ***creatObjMatrix***: This function generates a sorted matrix of all new individuals and their objective. This file is used multiple initial population implementation.
13. ***segDistance.m***: This function calculates the closest distance between two line segments.
14. ***Well_distance.m***: This function calculates the closest distance between two multi-lateral wells.
15. ***hcurve.m***: This function calculates the coordinated for a curved well.
16. ***ECL_CurvedWell.m***: This function creates the input file ECLIPSE when curved well description is used.
17. ***mainParallel.m***: This is the main program for performing continuous GA search for optimal multi-lateral wells using multiple initial population.
18. ***GPRS_well.m***: This function calculates the well connection factors for curved wells and writes the well file for GPRS.

The ***Inputs*** folder includes:

1. ***input.par***: This is a sample input file for our well placement optimization program.
2. ***ECL Input***: This folder contains inputs for ELCIPSE simulation on upscaled SPE10 model.
3. ***GPRS Input***: This folder contains inputs for GPRS simulation on $40 \times 40 \times 7$ channeled reservoir.

input.par:

```
 EclGArun       % Filename to store progress (.mat will be appended)
Spe10_2000     % eclipse data file name

-- Continuous Genetic algorithm parameters
0.06   0.2   2    % mutation (probability/factor/power)
0.2     0.2        % crossover (probability/factor)
2                  % ranking scale
0.3     0.8        % fraction of population selected as potential parents
30                 % initial population size
200                % Max number of Generations

-- Objective funtion

2                 % Objective function: 1-FOPT; 2-NPV; 3-UTILITY
1  0  1  0        % Number realizations / risk10 / risk50 / risk90

-- Economics

10                 % APR
60 -2.5 1 -3.5   % oil, water, gas selling prices, water injection cost ($/bbl)
2000000 3        % CAPEX($), OPEX($/bbl)
3000000          % cost of junction milling
1000000          % cost of prod and injector
3000             % cost per foot, drilling, completion etc
```

```
-- Wells
2               % # producers
1               % # injectors
0               % # junction points on main trunk

100   1000      % main bore length on xy plane (Min and Max)
0     500       % main bore length on z plane  (Min and Max)
100   300       % laterals length on xy plane  (Min and Max)
0     50        % laterals length on z plane   (Min and Max)
0.2   1         % lateral position on the well  (Min and Max)

-- Simulation
ECL           % Simulator

10  20  10    % NX NY NZ
120 110 68    % dX dY dZ

0             % Dynamic # of wells     (yes/no = 1/0)
0             % Dynamic # of Junctions (yes/no = 1/0)
0             % Dynamic Injection rate (yes/no = 1/0)

2000       % PROD BHP
0                                       % BHP controled injection  (yes/no = 1/0)
20000 12000 % INJE Rate target / BHP constraint
4          % Duration of simulation (YEARS)
-- END
```