

OPTIMIZATION OF WELL PLACEMENT
AND ASSESSMENT OF UNCERTAINTY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF PETROLEUM ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Barış Güyagüler
June 2002

© Copyright 2002
by
Barış Güyagüler

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Dr. Roland N. Horne
(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Dr. Khalid Aziz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Dr. Jef Caers

Approved for the University Committee on Graduate Studies:

Acknowledgements

First and foremost acknowledgements go to my advisor Roland N. Horne for his vision on optimization and consistent guidance and motivation. Also thanks to the Petroleum Engineering faculty for their valuable input throughout the course of this study.

Thanks to the financial support from SUPRI-D Research Consortium on Innovation in Well Testing and SUPRI-B Research Consortium on Reservoir Simulation without which this study would not have been possible.

Appreciation also goes to BP-Amoco and the DeepLook consortium for the permission to publish information from the Pompano field and to Elf for supplying their history matched dataset for the PUNQ-S3 reservoir.

Also thanks to Saudi Aramco for their financial support that made this work possible.

Abstract

Determining the best location for new wells is a complex problem that depends on reservoir and fluid properties, well and surface equipment specifications, and economic criteria. Various approaches have been proposed for this problem. Among those, direct optimization using the simulator as the evaluation function, although accurate, is in most cases infeasible due to the number of simulations required.

This study proposes a hybrid optimization technique (HGA) based on the genetic algorithm (GA) with helper functions based on the polytope algorithm and the kriging algorithm. Hybridization of the GA with these helper methods introduces hill-climbing into the stochastic search and also makes use of proxies created and calibrated iteratively throughout the run, following the idea of using cheap substitutes for the expensive numerical simulation. Performance of the technique was investigated by optimizing placement of injection wells in the Gulf of Mexico Pompano field. A single realization of the reservoir was used. It was observed from controlled experiments that the number of simulations required to find optimal well configurations was reduced significantly. This reduction in the number of simulations enabled the use of full-scale simulation for optimization even for this full-scale field problem. Well configuration and injection rate were optimized with net present value maximization of the waterflooding project as the objective.

The optimum development plan for another real world reservoir located in the Middle East was investigated. Optimization using the numerical simulator as the evaluation function for the field posed significant challenges since the model has half a million cells. The GA was setup in parallel on four processors to speed up the optimization process. The optimal deployment schedule of 13 predrilled wells that

would meet the production target specified by the operating company was sought. The problem was formulated as a traveling salesman problem and the order of wells in the drilling queue was optimized.

Ways to assess the uncertainty in the proposed reservoir development plan were also investigated since we never possess the exhaustive information about the *true* reservoir but rather we may have geostatistical realizations of the truth constructed from the information available. An approach that can translate the uncertainty in the data to uncertainty in terms of monetary value was developed. In this study the uncertainties associated with well placement were addressed within the utility theory framework using numerical simulation as the evaluation tool. The HGA was used to reduce the computational burden of making numerous numerical simulations. The methodology was evaluated using the PUNQ-S3 model, which is a standard test case that was based on a real field and was used for the PUNQ project in context of the EU-Joule program. Experiments were carried on 23 history-matched realizations and a truth case was also available. The results were verified by comparison to exhaustive simulations. Utility theory not only offered the framework to quantify the influence of uncertainties in the reservoir description in terms of monetary value but also provided the tools to quantify the otherwise arbitrary notion of the risk attitude of the decision maker.

To Onur, Ceyda and Tefik...

Contents

1	Introduction	1
1.1	Statement of the Problem	1
1.2	Literature Survey	4
1.2.1	Methods	4
1.2.2	Optimization of Reservoir Development	9
1.2.3	Survey Results	11
2	Theoretical Foundations	13
2.1	Genetic Algorithms	13
2.1.1	String Representation of Solutions	14
2.1.2	Population	15
2.1.3	Fitness Based Selection	15
2.1.4	GA Operators	15
2.1.5	The Schema Theorem	16
2.1.6	Deception	19
2.1.7	Practicalities	20
2.1.8	Extensions	21
2.2	Polytope Method	22
2.3	Proxy Construction	24
2.3.1	Kriging	24
2.3.2	Neural Networks	27
2.4	The Hybrid Genetic Algorithm (HGA)	30
2.4.1	Hybridization	30

2.4.2	Other Helper Techniques	33
3	Well Placement Optimization	35
3.1	The Synthetic Model	35
3.2	Producer Placement	35
3.3	Injector Placement	38
3.4	Well Indexing	40
4	Optimization of Waterflooding	45
4.1	Reservoir Description	45
4.2	Controlled Experimentation	47
4.3	Multiple Well Location and Rate Optimization	52
4.4	Comparison with the DeepLook Approach	58
4.5	The Quality Map Approach	59
4.5.1	Quality Map Definition	59
4.5.2	Optimization with the Quality Map	60
5	Real-World Field Development	68
5.1	Problem Definition	68
5.1.1	Optimization Objective	68
5.1.2	Production Constraint Handling	69
5.2	Optimization Approach	71
5.2.1	Formulation	71
5.2.2	Solution Representation	72
5.3	Results	75
6	Assessment of Uncertainty	78
6.1	Utility Framework	79
6.1.1	Decision Tree Construction	79
6.1.2	Utility Theory and Utility Functions	80
6.1.3	The Certain Equivalent	82
6.1.4	Axioms of Utility Theory	83

6.2	Random Function Formulation	84
6.3	Application to the PUNQ-S3 Reservoir	86
6.3.1	The PUNQ-S3 Model	87
6.3.2	Exhaustive Runs	89
6.3.3	Analysis of the Exhaustive Dataset	90
6.3.4	Application of the Utility Framework	95
6.3.5	Application of Random Function Formulation	96
6.4	Discussion	102
7	Sensitivity of Algorithm Parameters	104
7.1	GA Performance Sensitivity	104
7.1.1	Rule of Thumb Determination of GA Parameters	104
7.1.2	Crossover and Mutation Probabilities	105
7.1.3	Selection Type	110
7.2	Polytope Strategies	111
7.3	Discussion	112
8	Concluding Remarks	113
8.1	Conclusions	113
8.2	Suggestions for Future Work	114
A	Algorithm Parallelization	116
A.1	Value of Parallelization	116
A.2	Parallel Setup of Optimization	117
B	HGA Implementation	120
B.1	The Program	120
B.2	The Keywords	123
	Nomenclature	129
	Bibliography	132

List of Tables

2.1	The GA vocabulary.	14
3.1	Well indexing schemes.	42
4.1	Parameters for net present value calculations.	49
4.2	Means (m) and standard deviations (σ) of the number of simulations and the success percentage (%) to correctly locate global optimum for the composite algorithms: Genetic Algorithm (GA), Polytope algorithm (P), Neural Network (NN), Kriging (K) and Error analysis (E).	51
4.3	Search space sizes for the problems considered for the Pompano field.	55
4.4	Number of injectors (n_{inj}), pumping rates in MSTB/D, number of simulations (n_{sim}) and the resulting incremental NPVs in MM\$ for the HGA runs.	55
4.5	Comparison of resulting incremental net present value calculations for the quality map and direct optimization approaches.	65
6.1	Parameters for net present value calculations.	91
6.2	The percent differences from the actual mean NPV and the correlations between the GA sampled and actual means for the three cases where the GA was allowed 250, 500 and 1000 simulations.	100
7.1	Effect of selection type on algorithm performance; the mean and standard deviation of the number of simulations required to locate the optimum well location for 100 optimization runs.	111

7.2	Effect of polytope strategy on algorithm performance; the mean and standard deviation of the number of simulations required to locate the optimum well location for 100 optimization runs.	112
B.1	Keywords for the HybridGeneticAlgorithmParameters class.	125
B.2	Keywords for the PopulationParameters class.	126
B.3	Keywords for the PolytopeAlgorithmParameters class.	126
B.4	Keywords for the KrigingParameters class.	127
B.5	Keywords for the SimulationParameters class.	127
B.6	Keywords for the CostParameters class.	128

List of Figures

2.1	Simple GA flowchart.	16
2.2	A polytope and possible polytope steps in two dimensions.	24
2.3	A two-dimensional parameter space with irregularly spaced data (circles) and a location where value is to be estimated (cross).	25
2.4	A typical neural network unit.	28
2.5	The structure of a feed-forward neural network.	29
2.6	HGA flowchart with the conventional GA steps followed by the polytope steps and the proxy approach.	31
2.7	Illustration of the polytope constructed from the best solutions in the generation for a one-dimensional problem.	32
2.8	Illustration of the polytope constructed from the best solutions in the generation for a one-dimensional problem.	32
2.9	Illustration of the polytope constructed from the solutions closest to the best solution in the database for a one-dimensional problem.	33
3.1	The permeability distribution for the synthetic model.	36
3.2	Cumulative oil production surface for the single production well placement problem for the 32×32 synthetic case obtained by exhaustive simulation.	37
3.3	Comparison of the means (crosses) and the standard deviations (error bars) for different combinations of Genetic Algorithm (GA), Polytope method (P) and Kriging (K) for the producer placement problem on the 32×32 synthetic case.	38

3.4	Cumulative oil production surface for the single injection well placement problem for the 32×32 synthetic case obtained by exhaustive simulation.	39
3.5	Comparison of the means (crosses) and the standard deviations (error bars) for different combinations of Genetic Algorithm (GA), Polytope method (P) and Kriging (K) for the injector placement problem on the 32×32 synthetic case.	40
3.6	One dimensional objective functions generated by mapping from the 32×32 synthetic injector placement problem by different well indexing schemes.	41
3.7	The Hilbert curve used to map the 32×32 two-dimensional surface into a single-dimension.	43
3.8	Comparison of the efficiency of the HGA for different types of indexing; crosses indicate the means and the error bars indicate the standard deviations of 50 runs.	43
4.1	The numerical model for the Pompano field.	47
4.2	Illustration of the process for the evaluation of well configurations for the optimization of Pompano field development.	48
4.3	The incremental net present value surface for the single injector well placement problem generated by exhaustive simulations.	49
4.4	Comparison of the mean (crosses) and the standard deviation (error bars) for different combinations of Genetic Algorithm (GA), Polytope method (P), Kriging (K) and Error analysis (E).	51
4.5	Effect of invalid point interpretation (IPI) and local mutation (LM) on algorithm performace; crosses indicate the means and the error bars indicate the standard deviations of 50 runs.	52
4.6	Evolution of the kriging estimation around the global optimum for the single injector placement problem.	53
4.7	Optimum well location for the one injector placement problem for the Pompano field.	56

4.8	Optimum well location for the two injector placement problem for the Pompano field.	56
4.9	Optimum well location for the three injector placement problem for the Pompano field.	57
4.10	Optimum well location for the four injector placement problem for the Pompano field.	57
4.11	The 25 candidate well locations selected by the DeepLook consortium prior to optimization and the locations proposed by the HGA.	59
4.12	Resulting optimized single well location with the quality approach for the Pompano field; red-white crosses indicate the location of existing producers and the proposed well locations have unit cell-weight.	61
4.13	Resulting optimized two well locations with the quality approach for the Pompano field; red-white crosses indicate the location of existing producers and the proposed well locations have unit cell-weight.	62
4.14	Resulting optimized three well locations with the quality approach for the Pompano field; red-white crosses indicate the location of existing producers and the proposed well locations have unit cell-weight.	63
4.15	Resulting optimized four well locations with the quality approach for the Pompano field; red-white crosses indicate the location of existing producers and the proposed well locations have unit cell-weight.	64
4.16	Well locations suggested by the quality approach and direct optimization using the numerical simulator as the evaluation function for the Pompano field one injector placement problem.	65
4.17	Well locations suggested by the quality approach and direct optimization using the numerical simulator as the evaluation function for the Pompano field two injector placement problem.	66
4.18	Well locations suggested by the quality approach and direct optimization using the numerical simulator as the evaluation function for the Pompano field three injector placement problem.	66

4.19	Well locations suggested by the quality approach and direct optimization using the numerical simulator as the evaluation function for the Pompano field four injector placement problem.	67
5.1	The target rate and production forecast for the REAL field when no new wells are deployed.	69
5.2	The target rate and production forecast for the REAL field when new wells are deployed and the wells are shut when production target is exceeded.	70
5.3	The target rate and production forecast for the REAL field when new wells are deployed and the well rates are scaled back to meet the target rate in the case of overproduction.	71
5.4	Decision tree for the four well ordering problem.	74
5.5	Decision tree for the three well ordering problem, the integers on the decision nodes are the well indices.	74
5.6	The path and the well order resulting from the branching order of (1,2).	75
5.7	The algorithm to decode a branching order (\mathbf{b}) into a well order (\mathbf{w}) by modifying a vector of tasks that remain (\mathbf{r}).	76
6.1	The well placement decision tree.	80
6.2	The well placement decision tree with event nodes collapsed into expected utility.	81
6.3	The exponential utility function for different exponent values, R	82
6.4	Illustration of the suggested methodology to optimize in the presence of an uncertain reservoir model; the \sim indicates a random variable.	86
6.5	Histograms of the cumulative distributions for the 23 realizations of the PUNQ-S3 field, the cross indicates the true value.	87
6.6	Match of history with the truth case of the Elf data set for the PUNQ-S3 model, dotted lines were generated by numerical simulation on the 23 permeability realizations and the solid line was generated by numerical simulation on the true permeability field.	88
6.7	Truth case permeability fields for the PUNQ-S3 model.	89

6.8	Results of the PUNQ-S3 comparative study (plot taken from the Netherlands Institute of Applied Geoscience, TNO-PUNQ online presentation, http://www.nitg.tno.nl/punq).	90
6.9	NPV histogram of all the flow simulation runs on the PUNQ-S3 realizations.	91
6.10	Mean of NPVs over the 23 realizations for the PUNQ-S3 model.	92
6.11	Standard deviation of NPVs over the 23 realizations for the PUNQ-S3 model.	93
6.12	True NPVs (sorted) at the optimum locations of the realizations and the spreads of these locations for the PUNQ-S3 model.	94
6.13	Two-dimensional histogram of the optimum well locations of the 23 realizations for the PUNQ-S3 model.	94
6.14	The normal distribution of NPVs obtained by fitting to the histogram for a particular well location for the PUNQ-S3 field.	96
6.15	The gradual transformation of the decision makers attitude from extreme risk averse to extreme risk prone for the PUNQ-S3 problem.	97
6.16	The three-dimensional surfaces of the mean and standard deviations of the NPV maps of the 23 realizations for the PUNQ-S3 model.	98
6.17	Standard deviations (error bars) and expected NPVs (crosses) of optimal decisions as a function of the risk aversion coefficient, r	99
6.18	Histogram for the number of times the GA visited well locations for the random function formulation of the PUNQ-S3 well placement problem.	100
6.19	The NPVs of the solutions visited by the GA and the number of revisits for the random function formulation of the PUNQ-S3 well placement problem.	101
7.1	The average number of simulations that the GA consumed to locate the global optimum for different values of mutation probability (p_m) and crossover probability (p_c) for the Pompano problem.	106

7.2	The average number of simulations that the GA consumed to locate the global optimum for different values of mutation probability (p_m) and crossover probability (p_c) for the PUNQ-S3 problem.	107
7.3	The variance of the number of simulations that the GA consumed to locate the global optimum for different values of mutation probability (p_m) and crossover probability (p_c) for the Pompano problem.	108
7.4	The variance of the number of simulations that the GA consumed to locate the global optimum for different values of mutation probability (p_m) and crossover probability (p_c) for the PUNQ-S3 problem.	109
A.1	Cartoon analogy of how the process organizer works; two machines helping processes cross the river.	118
A.2	Parallelization setup.	119
B.1	C++ classes making up the HGA; arrows indicate inheritance.	121

Chapter 1

Introduction

1.1 Statement of the Problem

Decisions have to be made at every level of reservoir development. For many cases, optimal decisions are dependent on many nonlinearly correlated parameters, which makes intuitive judgement difficult. In such cases automated optimization is an option. The impact of development decisions on the success or failure of a project can be significant. Also, most of the time, a slightly better decision may lead to a considerable increase in the value of the project. Thus, decisions should be based on the most relevant and accurate tools available. However, such tools are most often computationally expensive.

Consider the problem of finding the optimum locations for infill wells in waterflooding. Numerical models are the basis for our decision since they are detailed predictive tools that are able to evaluate the complex interactions of the dependent parameters, such as reservoir and fluid properties, well and surface equipment specifications, and economic criteria. On the other hand, the numerical models used in the oil industry for reservoir evaluation are generally too CPU intensive to couple with an automated optimization scheme. Two things would make the optimization process feasible:

- Reduction of the necessary number of simulations.

- Approximation of the numerical model with a less expensive tool.

Various researchers have looked into these approaches. Reduction of the number of evaluations is the first idea that comes to mind. However special care should be taken to avoid local extrema that will lead to suboptimal decisions. Approximation of the full numerical model is also an option. However decisions based on an approximate tool, unless the approximation is either very good or fortunately correct, may not be optimal. This study proposes a methodical solution to the problem by utilizing both of these approaches together.

The following list summarizes the requirements that must be met by any solution approach that would optimize reservoir engineering problems as robustly as possible:

Computational feasibility - This requirement necessitates an efficient algorithm.

Ability to avoid local optima - Local optima can be avoided by a stochastic rather than deterministic algorithm. Stochastic algorithms also rely less on the initial guess.

Flexibility - The proposed approach should be able to handle continuous parameters as well as discrete. Such a flexibility allows for optimization of discrete decision variables together with continuous parameters. In addition, since different decisions may lead to different numbers of parameters to be optimized, the ability to handle different numbers of parameters simultaneously is also required.

Generality - The approach should not be problem-specific. The method should adapt to the characteristics of the problem as the optimization progresses.

We now summarize the tools investigated as ways to meet these requirements.

Genetic Algorithm (GA) was chosen as the basis of the algorithms. GAs provide a good start to meet some of the requirements listed earlier. GAs are stochastic algorithms that work with discrete parameters. Continuous parameters are handled by fine discretization. The simple GA was extended in this study with some improvements that have been suggested in the literature. The proposed GA also allows

solutions of different structures, for instance different number of parameters, to be modified simultaneously for optimality. GAs are also easily hybridized (coupled) with other techniques. Another favorable GA property is that they are easily parallelized since they modify a set of solutions simultaneously.

Some of the same traits that make the GA robust and powerful also make it slow and inexact in refinement of the solution. The GA typically has rapid initial progress during the search, but has problems locating the final optimal solution. Because of these limitations, this study investigated specific helper methods that significantly improve the efficiency. Hill-climbing effects were introduced, through the polytope method, in order to resolve the GA pitfall of poor refinement. Several other techniques were employed to further aid search.

The idea of using a cheaper approximation of the numerical model was also investigated through the proxy approach. A proxy of the numerical simulator is created and improved at every GA iteration. This approach has several advantages over creating a proxy with a big initial investment of numerical simulations. First of all, the determination of the number of simulations to create a sufficiently accurate proxy does not always have an intuitive basis. Besides, with iterative proxy improvement, the next solution vector to be simulated is chosen *intelligently* by the GA and is dependent on the results of the previous simulations. This dependence on previous simulations allows the algorithm to adapt itself to the characteristics of the problem as the run progresses. Ordinary kriging and neural networks were investigated as proxies in this study.

It was also realized that the numerical model with which the optimization algorithm evaluates the potential well configurations is constructed with data that are sparse thus numerical simulation forecasts are uncertain. These uncertainties further complicate the optimization problem. Decision analysis tools and the utility theory framework were utilized to deal with the uncertainty. The problem was represented as a decision tree and utility functions were used to quantify risk attitude.

In order to handle the cases where the utility framework becomes computationally infeasible, the problem was formulated as the optimization of a random function. This approach was demonstrated to be a quick and approximate way of dealing with the

uncertainties.

The algorithm developed here is referred to as the Hybrid Genetic Algorithm (HGA). The following chapters explain the tools that make up the HGA and present some synthetic cases and application to two real world cases.

1.2 Literature Survey

1.2.1 Methods

Many real world problems are not straightforward, and this is often true for reservoir development problems as well. Even some of the simple synthetic cases considered in this study had several local optima. The real reservoir problems we considered have many feasible but not optimal solutions, which would all lead to a suboptimal reservoir development plan. The existence and the need to avoid these local optima drove us to use stochastic optimization techniques. Some of the most significant stochastic optimization techniques are summarized here. Also, methods to construct proxy functions to aid optimization and uncertainty assessment tools are examined.

Optimization Techniques

Stochastic optimization algorithms gain an increasing amount of attention as we realize that even seemingly simple real world problems have local optima. Stochastic algorithms are effective at avoiding local optima by continuing to search somewhat randomly in areas beyond *attractive* zones. *Greedy* algorithms such as simple hill-climbers will tend to converge to the local optimum closest to the starting point. In many cases, the chance that this solution is the global optimum is very slim.

Stochastic optimization algorithms should not be confused with *random search*. Stochastic algorithms employ a *randomized* search; this does not imply directionless or unstructured search. Through algorithm parameters, it is possible to tune the balance between *exploitation* and *exploration*. Gradient-based algorithms can be seen as pure exploiters, they therefore lack robustness. In this section we will now introduce some robust stochastic optimization algorithms.

A very basic approach to avoid local optima would be to carry out gradient-based searches from random initial points. This naive approach is obviously *blind* in terms of seeing the big picture. One can also try to choose several starting points based on engineering judgement, however many problems have too many factors to evaluate manually. In particular, the time factor is hard to predict without physical simulation of the underlying process. Here we focus on structured and stochastic automatic optimization techniques that have been applied successfully to real world problems.

Simulated Annealing (SA) (Metropolis *et al.*, 1953) was inspired by an analogy to the thermodynamic process of freezing liquids and crystallization or metal cooling and annealing. When a liquid is cooled down slowly, a pure crystal is formed and this state is believed to be the minimum energy state of atoms. The interesting fact is that nature is able to find this minimum energy state when the system is cooled down sufficiently slowly. The analogy in optimization is that the search state is initially perturbed randomly and unlike in greedy algorithms a worsening step is sometimes accepted with a probability decreasing with the number of steps taken. The magnitude of perturbation also decreases as the algorithm progresses. This approach is able to avoid local optima and is often effective if its parameters are tuned optimally. SA parameters are often very problem-specific and require experimentation and fine tuning.

During the last 30 years, there has been a growing interest in problem-solving systems inspired from the principles of natural selection and genetics (Michalewicz, 1996). These optimization techniques are often referred to as evolutionary techniques and they are based on nature's way of finding the individual fittest to its environment. Evolutionary methods differ in the tools they employ and their applicability. Here, we discuss some of the most significant evolutionary techniques.

Fogel *et al.* (1966) introduced Evolutionary Programming. Briefly, Fogel *et al.* applied perturbation (mutation) and natural selection on a population of solutions, with increasing population size at each iteration.

Evolution strategies, introduced by Rechenberg (1973), evolved a single point rather than a population. At each iteration, the point is perturbed by an amount

drawn randomly from a Gaussian distribution with zero mean and the resulting solution is accepted if and only if it improves the objective function. This idea is borrowed from nature since large changes in nature occur less often.

Holland (1975) introduced Genetic Algorithms (GAs). Different from the previous evolutionary techniques, Holland employed ideas from Mendel's genetics as well as Darwin's theory of natural selection. GAs use natural selection, mutation and crossover to modify a set of solutions (population) simultaneously, in the effort to evolve the population to its globally optimal solution. GAs have mathematical foundations of how and why they work. Numerous improvements and modifications to GAs have been proposed since their debut.

Another interesting evolutionary algorithm was proposed by Koza (1992). This algorithm, named Genetic Programming, evolves hierarchically structured computer programs to solve the desired problem. The evolved programs consist of trees made up of terminals and functions.

For the purposes of reservoir development optimization, GAs seem to be a very appropriate choice. In comparing performance to SA, GAs were shown to solve problems with the same order of computational effort (Goldberg, 1989; Davis, 1991) for a wide range of problems, given that both algorithms are tuned near-optimally. This is not surprising since both algorithms provide the flexibility to adjust exploitation and exploration. However, considering the reservoir development problem, GAs have several advantages over the other stochastic techniques discussed here. First of all discrete parameters, that often show up in reservoir development optimization, are handled intuitively. Continuous parameters are handled by fine discretization, which is sufficient for our purposes. GAs can work with various data structures simultaneously, which enables us to optimize the problem definition itself. For instance, consider the well placement problem; in GAs, we can introduce the number of wells as an unknown. Parallelization of GAs is also straightforward, since they evaluate a population of solutions simultaneously. In addition adaptive GAs have been proposed, in which the algorithm parameters are adapted to the problem as the run progresses. GAs are also suitable for hybridization with other algorithms.

In some of the synthetic well placement problems considered in this study, the GAs

have very rapid initial progress towards the optimum, however they lack refinement. Hill-climbing effects have been introduced through the polytope method (Nelder and Mead, 1965), also known sometimes as the simplex method. The combination of GA and polytope methods has been used previously for the well placement problem by Bittencourt and Horne (1997). Algorithms that combine local gradient search and GAs may be referred to as hybrid GAs or sometimes as memetic algorithms. Bittencourt and Horne called the combined algorithm the Hybrid Genetic Algorithm (HGA), a term that will also be used here.

Proxy Generation

Since the appropriate evaluation function for reservoir development (e.g. the numerical simulator) is most often very expensive computationally, the idea of using a cheaper approximation of this expensive function is attractive. In the literature, this cheaper alternative is referred to as a proxy function or a surrogate model. This function can be seen as an interpolating and extrapolating function, or as multivariable regression. There are many candidates for the proxy. Some of the most widely used and efficient ones are discussed here.

Polynomial regression is the most common of the proxy generating methods. The idea is to fit a polynomial of degree n to a set of *observations* computed using the full evaluation function. The idea is extended easily to multiple dimensions. Problems arise when the data are erratic or the dimensions are high. In such cases, very high order polynomials are necessary for a reasonable fit to the data, which causes a lack of generalization. Splines have been introduced to approximate the data with local, low order polynomials that are continuous (first derivative exists). This works well, however multidimensional splines are very complex and this approach is still an area of research. Also polynomial regression is not necessarily data exact.

Kriging (Matheron, 1965) is an interpolation-extrapolation algorithm used widely in earth sciences to predict earth properties. With kriging, we have control over two-point statistics of our estimations through the variogram (although this variogram is not reproduced exactly by the estimations). Hence the configuration of data with respect to the point to be estimated plays an important role. Although not utilized

commonly in earth sciences practices, the algorithm can be extended to multiple dimensions without significant increase in complexity. Kriging estimations are also exact at data locations. Another favorable property is that, given a set of data, when using all data for estimation (i.e. global kriging, see Goovaerts, 1997), the covariance matrix has to be calculated only once. Afterwards, each estimation requires only a matrix multiplication. It is also possible to obtain estimation variances at estimated points that helps extend the search to the least visited areas of the search space.

Neural Networks (NNs) (Anderson, 1995) have also been utilized in function approximation. NNs have gained acceptance as powerful interpolation techniques in the engineering community. NNs are not data exact, but they are able to represent very complex functions, continuous or discrete. NNs can map multidimensional spaces into a lower-dimensional space, thus can be used easily for multivariable regression.

We have tested both kriging and NNs as proxy functions in this study. We choose kriging mainly for its exactness property and easy extension to multidimensional space. We choose NNs for their applicability in a very broad range of problems.

Uncertainty Assessment Tools

Decision analysis tools to quantify and manage risk have been utilized across a wide range of industries (Chacko, 1993). Specifically the utility framework provides an established framework that enables the quantification and management of uncertainty (DeGroot, 1970). The utility framework is intuitive and very useful since it honors the fact that every decision maker who is given options with probabilistic outcomes would act according to their own risk attitudes which may be very different. The utility theory provides the framework and the tools to quantify the rather abstract notion of risk attitude and helps in making decisions in the presence of uncertainty (Holloway, 1979).

There have been several applications of decision analysis tools in the petroleum industry as well (Simpson *et al.*, 2000; Thankur, 1995; Jonkman *et al.*, 2000; Erdogan *et al.*, 2001; Sarich, 2001). These applications of decision analysis tools were mostly used during exploration and initial development stages of reservoirs (Jonkman *et al.*, 2000). Application of the decision theory framework coupled with full field-scale

numerical simulation has not been common mainly due to computational infeasibility and the lack of involvement of the fields of petroleum engineering and management science.

The problem was also formulated as the optimization of a random function. The GA is known to be able to cope with random functions and there have been applications to problems in industries other than petroleum engineering (Goldberg, 1989).

1.2.2 Optimization of Reservoir Development

In the petroleum literature, the word *optimization* has often been misused to describe the act of *manually trying several things and picking the best*. Here, *optimization* refers to objective function maximization (or minimization) with a structured and automated algorithm that generates solution vectors according to the function values of previously evaluated solution vectors.

Previous researchers in the petroleum literature have approached the optimization of reservoir development in different ways. These approaches can be grouped into two major headings, direct optimization and use of a proxy.

Direct Optimization

Rian and Hage (1994) presented the basic setup for automatic optimization of well locations using a numerical simulator. They pointed out that optimization with conventional full-scale models has computational constraints due to the number of simulations required, thus they proposed a faster but limited front-tracking simulator as their objective evaluation tool.

Beckner and Song (1995) formulated the problem of optimization and scheduling of wells as a traveling-salesman problem and used simulated annealing to optimize well locations and drilling schedule. Their emphasis was on problem setup, rather than the methods used for optimization. They showed that the computational time was feasible for such automatic optimization, with the numerical model used ($36 \times 3 \times 3$ grid blocks). They also pointed out that optimization algorithms coupled with the numerical model has the potential to evaluate the nonlinear effects of the optimized

parameters.

Bittencourt and Horne (1997) investigated optimization of well placement using a hybrid of GA and polytope method. This method was used to estimate the optimal locations for the placement of 33 wells. Bittencourt and Horne indexed active cells only, arguing that (i, j) indexing is not suitable because the optimization algorithm could place wells in inactive regions. In the present study, several types of well-block indexing were investigated for a synthetic case and it was discovered that (i, j) indexing of wells is more suitable for optimization algorithms since other kinds of indexing may introduce artificial noise due to the discontinuities in the indexed search space.

Güyagüler and Gümrah (1999) optimized production rates for a gas storage field using GAs coupled with a numerical simulator. This study demonstrated the limitations of using approximated methods by comparing production schedules obtained by linear programming to those obtained by GAs. The approach used simple GAs, which are shown here and by other researchers to have a lot of room for improvement.

Use of a proxy

Rogers and Dowla (1994) trained a NN with a large initial computational investment of numerical simulations, and then utilized this NN proxy of the simulator to optimize groundwater remediation.

Aanonsen *et al.* (1995) selected a set of well configurations to simulate and then constructed response surfaces from the simulation results, using several regression techniques (including kriging). The study then estimated the well location on this response surface, also assessing uncertainty in a very simple manner. Aanonsen *et al.* demonstrated that in several relatively simple cases the regression-estimated response surface was sufficiently accurate to allow the proper location to be found.

Pan and Horne (1998), in an approach similar to Aanonsen *et al.*, used uniform design and kriging to decrease the number of simulations to less than an exhaustive search. This work proposed the idea of uniform refinement areas of search spaces with favorable objective function value.

Stoitsis *et al.* (1999) used a NN proxy to represent the components of the production system, and then used a simple GA to optimize production. Stoitsis *et al.*, demonstrated that rigorous optimization with the GA was superior to the conventional approach of sequential one-dimensional optimization for the production optimization problem.

Centilmen *et al.* (1999) used NN as a substitute for the numerical simulations. The NN was used to try to capture the nonlinear effects of geometric well configurations on productivity from the reservoir. The numerical model was replaced by the NN which is able to output production profiles from wells. The accuracy of such an approach for full field-scale models is not clear and such a proxy may be unnecessarily complex for the purposes of this study since we expect the proxy to estimate only the global objective function value, for instance Net Present Value (NPV).

Johnson and Rogers (2001) used a NN proxy for the numerical model in an effort to optimize water injector locations for the Pompano field. Prior to optimization, Johnson and Rogers preselected 25 candidate locations for injection wells based on injectivity criteria. The present study found that a preselection of locations based on criteria other than the objective function can be limiting.

1.2.3 Survey Results

Direct optimization is very often prohibitive in terms of CPU requirements. Proxies, on the other hand, are very fast, however they often require an initial investment in computation. The magnitude of this initial computational investment is unclear. Also the training points are chosen synchronously; that is, the choice of a particular point to be simulated is independent of the others even though in real life the choice of later *experiments* would be based on the experience of earlier *observations*.

These various issues demonstrate that there remains considerable room for the improvement of generality and robustness in the area of reservoir development. The HGA shows potential as a robust and general approach. The HGA employs direct optimization and proxy approaches simultaneously. The proxy ought to evolve *intelligently* as the GA iterates. This work investigated the design of such a composite and

adaptive algorithm, and tested its effectiveness in a range of artificial test problems and real field cases.

The presence of uncertainty for the well placement optimization problem has most often been addressed approximately or has been completely overlooked. Decision analysis tools were employed for the methodical assessment of uncertainty in this study. A quicker and approximate approach of formulation of the problem as the optimization of a random function was also experimented with.

Chapter 2

Theoretical Foundations

The theoretical foundations of the tools that make up the HGA for reservoir development optimization are discussed in this chapter.

2.1 Genetic Algorithms

Greedy gradient-following algorithms have local scope. These algorithms modify a single point and the optima they seek is within the neighborhood of the current point. Such methods have been studied extensively and are very efficient for smooth, unimodal (i.e. single optimum) problems. However, real world problems are often multimodal, noisy and discontinuous. Greedy algorithms lack robustness on such challenges. GAs on the other hand, although not as efficient as greedy algorithms for some problems, have reasonable performance for a very wide range of problems.

Genetic Algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics (Goldberg, 1989). These algorithms combine survival of the fittest among string structures (solution vectors) with a structured yet randomized information exchange. GAs modify a set of solution vectors simultaneously, as opposed to modifying a single point. In every iteration (generation), a new set of solution vectors (population) is created from the pieces of the fittest of the older solution vectors. An occasional new vector is tried out to introduce further variety. Some of the GA vocabulary is summarized in Table 2.1. The GA and engineering

Table 2.1: The GA vocabulary.

GA vocabulary	Engineering vocabulary
environment	objective function evaluator
population	set of solution vectors
chromosome, string	encoded solution vector
gene	an element of the encoded string
fitness	function value
individual	data structure
generation	GA iteration
reproduce	carry on to the next iteration

vocabulary will both be referred to in this discussion.

GAs were developed by Holland (1975). The central theme of GAs is *robustness*, which is the balance between exploitation and exploration that is necessary for survival in different and complicated environments as defined by Goldberg (1989). Exploitation constitutes to making the most out of the information available about the problem without further exploration. Such a balance is necessary for survival of species in different environments. GAs employ natural selection (selection based on fitness) as well as crossover for information exchange and mutation to introduce further variety (randomness) into search. We will now discuss why such a strategy may lead to the optimal solution. We will use the binary alphabet, $\{0,1\}$, however the demonstration is general to arbitrary alphabets.

2.1.1 String Representation of Solutions

Solutions in GAs have to be represented in the form of strings. Strings representing each parameter are encoded in some fashion and arranged linearly.

GAs work in discretized space. The length of the strings representing potential solutions to the problem depends on the range of the parameters (minimum and maximum) and also the precision with which the solutions are sought within this range. Binary strings with an alphabet size of two (0 and 1) are common, however

any alphabet size may be employed. Given the range (x_{min} and x_{max}), precision (p) and the alphabet size (N_a) the string length (l) has to satisfy the following relation:

$$N_a^{l-1} < \frac{x_{max} - x_{min}}{p - 1} < N_a^l - 1 \quad (2.1)$$

Once the string length is defined we can decode the real value from a given string:

$$x = x_{min} + \left[\sum_{i=1}^l N_a^{i-1} \cdot b_i \right] \left[\frac{x_{max} - x_{min}}{N_a^l - 1} \right] \quad (2.2)$$

When an alphabet size of two is employed Eqn. 2.2 reduces to binary encoding. Integers may be represented *as is* by having unit precision and the alphabet size as $N_a = x_{max} - x_{min} + 1$.

2.1.2 Population

One of the distinctive features of GAs is that they modify a set of solutions simultaneously rather than the more conventional way of modifying a single solution at a time. Simultaneous modification of a set of solutions enables more robustness since different areas of the parameter space are searched simultaneously. Populations also enable the crossover operator helping to combine better parts of multiple solutions to create an even better one.

2.1.3 Fitness Based Selection

The individuals are selected according to their fitness. Individuals with higher fitness have higher chances to reproduce. The selection process is based on the spin of a roulette wheel with slot sizes proportional to individual fitnesses. The simple GA flowchart is given in Fig. 2.1. Selection, crossover and mutation are the only tools required to implement a simple GA.

2.1.4 GA Operators

Before introducing their purpose, we introduce the mechanics of the GA operators and selection based on fitness.

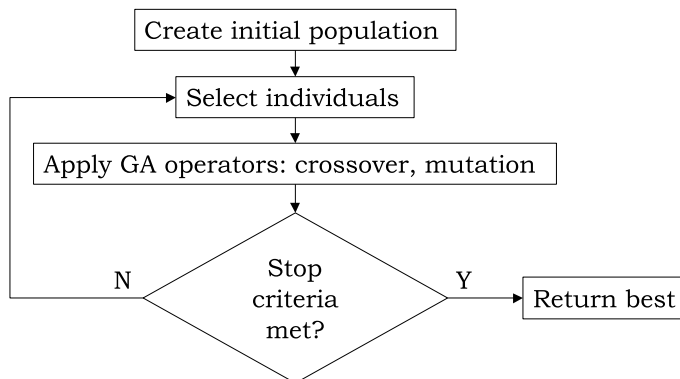


Figure 2.1: Simple GA flowchart.

The two GA operators are crossover and mutation. Two strings are crossed over from a randomly selected crossing point. For instance if the strings **00110010** and 10001101 were to be crossed over from point 3, the offsprings would look like this:

$$\begin{array}{ccc}
 \mathbf{00110010} & \rightarrow & \mathbf{00101101} \\
 10001101 & & \mathbf{10010010}
 \end{array}$$

This type of crossover is called single-point crossover. There are several types of crossovers in the literature (Goldberg, 1989) that are simply different ways of combining two strings. Crossover is carried out with a probability p_c .

Mutation on the other hand reverses a gene within a string, that is, makes it 1 if it is 0, and makes it 0 if it is 1, with the probability p_m .

2.1.5 The Schema Theorem

Remark It should be noted that the schema is conceptualized for demonstration purposes only and that the GA does not explicitly use the schema to solve a problem.

Holland demonstrated why GAs work with what he called the *Schema Theorem* (Holland, 1975). The concept of schema is now introduced.

Schema A schema (Holland, 1975), H , is a similarity template describing a subset of strings with similarities at certain string positions. In a binary alphabet $\{0,1\}$,

the schema is introduced by adding the * i.e. the *don't care* symbol; $\{0,1,*\}$. A schema is a pattern matching template. In a schema, 1 matches a 1, 0 matches a 0, and * matches either. For instance schema $H = 1 * 0$ matches two strings, $\{100,110\}$, and $H = 0 * 1 *$ matches four strings, $\{0010,0110,0011,0111\}$.

Two important properties of a schema are its *order* and *defining length*.

Order, $o(H)$ Order of a schema is the number of fixed positions.

$$o(11 * 01 * *) = 4$$

Defining length, $\delta(H)$ Defining length of a schema is the distance between the first and last fixed string position.

$$\delta(11 * 01 * *) = 5 - 1 = 4$$

Having these definitions, we can now quantify the effects of selection and reproduction. Let $m(H, t)$ be the number of strings matching a particular schema H contained within the population $\mathbf{A}(t)$ at generation t .

The i th string at generation t , $A_i(t)$, gets selected with probability $p_i(t)$ proportional to its relative fitness

$$p_i(t) = \frac{f_i(t)}{\sum_{j=1}^n f_j(t)} \quad (2.3)$$

where n is the population size and $f_i(t)$ is the fitness of string $A_i(t)$. At generation $t + 1$ we expect to have $m(H, t + 1)$ representatives of schema H given by

$$m(H, t + 1) = m(H, t) \frac{n \cdot f(H, t)}{\sum_{j=1}^n f_j(t)} = m(H, t) \frac{f(H, t)}{\bar{f}(t)} \quad (2.4)$$

where $f(H, t)$ is the average fitness of the strings representing schema H at time t . Equation 2.4 means that the number of schemata (plural of schema) with fitness above the population average will increase (grow), similarly number of schemata with fitness below population average will decrease (decay). This applies to every schema in population $\mathbf{A}(t)$ in parallel. Note that the number of processed schemata in a

population of size n is much larger than n , to be more specific it is in the order of $O(n^3)$ (Holland, 1975). This is an important concept known as *implicit parallelism*, which tells us that although only n strings are processed explicitly at each generation, the GA implicitly processes around n^3 schemata in parallel. This is information processed implicitly at no cost, since the computation time is proportional to only the population size, n .

If a schema H has fitness $f(H, t) = \bar{f}(t) + c\bar{f}(t)$ where c is a constant and $c > 0$, Eq. 2.4 can be rewritten as

$$m(H, t + 1) = m(H, t) \frac{\bar{f}(t) + c\bar{f}(t)}{\bar{f}(t)} = (1 + c) \cdot m(H, t) \quad (2.5)$$

Writing everything in terms of $m(H, 0)$ and assuming c stays constant over several generations, we obtain

$$m(H, t) = m(H, 0) \cdot (1 + c)^t \quad (2.6)$$

which states reproduction based on fitness gives above average schemata exponentially increasing trials over generations where c remains a constant.

If only selection was utilized, the probability of survival of schema would be 1.0, $p_s(H) = 1.0$. However reproduction alone does not introduce new strings into the population. The GA operators crossover and mutation are introduced for this purpose. Crossover is a means of random information exchange between strings. However, crossover will have disruptive effect on the above average schemata growth (Eq. 2.6). The probability of survival of schema H in the presence of crossover is given by

$$p_s(H) = 1 - p_c \frac{\delta(H)}{l - 1} \quad (2.7)$$

where l is the string length. It is intuitive that schema with large defining length (e.g. $\delta(1****0) = 6$) have a higher probability of crossover disruption than ones with smaller defining lengths (e.g. $\delta(**01**) = 1$).

Mutation is utilized in order to introduce additional variety into the population. Mutation also has a disruptive effect on exponential growth. Survival probability of schema H in the presence of both crossover and mutation is given by

$$p_s(H) = 1 - p_c \frac{\delta(H)}{l-1} - p_m \cdot o(H) \quad (2.8)$$

where $p_s(H)$ is reset to zero if $p_c \frac{\delta(H)}{l-1} + p_m \cdot o(H) > 1.0$

Equation 2.4 is then modified to give the actual schema growth equation in a GA run:

$$m(H, t+1) = m(H, t) \frac{f(H, t)}{f(t)} \left[1 - p_c \frac{\delta(H)}{l-1} - p_m \cdot o(H) \right] \quad (2.9)$$

What Eq. 2.9 tells us is referred to as the *Schema Theorem* (Michalewicz, 1996):

Schema Theorem Short, low order, above average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm.

The *Building Block Hypothesis* naturally follows the schema theorem:

Building Block Hypothesis A genetic algorithm seeks near optimal solutions through the juxtaposition of short, low-order, high-performance schemata, which are called the *building blocks* of the solution.

2.1.6 Deception

One should understand the GA phenomenon known as *deception*, to be able to use them efficiently. In the previous section we showed that the genetic algorithm brings together fit, short, low-order building blocks in the effort to create a fitter individual. Consider a synthetic case where the solution vectors are made up of eight binary strings. Assume the following schemata have above population average fitness:

$$\begin{aligned} H_1 &= 111**** & f(H_1) &> \bar{f} \\ H_2 &= *****11 & f(H_2) &> \bar{f} \end{aligned}$$

Also assume that their combination has much less fitness than a third schema, $H_3 = 000***00$:

$$\begin{aligned} \text{crossover}(H_1, H_2) &= H_{1 \times 2} = 111***11 \\ H_3 &= 000***00 & f(H_3) &>> f(H_{1 \times 2}) \end{aligned}$$

Further assume that the global optimum is $A_g = 11111111$. The GA would have a hard time converging into A_g due to the fact that $f(H_3) \gg f(H_{1 \times 2})$, and might converge to the the strong suboptimal of solution $A_s = 00011100$. The phenomenon, where fit building blocks mislead the search, is called deception (Michalewicz, 1996).

Deception generally occurs when related string pieces are placed far apart in the chromosome. The intuitive approach to avoid deception is then to place such physically related string pieces closer to each other. This requires some prior knowledge about the objective function of the problem. However it is most often clear, through expertise or experimentation, which parameters are related. For instance, consider a reservoir development problem of finding the optimum location and injection rate of multiple water injectors. For such a problem, one should not put the strings that encode the well location, and the strings that encode its well rate, very far apart in the chromosome.

2.1.7 Practicalities

Here we discuss some of the practical aspects of GA applications.

In a GA optimization run, the parameters to be varied are encoded and arranged linearly into a string. To decrease the occurrence of deception, the related strings should be placed close to each other.

The GA parameters have to be defined prior to optimization. For binary alphabets Goldberg (1989) suggests an appropriate population size and De Jong (1975) suggests appropriate GA operator probabilities based on experimentation over problems of different nature:

Population size - same as total string length, $n = l$

Crossover probability - $p_c = 0.6$ works well for a wide range of problems

Mutation probability - inverse of population size, $p_m = 1/n$

Another way to adjust GA parameters is to let the GA adjust its own parameters, which is referred to as *parameter adaptation* (Hinterding *et al.*, 1997). Parameter

adaptation is intuitive in that the algorithm itself evolves along with evolving strings. Such dynamic parameter adjustment also takes into consideration the fact that the best set of GA parameters changes as the run progresses. In other words, the optimum GA operator probabilities are different at the initial and final stages of a GA run. There are two ways to dynamically adjust parameters during a GA run:

Adaptive - The probability of the GA operators are adjusted according to their previous performances (Davis, 1989). For instance, if the increase in mutation probability increases the performance of the GA, then the mutation probability is increased further.

Self-adaptive - The GA parameters are attached to the end of the string and are evolved together with the rest of the problem parameters (Bäck, 1992). In this way the operator probabilities, that produce fitter individuals, dominate.

Problem-specific knowledge should also be employed where appropriate. Some problems necessitate data structures other than linear binary strings (e.g. trees). Custom mutation and crossover operators have to be defined for such problems. These custom GA operators have the potential to take advantage of problem specific engineering intuition. However caution should be taken not to exert extreme bias on the algorithm since *alternative* or *good* solutions that contradict engineering judgement may exist or such contradicting solutions may contain the building blocks of an optimal solution.

It is also straightforward to parallelize GAs since at each generation the evaluation of the individuals are independent of each other. The parallelization setup of the algorithm is discussed in Appendix A.

2.1.8 Extensions

Numerous extensions have been proposed to the simple GA given in Fig. 2.1.

Elitism - The best of the older generation is copied to the next generation without alteration. This ensures that the good solutions found earlier are not lost.

Fitness scaling - The fitness of individuals are scaled, that is the relative fitness differences among individuals are increased or decreased by exponentiating the raw fitnesses. Scaling enables the control over selective pressure. Selective pressure refers to the relative selection probability of individuals.

Tournament selection - Some number k of individuals is selected and the best of them is reproduced (Goldberg *et al.*, 1991). Increasing values of k increase the selective pressure. Tournament selection is equivalent to scaling the fitness.

Rank-based selection - The rank of the individuals in the population determines its fitness (Whitley, 1989). Ranking eliminates the problem of unfavorably scaled fitness values.

Gray coding - The problem with straightforward binary encoding is that, the strings that differ with only a single gene are not adjacent in decoded space. Gray-coded strings have the property that, string that have a single different gene are adjacent in the decoded space. This was shown to improve performance in some cases (Mathias and Whitley, 1994).

2.2 Polytope Method

The polytope method is a hill-climbing algorithm that does not require derivative information. The polytope method is also known sometimes as the simplex method, not to be confused with the simplex algorithm used to solve linear programming problems.

The polytope algorithm is a direct search hill-climbing method that can be utilized with $n + 1$ linearly independent points in an n -dimensional search space (Gill *et al.*, 1981). The method was first established by Nelder and Mead (1965) and was used for maximization in this study. At each stage of the algorithm the decision variables x_1, x_2, \dots, x_{n+1} are arranged in the order of decreasing objective function value, $f_1 \geq f_2 \geq \dots \geq f_{n+1}$. Initially a trial point is generated by a single reflection step:

$$x_r = c + \alpha(c - x_{n+1}) \quad (2.10)$$

where c is the centroid of the $n+1$ points:

$$c = \frac{1}{n} \sum_{i=1}^n x_i \quad (2.11)$$

and α ($\alpha > 0$) is the reflection coefficient. The function is evaluated at the new point x_r , revealing the function value f_r . The next step is taken according to the value of f_r :

- If $f_1 \geq f_r \geq f_n$ then replace the worst point (x_{n+1}) with x_r and proceed to the next iteration.
- If $f_r > f_1$ then x_r is the new best point. Assuming the direction of reflection is towards the optimum, an expansion step is carried out. The expanded point given by:

$$x_e = c + \beta(x_r - c) \quad (2.12)$$

is evaluated to find f_e , where β ($\beta > 1$) is the expansion coefficient.

- If $f_e > f_r$ then expansion is successful and x_e replaces x_{n+1} .
- If $f_e < f_r$ then expansion fails and x_r replaces x_{n+1} .
- If $f_r < f_n$ then the polytope is assumed to be too large thus is contracted:
 - If $f_r \geq f_{n+1}$ then

$$x_c = c + \gamma(x_{n+1} - c) \quad (2.13)$$

- If $f_r < f_{n+1}$ then

$$x_c = c + \gamma(x_r - c) \quad (2.14)$$

where γ ($0 < \gamma < 1$) is the contraction coefficient. The function is evaluated at x_c to find f_c . If f_c is greater than both f_r and f_{n+1} , x_c replaces x_{n+1} , otherwise further contraction steps are carried out.

The graphical representation of a polytope constructed in two-dimensional space is shown in Fig. 2.2.

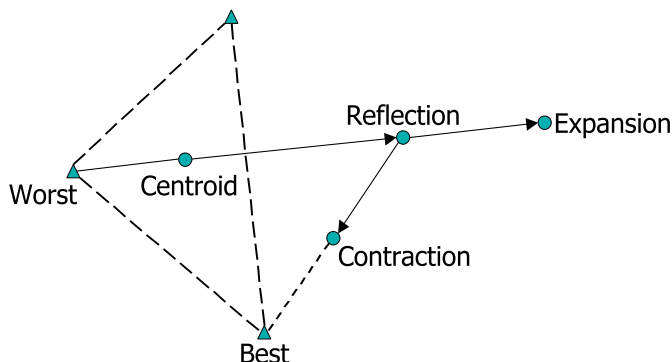


Figure 2.2: A polytope and possible polytope steps in two dimensions.

2.3 Proxy Construction

The appropriate evaluation function for reservoir development (e.g. numerical simulator) is most often very expensive computationally, thus the idea of using a cheaper substitute to this expensive evaluation function is attractive. Kriging and NNs have been considered as proxies in this work.

2.3.1 Kriging

The kriging algorithm is based on the theory of regionalized variables (Matheron, 1965). A regionalized variable refers to a phenomenon which is spread out in space or time. Actually any variable can be considered to be spread out in the related parameter space. From this point of view, kriging becomes nothing more than an interpolation algorithm assuming correlation among data points. Kriging has been used widely in earth sciences, particularly in geostatistics. Geostatistical applications are usually limited to three dimensions, however the algorithm can be extended to n -dimensions, thus kriging can be used for interpolation of multivariate functions.

Ordinary kriging, which is the most widely used kriging method, was used in this study. Ordinary kriging is used to estimate a value at a point of a region for which the variogram is known, without prior knowledge about the mean. Furthermore, ordinary kriging implicitly evaluates the mean in the case of a moving neighborhood.

Suppose the objective function value at point x_0 in Fig. 2.3 is to be estimated

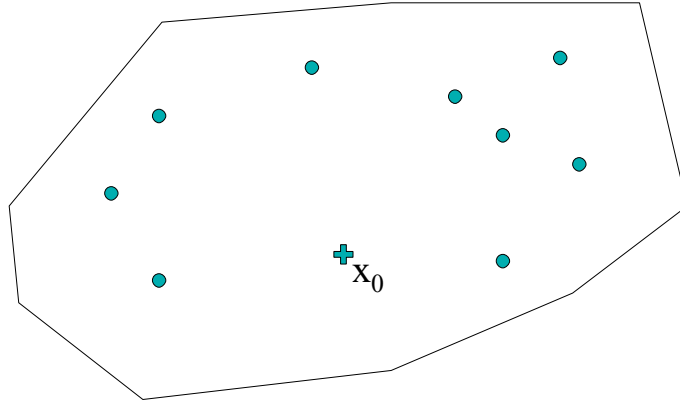


Figure 2.3: A two-dimensional parameter space with irregularly spaced data (circles) and a location where value is to be estimated (cross).

with the n data points. Combining function values at the data points linearly with weights λ_i (Wackernagel, 1998):

$$Z^*(x_0) = \sum_{i=1}^n \lambda_i Z(x_i) \quad (2.15)$$

The weights are constrained to add up to one. In ordinary kriging, this constraint is imposed by Lagrange formalism, with the introduction of the Lagrange parameter μ . In simple kriging, which is the fundamental kriging algorithm, there is no need for such a constraint since the weights are made to sum up to one by giving the weight λ_{n+1} to the mean, hence in simple kriging, prior knowledge of the mean is necessary.

The data are assumed to be part of a realization of an intrinsic random function $Z(x)$ with the variogram $g(h)$. In a geostatistical study, the variogram that has a best fit to the data would be chosen.

The unbiasedness is assured with unit sum of the weights:

$$\sum_{i=1}^n \lambda_i E[Z(x_i) - Z(x_0)] = 0 \quad (2.16)$$

The estimation variance is given by:

$$\begin{aligned}\sigma_{\text{E}}^2 &= \text{E} \left[(Z^*(x_0) - Z(x_0))^2 \right] \\ \sigma_{\text{E}}^2 &= -\gamma(x_0 - x_0) - \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j \gamma(x_i - x_j) + 2 \sum_{i=1}^n \lambda_i \gamma(x_i - x_0)\end{aligned}\quad (2.17)$$

The ordinary kriging system is obtained by minimizing the estimation variance with the constraint on weights:

$$\begin{bmatrix} \gamma(x_1 - x_1) & \cdots & \gamma(x_1 - x_n) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \gamma(x_n - x_1) & \cdots & \gamma(x_n - x_n) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu \end{bmatrix} = \begin{bmatrix} \gamma(x_1 - x_0) \\ \vdots \\ \gamma(x_n - x_0) \\ 1 \end{bmatrix}\quad (2.18)$$

The left hand side describes the dissimilarities among data points while the right hand side describes the dissimilarities between the data points and the estimation point.

Performing the matrix multiplication, the ordinary kriging system can be rewritten in the form:

$$\begin{cases} \sum_{j=1}^n \lambda_j \gamma(x_i - x_j) + \mu = \gamma(x_i - x_0) \\ \sum_{j=1}^n \lambda_j = 1 \end{cases} \quad i = 1, \dots, n \quad (2.19)$$

Ordinary kriging is an exact interpolator, in other words, at the point of the data, the estimated value is equal to the data value.

If all data are used for each estimation point, only the right hand side vector in Eq. 2.18 is modified, while the left hand side matrix remains unchanged. Thus given a data configuration, the left hand side matrix in Eq. 2.18 has to be inverted once. The left hand side matrix in Eq. 2.18 has the size $n + 1$, n being the number of data points. After the inversion, every estimation requires a single matrix multiplication.

The correlation structure of the underlying function can be extracted by fitting a variogram model to the data produced by this function, although the variogram is not reproduced by the kriging estimates. However if the variogram used has a very short correlation range, the ordinary kriging estimates will be equal to the mean for

points not close to the data locations. The estimated surface would resemble *circus tents* in the case of short correlation ranges. This would be unsatisfactory in that it would never compute a value higher than the values at the known data points. Therefore the variogram used for the optimization algorithm must have a very long correlation range.

2.3.2 Neural Networks

The principles of Neural Networks (NNs) were influenced from how the human brain works (Russell and Norvig, 1995). In the human brain, the fundamental functional unit, called a neuron, consists of a cell body with dendrites and a long fiber called the axon which has synapses at its end. Neurons are connected to each other through the links of dendrites and synapses. Signals are propagated from neuron to neuron by a complicated electrochemical reaction. A neuron *fires* current through its synapses to the dendrites of the other neurons it is connected to. When the incoming current raises the electrical potential of a neuron over a certain threshold, the neuron fires. Although it is not fully understood how exactly the brain works, it is believed that such a configuration leads to learning and consciousness.

The mathematical NNs consist of a number of nodes connected by links. Each link has a weight value associated with it. Weights are the means of adapting the NNs to the problem. Learning in NNs takes place by updating the weights. Some of the nodes are connected to external points for input and output purposes. The weights are modified in order to tune the NN behavior to match the imposed inputs and resulting outputs.

Each node has input links from other nodes and output links to other nodes. Each node makes a local computation, based on the cumulative input from its incoming links and outputs, of a value based on its activation function. An illustration of a typical NN unit is given in Fig. 2.4. Each unit has a threshold, conveniently represented by a fictitious input to every node. The activation function used in this study is the sigmoid function given by

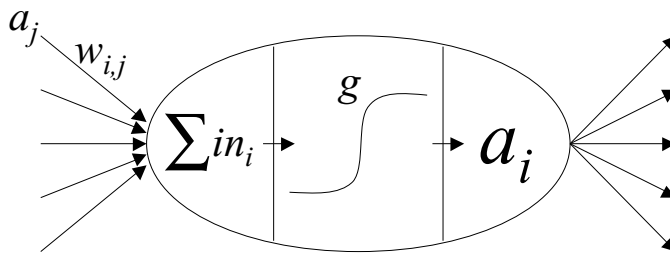


Figure 2.4: A typical neural network unit.

$$a_i = g' \left(\sum_i in_i \right) = \frac{1}{1 + e^{-\sum in_i}} \quad (2.20)$$

Multilayer feed-forward NNs are capable of representing nonlinear functions of any complexity. The structure of such a NN is illustrated in Fig. 2.5.

The NN can be trained by back-propagation learning. Weights are initialized arbitrarily (randomly). Example inputs are presented to the network and if there is an error between the example and the NN output, the weights are adjusted to reduce this error. The trick for back-propagation learning is to assess the blame for an error and divide it among the contributing weights (Russell and Norvig, 1995). The weight for the link connecting hidden node j to an output node i is updated by

$$w_{j,i}^{t+1} = w_{j,i}^t + \alpha a_j \Delta_i \quad (2.21)$$

where α is a constant called the learning rate, a_j is the output of node j , and Δ_i is given by

$$\Delta_i = (T_i - a_i) g' \left(\sum_i in_i \right) \quad (2.22)$$

where T_i is the true output, a_i is the node output and g' is the derivative of the activation function. The learning rate is usually close to unity. The learning rate determines the convergence behavior of the back-propagation algorithm. The optimum value of the learning rate is problem specific.

The weight between an input unit k and a hidden layer unit j is updated by

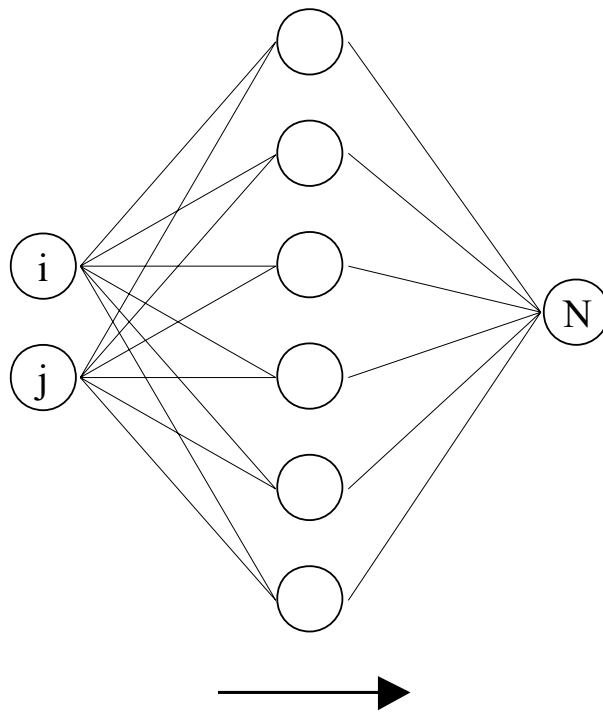


Figure 2.5: The structure of a feed-forward neural network.

$$w_{k,j}^{t+1} = w_{k,j}^t + \alpha a_k \Delta_j \quad (2.23)$$

where Δ_j is defined as

$$\Delta_j = g' \left(\sum_j in_j \right) \sum_i w_{j,i}^t \Delta_i \quad (2.24)$$

The back-propagation algorithm can also be interpreted as a gradient-descent search to minimize NN error in weight space.

Practically, the example set is partitioned into a training set and a test set. The training set is used to update the weights and the NN performance is tested with the test set. Such an approach provides an indication of whether the NN is generalizing to the whole example set and not overfitting the training set.

2.4 The Hybrid Genetic Algorithm (HGA)

In this study GA was hybridized with the polytope method. In addition the proxy approach was integrated within the steps of a conventional GA. The proxy was improved iteratively at each generation. The resulting algorithm is referred to as the Hybrid Genetic Algorithm (HGA). The HGA flowchart is given in Fig. 2.6.

2.4.1 Hybridization

A polytope is constructed from $n_d + 1$ individuals, n_d being the dimensions of the problem. The polytope method is then used to generate a new potential member of the population. Three different strategies to construct a polytope were considered:

- Polytope construction from the $n_d + 1$ fittest individuals in the current population (Fig. 2.7).
- Polytope construction from the $n_d + 1$ fittest individuals ever encountered in all populations so far (Fig. 2.8).

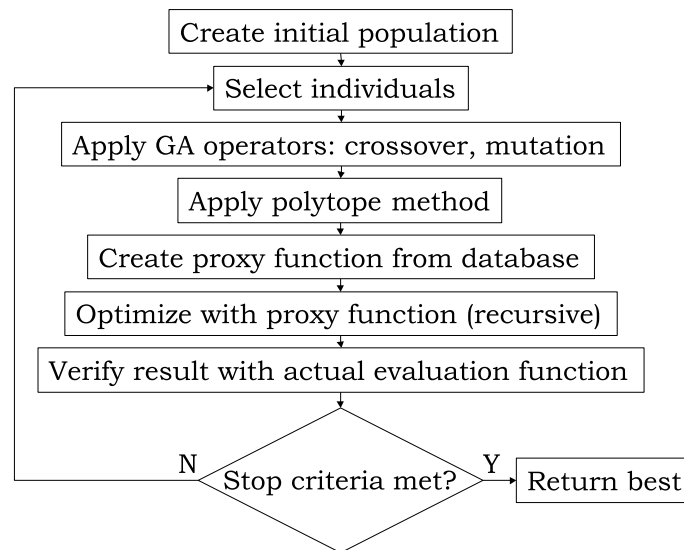


Figure 2.6: HGA flowchart with the conventional GA steps followed by the polytope steps and the proxy approach.

- Polytope construction from the fittest ever encountered individual and the n_d closest individuals to this individual (Fig. 2.9).

The first two methods of constructing a proxy are more of a mutation operator than a refinement. The last method is the best way to construct a polytope for the purpose of local refinement. The effect of polytope strategy on algorithm performance was investigated and is presented in Appendix 7.

As a result of the polytope steps, if the solution is improved, then the individual with the lowest function value (worst solution) is replaced and the GA is resumed. The polytope method will require several (one to three) function evaluations per polytope step.

To integrate the proxy method, a database of visited points is constructed and updated whenever a function evaluation is made. This database is used to construct the proxy, which replaces the actual evaluation function. Using this proxy, estimations are made for the unevaluated points. The maximum point of the proxy representation of the evaluation function is found by a recursive call to the HGA with the proxy replacing the numerical model. This proxy maximum is then evaluated by the actual

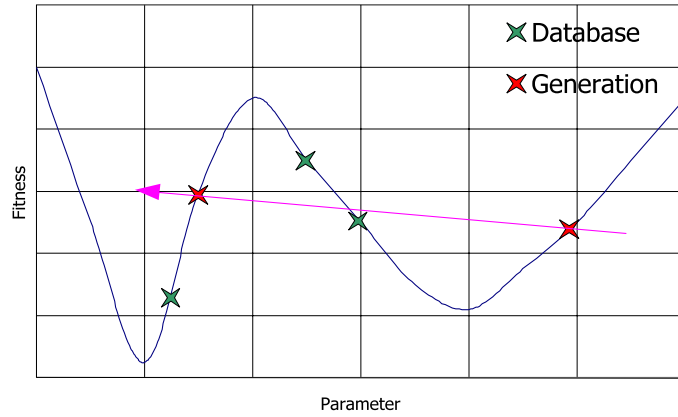


Figure 2.7: Illustration of the polytope constructed from the best solutions in the generation for a one-dimensional problem.

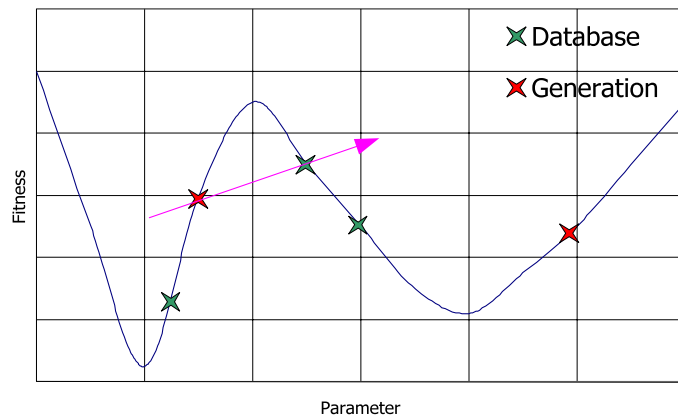


Figure 2.8: Illustration of the polytope constructed from the best solutions in the generation for a one-dimensional problem.

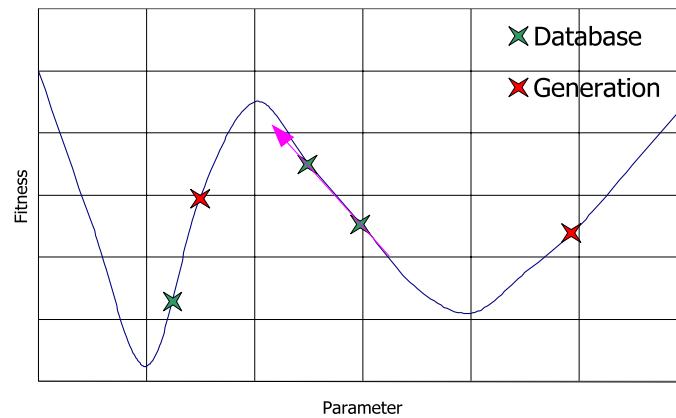


Figure 2.9: Illustration of the polytope constructed from the solutions closest to the best solution in the database for a one-dimensional problem.

evaluation function for verification, and if better than the best solution, the worst solution in the generation is replaced. The proxy method requires only one function evaluation per generation.

Error Analysis

An additional step may be employed when using the kriging proxy in order to further ensure robustness. The kriging framework enables the computation of the estimation variance (Eq. 2.17). The location in the search space with the maximum kriging variance is the location furthest away from all data taken together. Thus making an additional simulation at this location ensures that the proxy is not missing a significant optimal region, making the overall hybrid algorithm more robust. This approach will be referred to as Error Analysis.

The HGA was implemented in C++ with object-oriented design criteria. The structure of the code and implementation details are given in Appendix B.

2.4.2 Other Helper Techniques

Some additional techniques were investigated as ways to further enhance the HGA performance.

Invalid Point Interpretation - A point in the search space is considered invalid when it does not have a physical description. An example would be the placement of a well in an inactive grid block. Invalid points could be given a fitness of 0. However such an approach disrupts the effectiveness of the polytope and the proxy and also makes it harder for the optimization algorithm if the optimal solution happens to be adjacent to the invalid solution. Invalid points in this study were assigned a fraction of the fitness of the closest valid point, which enables easier proxy calibration and better located polytopes. This approach is analogous to the penalty function approach used in constrained optimization.

Local Mutation - In order to further resolve the issue of refinement, local mutation is proposed. The best solution at each generation is perturbed within a given range and probability. This lowers the chance of missing better points close by. Local mutation is similar to the perturbation made during simulated annealing, although here the range and probability of the perturbation is kept constant throughout successive generations.

Chapter 3

Synthetic Model - Well Placement Optimization

The performance of the HGA was tested on a synthetic model. Controlled experiments were carried out to compare the performance to simple GAs. Schlumberger-Geoquest's ECLIPSE was used for numerical simulation.

3.1 The Synthetic Model

A heterogeneous, two-dimensional (32×32) numerical model was constructed. The permeability field was established by sequential Gaussian simulation conditioned to arbitrary permeability data. The permeability distribution is given in Fig. 3.1. The permeability values range from 9 md to 47 md. Porosity was related to permeability through a logarithmic function given in Eq. 3.1.

$$\phi_i = 11.889 + 2.277 \log(k_i) \quad (3.1)$$

3.2 Producer Placement

The optimum location of a producer on the 32×32 synthetic field was determined by exhaustive search. The purpose in doing this was to know the true global optimum

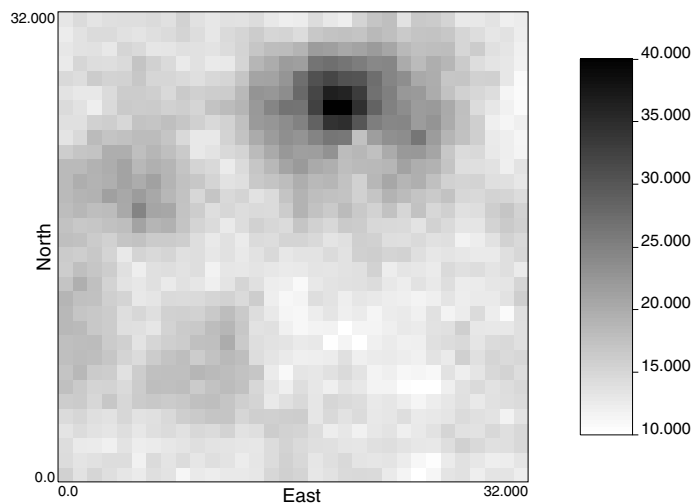


Figure 3.1: The permeability distribution for the synthetic model.

and hence be able to evaluate the effectiveness of the algorithm in finding it. The example reservoir already had three existing producers. The wells were controlled by bottom hole pressure. The cumulative oil production surface produced by exhaustive simulation is given in Fig. 3.2. Possession of the objective function value at every possible location of the grid enables the test runs to be made without any further simulation. Thus sensitivity studies were carried out with minimal computational resources. The effectiveness of the GA operators and hybridization were studied by comparing to the exhaustive search result.

This problem was influenced strongly by the fluid properties, boundary conditions and the existing production well locations as well as heterogeneity. It is seen from Fig. 3.2 that the optimum location is (20,1). This grid block is adjacent to a no-flow boundary, however it is less affected by the other producers. The block (32,1) would probably be least affected by the producers, however due to its adjacency to two no-flow boundaries, the well would switch to bottom hole pressure control too early in the simulation, lowering the cumulative oil production from the field. After the system reaches pseudosteady state, the shape of the surface (Fig. 3.2) will not change significantly, therefore the duration of production from the field is not crucial for the optimum location of the production well.

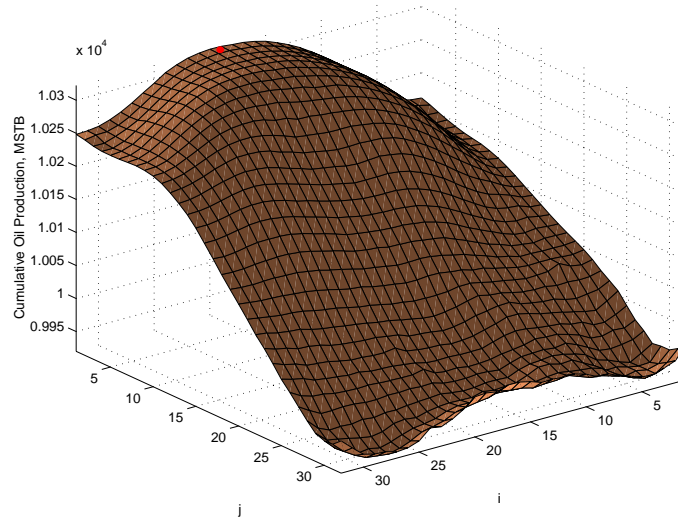


Figure 3.2: Cumulative oil production surface for the single production well placement problem for the 32×32 synthetic case obtained by exhaustive simulation.

Experiments were carried out to test HGA performance. Three experiments were made and 50 optimization runs were made for each experiment. Each optimization run in these experiments was terminated once the algorithm found the optimal solution and the number of function evaluations to get to the optimum were recorded. First simple GA was used, then GA and polytope method, and finally the full HGA, which is GA, polytope method and the kriging proxy applied together. Binary encoding of the optimized parameters, (i, j) locations of the optimized wells, would require a total of 10 strings and the binary mapping would be exact. In this work integer genes were used since integer encoding has the ability to avoid the inexact mapping of binary strings for parameters that are not discretized to powers of 2. According to suggestions of Goldberg (1989) and De Jong (1975) the following sets of basic GA parameters were used for all runs:

$$n = l_{binary} = 10$$

$$p_c = 0.6$$

$$p_m = 1/n = 0.1$$

where n is the population size, l_{binary} is the length of the binary string, p_c is the

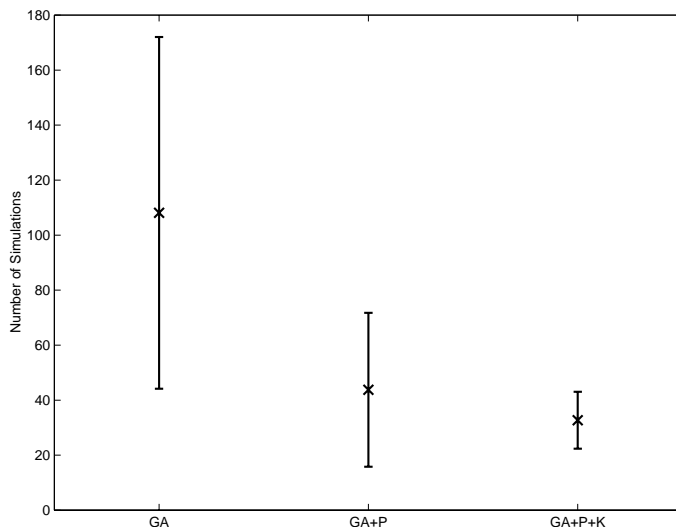


Figure 3.3: Comparison of the means (crosses) and the standard deviations (error bars) for different combinations of Genetic Algorithm (GA), Polytope method (P) and Kriging (K) for the producer placement problem on the 32×32 synthetic case.

crossover probability and p_m is the mutation probability.

The necessary number of simulations required to find the global optimum for each run was recorded. The means and the standard deviations of the number of simulations for the three experiments are given in Fig. 3.3.

It was observed that the HGA reduced the required number of simulations by a factor of 3.3 (108.1 to 32.7 simulations) compared to the simple GA with significantly less uncertainty about the run performance (Fig. 3.3).

3.3 Injector Placement

The optimum location for an injection well was investigated on the 32×32 heterogeneous model. Three existing producers were placed at the same locations as the previous run. The cumulative oil production surface given by exhaustive runs is shown in Fig. 3.4.

This optimization problem is influenced by fluid and rock properties, especially

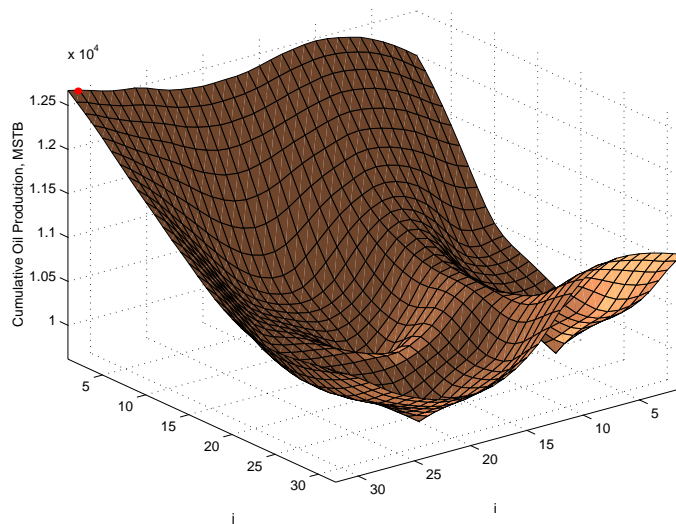


Figure 3.4: Cumulative oil production surface for the single injection well placement problem for the 32×32 synthetic case obtained by exhaustive simulation.

relative permeability, the existing producer locations, the boundaries, and the duration of injection and production from the field. The duration of the simulation in this case is important since the water breakthrough time at the producers will be the determining factor for their productivity. Figure 3.4 suggests that the optimum location for the injector well is $(32,1)$, which is the block most distant from the three producers, which means that time of water breakthrough at producers dominates other factors.

Experiments were carried out with the same parameters as the producer placement problem. The means and the standard deviations of the number of simulations required to find the global optimum of Fig. 3.4 for the three experiments are given in Fig. 3.5.

It was observed that the HGA reduced the required number of simulations by a factor of 2.2 (38.8 to 17.8 simulations) compared to the simple GA with much less uncertainty about the run performance (Fig. 3.5).

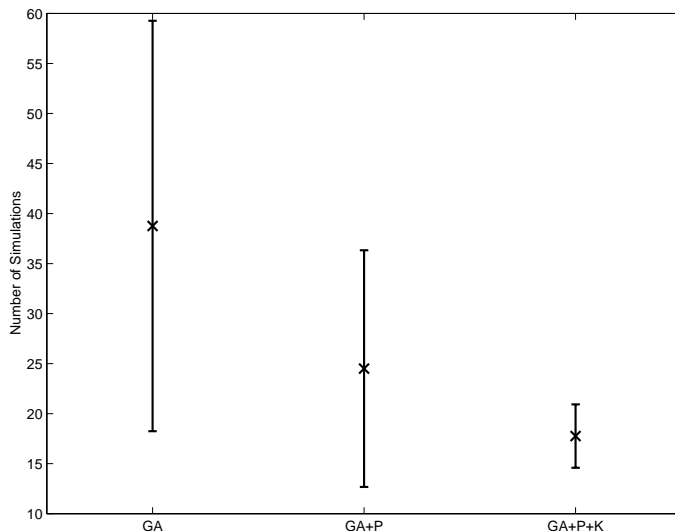


Figure 3.5: Comparison of the means (crosses) and the standard deviations (error bars) for different combinations of Genetic Algorithm (GA), Polytope method (P) and Kriging (K) for the injector placement problem on the 32×32 synthetic case.

3.4 Well Indexing

The way the well-blocks are indexed is important since it determines the behavior of the evaluation function. Bittencourt and Horne (1997) indexed active cells only, arguing that (i, j) indexing is not suitable because the optimization algorithm could place wells in inactive regions. Several types of well-block indexing were investigated for the 32×32 synthetic case, injector placement problem. Well indexing types investigated are given in Table 3.1.

The indexing schemes in Table 3.1 map the two-dimensional surface (Fig. 3.4) into a single dimension. The resulting one-dimensional functions are given in Fig 3.6.

All of the indexing schemes except the Hilbert curve are easy to generate. The Hilbert curve was first described by David Hilbert in 1892. The Hilbert curve is a space filling curve that visits every point in a square grid with a size of any power of 2. The useful feature of Hilbert curves is that two points close to each other on the Hilbert curve are also close in the two-dimensional space. The Hilbert curve mapping

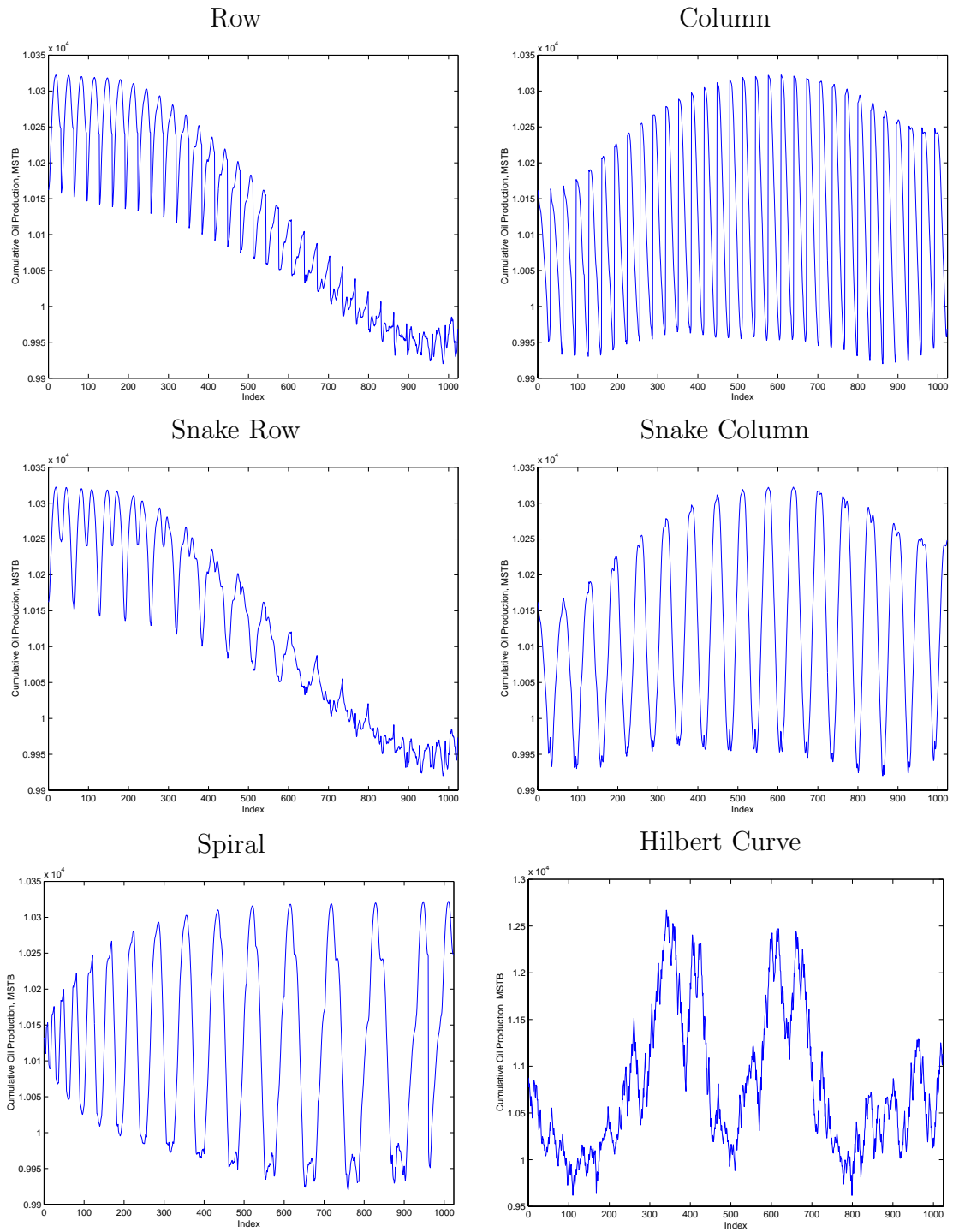
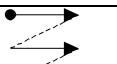


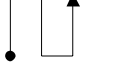
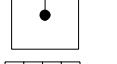
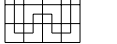


Figure 3.6: One dimensional objective functions generated by mapping from the 32×32 synthetic injector placement problem by different well indexing schemes.

Table 3.1: Well indexing schemes.

Index name	Scheme
Row	
Column	
Snake row	
Snake column	
Spiral	
Hilbert curve	

results in a smaller number of large scale local optima but the one-dimensional function has a lot of small scale noise (Fig. 3.6). The Hilbert curve has links to the area of fractals and the generation of the Hilbert curve requires a recursive algorithm. The one limitation in our case is that the Hilbert curve mapping works only for square grids with a size of any power of 2. The 32×32 Hilbert curve, also referred to as the level 5 Hilbert curve ($2^5 = 32$), is shown in Fig. 3.7;

After mapping with the given indexing schemes, the smooth two-dimensional function (32×32) given in Fig. 3.4 is replaced by noisy one-dimensional functions (1024×1) given in Fig 3.6. This artificial noise is problematic from an optimization point of view. The HGA was applied to the indexed one-dimensional functions (Fig 3.6) and the results are given in Fig. 3.8.

It is seen from Fig. 3.8 that (i, j) indexing of wells is the most suitable for optimization algorithms for the well placement problem since other kinds of indexing may introduce artificial noise due to the discontinuities in the indexed search space. Oddly,

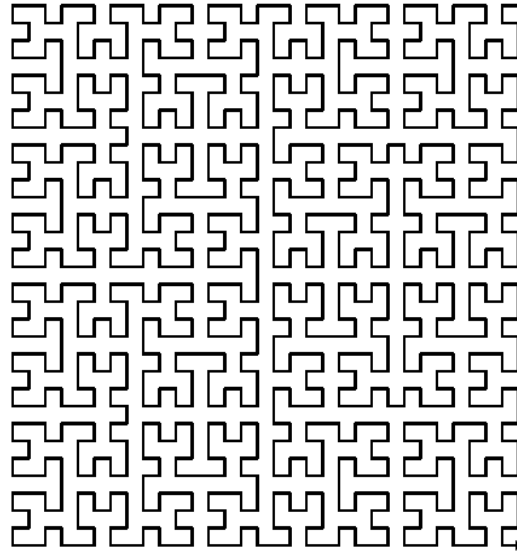


Figure 3.7: The Hilbert curve used to map the 32×32 two-dimensional surface into a single-dimension.

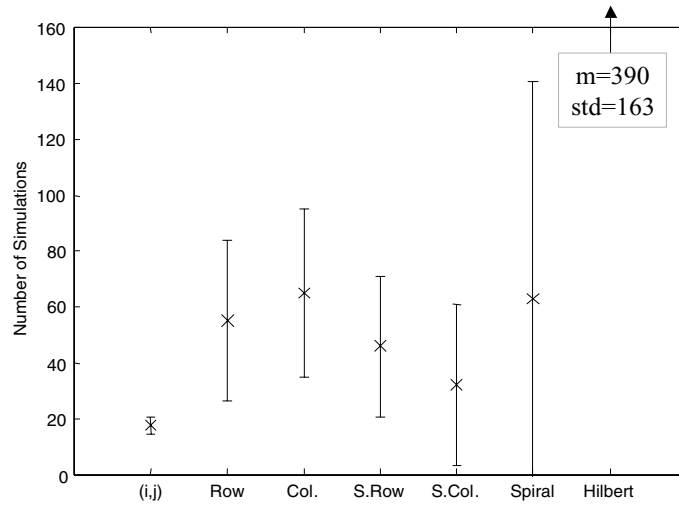


Figure 3.8: Comparison of the efficiency of the HGA for different types of indexing; crosses indicate the means and the error bars indicate the standard deviations of 50 runs.

the promising Hilbert curve indexing scheme results in the worst optimization performance which implies that for this problem, small-scale noise is more of a hinderance to optimization with GAs than large-scale noise. A solution to reduce this small-scale noise might be to sample a lesser number of points from the two-dimensional representation of the search space. This would result in a smoother one-dimensional line that would be an approximate representation of the real search space, but would be easier to optimize.

Chapter 4

Optimization of Waterflooding in the Pompano Field

The HGA was applied to the Pompano field which is an offshore development in the Gulf of Mexico. The operating companies decided to carry out waterflooding after 2.6 years of production. The optimum location and rate of water injection wells were sought.

4.1 Reservoir Description

The Pompano field extends over five Gulf of Mexico Blocks, Mississippi Canyon (MC) 27, 28, 72, and Viosca Knoll (VK) 989, 990 located approximately 24 miles Southeast of the Mississippi River Delta (Güyagüler *et al.*, 2000). BP Amoco and Kerr McGee hold 75% and 25% equity respectively. The Pompano platform sits in block VK 989 in 1290 feet of water and is a 40-slot fixed-leg platform. This platform is the second tallest fixed structure in the Gulf of Mexico. The platform receives production from three reservoirs: Uplifted Pliocene, Downthrown Pliocene, and the Miocene. In this study we focused on the Miocene reservoir which comprises two thirds of the field reserves.

The Miocene sands are located in blocks MC 28, and 72. Developing these sands

has proved challenging given the large distance from the platform. The longest drillable platform wells (about 18,000 ft of lateral step-out) can only reach the northern one third of the reservoir. Thus the remaining two thirds in the south were developed from a 10-slot subsea template located in block MC 28 in 1800 ft of water. The template is tied back to the VK989 platform via two 8 inch flowlines approximately 4.5 miles in length.

Production from the Miocene commenced in April 1995. The oil has very favorable properties which help in achieving high production rates. API gravity is 32, viscosity is 0.38 cp, and the gas-oil ratio (GOR) was initially 1037 SCF/STB and climbing with increased production. The very restricted range of variability in the producing wells emphasizes the connectivity in the Miocene reservoirs. There are 12 wells in operation, five drilled from the platform to the north during Phase I, and seven drilled from the template during Phase II. The average initial flow rate was 7880 STB/D for the five Phase I wells and 6343 STB/D for the seven Phase II wells.

The Miocene sands were deposited as mid-slope turbidites in a large aggradational channel complex. There is significant connectivity between channels as younger channels frequently eroded into previously deposited ones. Pressure depletion in successively drilled wells suggests that most of the reservoir complex is in pressure and fluid continuity. Grain size ranges from very fine to medium, with the bulk being fine grained. The average thickness of the Miocene sand is 50 net ft of oil (NFO) in a vertical target interval of 300 to 400 ft, and the thickest sand penetrated is 110 ft NFO in a single sand.

A north-south trending channel system draped over east-west trending structural nose forms the trap. The channel sands are laterally confined by the shales and silty shales of the overbank deposits. An oil-water contact at -10,200 ft true vertical depth subsea (TVDS) has been penetrated on the southern edge of the field and is implicated on the north/northwest end by seismic interpretation and water production. Maximum hydrocarbon column height is approximately 600 ft. The large aquifer system below, estimated to be three times larger than oil in place, is judged to be an advantage to help offset pressure losses during reservoir depletion.

A geological reservoir model based on seismic data (acoustic impedance) and tied

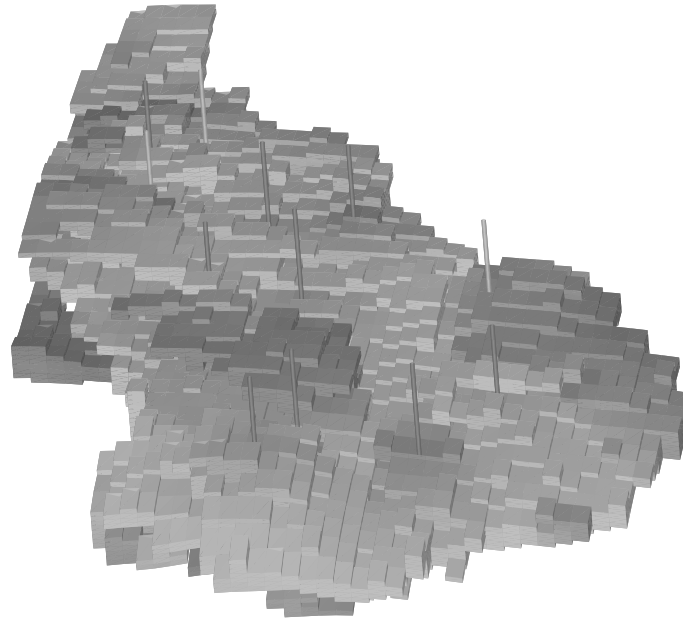


Figure 4.1: The numerical model for the Pompano field.

to appraisal wells was built. The heterogeneous anticlinal turbidite reservoir was discretized into an approximately three million cell blocks at seismic resolution. It was then scaled up to a 40,000 cell block simulation model, with dimensions of $40 \times 40 \times 25$ having 7,533 active cells. The three-dimensional visualization of the grid for the numerical model is shown in Fig. 4.1. Sensitivity runs with the simulation model suggested that an injector placed in the northern Miocene, (where aquifer size is believed to be more limited than in the south) had the potential of adding an additional 10 MMBOE of recoverable reserves. A more detailed investigation of the location and pumping rates of injectors was studied in this work.

4.2 Controlled Experimentation

The HGA was used first to optimize the location of one injection well injecting water at constant rate. Fixing the pumping rate decreased the size of the search space and the relatively small search space enabled the exhaustive simulation of every possible well location. The exhaustive run for this problem was made by carrying out a

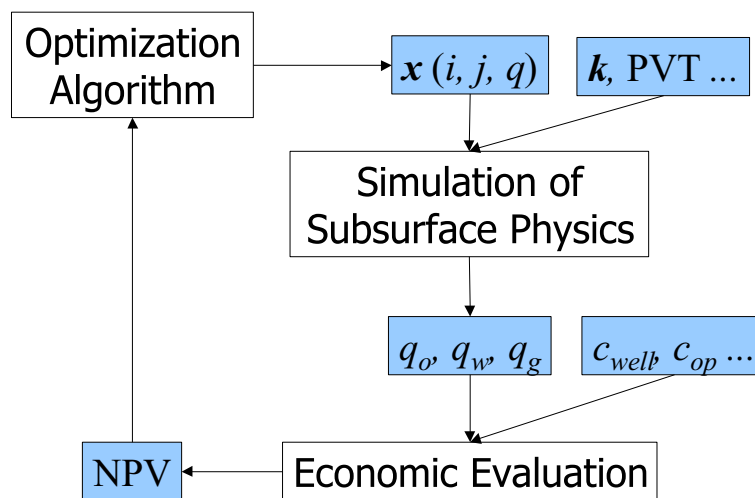


Figure 4.2: Illustration of the process for the evaluation of well configurations for the optimization of Pompano field development.

simulation at every possible active cell in the numerical model. The corresponding incremental NPVs from the no-injection case for each well location were calculated from the simulation output. The process of evaluation of a well configuration is illustrated in Fig. 4.2. The parameters for NPV calculations are given in Table 4.1. The resulting incremental NPV surface is shown in Fig. 4.3. This NPV surface is very noisy. Gradient-based techniques would fail miserably in locating the global maximum for this surface. The global maximum on this surface is at (25, 12) with an incremental NPV of \$21.9 million. Note that some injector well locations result in loss. Possession of such an exhaustive run provided the means to carry out sensitivity analysis without further simulation. When the HGA placed a well on the numerical model, the corresponding NPV could be looked up from a table.

While it is the simplest, this problem offers the most control in overall performance assessment because the exhaustive run is possible. The exhaustive run also provides insight about the more complex well optimization problems of similar sort. 100 runs were made for each case considered since the process is stochastic. A mean and standard deviation value was obtained for each case.

The total simulation time was 4.6 years with 2 years of water injection. Various combinations of the proposed methods were experimented with for this problem. The

Table 4.1: Parameters for net present value calculations.

Parameter	Value
Discount rate, %	10.0
Oil price, \$/bbl	25
Gas price, \$/MSCF	3
Water handling cost, \$/bbl	1
Operation cost, \$/day	40,000
Well cost, \$/well	20,000,000
Capital expenditures, \$	50,000,000

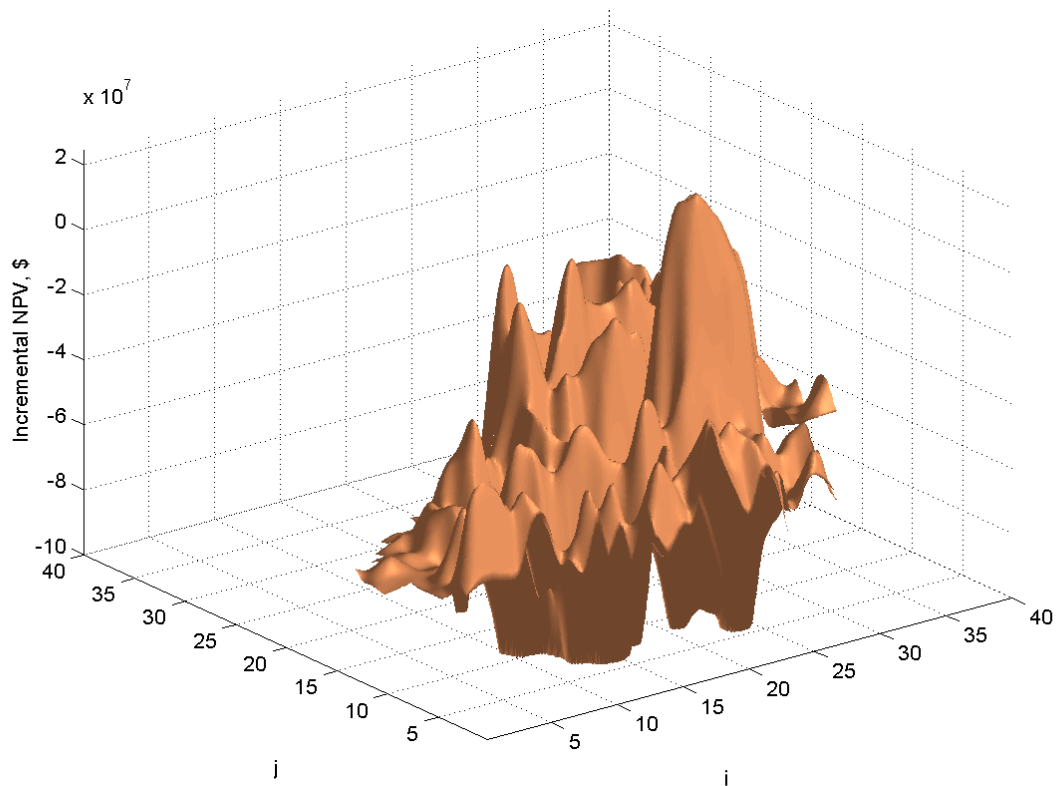


Figure 4.3: The incremental net present value surface for the single injector well placement problem generated by exhaustive simulations.

rule of thumb values for HGA parameters were utilized. A more detailed study of sensitivity of optimization performance algorithm parameters is presented in Chapter 7.

Performance of the different combinations of the proposed algorithms were compared. A maximum of 300 evaluations and 100 iterations were allowed for all the algorithms. The number of simulations consumed to correctly locate the optimum well location are given in Table 4.2 and visualized in Fig. 4.4. Some runs fail to achieve the global optimum given 300 evaluations and 100 iterations. The success percentage of the algorithms is also given in Table 4.2. It is observed that ordinary GA performs poorly with an average of 182.0 simulations and fails to locate the global optimum in 36 of the 100 runs carried out. Utilization of the polytope method with the GA improves the search efficiency and the average number of simulations required to correctly locate the optimum location is decreased to 141.0. Utilization of kriging further improves the algorithm efficiency and the average number of simulations is 114.0 for this case. The introduction of the error analysis step does not significantly improve the performance and the average number of simulations is 115.0. Introduction of the kriging proxy greatly improves the success percentage of the hybrid.

The hybrid employing the neural network is less efficient with an average of 133.2 simulations. It should be noted that this is still better than the ordinary GA and the GA-polytope composite. However several issues influence the efficiency of the neural network proxy. The size and structure of the network and the number of training iterations were kept constant throughout the optimization process. The neural network estimates are poor initially since the network overfits the small number of points at the earlier generations. Network estimates are also poor at later generations when there are a lot of data points for an accurate estimation. In other words the network does not generalize at early generations and is not representative enough at later generations. This problem might be overcome by varying the size and structure of the networks and also the number of training iterations as the optimization algorithm progresses.

The effect of invalid point interpretation and local mutation was also investigated. It was observed that when invalid point interpretation or local mutation were not used

Table 4.2: Means (m) and standard deviations (σ) of the number of simulations and the success percentage (%) to correctly locate global optimum for the composite algorithms: Genetic Algorithm (GA), Polytope algorithm (P), Neural Network (NN), Kriging (K) and Error analysis (E).

	GA	GA+P	GA+P+NN	GA+P+K	GA+P+K+E
m	182.0	141.0	133.2	114.0	115.0
σ	92.1	94.1	66.6	72.4	72.3
%	74	80	91	94	95

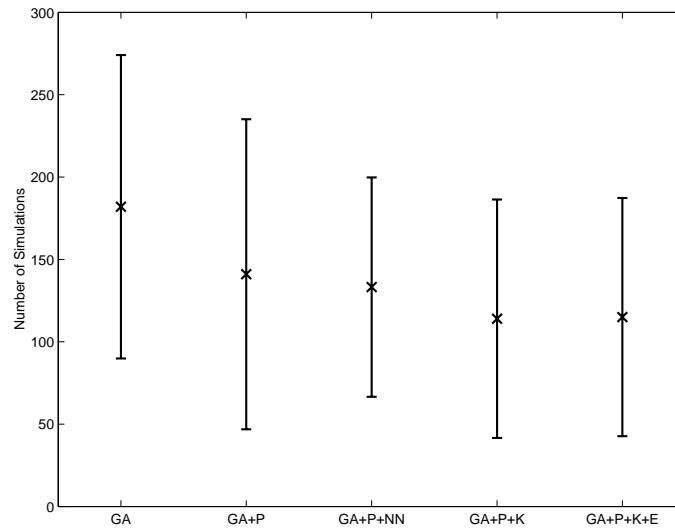


Figure 4.4: Comparison of the mean (crosses) and the standard deviation (error bars) for different combinations of Genetic Algorithm (GA), Polytope method (P), Kriging (K) and Error analysis (E).

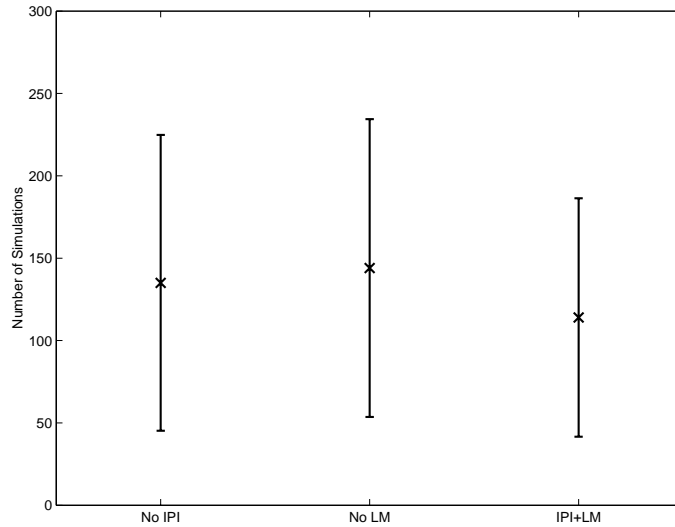


Figure 4.5: Effect of invalid point interpretation (IPI) and local mutation (LM) on algorithm performance; crosses indicate the means and the error bars indicate the standard deviations of 50 runs.

the helper methods were less efficient. The effect of invalid point interpretation and local mutation on algorithm performance is summarized in Fig. 4.5.

The evolution of the kriging estimate of the NPV surface for one of the runs using GA, polytope method and the kriging proxy is shown in Fig. 4.6. It is seen that as generations progress and more points are visited, the kriging estimates get better and eventually at generation 6, the algorithm correctly locates the optimum well location resulting in the highest NPV.

4.3 Multiple Well Location and Rate Optimization

The configuration and pumping rates of up to four wells were optimized using the HGA. In these cases it is not feasible to carry out exhaustive runs since the search space size is very large (Table 4.3) and a very large number of simulations would be necessary. The pumping rate was discretized into seven intervals from 0 STB/D to 30,000 STB/D which allows optimization of rate with a precision of 5,000 STB/D. Search space sizes for the problems considered are shown in Table 4.3. Runs were

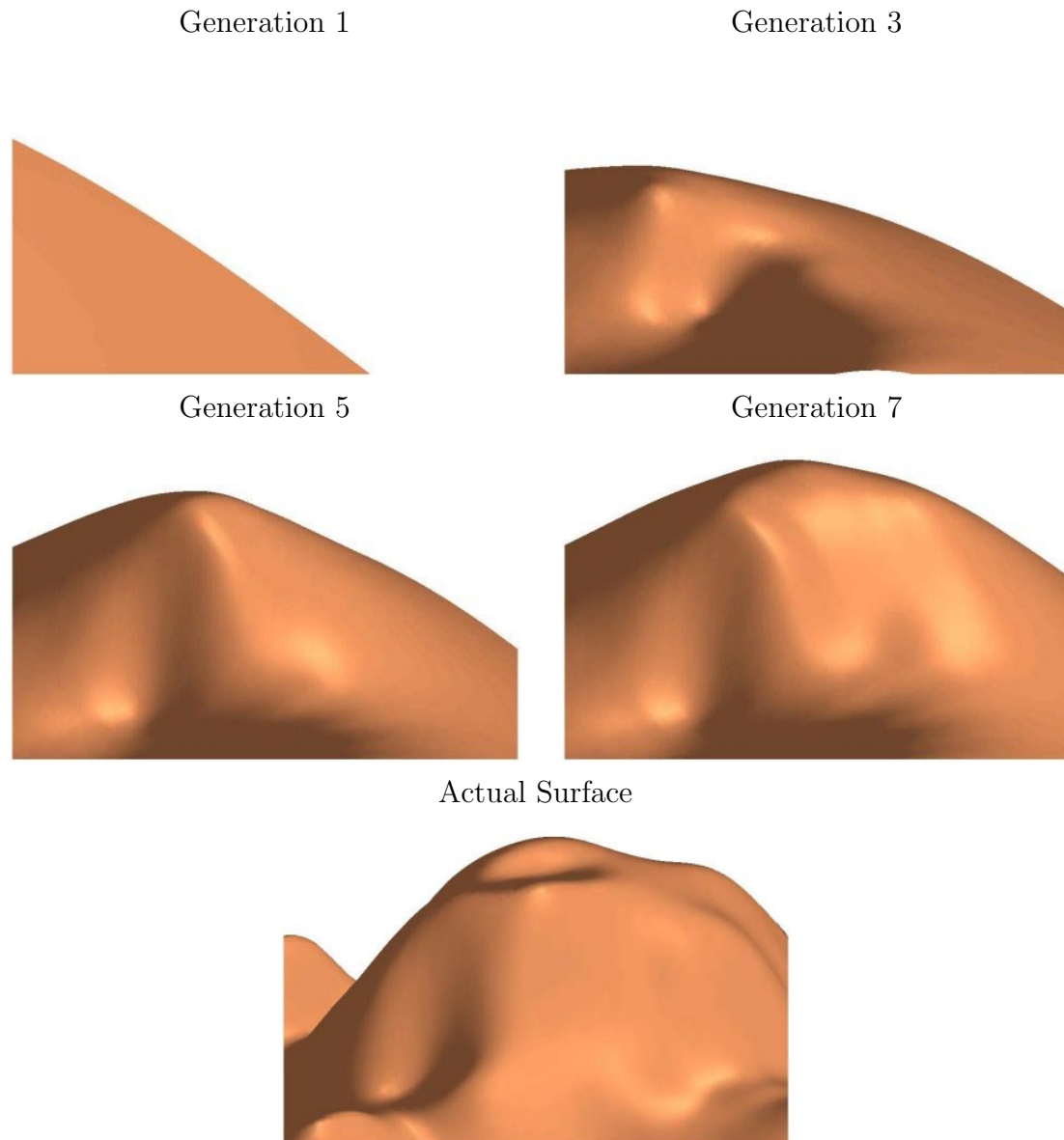


Figure 4.6: Evolution of the kriging estimation around the global optimum for the single injector placement problem.

allowed to consume a maximum of 300 evaluations before termination. Simulation times were a total of 7.6 years with 5 years of injection. Results are given in terms of incremental NPV from the base case, where there is no injection. The configurations and rates of injection wells proposed by the HGA and the resulting NPVs are given in Table 4.4. The locations of the existing producers and the proposed injector locations are plotted in Fig. 4.7 through Fig. 4.10. From the results in Table 4.4, it is observed that water injection increases profit in all cases and that three injectors with the proposed configuration is the most profitable development plan.

One Injector The algorithm converges to an (i, j) location of $(24, 13)$ and a pumping rate of 30,000 STB/D. Well location is shown in Fig. 4.7. This location has the best balance between injectivity, pressure maintenance and water cut. The pumping rate hits its upper limit suggesting that the pressure maintenance factor was dominating. The incremental NPV is \$63 million.

Two Injectors The locations for two injection wells proposed by the HGA for highest NPV are shown in Fig. 4.8. These locations have favorable rock properties and are strategic for better sweep and pressure maintenance. In this case the pumping rates of the two wells are 25000 STB/D. The pumping rates do not hit the upper limit in order to prevent high water cut. The HGA solution is intuitive in that it provides a good balance between pressure maintenance and water production. The incremental NPV is \$115 million for the two wells case.

Three Injectors The well locations proposed by the HGA for the placement of three injection wells are shown Fig. 4.9. The proposed configuration results in an incremental NPV of \$154 million. Among the configurations considered, this well configuration results in the highest NPV value for the Pompano field thus the optimum number of wells is three.

Four Injectors For the four well placement problem, the HGA places the wells as shown in Fig. 4.10. Introduction of the additional well does not pay off and the resulting incremental NPV of \$151 million is less than the three-well scenario.

Table 4.3: Search space sizes for the problems considered for the Pompano field.

Number of Injectors	Search space size
1	6,090
2	18,522,735
3	37,514,712,620
4	56,919,197,722,695

Table 4.4: Number of injectors (n_{inj}), pumping rates in MSTB/D, number of simulations (n_{sim}) and the resulting incremental NPVs in MM\$ for the HGA runs.

n_{inj}	n_{sim}	NPV	(i_1, j_1)	q_1	(i_2, j_2)	q_2	(i_3, j_3)	q_3	(i_4, j_4)	q_4
Base Case	1	0								
1 Well	106	63	(20,4)	15						
2 Wells	155	115	(25,14)	25	(17,20)	25				
3 Wells	275	154	(30,22)	15	(33,26)	15	(23,13)	20		
4 Wells	254	151	(28,12)	30	(34,21)	30	(17,4)	20	(23,14)	30

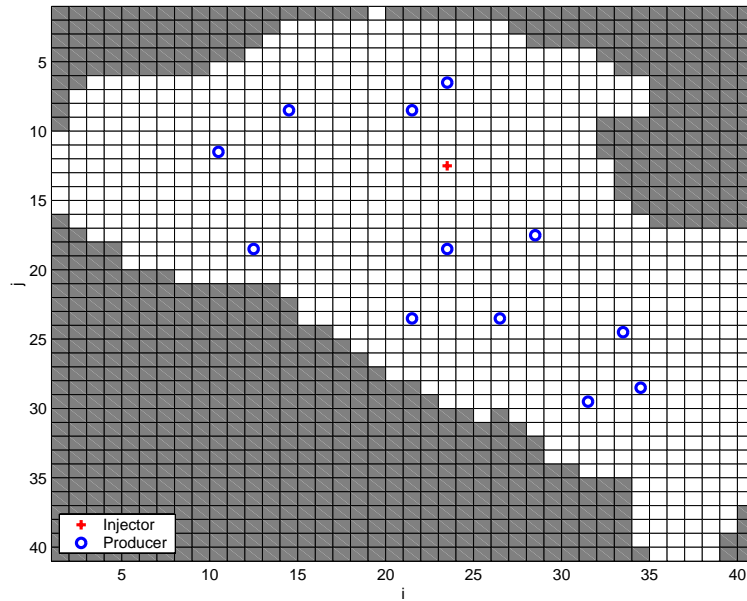


Figure 4.7: Optimum well location for the one injector placement problem for the Pompano field.

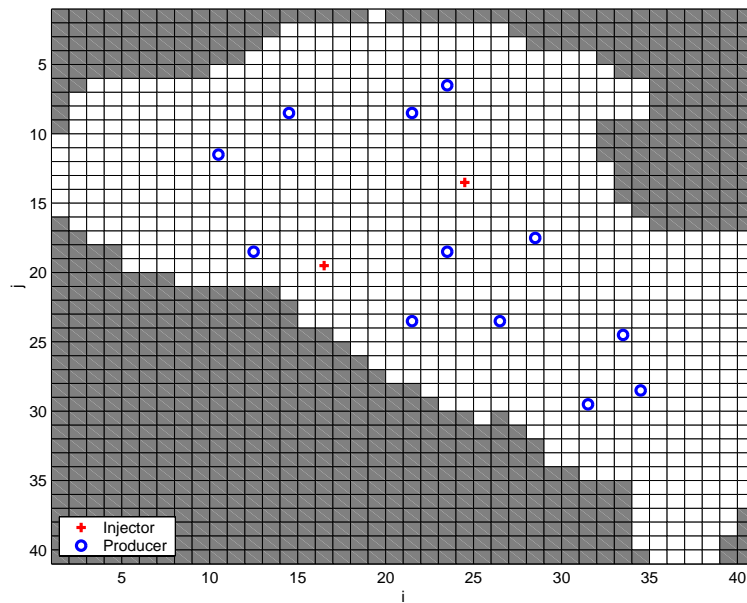


Figure 4.8: Optimum well location for the two injector placement problem for the Pompano field.

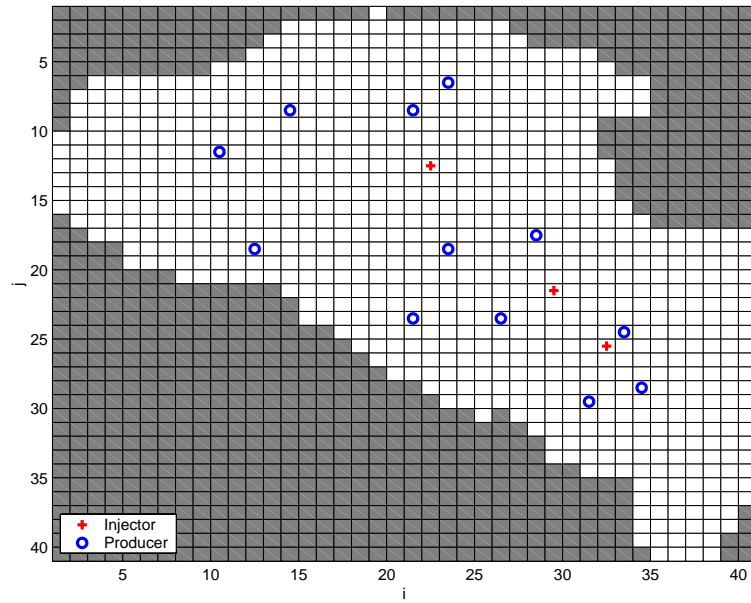


Figure 4.9: Optimum well location for the three injector placement problem for the Pompano field.

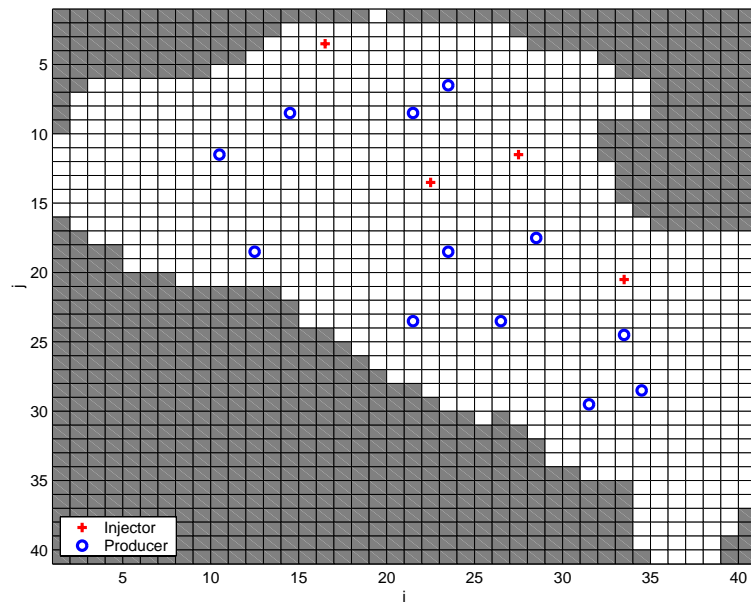


Figure 4.10: Optimum well location for the four injector placement problem for the Pompano field.

In all the cases considered the HGA proposed intuitive solutions. A reservoir engineer would not oppose the well locations and pumping rates proposed by the HGA. It should be noted that, different from a human being, the optimization procedure is able to evaluate all the effects of possibly hundreds of factors in a straightforward and precise manner. Some of these factors are rock and fluid properties, physics of flow through porous media, economic parameters, etc. Most of these factors have nonlinear and implicit effects on the objective function, which are hard to evaluate manually. In particular it should be noted that adding more injectors changes the optimal location of earlier injectors; this phenomenon would prevent the application of successive or sequential optimization methods such as dynamic programming.

4.4 Comparison with the DeepLook Approach

The well placement problem for the Pompano field has also been studied by the DeepLook consortium (Chawathe, 1999). Johnson and Rogers (2001) used a neural network proxy for the numerical model for the Pompano field and optimized using GA. Although the reservoir simulator and the economic parameters for NPV calculations were different in this study, a qualitative comparison is noted. Johnson and Rogers proposed 25 candidate locations for injection wells based on injectivity concerns. The determination of these 25 locations was based on reservoir properties thus was independent of the objective function (NPV). These preselected 25 candidate locations and the HGA proposed well locations are shown in Fig. 4.11. The optimum location for the single injector placement problem, for which the optimum was found explicitly here through exhaustive simulation, does not fall onto any of the 25 preselected locations. Also comparison was made for the three injector placement case. The top ranked three-well optimum configuration was run through the HGA for pumping rate optimization. The resulting NPV of the project was \$159 million less than the solution proposed by the HGA for the three well configuration. Thus there is evidence that a preselection of locations based on criteria other than the objective function can be limiting. Such preselection will especially be problematic in multiwell scenarios where well interactions become very important.

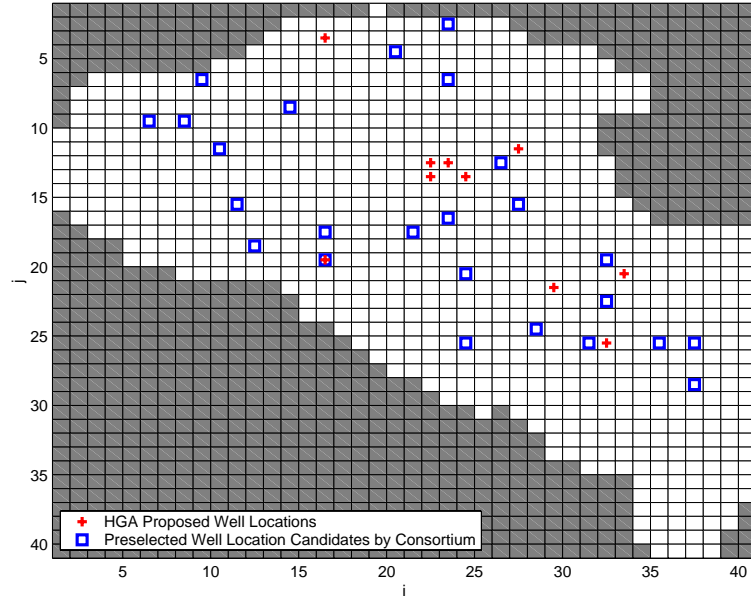


Figure 4.11: The 25 candidate well locations selected by the DeepLook consortium prior to optimization and the locations proposed by the HGA.

4.5 The Quality Map Approach

The quality map was introduced by da Cruz *et al.* (1999) in an effort to quantify the complex interactions and reservoir response into a simple two-dimensional representation. da Cruz *et al.* also suggested an approach to locate single and multiple wells using the quality map. This suggested approach of well placement with the quality map was improved and applied to the Pompano field injector placement problem.

4.5.1 Quality Map Definition

The objective is to maximize the total quality (Q_t) given the quality values for each well (Q_w):

$$Q_t = \sum_{c=1}^{n_w} Q_w \quad (4.1)$$

where n_w is the number of wells.

The quality of each well can be calculated by adding the inverse distance weighed qualities of cells belonging to that well:

$$Q_w = \sum_{c=1}^{n_{cw}} Q_c \cdot w_c \quad (4.2)$$

where n_{cw} is the number of cells belonging to the well.

The cell weights (w_c) are given by:

$$w_c = \frac{1}{a \cdot d_{wc}^b} \quad (4.3)$$

where d_{wc} is the well-cell distance and a , b are constants and $w_c = 1$ for $d_{wc} = 0$.

da Cruz *et al.* proposed $a = 2$ and $b = 2$ as the result of some sensitivity studies. These values were also used here.

The advantage of the quality map approach is that no simulations are performed beyond those used to generate the map. This may make the approach attractive when simulations are expensive. In effect the quality map is used as a form of proxy.

4.5.2 Optimization with the Quality Map

da Cruz *et al.* also proposed a simple optimization heuristic for the well placement problem which has a high likelihood of delivering suboptimal solutions. In this study the HGA was utilized for the optimization of total quality for single and multiple well locations. The exhaustive NPV map for the Pompano field for the single injector placement problem injecting with constant rate was used as the quality map. Up to four well locations were optimized. The results are given in Figures 4.12 through 4.15. Injection rates were fixed because including variable injection rates is not straightforward with the quality map approach, although some kind of ad hoc scaling might usefully be applied.

In order to evaluate the results obtained by the quality map approach, direct optimization of well locations with constant injection rates was carried out. As shown in Figures 4.16 through 4.19 the well locations proposed by the quality map approach are significantly different from those obtained by direct optimization using numerical simulation for the evaluation of well configurations. Numerical simulation with the

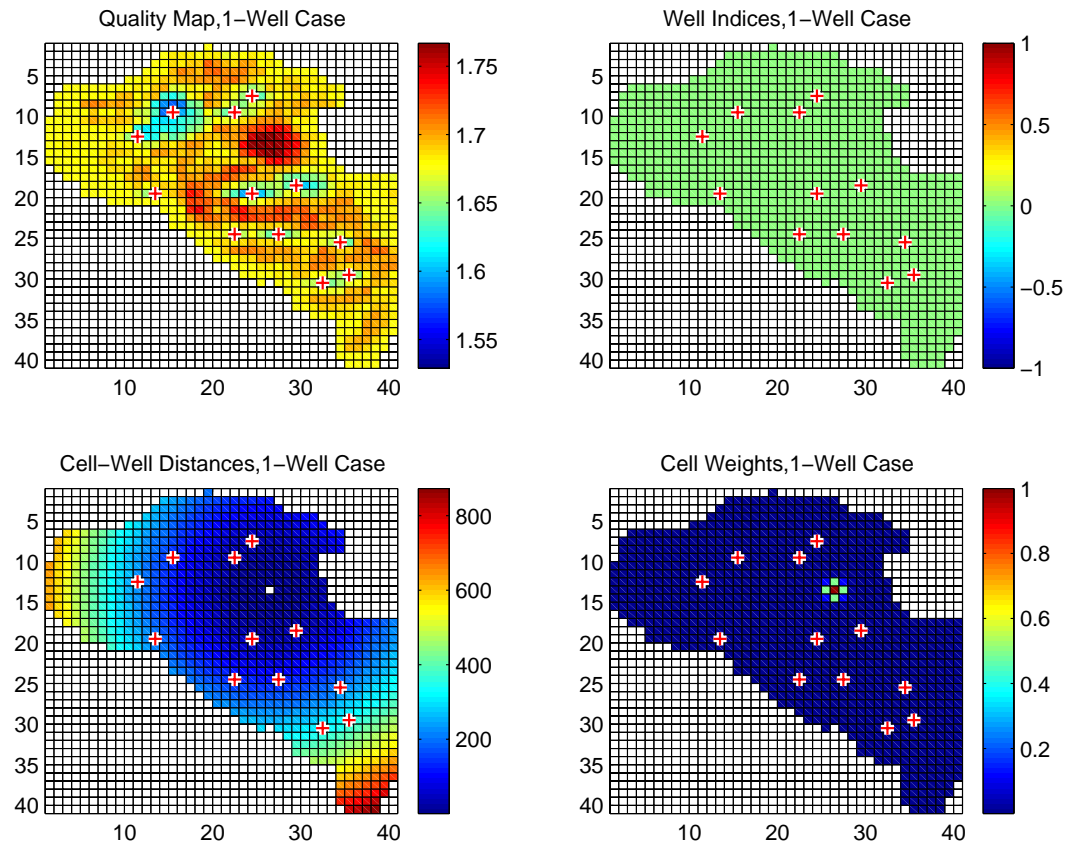


Figure 4.12: Resulting optimized single well location with the quality approach for the Pompano field; red-white crosses indicate the location of existing producers and the proposed well locations have unit cell-weight.

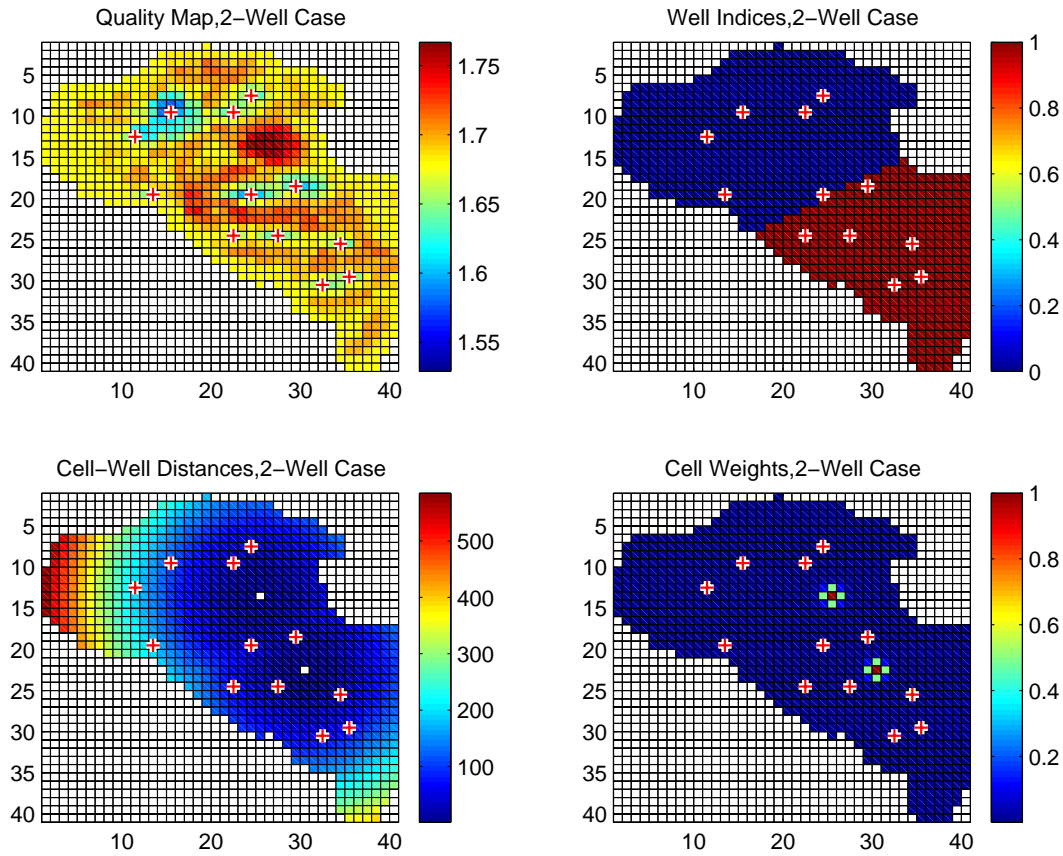


Figure 4.13: Resulting optimized two well locations with the quality approach for the Pompano field; red-white crosses indicate the location of existing producers and the proposed well locations have unit cell-weight.

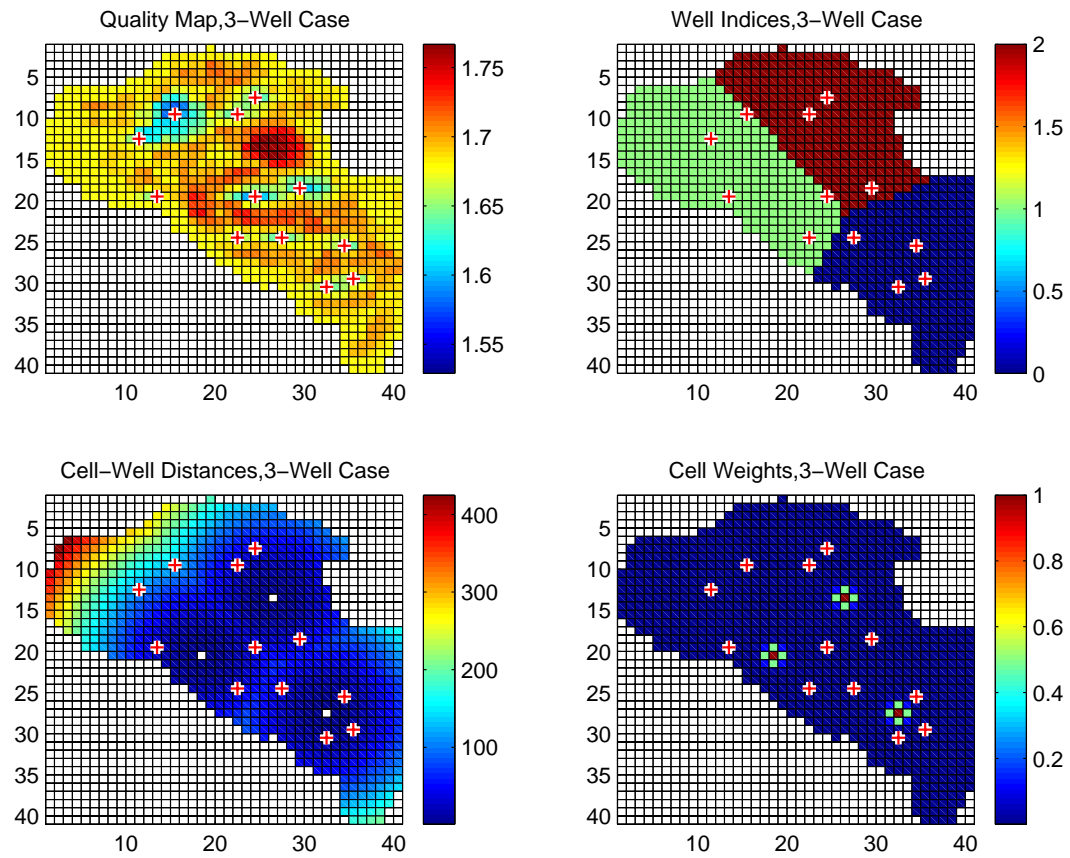


Figure 4.14: Resulting optimized three well locations with the quality approach for the Pompano field; red-white crosses indicate the location of existing producers and the proposed well locations have unit cell-weight.

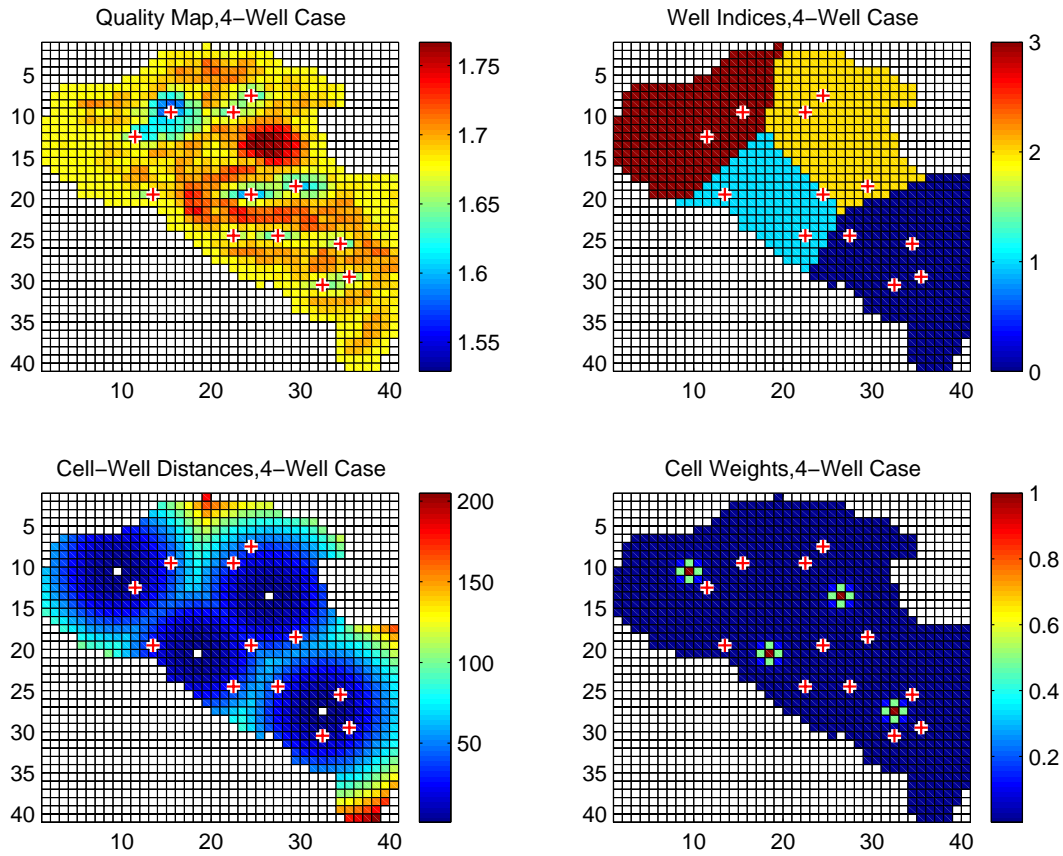


Figure 4.15: Resulting optimized four well locations with the quality approach for the Pompano field; red-white crosses indicate the location of existing producers and the proposed well locations have unit cell-weight.

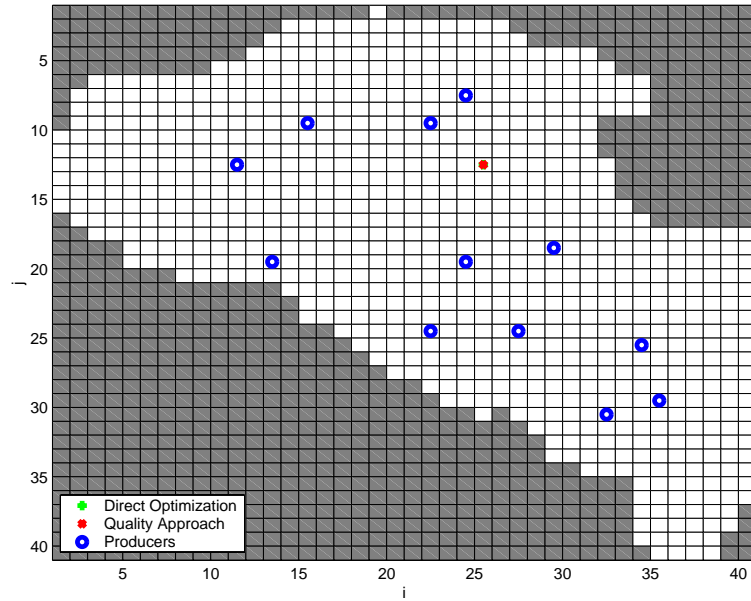


Figure 4.16: Well locations suggested by the quality approach and direct optimization using the numerical simulator as the evaluation function for the Pompano field one injector placement problem.

well configurations suggested by the quality map approach were carried out. Quality map approach results in less incremental NPV for multiple well locations as given in Table 4.5.

The well locations obtained with the quality map approach using the HGA for

Table 4.5: Comparison of resulting incremental net present value calculations for the quality map and direct optimization approaches.

	Quality NPV, \$ $\times 10^7$	Direct Optimization NPV, \$ $\times 10^7$
No Wells	0.00	0.00
1 Well	2.19	2.19
2 Wells	-2.84	3.05
3 Wells	2.82	7.59
4 Wells	1.40	10.00

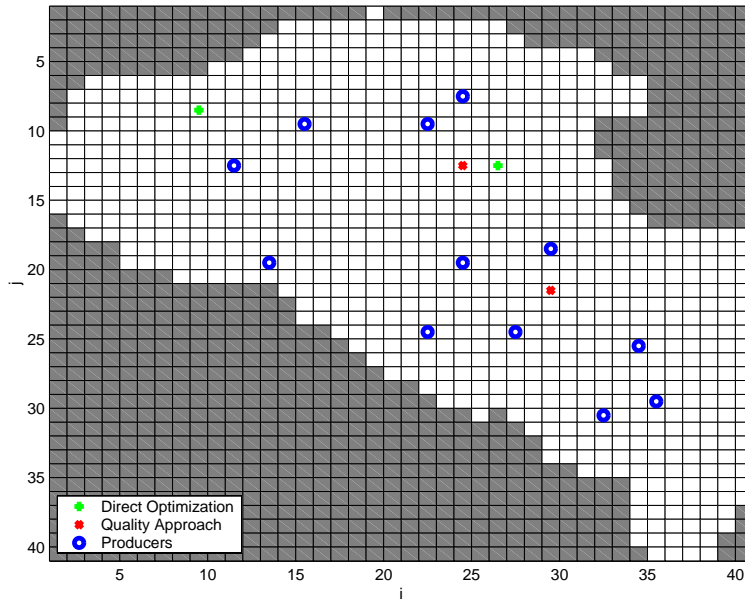


Figure 4.17: Well locations suggested by the quality approach and direct optimization using the numerical simulator as the evaluation function for the Pompano field two injector placement problem.

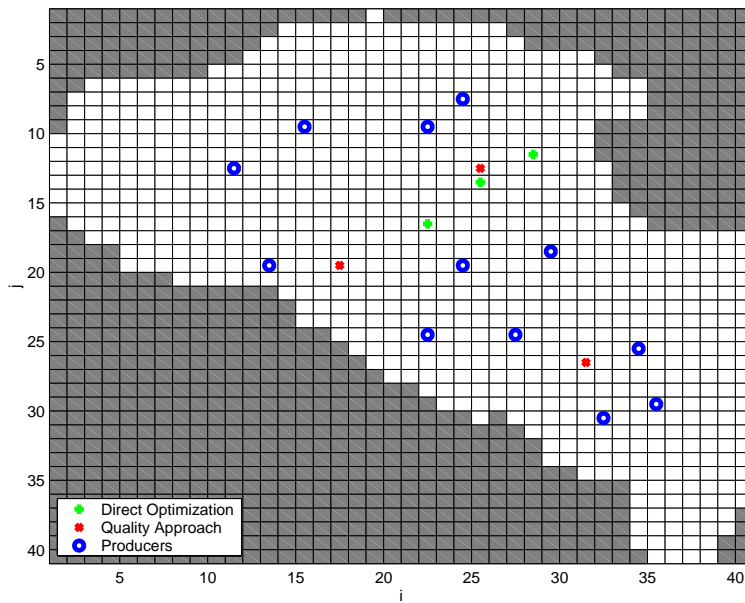


Figure 4.18: Well locations suggested by the quality approach and direct optimization using the numerical simulator as the evaluation function for the Pompano field three injector placement problem.

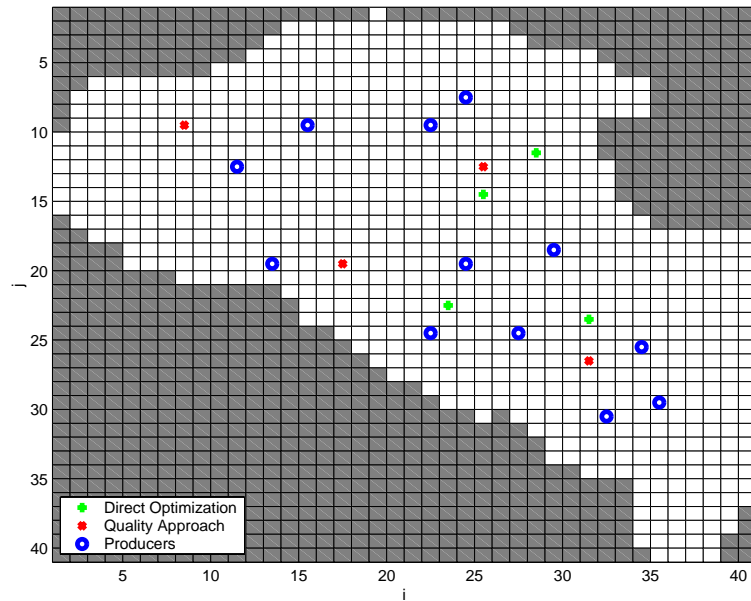


Figure 4.19: Well locations suggested by the quality approach and direct optimization using the numerical simulator as the evaluation function for the Pompano field four injector placement problem.

optimization appear to be sensible when one considers the relative locations of producers and other injectors (Figures 4.12 through 4.15). However the well interactions are modeled implicitly with a very simple inverse distance weighing scheme which will not be sufficiently accurate for real-world decisions. This can be observed in Figures 4.12 through 4.15 where it is seen that introduction of additional injectors does not significantly change the locations of previous injectors. Although the optimization procedure is simultaneous the results are as if the injectors were introduced sequentially. Consequently the quality approach may be useful for initial approximate evaluation of the well placement problem or perhaps as a helper method for full and direct optimization but not as a stand-alone decisive method. The significantly lesser incremental NPV values obtained by the quality approach (Table 4.5) for the multiple well placement problems indicates the importance of well interactions.

Chapter 5

Real-World Field Development Optimization

Optimization of field development of a real-world reservoir has been undertaken. The reservoir will be referred to as REAL. The numerical model for the REAL field has approximately half a million active cells, thus optimization by using the numerical model as the evaluation function poses a significant challenge. The optimum deployment schedule of 13 potential producers that would meet a production schedule was sought.

5.1 Problem Definition

5.1.1 Optimization Objective

The objective of the full field optimization problem is defined as follows:

Objective - Maximization of profit (NPV) by scheduling the deployment of the 13 predrilled candidate producers to meet a specific production target schedule.

There are 13 predrilled producers ready for deployment. Each producer is associated with a different deployment cost. There is a field oil production schedule imposed by the operating company. The numerical simulation forecast with no new

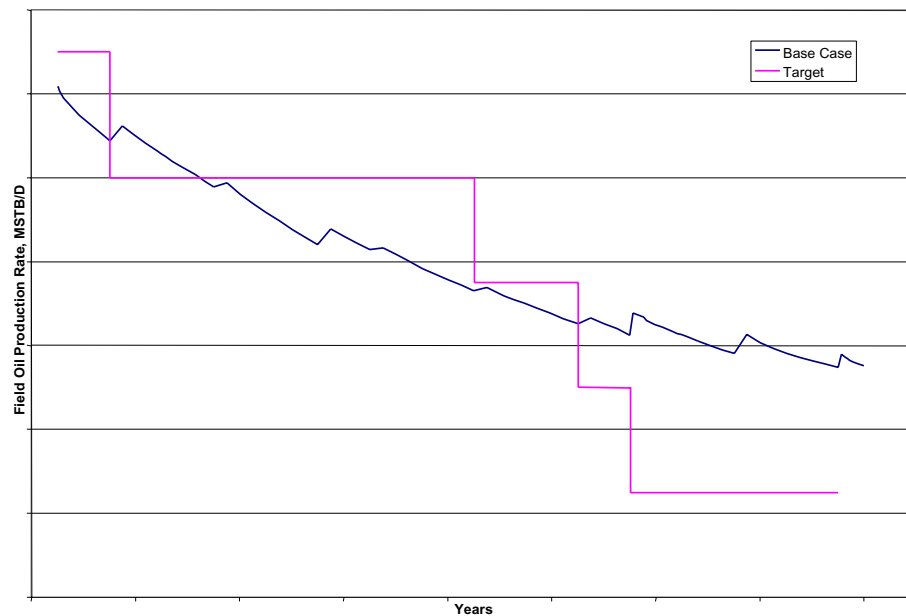


Figure 5.1: The target rate and production forecast for the REAL field when no new wells are deployed.

wells fails to meet this imposed production schedule for some periods (Fig. 5.1) which necessitates the deployment of some or all of the new wells.

The optimal deployment schedule of the 13 predrilled producers was sought. Due to the imposed production constraint, maximization of NPV is equivalent to the minimization of the cost of meeting the production target constraint. This is due to the fact that, out of two deployment schedules that both meet the production target at all times, the one that costs less to implement will have a higher NPV. The NPV of a deployment schedule will depend on the flow forecasts made by the numerical simulator and also the deployment dates and the costs of the wells used.

5.1.2 Production Constraint Handling

During the production forecast, when the production rate exceeds the imposed production constraint, the numerical simulator has to decrease the field production. Two different strategies to achieve this were considered. One strategy would be to shut the

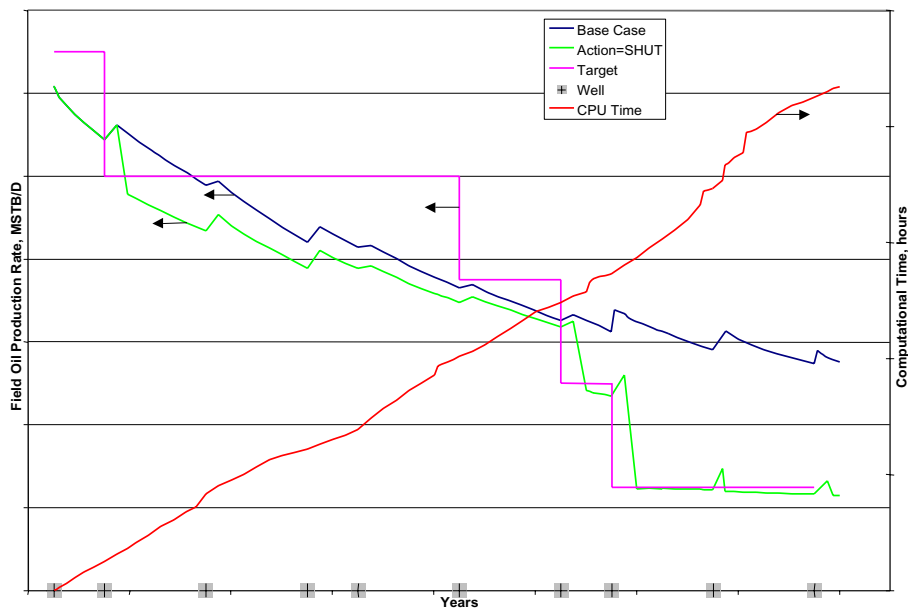


Figure 5.2: The target rate and production forecast for the REAL field when new wells are deployed and the wells are shut when production target is exceeded.

more prolific wells when the field production has to be reduced. The numerical simulation results when highest producing wells are shut once the field production exceeds the target is given in Fig. 5.2. It is observed from Fig. 5.2 that the field production is quickly cut below the production target when the target is exceeded, however there is no simulator feature that would turn the well back on once it is needed again. Also it does not make practical sense to shut wells and then to introduce new wells to match the production targets.

A second strategy would be to choke back producer rates until the field production target is matched exactly. Numerical simulation forecasts when the numerical simulator scales back simulation rates in the case of overproduction is given in Fig. 5.3. It is observed from Fig. 5.3 that overproduction is scaled back to the target rate. It is also observed that computational time increases drastically when the numerical simulator tries to scale back the significant overproduction during the last two periods of the target production schedule. This is due to convergence problems associated with rate scaling. Fortunately in this case, optimization for the last two periods is

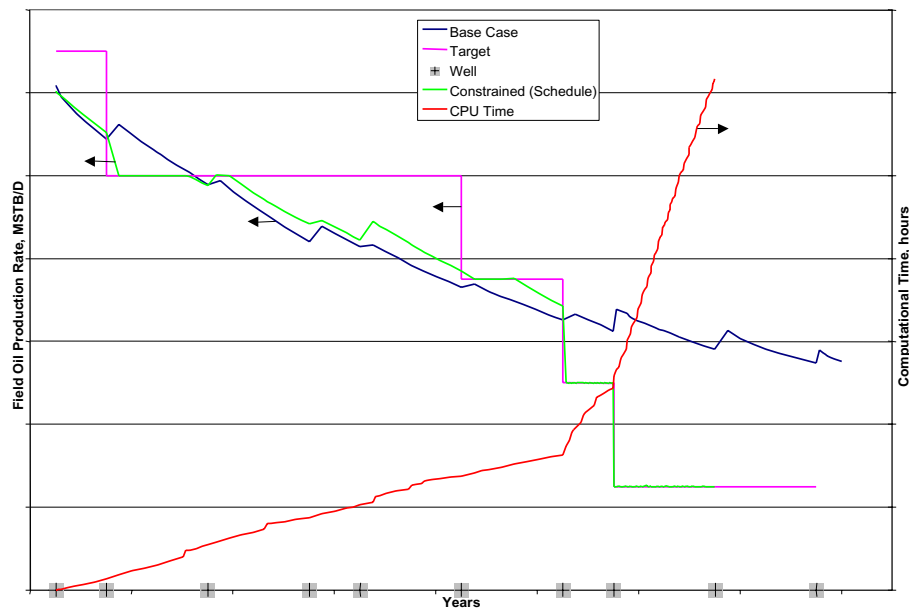


Figure 5.3: The target rate and production forecast for the REAL field when new wells are deployed and the well rates are scaled back to meet the target rate in the case of overproduction.

not necessary since the field is able to meet the target rates even when none of the wells are employed (Fig. 5.1), thus numerical simulations and the optimization search space do not need to cover these last two periods.

The second strategy of scaling rates was used for the REAL problem since this strategy is less problematic and optimization is expected to lead to more practical results. Also the convergence problems may be minimized in this particular case by excluding the last two periods of the production rate target for which these problems are intense.

5.2 Optimization Approach

5.2.1 Formulation

The well deployment scheduling problem for the REAL field may be formulated in several different ways. One can optimize the deployment date of every individual well.

This will result in a search space size given by $(n_{time})^{13}$ where n_{time} is the number of time points at which the wells are to be deployed. For instance if the deployment dates were to be optimized over a period of 9 years and the precision of dates were to be one year, then we would have 10 time points and the search space size would be 10^{13} .

Another approach would be to make use of the numerical simulator feature that allows the user to specify a drilling queue from which the wells are deployed in the order specified only when the production rate from the field fails to meet the specified target. Hence the item to optimize becomes the order of wells in the drilling queue. Different ordering of wells results in a different deployment schedule which is determined by the numerical simulator. Formulation of the problem as the ordering of wells in the drilling queue results in a search space size of $13! = 6,227,020,800$ which is a $1/1606$ fraction of search space size of 10^{13} resulting from the previous formulation of optimization of the deployment date of every individual well.

Drilling queue optimization eliminates solutions that would propose deployment of wells even when the target rates have already been met, hence any deployment schedule obtained by using the drilling queue can be considered *preoptimized*. Given a well order the deployment schedule is optimized relying on a simple heuristic: *deploy the next well in the drilling queue when field production falls below target*.

This formulation also utilizes only the necessary number of wells, that is if the production target can be met with a subset of the 13 wells, the remaining wells will not be deployed by the numerical simulator. This also minimizes the convergence problems associated with rate scaling since the production target is never significantly exceeded. Another advantage is that the time dimension does not need to be discretized, in other words there is no dependence on n_{time} .

5.2.2 Solution Representation

Formulation of the approach as the ordering of wells within a drilling queue makes the optimization analogous to a Traveling Salesman Problem (TSP). In a TSP the task is to determine the order of cities the salesman is to visit while minimizing his traveling

costs. Every city has to be visited exactly once. In our REAL world problem we need to determine the order of wells which maximizes NPV and each well has to appear in the drilling queue exactly once. Although the objective is completely different than that of a TSP, the data structure will be similar.

There are several ways to represent TSP solutions within an optimization context. One can employ sequential representation where the cities are listed in the order in which they are visited. For instance a path that includes three cities might look like this: (2,1,3).

Such a representation is problematic in a GA context since straightforward crossover and mutation may result in a path where some cities are visited more than once and some not visited at all. Consider a crossover of two paths from the first crossing point:

$$(2, 1, 3) \times (1, 3, 2) = (2, 3, 2) \text{ and } (1, 1, 3)$$

Resulting individuals (2,3,2) and (1,1,3) are infeasible paths. Crossover and mutation operators customized for the TSP have been suggested (Grefenstette *et al.*, 1985). GA-customized TSP operators generally rely on local search heuristics which require the evaluation of costs of travel between individual cities. These TSP operators are often referred to as greedy operators. For instance, evaluation might be required to determine whether a subsection of the path would be cheaper if reversed. This is easily done for the TSP problem since costs associated with a subsection of the path can be evaluated without the evaluation of the complete path. However in the case of drilling queue optimization, there is no way to determine the effect of changing a subsection of the queue without carrying out full-field numerical simulation.

Other crossover or mutation operators that do not utilize local search or repair algorithms that correct an invalid path may be used. However these approaches require major customization of the GA without apparent gain in efficiency.

One can also employ specialized decision trees as the path representation in TSPs. The decision nodes at any point in this decision tree represent the next well (or city) in the queue. Instead of working on the well indices, we modify and optimize the branching order on the decision tree. These specialized decision trees start with a single decision node and initially branch to n_{well} decision nodes. Each of these decision

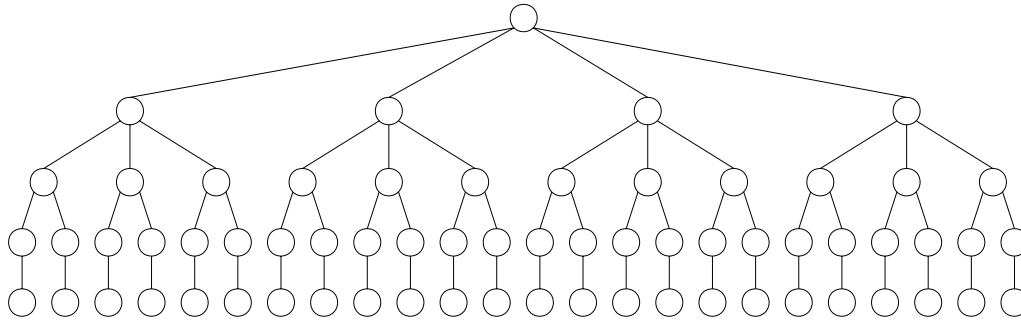


Figure 5.4: Decision tree for the four well ordering problem.

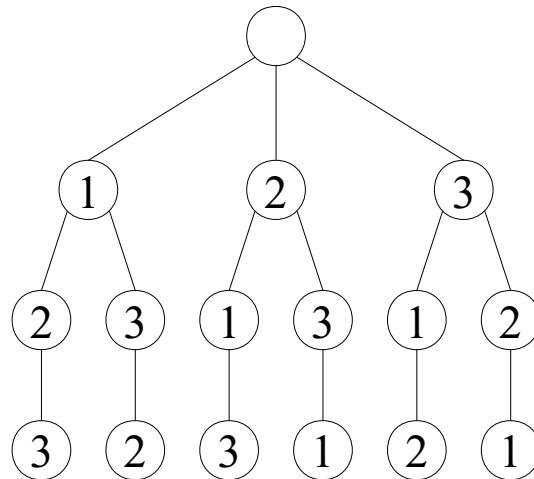


Figure 5.5: Decision tree for the three well ordering problem, the integers on the decision nodes are the well indices.

nodes branch to one less number of nodes with increasing tree depth until only one decision node is left. A sample decision tree for a problem involving four wells is illustrated in Fig. 5.4.

The order of n_{well} wells can be determined by $(n_{well} - 1)$ integers which represent the branching order on the decision tree. This will be demonstrated for three wells ($n_{well} = 3$). Consider the decision tree for the three-well ordering problem, shown in Fig. 5.5. The integers at the decision nodes in Fig. 5.5 are the well indices. The six ($n_{well}!$) possible well orders can be represented with this decision tree.

The order of three wells can be determined by two integers representing the branching order. Consider the following branching order: (1,2). As shown in Fig.

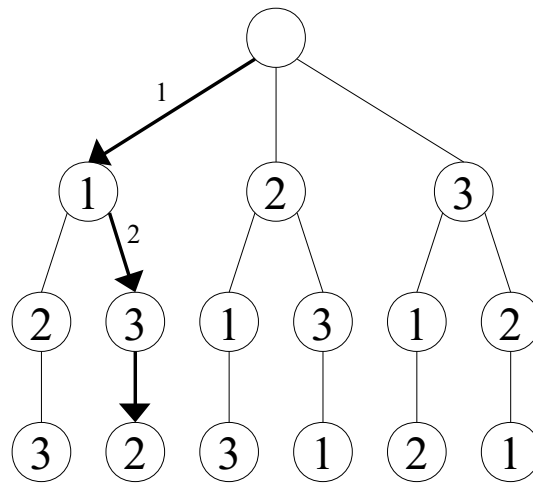


Figure 5.6: The path and the well order resulting from the branching order of (1,2).

5.6 if a branching order of (1,2) is used the resulting well order is: (1,3,2).

A decision tree for 13 wells was utilized for the REAL field deployment schedule optimization. It should be noted that one does not need to explicitly construct the decision tree which would be a computationally challenging task. Instead a simple algorithm to decode a branching order into a well order can be used (Fig. 5.7).

5.3 Results

The GA has been used to optimize deployment schedule since the hybrid components of the HGA are not suitable for the decision tree data structure. The GA was allowed to consume 100 numerical simulations. The optimization process was run in parallel on four processors. Refer to Appendix A for parallelization details.

The computational effort that went into the optimization process was significant, approximately 100 CPU-days on a single 2GHz Xeon processor, or approximately 25 days in parallel on four processors (two 2GHz and two 1.8GHz Xeon). However, considering the importance and possible implications of the decision and also keeping in mind that computer processors are the entities that do the work and not engineers, the amount of computational effort is reasonable.

The best solution found utilized all the wells to meet the production target. The

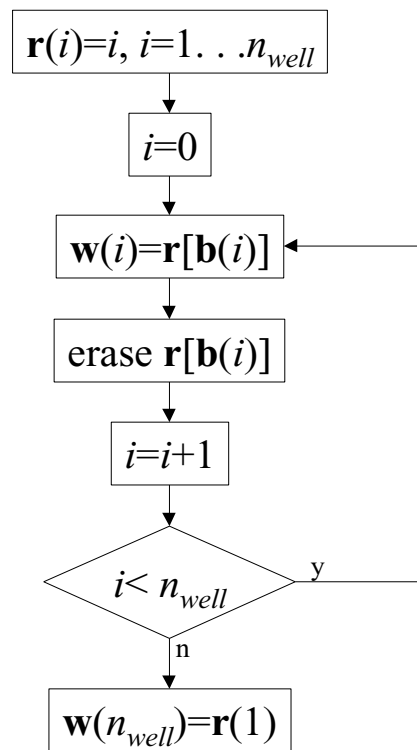


Figure 5.7: The algorithm to decode a branching order (\mathbf{b}) into a well order (\mathbf{w}) by modifying a vector of tasks that remain (\mathbf{r}).

best solution resulted in a 15.23% increase in NPV compared to the base case where no new wells were deployed. Also the best solution resulted in a 0.70% increase in NPV compared to the case where all the new wells were deployed at the beginning of the forecast. These percentage improvements translate into significant monetary value which justifies the computational effort spent in finding the optimal development plan.

Chapter 6

Assessment of Uncertainty

It is realized that the numerical model, on which we chose to base the well placement decision, relies on data that is incomplete thus the numerical simulation forecasts are uncertain. Optimization in such an uncertain case has many additional complications. A deterministic global solution is not available in the case of uncertainty. Given the data available, what we hope to achieve is to estimate the expected outcome of any proposed decision and also the risks associated with it. The established framework of decision and utility theory enables us to manage uncertainty (DeGroot, 1970). Realizing the fact that every decision maker would act differently given options with probabilistic outcomes, the framework also provides the tools necessary to quantify the risk attitudes of the decision maker (Holloway, 1979).

Decision theory framework has been used extensively and successfully in a wide range of industries (Chacko, 1993) including the petroleum industry (Simpson *et al.*, 2000; Thankur, 1995; Jonkman *et al.*, 2000; Erdogan *et al.*, 2001; Sarich, 2001). However it was observed that, in the petroleum industry, decision analysis tools are generally used during exploration and initial development stages (Jonkman *et al.*, 2000). Application of decision analysis tools to the reservoir development process has been limited due to the exponentially increasing number of options with added uncertainty in the decision trees. Due to this exponential growth property of the decision trees, decision makers have been forced to use approximate evaluation tools. In this study

we insist on using full numerical simulation as the evaluation tool. The use of numerical simulation was rendered computationally feasible by transforming the problem into a deterministic problem through utility functions that quantify risk attitudes and by using the HGA for optimization.

The utility framework requires the outcome probabilities for proposed well configurations. In some cases the determination of outcome probabilities might be computationally infeasible, particularly for very large numerical models. A second approach is presented in which the well placement problem has been formulated as the optimization of a random function which does not require the prior knowledge of outcome probabilities. The GA was used for optimization.

6.1 Utility Framework

The problem of well placement can be studied within the decision analysis framework since the problem consists of the decision of the location to drill the new well and the probable events thereafter.

6.1.1 Decision Tree Construction

The decision tree for the well placement problem is visualized in Fig. 6.1. Decisions are made at the decision nodes (square nodes). The decision in this study corresponded to the selection of a particular well configuration. Due to imperfect information about the truth, each decision leads to an event, $Event_i$, with different probabilities of occurrence, P_i . The outcome of an event is the Net Present Value, NPV_i . Each event is also assigned a utility value, U_i , which is a measure of the satisfaction of the decision maker over the possible range of outcomes. Satisfaction for any given outcome depends on the risk attitude of the decision maker. The risk attitude is quantified in the form of a mathematical function called the utility function which is used to translate an outcome (NPV) into utility. The utility function simply returns a utility value given the NPV. The final step is to calculate the expected utility by:

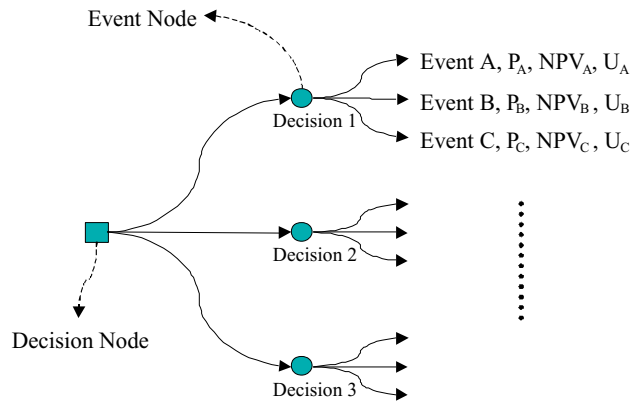


Figure 6.1: The well placement decision tree.

$$E \{U (decision)\} = \sum_{i=1}^{N_{event}} U_i \cdot P_i \quad (6.1)$$

The event nodes are collapsed into a single expected utility value as shown in Fig. 6.2. The decision with the maximum utility is then chosen. The number of decisions, that is, the number of possible well configurations is typically too many to evaluate exhaustively. Thus optimization by evaluating a subset of the decisions can be carried out to determine the decision that leads to maximum expected utility.

Expected value is used in this case since most of the histograms look symmetric and a consistent skewed distribution is not observed. If there is an evident basis that expected value is not representative of the distribution, one could consider using different statistical measures.

6.1.2 Utility Theory and Utility Functions

The whole process of decision tree construction and definition of the problem as the maximization of expected utility rather than the monetary value constitutes a transformation of the problem according to the decision makers attitude towards risk. Utility or preference theory explains how this transformation is possible (Holloway, 1979).

The utility function is the tool to quantify the decision maker's risk attitude

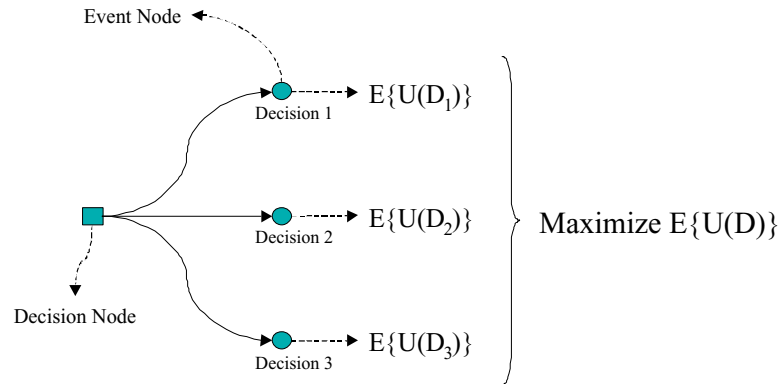


Figure 6.2: The well placement decision tree with event nodes collapsed into expected utility.

(Rubinstein, 1975). The shape of the utility function determines whether the decision maker is risk-neutral, risk-averse or risk-prone. A risk-neutral decision maker has a linear utility function which is equivalent to basing decisions purely on monetary value (NPV). A risk-averse decision maker has a concave utility function which corresponds to the avoidance of uncertain areas of the search space even if they might have the possibility of greater financial gain. The risk-prone decision maker has a convex utility function which represents a decision maker who is willing to take some risk for the chance of greater financial gain (Holloway, 1979).

A simple analytical utility function has the exponential form:

$$U(x) = a + be^{-rx} \quad (6.2)$$

where x is the objective function value which is NPV in this case.

A normalized version of Equation 6.2, with $a = 1$ and $b = -1$, is visualized in Fig. 6.3.

The curvature of the utility function determines the risk attitude of the decision maker as illustrated in Fig. 6.3. The magnitude of risk aversion of a given utility function, U , is given by:

$$R(x) = -\frac{U''(x)}{U'(x)} \quad (6.3)$$

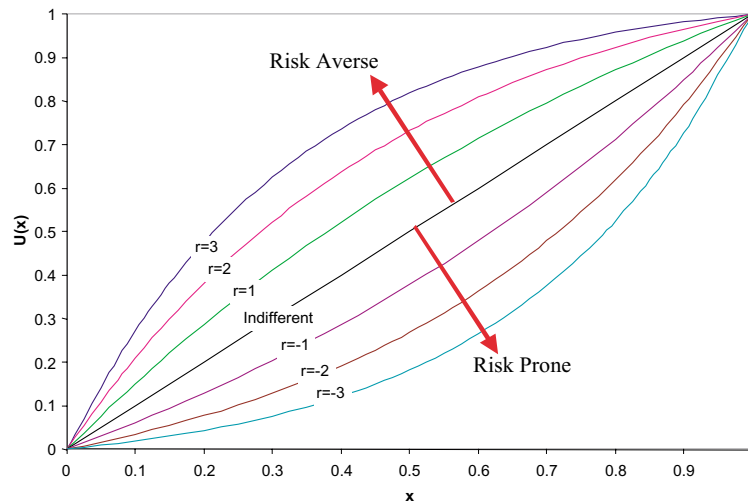


Figure 6.3: The exponential utility function for different exponent values, R .

The term $R(x)$ in Equation 6.3 is also referred to as the Arrow-Pratt measure of absolute risk aversion or the risk aversion coefficient (Holloway, 1979). The risk aversion coefficient is a constant for the exponential utility function and is equal to the exponent r in Equation 6.2.

6.1.3 The Certain Equivalent

The implications of the steps to construct the utility framework for any given problem are subtle. However the utility framework can be explained intuitively with the notion of certain equivalent. Consider the set of probable outcomes of a decision as a lottery. The certain equivalent of this lottery is the amount of certain monetary value for which the decision maker is willing to give up participating in the lottery, in other words, the decision maker's selling price of the lottery. This amount will be different for different decision makers with different risk attitudes. The utility of the certain equivalent of a lottery is equal to the utility of the lottery:

$$U(CE(L)) = U(L) \quad (6.4)$$

Thus the certain equivalent is given by:

$$CE(L) = U^{-1}(U(L)) \quad (6.5)$$

The certain equivalent or the selling price of a given lottery will depend on the decision maker's risk attitude. A risk-averse decision maker's selling price for the lottery will be less than the risk-prone decision maker's selling price since a risk-averse person prefers certainty to risk.

6.1.4 Axioms of Utility Theory

von Neumann and Morgenstern (1947) developed an axiomatic approach to utility theory. The decision maker should accept these axioms that the utility theory is based on. If all five of the axioms are applicable to the decision maker, the decision with the maximum expected utility will honor the decision maker's risk preferences. These axioms are:

1. **Orderability of Outcomes:** The decision maker must be able to rank the outcomes according to his/her preferences, thus transitivity among preferences must hold:

$$A \succ B \quad B \succ C \quad A \succ C$$

where \succ means *preferred over*.

2. **Continuity:** Given $A \succ B \succ C$, the decision maker can determine a probability p^* such that he/she is indifferent between receiving B for sure or facing a lottery where the possible outcomes are the best and the worst outcomes with respective probabilities of p^* and $1 - p^*$:

$$\begin{aligned} &\exists p^* : \\ &\{p = 1 \rightarrow B\} \sim \left\{ \begin{array}{l} p = p^* \rightarrow A \\ p = 1 - p^* \rightarrow C \end{array} \right\} \end{aligned}$$

where \sim indicates *indifference*. This axiom implies the following equality:

$$U(B) = p^*U(A) + (1 - p^*)U(C)$$

3. **Substitutability:** A lottery and its certain equivalent are interchangeable from the decision makers point of view.
4. **Monotonicity:** Given two event nodes (lotteries) with the same outcomes, the decision maker must prefer the lottery with the higher probability of obtaining the preferred outcome.
5. **Decomposability:** The decision maker is indifferent to the differing decision sequences of an otherwise equivalent outcome.

These axioms are all satisfied for the well placement problem, thus the decisions based on the utility framework will represent the risk attitude of the decision maker.

6.2 Random Function Formulation

An alternative approach is proposed for the cases where computational requirements of determining the outcome probabilities of well configurations is intractable. The well placement problem is formulated as the optimization of a random function. The GA was used as the optimization tool. Hill-climbing will not be effective in this case since the evaluation function value changes every time an evaluation is made at a specific point resulting in an uneven surface.

Each time a well configuration was to be evaluated a different realization of the reservoir properties was selected randomly from the set of realizations which all honor the geologic and dynamic data available from the reservoir. Numerical simulation was then carried out with this randomly selected realization to calculate the NPV and the objective function value was updated for this well configuration. For instance if the objective was to be to maximize expected NPV, the expected NPV for a specific well configuration would be updated every time the algorithm revisited this configuration. This results in a setup where the GA visits the feasible well configurations more frequently and the objective function values are more accurately computed for these feasible configurations. Furthermore the utility framework can be established within the random formulation context. Such a setup constitutes the gradual update of the expected utility value at the well configuration suggested by the GA.

The random function formulation is significantly different than optimization by calculating the objective function value (e.g. expected NPV) by carrying out simulations on all realizations whenever the GA proposes a solution. More numerical simulations are carried out for feasible well configurations with the random function formulation approach. The objective function value for the infeasible well configurations is not fully determined. Consider the following scenario:

The objective is to maximize expected NPV. The GA proposes a well configuration. Suppose that if numerical simulation on all realizations were made and the NPVs were calculated, the resulting expected NPV would be significantly below average. However we do not carry out simulation on all realizations, instead a single numerical simulation is carried out with a randomly selected realization at this well configuration and the NPV is calculated. Unless the choice of this realization was unlucky, the NPV resulting from the randomly selected realization will be below average and the GA will be less likely to visit this configuration later in the run. Suppose the choice of the realization was unlucky and we sampled a NPV from the higher tail of the NPV distribution. In this case the GA will revisit this point and another numerical simulation will be carried out with a randomly selected but *different* realization. Unless we are unlucky every time the GA visits this configuration the NPV will be sampled to be below average before numerical simulations on all realization are made. The same scenario can be considered for feasible well configurations. In this case the GA will revisit the feasible locations until the objective function value is determined absolutely.

The basic steps for the random function formulation follows:

- GA proposes a well location.
- Numerical simulation is carried out with a randomly selected realization and the objective function value is calculated.

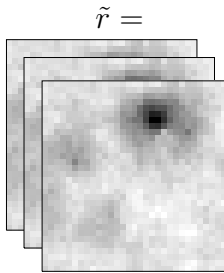
$$\begin{array}{cc}
 \vec{x}_1 & f_1 \\
 \vec{x}_2 & f_2 \\
 \vdots & \vdots \\
 \vec{x}_{n_p} & f_{n_p}
 \end{array}
 \rightarrow f_i = \tilde{f}(\vec{x}_i, \tilde{r})$$


Figure 6.4: Illustration of the suggested methodology to optimize in the presence of an uncertain reservoir model; the $\tilde{\cdot}$ indicates a random variable.

- If this is a feasible well configuration, then the GA will revisit it; another numerical simulation is then made with a randomly selected realization (different from previously simulated realizations) and the objective function value is updated.

The procedure is illustrated in Fig 6.4. The GA is known to be able to cope with such functions of random nature (Goldberg, 1989), that is the GA does not breakdown with changing objective function values as the algorithm progresses. The GA is anticipated to eventually converge to the well configuration with maximum objective function value. It should be noted once again that the problem can be set up to maximize expected utility (Fig. 6.1) and all the advantages of the utility framework can be gained. This is simply achieved by recalculating the expected utility value (Equation 6.1) for a well configuration whenever the GA revisits it.

6.3 Application to the PUNQ-S3 Reservoir

The proposed methodology for the uncertainty assessment of well placement optimization was tested in an application to the PUNQ-S3 reservoir. The PUNQ-S3 problem was evaluated within the utility framework assessing the effects of uncertainty. For comparison, the PUNQ-S3 problem was also formulated as the optimization of a random function.

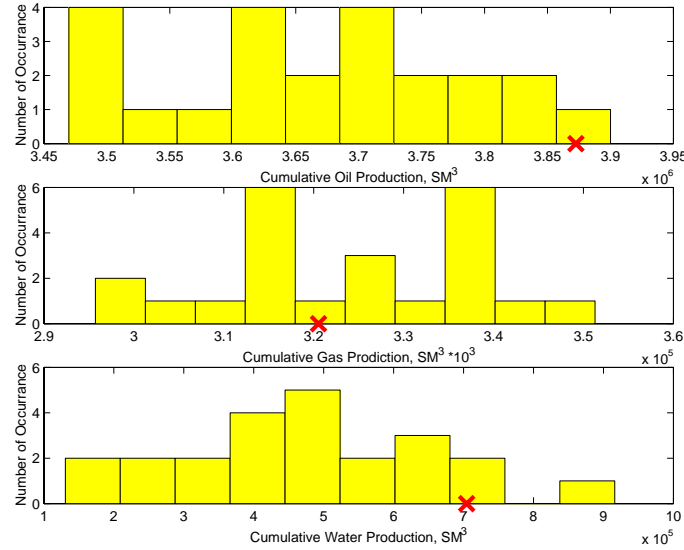


Figure 6.5: Histograms of the cumulative distributions for the 23 realizations of the PUNQ-S3 field, the cross indicates the true value.

6.3.1 The PUNQ-S3 Model

The PUNQ-S3 model is a standard test case that was based on a real field and was used for the PUNQ (Production forecasting with UNcertainty Quantification) project in context of the EU-Joule program (Barker *et al.*, 2000). PUNQ-S3 is a small-size industrial reservoir model based on a real field operated by Elf. The numerical model ($19 \times 28 \times 5 = 2660$ blocks, 1761 of them active) and 23 history-matched realizations were provided to us by Elf. The truth case from which these realizations were generated was also available. The dataset brackets the true cumulative properties, however is not a complete match to individual well histories (Figures 6.5 and 6.6). The truth case permeability fields for the five layers of the PUNQ-S3 model are shown in Fig. 6.7. The fields are very heterogeneous with connected high permeability streaks observed in all layers. The streaks are also observed in all the realizations. The field has six producers with 16.5 years of production history. The proposed development plan is an additional 5 years of water injection from a single infill well. Here we investigated the problem of finding the optimum location of this infill injector and assessing the uncertainty associated with it.

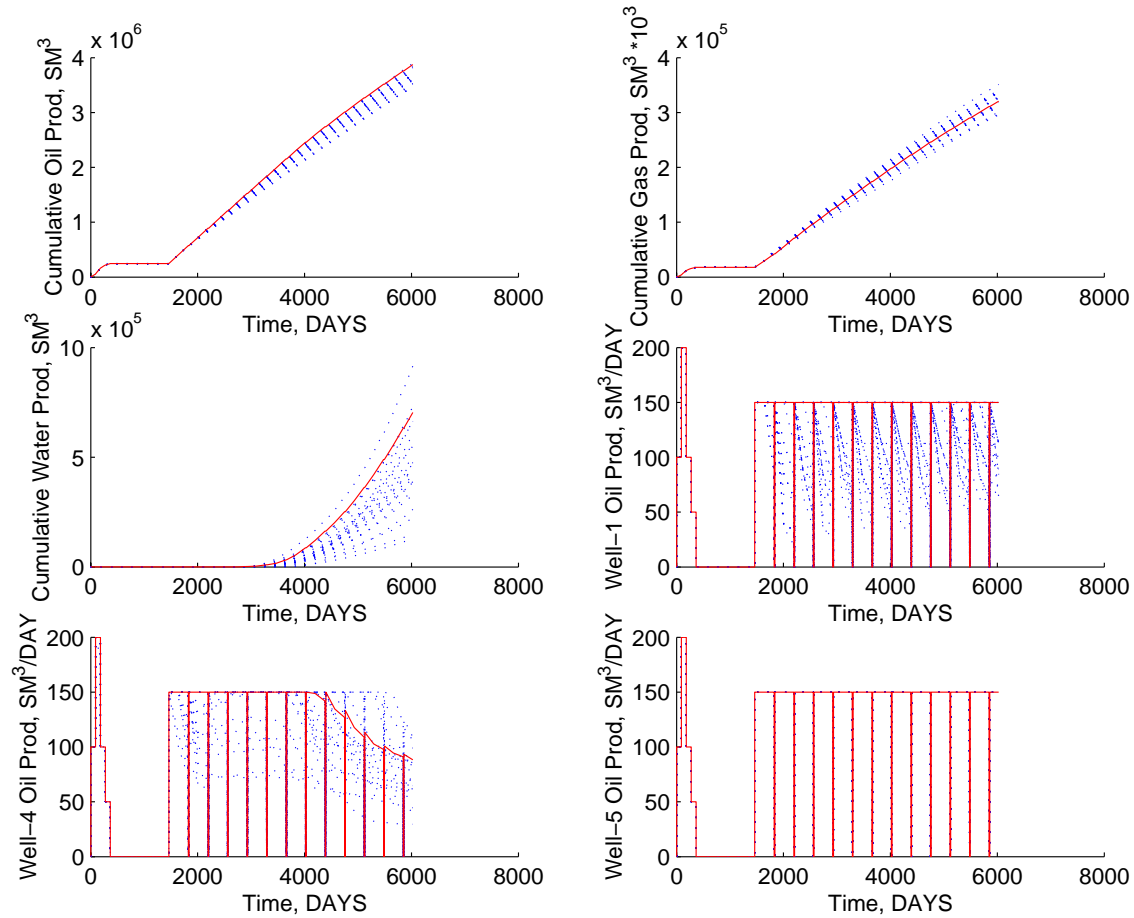


Figure 6.6: Match of history with the truth case of the Elf data set for the PUNQ-S3 model, dotted lines were generated by numerical simulation on the 23 permeability realizations and the solid line was generated by numerical simulation on the true permeability field.

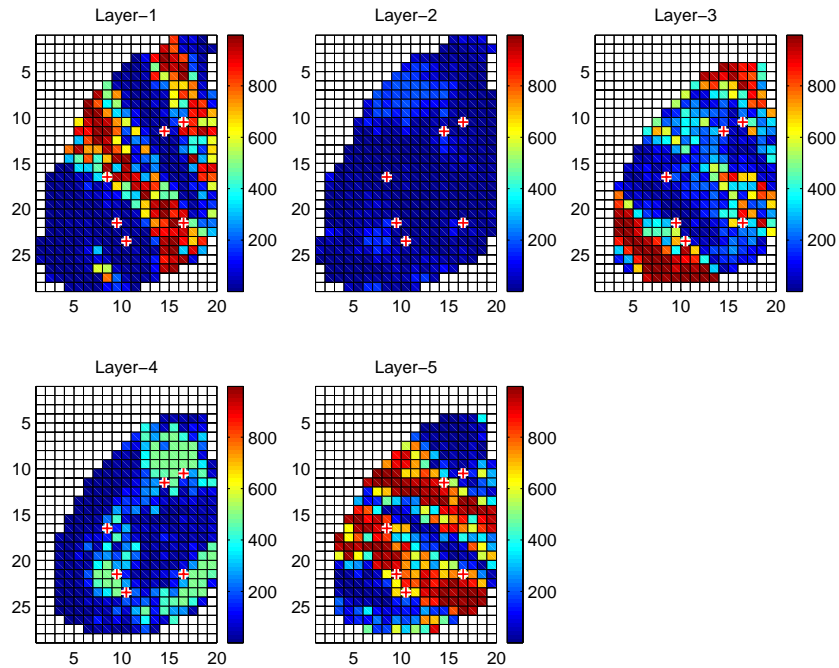


Figure 6.7: Truth case permeability fields for the PUNQ-S3 model.

The results of the comparative study are given in Fig. 6.8. The histograms of the cumulative parameters for the flow simulation results of the realizations and the truth case are given in Fig. 6.5. Although the true cumulative oil production is on the edge of the distribution, it is observed that the Elf data set is generally doing a good job in bracketing the uncertainty on cumulative properties despite the mismatch in individual well histories as shown in Fig. 6.6.

6.3.2 Exhaustive Runs

Exhaustive simulations were carried out on all the realizations and the true case. The generation of this exhaustive dataset was for research purposes and was carried out in order to enable controlled experimentation and sensitivity studies. The problem of locating a single constant-rate injector well was investigated. The dataset was constructed by placing the well at every active grid location on every realization, carrying out a simulation, and calculating the NPV of the particular well configuration

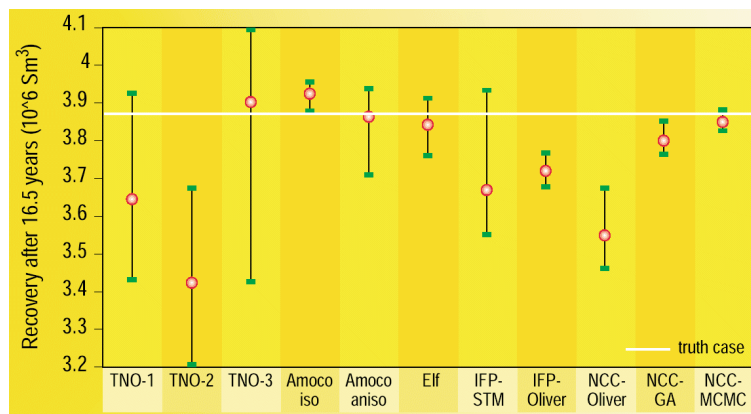


Figure 6.8: Results of the PUNQ-S3 comparative study (plot taken from the Netherlands Institute of Applied Geoscience, TNO-PUNQ online presentation, <http://www.nitg.tno.nl/punq>).

with the resulting flow history generated by the simulator. The same was done for the truth case as well. The economical parameters used for the NPV calculations are given in Table 6.1. The total number of simulations performed was 9912. The flow results of each simulation were converted into a single monetary value through NPV calculations hence NPV maps could be constructed for all the realizations as well as for the truth case. The NPV histogram of all the flow simulation runs on the realizations is given in Fig. 6.9. A wide range of NPVs was observed. The mean and standard deviation of NPVs over the 23 realizations is given in Figures 6.10 and 6.11 respectively.

6.3.3 Analysis of the Exhaustive Dataset

The mismatch and uncertainty in the flow history between the realizations and the truth case can be shown on the NPV maps as well. Fig. 6.12 demonstrates the extent of this mismatch. The true NPVs at the optimum locations of the realizations are sorted and plotted in Fig. 6.12. This shows, for example, that if a particular realization had been chosen and a well was drilled at the optimum location suggested by this realization, we could be 15% to 1% off the true optimum, thus the choice of the realization makes a big difference even though geostatistically all realizations

Table 6.1: Parameters for net present value calculations.

Parameter	Value
Discount rate, %	10.0
Oil price, \$/bbl	25
Gas price, \$/MSCF	3
Water handling cost, \$/bbl	1
Operation cost, \$/day	40,000
Well cost, \$/well	20,000,000
Capital expenditures, \$	500,000,000

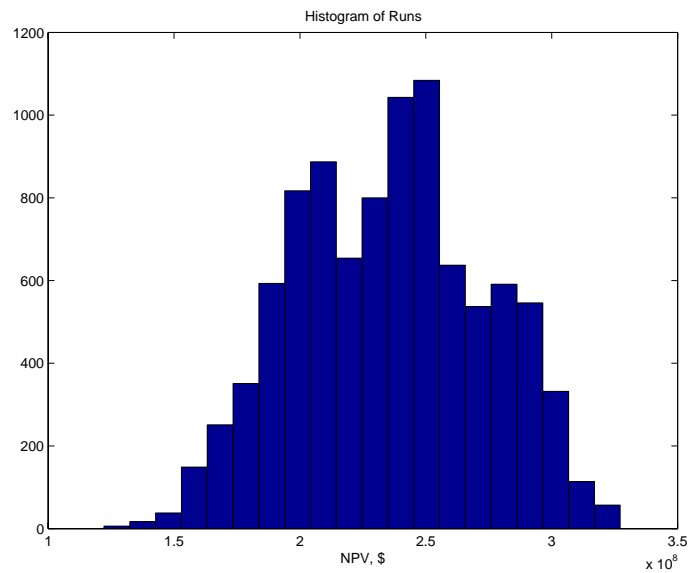


Figure 6.9: NPV histogram of all the flow simulation runs on the PUNQ-S3 realizations.

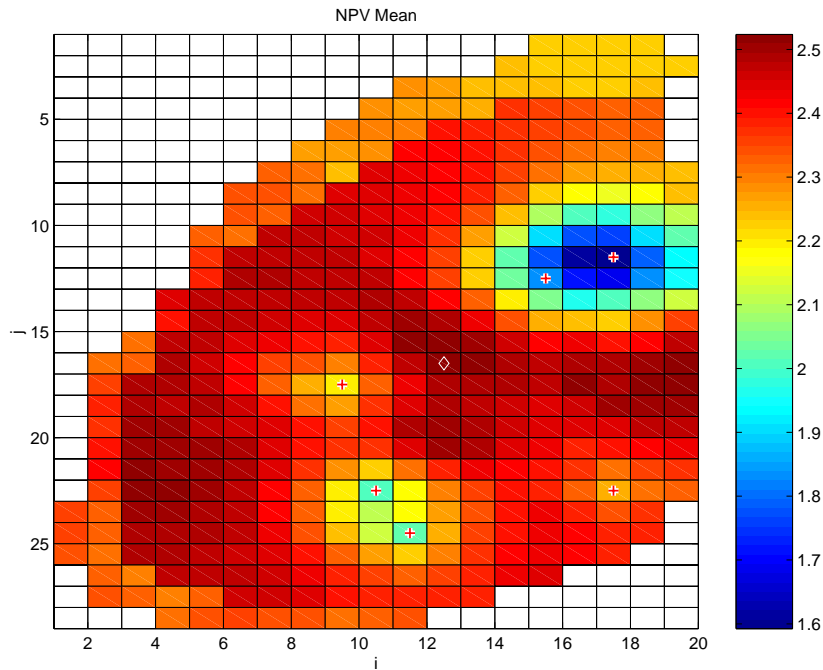


Figure 6.10: Mean of NPVs over the 23 realizations for the PUNQ-S3 model.

are equiprobable. The means and standard deviation of the NPVs at the optimum locations on each of the 23 realizations are also plotted in Fig. 6.12. The center and spread of the NPVs are very close for all the realization optima which suggests that these optima are the extreme NPVs for particular well locations. The essential message of Fig. 6.12 is that a decision based on a single realization may differ substantially from the true optimum.

Examining the optimum location histogram (Fig. 6.13), which is the two-dimensional histogram of the optimum well locations on the 23 realizations, we observe that the scatter of the optimum locations is unstructured and none of the realization optimal well locations coincide with the truth-case optimum. There is no clear definition of what the *optimum* well location is and how one would approach such a problem. However, keeping in mind that the truth case is never known, given the realizations one should not expect to discover the true optimum well location exactly and deterministically, but rather one would like to be able to evaluate the outcomes and risks associated with any given well configuration.

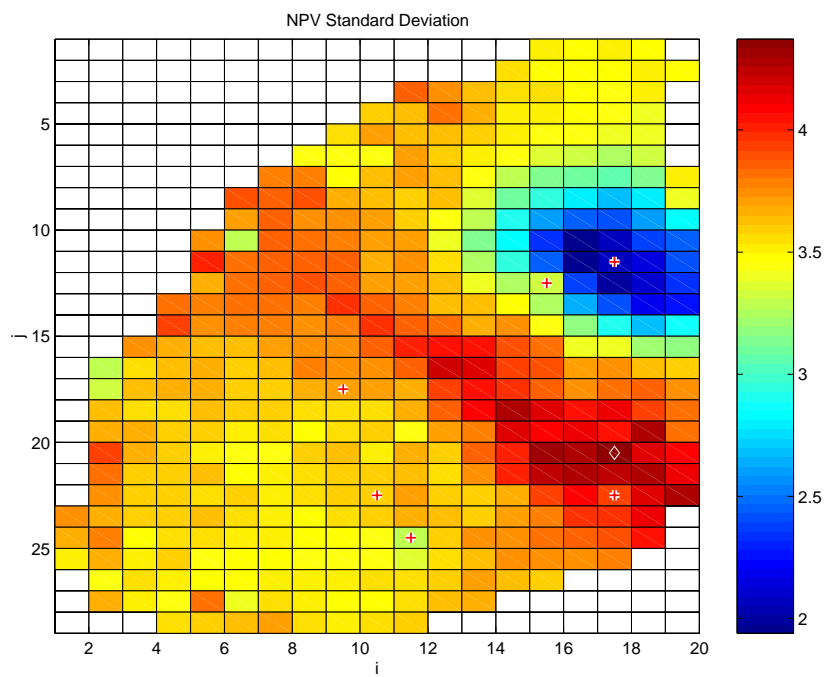


Figure 6.11: Standard deviation of NPVs over the 23 realizations for the PUNQ-S3 model.

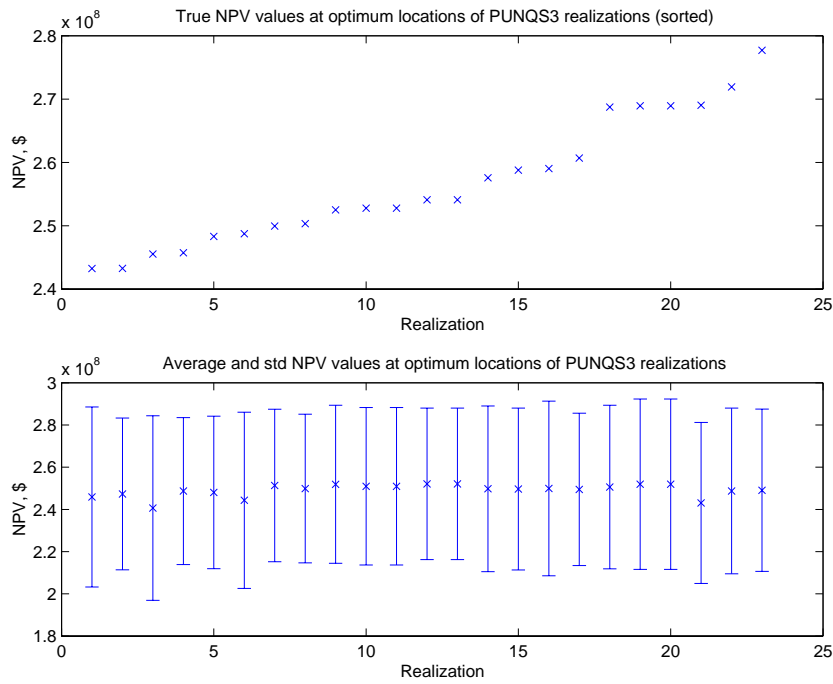


Figure 6.12: True NPVs (sorted) at the optimum locations of the realizations and the spreads of these locations for the PUNQ-S3 model.

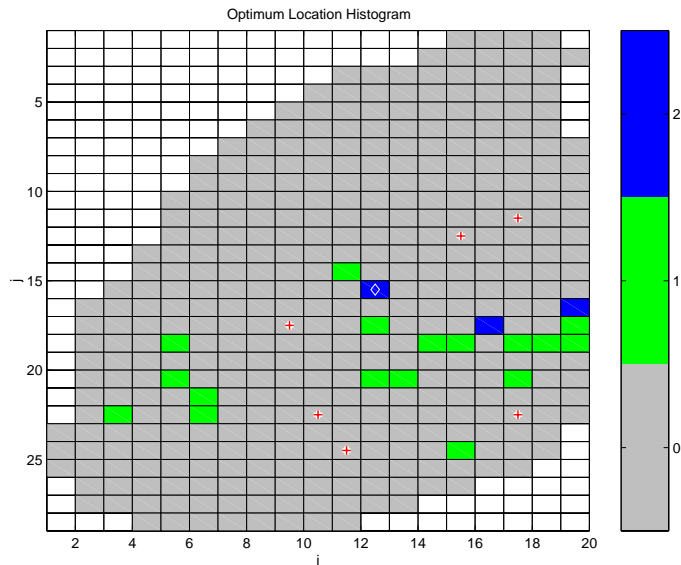


Figure 6.13: Two-dimensional histogram of the optimum well locations of the 23 realizations for the PUNQ-S3 model.

So how does one make a decision with uncertainty? There is no defined global optimum solution to the problem in the case of uncertainty. The optimum depends on the amount of risk the decision maker is willing to take, some play it safe, while others like to gamble and are willing to take some risk for the possibility of having a better outcome. As explained earlier, the utility framework addresses this problem.

6.3.4 Application of the Utility Framework

The utility framework was applied to the PUNQ-S3 problem. The exponential utility function given in Equation 6.2, with $a = 1$ and $b = -1$, was used. The effect of the risk aversion coefficient (Equation 6.3) was investigated. The risk aversion coefficient was varied from the extreme risk-prone value of -10 to the extreme risk-averse value of 10 with unit increments. It should be noted that Equation 6.2 is a scalar for $r = 0$, but we assumed that $r = 0$ corresponds to the risk-indifferent case where the utility function is linear. Normal NPV distributions were assumed at each well location. The NPV distributions were obtained by fitting a normal distribution on the NPV histogram at the particular location (Fig. 6.14). The NPV histograms for each well location are constructed from 23 samples (realizations) and they do not pose indication of skewness. The utility surfaces were generated for each case through the procedure illustrated in Figures 6.1 and 6.2. The gradual transformation of the decision maker's attitude from extreme risk-prone to extreme risk-averse is plotted in Fig. 6.15. The NPV mean and standard deviation maps are plotted as three-dimensional surfaces in Fig. 6.16. Examining Figures 6.15 and 6.16 simultaneously we observe that the risk-prone decision maker does not mind the risk involved as long as the expected outcome is high whereas the risk-averse decision maker is willing to give up high incomes for the sake of smaller risk. The point with maximum utility on the surface of the extreme risk-prone decision maker ($r = -10$) corresponds to a high NPV and also high standard deviation whereas the maximum point for the extreme risk-averse decision maker ($r = 10$) has very low standard deviation and also low expected NPV. Fig. 6.17 shows the transformation from risk-averse to risk-prone as a function of the risk aversion coefficient. Fig. 6.17 is a step function since the decision variables,

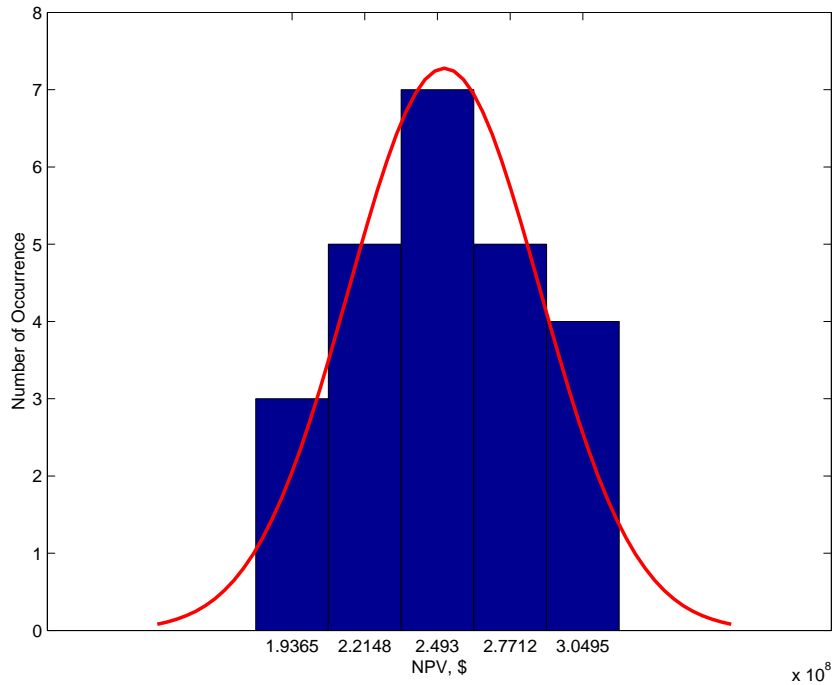


Figure 6.14: The normal distribution of NPVs obtained by fitting to the histogram for a particular well location for the PUNQ-S3 field.

(i, j) coordinates on the numerical model, are discrete. It should be noted that the tails of the NPV distributions of the risk-prone decision maker reach beyond those of the more risk-averse decision maker resulting in higher probabilities of having lesser NPVs for the risk-prone decision maker.

6.3.5 Application of Random Function Formulation

The random function formulation illustrated in Fig. 6.4 was established for the PUNQ-S3 problem. The objective was to maximize expected NPV which corresponds to a linear utility function if considered within the utility framework. The expected NPV was constructed iteratively as the GA sampled different points in the search space. The GA was allowed 250, 500 and 2500 simulations and ten runs were made for each case. The percent differences from the actual maximum expected NPV (Fig. 6.16) and the correlations between the actual means and the GA established means

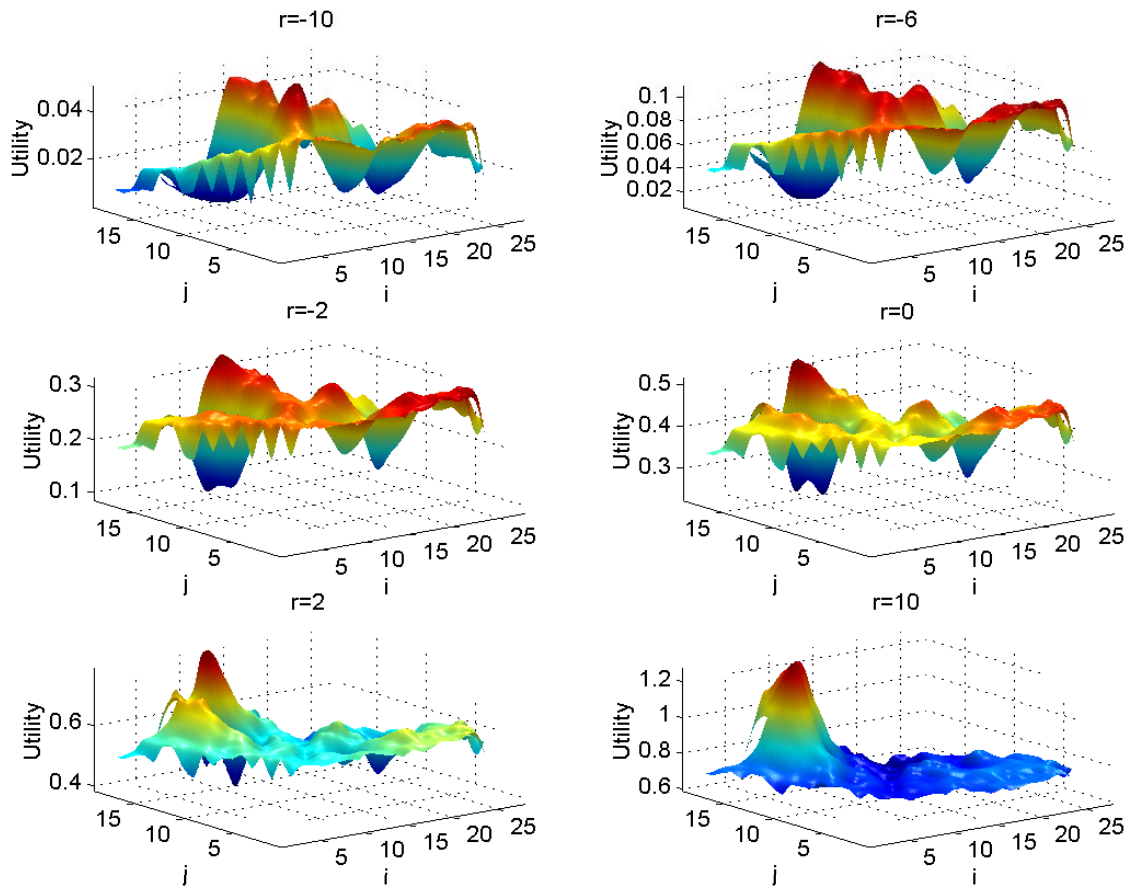


Figure 6.15: The gradual transformation of the decision makers attitude from extreme risk averse to extreme risk prone for the PUNQ-S3 problem.

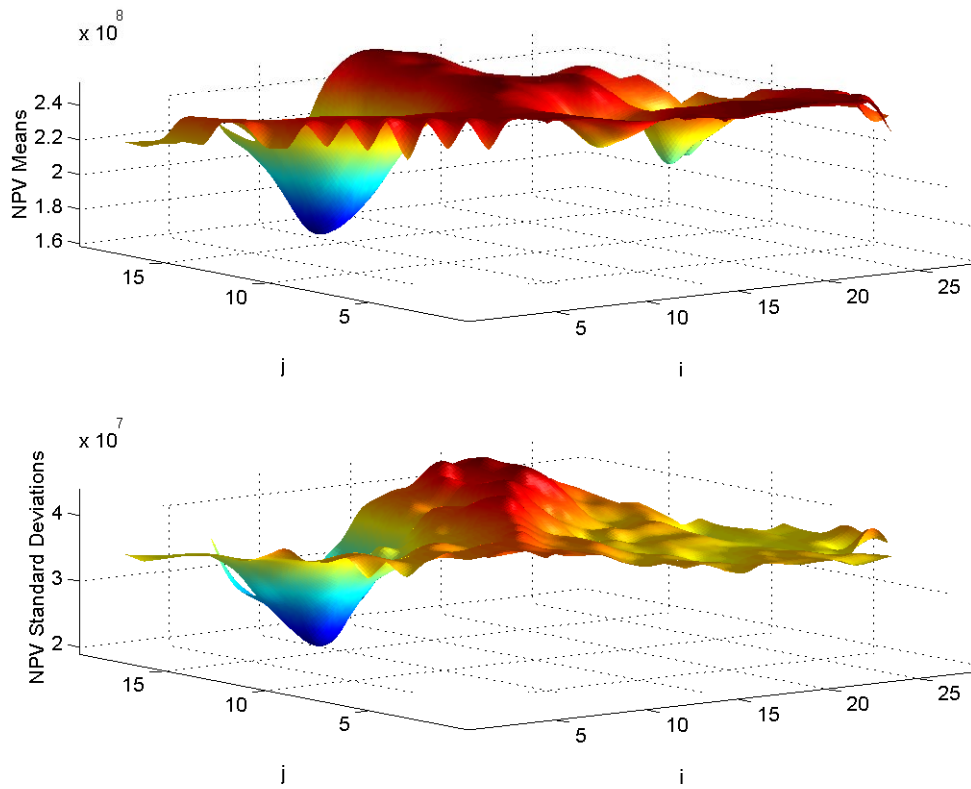


Figure 6.16: The three-dimensional surfaces of the mean and standard deviations of the NPV maps of the 23 realizations for the PUNQ-S3 model.

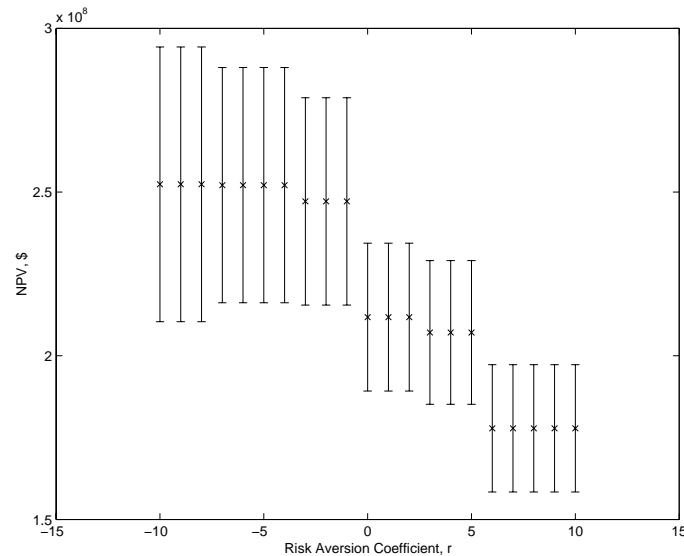


Figure 6.17: Standard deviations (error bars) and expected NPVs (crosses) of optimal decisions as a function of the risk aversion coefficient, r .

are given in Table 6.2. It was observed that for this specific problem the number of simulations allowed for the optimization does not strongly affect the accuracy of the results. This might be due to the plateau-like surface around the optimum (Fig. 6.16). The GA in this case pointed out this area around the optimum but rarely was able to locate the global optimum given the limited number of simulations (Table 6.2).

The number of times the GA visits well locations is shown as a two-dimensional histogram given in Fig. 6.18. Examining Fig. 6.18 and the mean NPV surface (Fig. 6.10) it is observed that the GA samples the well locations where the mean NPV is higher more than locations with lower mean NPV, thus spends less computational time on poorer areas of the search space. This can also be observed by plotting the NPVs of the solutions and the number of times the GA visited these solutions (Fig. 6.19).

Table 6.2: The percent differences from the actual mean NPV and the correlations between the GA sampled and actual means for the three cases where the GA was allowed 250, 500 and 1000 simulations.

Run	250 Simulations		500 Simulations		1000 Simulations	
	Off, %	Corr.Coeff.	Off, %	Corr.Coeff.	Off, %	Corr.Coeff.
1	2.5675	0.3815	0.0000	0.3154	1.9615	0.6490
2	0.1842	0.4377	1.3775	0.5486	0.5583	0.7532
3	1.7428	0.4735	1.3032	0.5350	2.2383	0.6692
4	3.9201	0.4440	3.1891	0.4278	2.1924	0.6812
5	1.4379	0.4915	1.3775	0.4861	0.4275	0.6583
6	1.3775	0.5325	1.7428	0.4945	1.2690	0.6691
7	0.1165	0.4177	1.1411	0.4542	1.9615	0.5979
8	1.2762	0.2354	2.4685	0.5207	0.9611	0.6680
9	1.2690	0.5444	0.8612	0.5008	0.9611	0.6771
10	0.9611	0.2844	3.1891	0.4766	1.3775	0.6828
Average	1.4853	0.4243	1.6650	0.4760	1.3908	0.6706

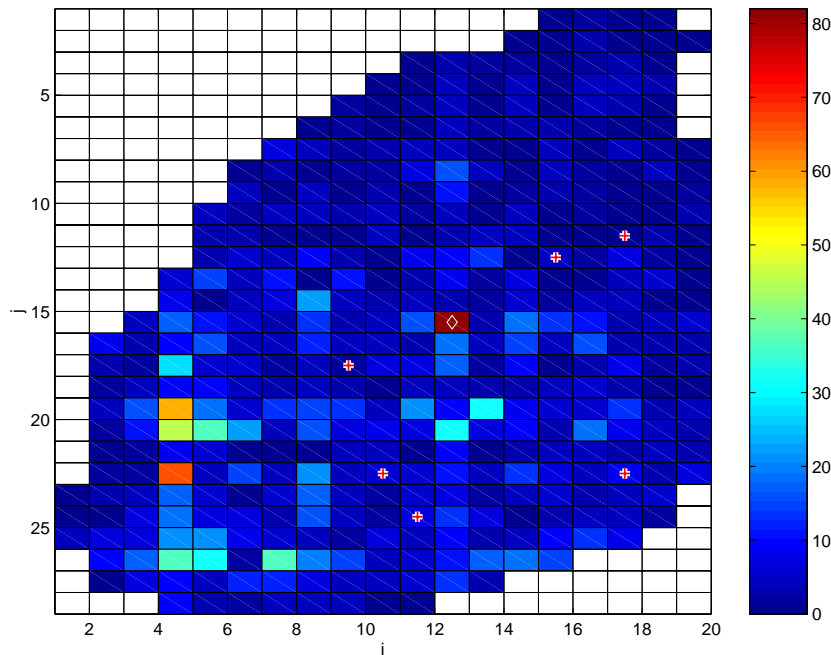


Figure 6.18: Histogram for the number of times the GA visited well locations for the random function formulation of the PUNQ-S3 well placement problem.

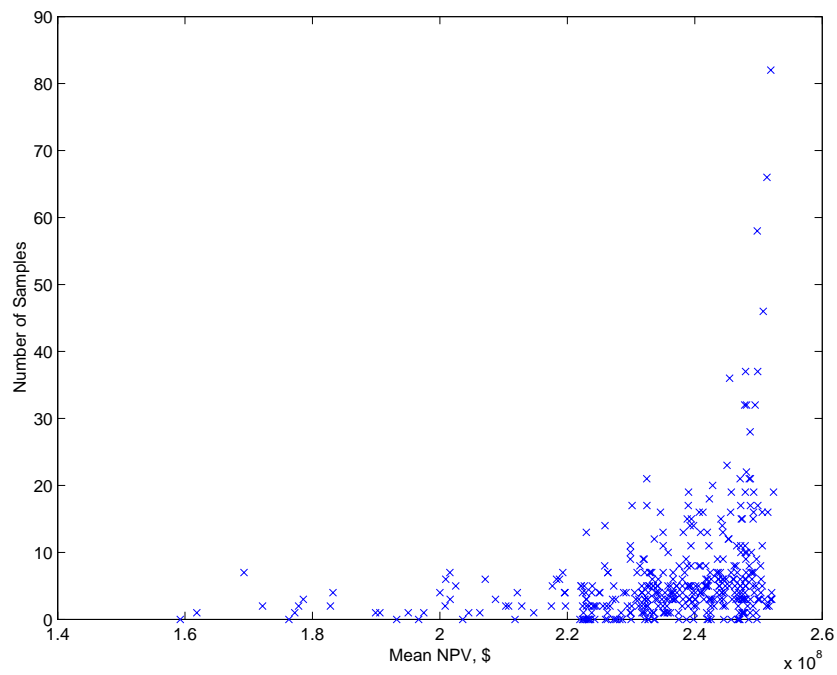


Figure 6.19: The NPVs of the solutions visited by the GA and the number of revisits for the random function formulation of the PUNQ-S3 well placement problem.

6.4 Discussion

The usefulness of the utility framework comes from the fact that it allows a clear definition of the otherwise arbitrary notion of an optimum solution to an uncertain problem. Decision makers who decide not to use the utility framework must define their objectives clearly since the *optimum* is not clear in case of *uncertainty*. One might choose to base decisions on expected value of monetary outcomes and carry out optimization. However the action to take is not clear if the maximum expected outcome turns out to have a risk higher than the decision maker is willing to take. One could penalize risky solutions and redo the optimization, however the extent of these penalties are unclear. Essentially the utility framework enables the decision maker to balance the risks and outcomes according to his/her specific risk attitude.

The utility framework has been applied extensively in many real world cases. However the application of the utility framework has not been widespread in the petroleum industry (Simpson *et al.*, 2000). One reason for this may be the difficulty in defining a utility function to represent the risk attitudes of a company. Another reason could be that the engineering and management fields do not have strong links. These problems can be overcome by some experimentation with the decision analysis methodology and a closer look at the theory behind the utility framework. The framework constructed here for the well placement problem can be established for other problems as well. The utility framework gives decision makers the ability to quantify their strategies when making an investment, thus should be used. If the risk attitudes are not taken into consideration, the problem would be difficult to construct and the decisions might not represent the company's intentions.

The HGA was able to determine the optimum well configuration in an average of 64 simulations for the case with the linear utility function. The utility framework, as constructed in this study, requires the knowledge of the distributions of the objective function for the well configurations proposed. This can be obtained by carrying out a simulation on each realization. If all realizations were used each time the GA proposed a well configuration the total number of simulations would be $64 \times 23 = 1472$, which may not be computationally feasible for some large numerical models. Alternatively,

the number of simulations required to construct the objective function distributions can be reduced by selecting a limited number of representative realizations based on some heuristics, such as injectivity. For instance if five realizations could be selected that roughly represent the NPV distributions for well placement, the total number of simulations required to determine the optimum well configuration would be $64 \times 5 = 320$.

Another approach to optimization under uncertainty, the random function formulation of the problem, provided approximate results that were around 1.5% off the true optimum on the average (Table 6.2) for the PUNQ-S3 model. However it should be noted that this may be due to the plateau nature of the optimized surface around the optima (Fig. 6.16), hence results might not be as satisfactory for problems with bumpier surfaces. Nevertheless, the random function approach may be useful in that it requires less computational time. Several hundreds of simulations was sufficient to get very close to the optimum for the PUNQ-S3 problem (Table 6.2).

Chapter 7

Sensitivity of Optimization Algorithm Parameters

The sensitivity of HGA performance to algorithm parameters were examined for the Pompano and PUNQ-S3 injector placement problems.

7.1 GA Performance Sensitivity

7.1.1 Rule of Thumb Determination of GA Parameters

The GA parameters used in this study were based on rule-of-thumb values from the literature. For binary alphabets Goldberg (1989) suggests an appropriate population size and De Jong (1975) suggests appropriate GA operator probabilities based on experimentation over problems of different nature:

Population size - same as total string length, $n = l$

Crossover probability - $p_c = 0.6$ works well for a wide range of problems

Mutation probability - inverse of population size, $p_m = 1/n$

For the Pompano and PUNQ-S3 problems integer data structures were used. For the Pompano problem a population size of 12 was used since the length of the equivalent binary string would have been 12. Since De Jong suggests a mutation probability

of 1/12 per bit the equivalent of this for integer data structures is 0.5 mutation probability. The same procedure applied to the PUNQ-S3 problem resulted in a mutation probability of 0.5 as well.

7.1.2 Crossover and Mutation Probabilities

Sensitivity of the GA performance to mutation and crossover probabilities was determined by varying these probabilities from 0.0 to 1.0 with 0.01 increments. The mean and variance of the necessary number of simulation runs consumed by the GA to locate the global optimum for the single injector placement problem for the Pompano and PUNQ-S3 fields was determined by carrying out 100 GA optimizations for each mutation and crossover probability pairs.

The dependency of the GA to mutation and crossover probability is shown in Fig. 7.1 for Pompano and in Fig. 7.2 for PUNQ-S3 which shows the average number of simulations that was required to locate the global optimum. The total number of GA optimizations made to construct the sensitivity plots (Fig. 7.1 and 7.2) were:

$$(101 p_m \text{ values}) \times (101 p_c \text{ values}) \times (100 \text{ GA runs per } p_c - p_m \text{ pair}) = (1,020,100 \text{ optimization runs per plot})$$

where p_m is the mutation probability and p_c is the crossover probability.

The variance of the number of simulations that the GA consumed to locate the global optimum is given in Fig. 7.3 for Pompano and in Fig. 7.2 for PUNQ-S3. In each of the 1,020,100 optimization runs carried out, the GA was allowed to consume at most 1000 simulations, otherwise the run was terminated after 1000 generations. Such an extensive sensitivity analysis was made possible by the construction of the exhaustive data sets for the Pompano and PUNQ-S3 single injector problems.

The result acquired from Fig. 7.1 and Fig. 7.3 is rather surprising at first glance: *GA optimization performance does not depend on crossover probability for the single injector placement problem for the Pompano and PUNQ-S3 fields.* However this result is reasonable when one evaluates the single injector placement problem more carefully. Optimized parameters are the i and j location of the proposed injector and there is no insight suggesting that a particular i or j value contributes to the fitness of an

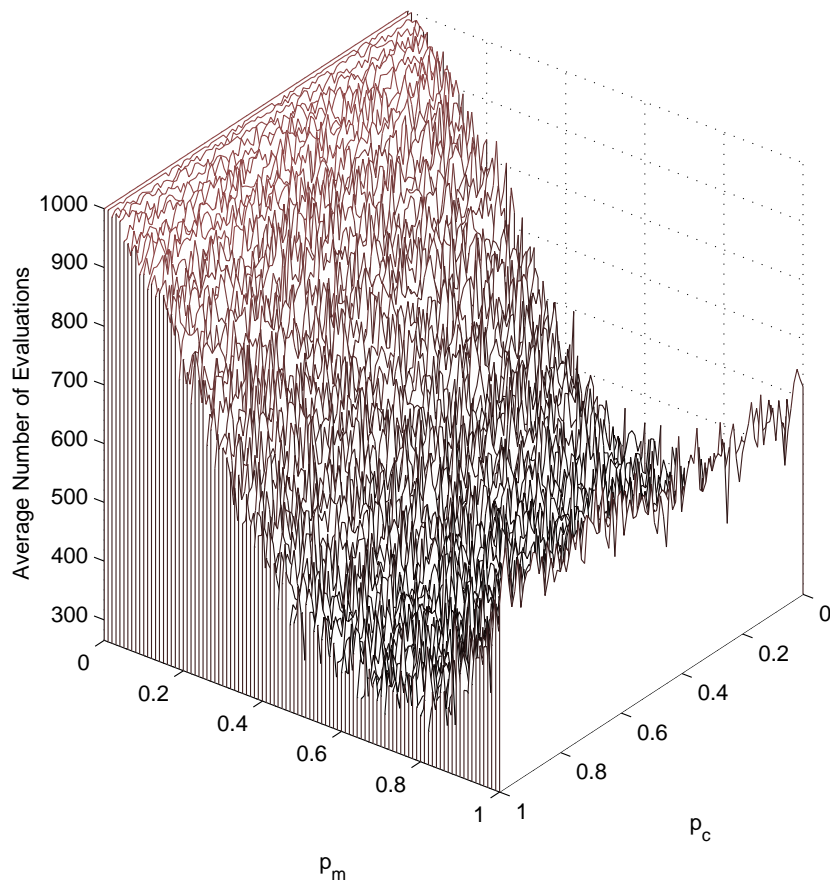


Figure 7.1: The average number of simulations that the GA consumed to locate the global optimum for different values of mutation probability (p_m) and crossover probability (p_c) for the Pompano problem.

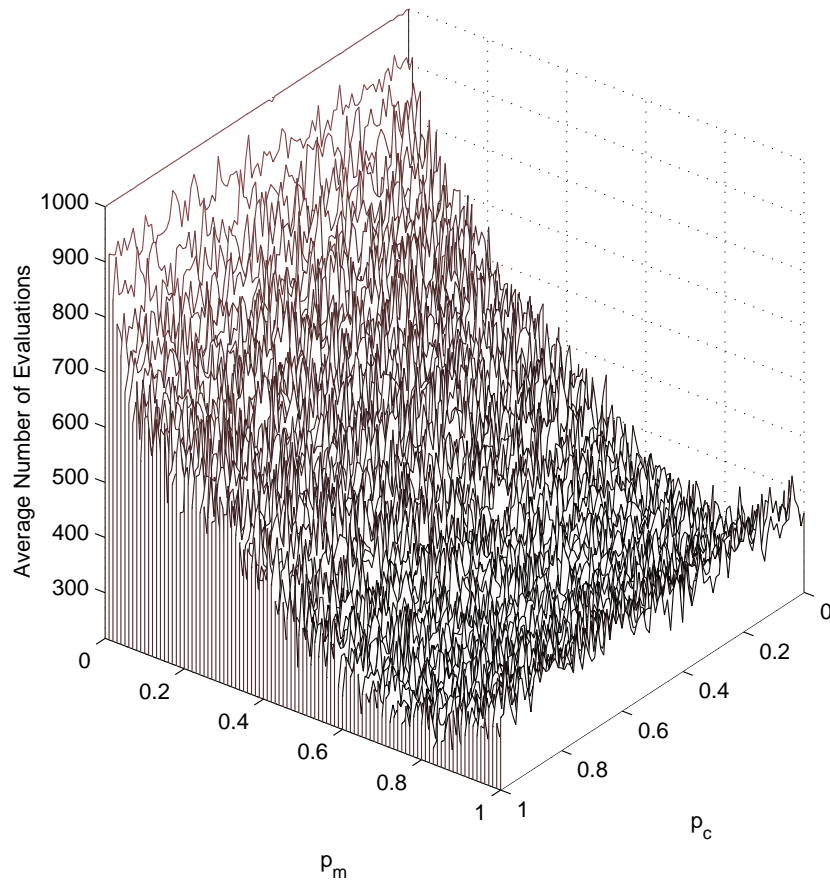


Figure 7.2: The average number of simulations that the GA consumed to locate the global optimum for different values of mutation probability (p_m) and crossover probability (p_c) for the PUNQ-S3 problem.

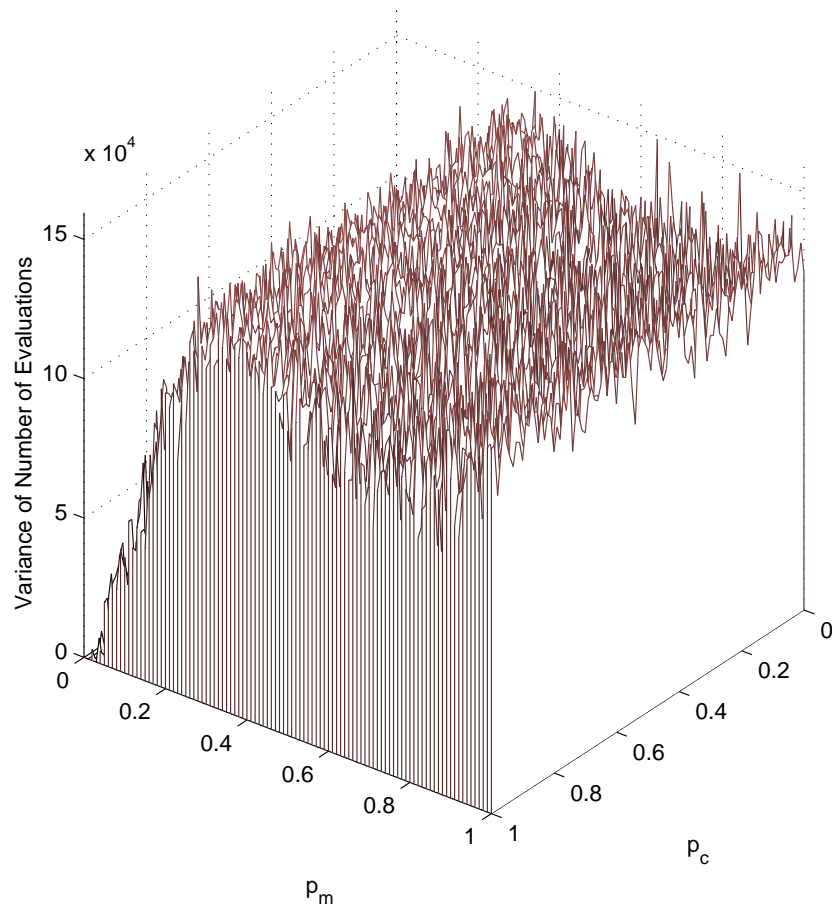


Figure 7.3: The variance of the number of simulations that the GA consumed to locate the global optimum for different values of mutation probability (p_m) and crossover probability (p_c) for the Pompano problem.

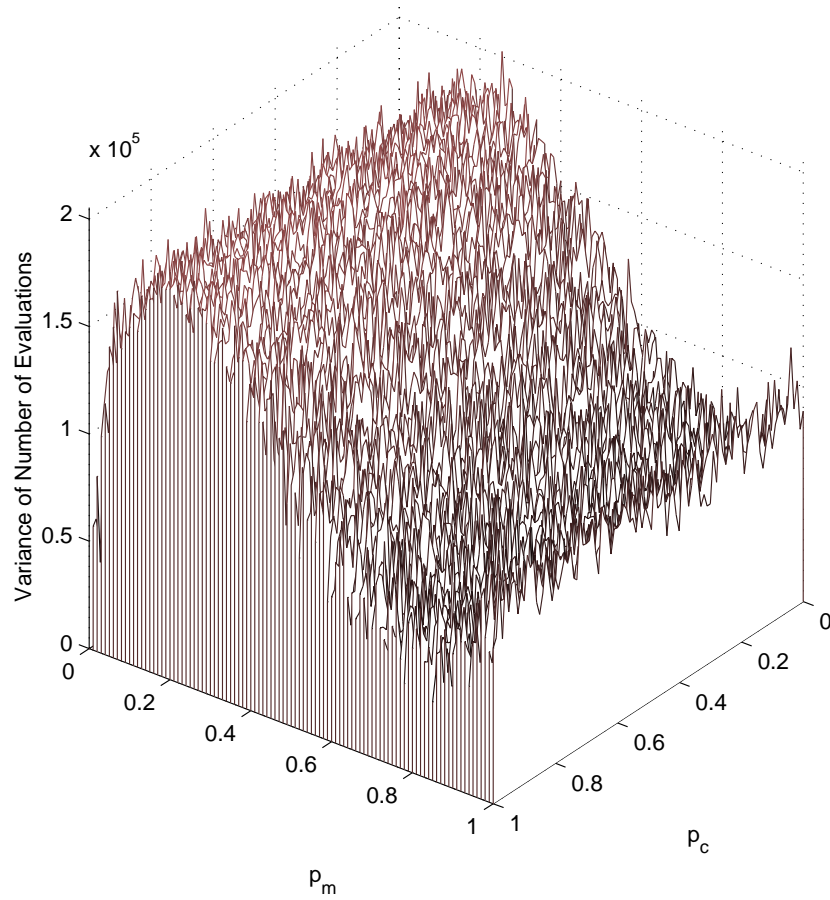


Figure 7.4: The variance of the number of simulations that the GA consumed to locate the global optimum for different values of mutation probability (p_m) and crossover probability (p_c) for the PUNQ-S3 problem.

individual. In other words the combination of i and j is what makes an individual fit, i or j alone is not correlated with the fitness of a given well configuration. This can also be observed by looking at the search space (Fig. 4.3). Crossover would make a difference if there was a clear trend on the search space. For instance if the NPV values increased with increasing i values, crossover would make a difference even for the one-well case.

Also the mutation probability obtained with the method suggested by De Jong ($p_m = 0.5$) is observed to be a reasonable value.

Although we cannot prove this empirically due to computational limitations, crossover will play an important role when more than one well is considered and rate is optimized together with location. This is simply due to the fact that the fitness of a well configuration is correlated with the locations and rates of individual wells and fitter individuals might be obtained by combining individuals with fit (i, j) pairs.

7.1.3 Selection Type

Effect of selection type on algorithm performance was investigated for the Pompano field, single injector placement problem. Three types of selection were examined:

- *Raw Fitness* - Raw fitness of the individual is used for selection.
- *Windowing* - The fitness of the least fit individual is subtracted from all the individuals in the generation.
- *Rank-Based* - The individuals are ranked in the order of increasing fitness and $(n - \text{rank})$ is assigned as the fitness, n being the population size. Thus, the best individual in the generation has a fitness of $(n - 1)$ and the worst individual in the generation has a fitness of zero.

In order to assess the effects of type of selection, 100 runs were carried out for each of the three types of selection. The GA and the polytope method were utilized. The results are given in Table 7.1. It is observed that, although the performances are very similar, rank-based selection performs the best and selection based on raw

Table 7.1: Effect of selection type on algorithm performance; the mean and standard deviation of the number of simulations required to locate the optimum well location for 100 optimization runs.

Selection Type	Mean	Standard Deviation
Raw Fitness	152	97
Windowing	147	97
Rank-Based	141	94

fitness performs the poorest. This is due to scaling issues associated with using raw fitness values. Windowing partially solves the scaling problem and rank-based selection eliminates it.

7.2 Polytope Strategies

The effect of polytope strategies, explained in Section 2.4.1, was investigated for the Pompano field, single injector placement problem. The GA and the polytope method were used and 100 runs were carried out for each polytope strategy. The results are given in Table 7.2. It is observed that the choice of the polytope strategy does not make much of a difference for this problem. The polytopes constructed with the best of the generation or the best of the database are expected to have more of a global hill-climbing behavior and also a mutation effect since the points in the polytope are not necessarily close to each other in the search space thus may not belong to different peaks in the search space. On the other hand the polytopes constructed with the closest to the best points in the database are expected to have a more deterministic behavior and will aid in local refinement of the search.

Table 7.2: Effect of polytope strategy on algorithm performance; the mean and standard deviation of the number of simulations required to locate the optimum well location for 100 optimization runs.

Selection Type	Mean	Standard Deviation
Best in Generation	141	76
Best in Database	144	92
Closest to Best in Database	141	94

7.3 Discussion

The results given here are for a limited number of problems. In order to generalize, a wide range of problems with different characteristics have to be considered. Nevertheless the results may be valuable as a starting point for further studies of similar sort.

Chapter 8

Concluding Remarks

8.1 Conclusions

A hybrid method of GA, polytope and proxy methods was developed. The component methods in the hybrid compensated for the weaknesses of each other, resulting in an algorithm that outperformed the individual methods.

The ordinary kriging algorithm was used as a proxy to numerical simulation. Kriging has the favorable properties of being data-exact and suitable for multiple dimensions. In addition, the kriging estimation variance enabled error analysis and helped ensure robustness.

The HGA was applied to the injector placement problem for the Pompano field in the Gulf of Mexico. The HGA proposed intuitive solutions. Controlled experimentation for the single well, constant rate case showed that the HGA was able to reduce the number of numerical simulations required. Waterflooding with three injectors was the most profitable option with an incremental NPV of \$154 million compared to the no-injection case.

The deployment schedule for a real-world field was optimized. The numerical model had approximately half a million cells and optimization posed a significant challenge. Optimization was made in parallel on four processors. The problem was formulated as a TSP and the branching order of a task tree was used as the data structure. Optimized well deployment schedule improved the NPV from the field by

15.23% compared to the base case where no new wells were deployed. Also the best solution resulted in a 0.70% increase in NPV compared to the case where all the new wells were deployed at the beginning of the forecast.

Realizing the uncertainty in numerical simulation forecasts, the utility framework was utilized to assess this uncertainty. The utility framework provides the tools to quantify risk attitudes through utility functions. The proposed utility framework for the well placement problem was applied to the PUNQ-S3 problem. It was observed through controlled experiments that the utility framework gave results that were consistent with the intent implied through the utility functions. The HGA performed significantly better than its individual components for the PUNQ-S3 problem as well.

An alternative approach to the assessment of uncertainty, the random function formulation of the problem, was also investigated. It was observed that this approach, although not very accurate, gave satisfactory results with smaller computational effort.

8.2 Suggestions for Future Work

Several topics are suggested as potential areas of future research.

Physical Proxies - In this study kriging and neural networks were evaluated as proxies. Both of these interpolation algorithms are purely mathematical and have no consideration of the physics underlying the process. A proxy that can be calibrated and one that honors physics would be very useful for the purpose of optimization of petroleum reservoirs. Perhaps a simpler version of streamline simulation might be appropriate. Possibly a mathematical structure of nodes and connections with nodes being the wells and the boundaries and the connections representing the interactions among wells and wells with boundaries. Possibly Buckley-Leverett equations may be employed to represent these interactions.

Further Applications - The algorithm may be utilized for the optimization of various sorts of reservoir engineering problems. Optimization on other real world

oil fields will also enhance our understanding of reservoir optimization.

Algorithm Performance Sensitivity - The sensitivity of the optimization performance to a limited number of algorithm parameters were examined in this study. Sensitivity of other parameters remains to be investigated. Also it is not possible to generalize the performance results based on only a few problems. Hence similar studies for other types of optimization problems should be carried out.

Neural Networks - When one works with optimization of discrete design decisions, the kriging proxy is not sufficient. NNs have the ability to represent such problems. However when NNs are used as proxies within the HGA framework as described in this study, the issues stated in Section 4.2 have to be addressed.

Appendix A

Algorithm Parallelization

A.1 Value of Parallelization

The HGA has been parallelized in this study. The basic setup of GAs enable straightforward parallelization since the individuals at any given generation can be evaluated independently from each other. Moreover the computational time scales back linearly with the increasing number of computers in the parallelization process. Assuming that the evaluation of solutions takes an equal amount of time on all the computers and that the overhead time is negligible, then the following inequality holds:

$$T_{N_c} \leq T_1 \frac{\min \left\{ z : z \geq \frac{n_i}{N_c}, z \in Z \right\}}{n_i} = T_1 \frac{\text{ceiling} \left\{ \frac{n_i}{N_c} \right\}}{n_i} \quad (\text{A.1})$$

where N_c is the number of machines available for the parallel run, T_i is the time it takes to complete the optimization run with i machines, and n_i is the number of individuals that is to be evaluated at each generation.

For instance for a run with three machines and assuming 10 evaluations are made at each generation, Eq. A.1 states that the run will be at least 10/4 times faster or for a run with five machines for which 20 evaluations are made for each generation, the run will be five (20/4) times faster.

With differing evaluation time among computers and overhead time taken into account, the scaling may be less favorable than that shown in Eq. A.1. Nevertheless

if a machine is not extremely slow compared to others, scaling close to Eq. A.1 will be achieved.

Notice that the value of parallelization is dependent on the number of evaluations that are carried out at each generation (Eq. A.1) which decreases the advantages of parallelization. For instance in the extreme case when only one individual is to be evaluated at each generation parallelization will have no benefits since all the processors would need to wait for the evaluation of this single individual. This might become a problem at the later stages of a GA run when the algorithm visits a smaller number of new points at each generation due to convergence.

Fewer new points will be visited at each generation especially when the population size is relatively small. Hence an approach would be to use a larger population size than rule-of-thumb values (Chapter 7). An alternative way to increase the number of newly visited points in a generation would be to increase the mutation probability. One could also make use of the wasted CPU power by evaluating an unvisited point with the processors that wait. This point may be generated randomly or could be located around the best solution obtained so far in the run for the purpose of local search.

A.2 Parallel Setup of Optimization

A master program organizes processes. This master program essentially takes each unprocessed solution *across the river*, as illustrated in Fig. A.1. At most one solution is assigned to a processor at any time. The HGA routine updates a pool of solutions without making the actual evaluation each time an evaluation is to be made within the HGA iteration. This pool is sent to the process organizer at the end of each iteration. The process organizer then assigns the solutions in this pool to machines for evaluation and returns the results back to HGA when all solutions have been evaluated.

Communication between the master process organizer and individual processes is carried out through network sockets (internet). Thus all the computers do not have to be connected through a local network, they just have to be able to access the

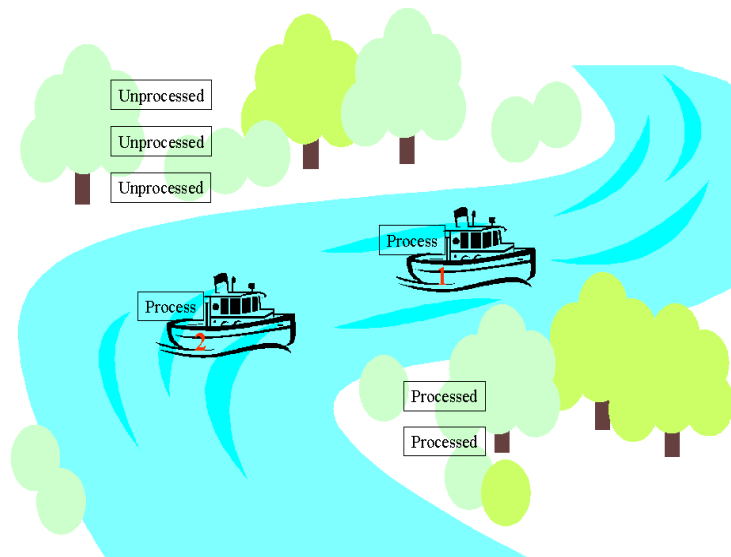


Figure A.1: Cartoon analogy of how the process organizer works; two machines helping processes cross the river.

internet and not be behind a firewall. The parallel setup is illustrated in Fig. A.2. Fig. A.2 illustrates a single remote server, however there would be more than one remote servers for a parallel run.

The process organizer and the communicating servers at remote computers were written in the Java language. The HGA and process launcher were written in C++.

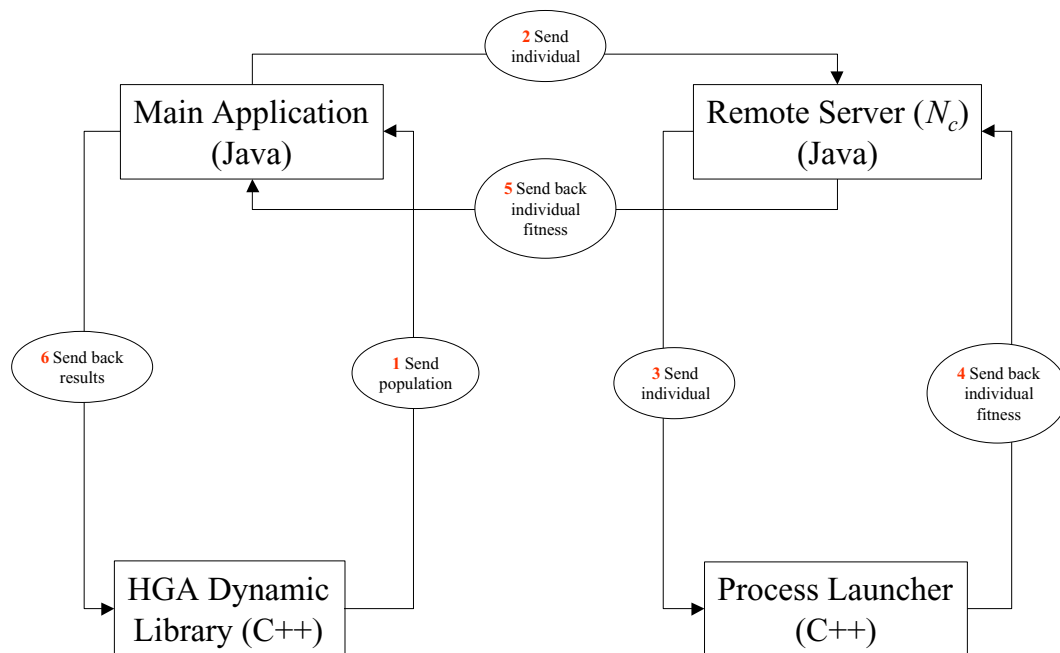


Figure A.2: Parallelization setup.

Appendix B

HGA Implementation

B.1 The Program

The algorithms developed and presented in this study were implemented in C++ using object-oriented design. Some of the classes making up the HGA can also be used as standalone objects. The class design is given in Fig. B.1. The arrows in Fig. B.1 indicate inheritance.

The following objects are required to make an HGA run:

- A HybridGeneticAlgorithmParameters object encapsulating the HGA parameters.
- A method implementing the evaluation function. This has to be in the following form:

```
double EvaluationFunction(vector<double>& solution, vector<bool>& flags)
```

The flags[0] should be set to *true* if evaluation was made and to *false* otherwise.

- A function implementing the processing of a solution before evaluation. This function has to be in the following form:

```
void ProcessPoint(vector<double>& solution)
```

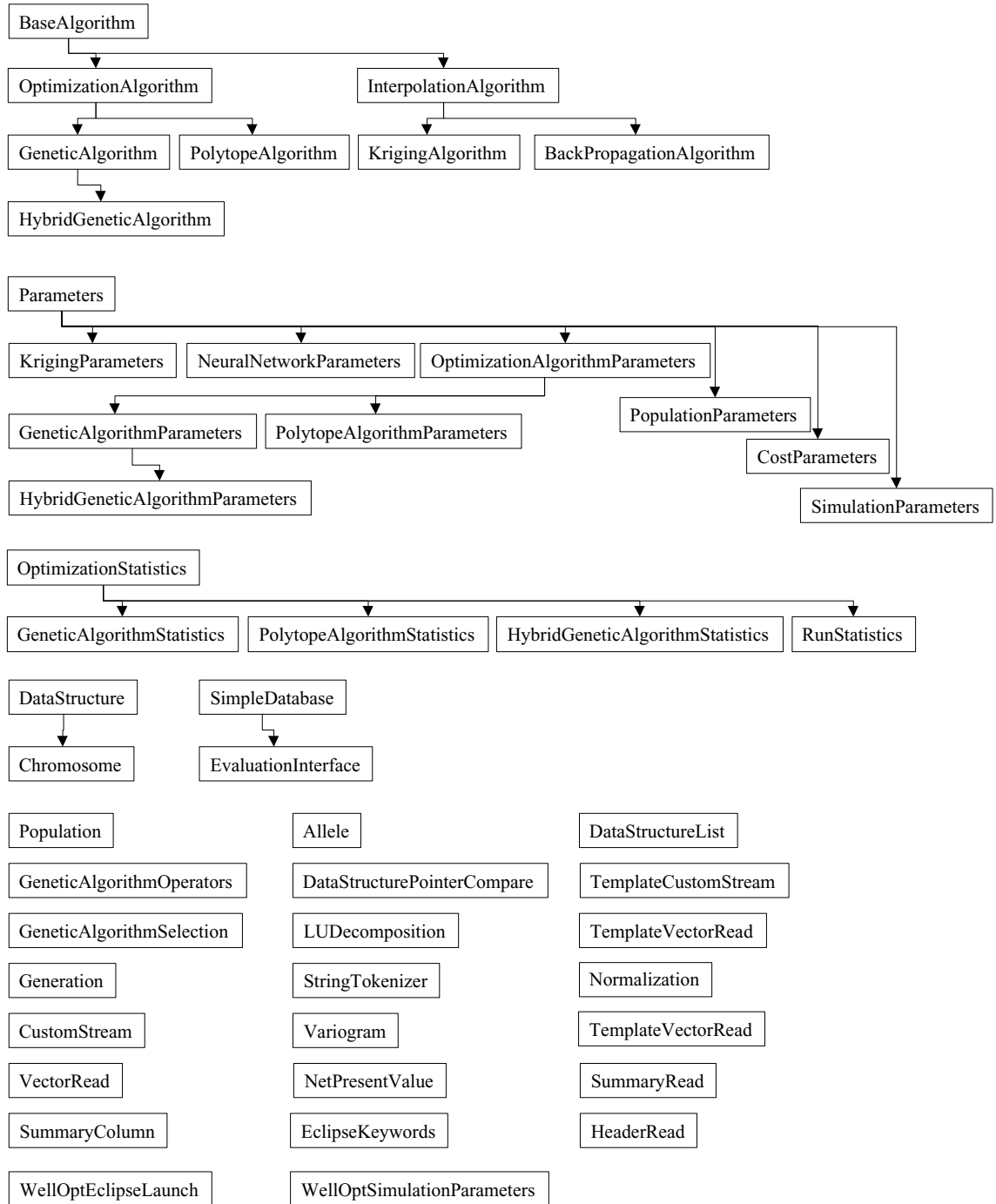


Figure B.1: C++ classes making up the HGA; arrows indicate inheritance.

It is straightforward to set up an optimization run with the HGA. The HGA is completely indifferent to what is being done within the problem-specific evaluation method. The calculations within the function evaluation method can be as simple as simple mathematical methods, or as complex as calling a numerical simulator and calculating economics from generated flow history.

The following piece of code sets up a nonparallel HGA run whose objective is to maximize the sum of the parameters:

```
#include <iostream>
#include <vector>
#include "hybridgeneticalgorithmlib.h"

double EvaluationFunction(vector<double>& solution, vector<bool>& flags)
{
    double sum = 0;
    for (int i=0; i<solution.size(); i++) sum += solution[i];
    flags[0] = true;
    return sum;
}

// This function does nothing in this case
void ProcessPoint(vector<double>& solution) {}

int main()
{
    HybridGeneticAlgorithmParameters parameters;

    // Parameters are read in from a file called "parameters.dat"
    ifstream in("parameters.dat");
    in >> parameters;

    HybridGeneticAlgorithm hga(parameters,EvaluationFunction,ProcessPoint);
    hga.Run();

    return 0;
}
```

B.2 The Keywords

A `HybridGeneticAlgorithmParameters` object which encapsulates the optimization parameters is required to make an HGA run. Following is a sample data file containing the HGA keywords:

```

HYBRIDGENETICALGORITHM
{
    CLOSESTSCALINGFACTOR 9.000000e-001
    CONVERGENCE TOLERANCE 0.000000e+000
    CONVERGENCE VALUE 1511180000
    FITNESS SCALINGFACTOR 1.000000e+000
    NEIGHBORHOOD RANGE 3.000000e+000
    PCROSSOVER 5.000000e-001
    PMUTATION 2.000000e-001
    PMUTATION NEIGHBORHOOD 0.0
    CONVERGENCE CRITERIA 1
    MAX NUM EVALUATIONS 500
    MAX NUM ITERATIONS 100
    MAX TIME 1000000
    NUM POPULATION 1
    NUM RUN 10
    NUM TOURNAMENT 2
    RANDOM SEED 1975
    SELECTION TYPE 2
    DOKRIGING 0
    DOKRIGING ERROR ANALYSIS 0
    DOPOLYTOPE 0
    EVALUATE AT END 0
    OUTPUT FILES 0
    USE CLOSEST DATA 0
    USE DATABASE 1
    USE ELITISM 1
    KRIGING RECURSIVE FILENAME SAME-10
    LOAD DATABASE 1
    OLD DATABASE FILENAME database.dat
    RUN DESCRIPTION ResOpt Run
    RUN NAME ResOptRun
    KRIGING ALGORITHM
{
    VARIOGRAMA 1.750000e+000
    VARIOGRAMC 1.000000e+000
    NUM KRIG DATA 1000
    VARIOGRAM TYPE 3
    OUTPUT FILES 0
    RUN NAME ResOptKriging

```

```

}
    POLYTOPEALGORITHM
{
    CLOSESTSCALINGFACTOR 9.000000e-001
    CONTRACTIONCOEFFICIENT 5.000000e-001
    CONVERGENCE TOLERANCE 0.000000e+000
    CONVERGENCEVALUE 0.000000e+000
    EXPANSIONCOEFFICIENT 2.000000e+000
    REFLECTIONCOEFFICIENT 1.500000e+000
    CONVERGENCECRITERIA 0
    MAXIMUMNUMCONTRACTION 0
    MAXNUMEVALUATIONS 10
    MAXNUMITERATIONS 1
        MAXTIME 1000000
    EVALUATEATEND 0
    OUTPUTFILES 0
    USECLOSESTDATA 1
    USEDATABASE 1
    RUNDESCRIPTION ResOpt Polytope
        RUNNAME ResOptPolytope
    INITIALSOLUTION { 1.000000e+000 1.000000e+000 }
        MAXVALUE { 4.000000e+001 4.000000e+001 }
        MINVALUE { 1.000000e+000 1.000000e+000 }
}

    POLYTOPETYPE { 2 }
    POPULATION
{
    POPULATIONSIZE 10
    OUTPUTFILES 0
    UNIFORMINITIALIZATION 1
    POPULATIONNAME ResOptPop
        RUNNAME ResOptPopRun
        VALUENAME { i j }
        ALPHABETSIZE { 19 28 }
        MINVALUE { 1.000000e+000 1.000000e+000 }
        MAXVALUE { 4.000000e+001 4.000000e+001 }
        PRECISION { 1.000000e+000 1.000000e+000 }
}
}

```

In addition to these keywords CostParameters and SimulationParameters are also required for well placement optimization. The description for all the keywords is given in Tables B.1 to B.6.

Table B.1: Keywords for the HybridGeneticAlgorithmParameters class.

Keyword	Description
CLOSESTSCALINGFACTOR	(double) Power scaling of the invalid points
CONVERGENCECRITERIA	(int) Flag indicating the criteria for convergence, 0:MAXNUMITERATIONS, 1:CONVERGENCEVALUE
CONVERGENCECTOLERANCE	(double) Percent (of the CONVERGENCEVALUE) tolerance to test convergence
CONVERGENCEVALUE	(double) The minimum sought evaluation function value, if found an equal or better solution, the algorithm quits
DOKRIGING	(bool) If true, the kriging proxy is constructed after each generation
DOKRIGINGERRORANALYSIS	(bool) If true, the kriging error analysis is carried out after each generation
DOPOLYTOPE	(bool) If true, the polytope algorithm is carried out after each generation
EVALUATEATEND	(bool) Flag to wait until the end of the iteration before making the function calls. If a parallel run is to be made, this flag should be set to 1 and the user supplied evaluation method should be able to manage the parallel evaluation of the group of individuals created at the end of each HGA iteration.
FITNESSSCALINGFACTOR	(double) Power scaling factor of the fitness value
KRIGINGALGORITHM	(KrigingParameters) Parameters for the kriging algorithm
KRIGINGRECURSIVEFILENAME	(string) Filename with the necessary HGA parameters to be used for the recursive run with the kriging proxy
LOADDATABASE	(bool) Flag to load a database and use it for the optimization
MAXNUMEVALUATIONS	(int) Maximum number of function evaluations after which the algorithm quits
MAXNUMITERATIONS	(int) Maximum number of iterations after which the algorithm quits
MAXTIME	(int) Maximum time in seconds after which the algorithm quits
NEIGHBORHOODRANGE	(double) Range of neighborhood mutation
NUMPOPULATION	(int) Number of sub populations
NUMRUN	(int) Number of runs to carry out
NUMTOURNAMENT	(int) Number of individuals to participate in the tournament selection
OLDDATABASEFILENAME	(string) The database file to load
OUTPUTFILES	(bool) Flag to output files, 0:no output, 1:output to files
PCROSSOVER	(double) Probability of crossover for each individual, [0,1]
PMUTATION	(double) Probability of mutation for every single gene, [0,1]
PMUTATIONNEIGHBORHOOD	(double) Probability of neighborhood mutation, [0,1]
POLYTOPEALGORITHM	(PolytopeParameters) Parameters for the polytope algorithm
POLYTOPETYPE	(vector<int>) Flag indicating how the initial polytope is constructed, list one or several of the choices, from 0:best of generation, 1:best of database, 2:closest to the best
POPULATION	(PopulationParameters) Parameters for the population
RANDOMSEED	(int) Random seed for the run, 0 indicates a random seed based on CPU time
RUNDESCRIPTION	(string) Description of the run
RUNNAME	(string) Name of the run, no spaces
SELECTIONTYPE	(int) Criteria to construct the roulette wheel, 0:fitness based, 1:rank based, 2>windowing
USECLOSESTDATA	(bool) Flag to use the scaled closest data function value for invalid points, 0:assign 0, 1:assign scaled function value of closest data to invalid points
USEDATABASE	(bool) Flag to use the database, 0:no database, 1:use database
USEELITISM	(bool) Flag to use elitism, 0:no elitism, 1:use elitism

Table B.2: Keywords for the PopulationParameters class.

Keyword	Description
ALPHABETSIZE	(vector<int>) Vector of the respective alphabet sizes of the variable
MAXVALUE	(vector<double>) Vector of the minimum bounds on the variables
MINVALUE	(vector<double>) Vector of the minimum bounds on the variables
OUTPUTFILES	(bool) Flag to output files, 0:no output, 1:output to files
POPULATIONNAME	(string) Descriptive name of the population
POPULATIONSIZE	(int) Number of individuals in the population
PRECISION	(vector<double>) Vector of the minimum discretization intervals of each variable
RUNNAME	(string) Name of the run, no spaces
UNIFORMINITIALIZATION	(bool) Flag to initialize the population by uniform design, 0:random, 1:uniform
VALUENAME	(vector<string>) Vector of descriptive names of the variables to be optimized

Table B.3: Keywords for the PolytopeAlgorithmParameters class.

Keyword	Description
CLOSESTSCALINGFACTOR	(double) Power scaling of the invalid points
CONTRACTIONCOEFFICIENT	(double) Contraction coefficient of the polytope contraction step, $0 < \dots < 1$
CONVERGENCECRITERIA	(int) Flag indicating the criteria for convergence, 0:MAXNUMITERATIONS, 1:CONVERGENCEVALUE
CONVERGENCE TOLERANCE	(double) Percent (of the CONVERGENCEVALUE) tolerance to test convergence
CONVERGENCEVALUE	(double) The minimum sought evaluation function value, if found an equal or better solution, the algorithm quits
EXPANSIONCOEFFICIENT	(double) Expansion coefficient of the polytope expansion step, > 1
INITIALSOLUTION	(vector<double>) The initial solution from which the initial polytope is constructed
MAXIMUMNUMCONTRACTION	(int) The maximum number of contraction steps that will be allowed
MAXNUMEVALUATIONS	(int) Maximum number of function evaluations after which the algorithm quits
MAXNUMITERATIONS	(int) Maximum number of iterations after which the algorithm quits
MAXTIME	(int) Maximum time in seconds after which the algorithm quits
MAXVALUE	(vector<double>) Vector of the minimum bounds on the variables
MINVALUE	(vector<double>) Vector of the minimum bounds on the variables
OUTPUTFILES	(bool) Flag to output files, 0:no output, 1:output to files
REFLECTIONCOEFFICIENT	(double) Reflection coefficient of the polytope reflection step, > 0
RUNDESCRIPTION	(string) Description of the run
RUNNAME	(string) Name of the run, no spaces
USECLOSESTDATA	(bool) Flag to use the scaled closest data function value for invalid points, 0:assign 0, 1:assign scaled function value of closest data to invalid points
USEDATABASE	(bool) Flag to use the database, 0:no database, 1:use database

Table B.4: Keywords for the KrigingParameters class.

Keyword	Description
NUMKRIGDATA	(int) The number of data to be used to construct the kriging system
OUTPUTFILES	(bool) Flag to output files, 0:no output, 1:output to files
RUNNAME	(string) Name of the run, no spaces
VARIOGRAMA	(double) The power coefficient
VARIOGRAMC	(double) The multiplier coefficient
VARIOGRAMTYPE	(int) Type of variogram to use, 0:spherical, 1:exponential, 2:gaussian, 3:power, 4:hole effect, 5:auto

Table B.5: Keywords for the SimulationParameters class.

Keyword	Description
ACTIVECELLSFILENAME	(string) Name of the file which has iSize*jSize 0,1's indicating if cell is active, i cycles faster (Eclipse format). Required if ALLCELLSACTIVE=0
ALLCELLSACTIVE	(bool) Flag indicating all wells are active, 0:all cells not active (requires input of ACTIVECELLSFILENAME), 1:all cells active
ECLIPSEDATAFILENAME	(string) Name of the Eclipse data file without the '.DATA'
ECLIPSEINCLUDEFILENAME	(string) Name of the Eclipse include file that will have the well location parameters
ISIZE	(int) Number of grid blocks in the x-direction
JSIZE	(int) Number of grid blocks in the y-direction
NUMWELLS	(int) Number of well locations to optimize
OUTPUTFILES	(bool) Flag to output files, 0:no output, 1:output to files
RUNNAME	(string) Name of the run, no spaces
WELLDEPTH	(vector<double>) Well depth, ft
WELLKLOWER	(vector<int>) End of the perforation interval (k2)
WELLKUPPER	(vector<int>) Beginning of the perforation interval (k1)
WELLNAME	(vector<string>) Name of the well
WELLPWFMIN	(vector<double>) Minimum flowing pressure of well, psia
WELLRATE	(vector<double>) Rate of the well (production or injection). Negative rate indicates that the rate is being optimized
WELLRW	(vector<double>) Well radius, ft
WELLSKIN	(vector<double>) Well skin
WELLTYPE	(vector<int>) Type of the well, 0:producer, 1:injector

Table B.6: Keywords for the CostParameters class.

Keyword	Description
DISCOUNTRATE	(double) Discount rate, fraction
FIXEDCOST	(double) Initial fixed investment amount (CAPEX), \$
GASCOST	(double) Price/cost of gas, \$/MSCF
OILCOST	(double) Price of oil, \$/bbl
OPERATINGCOST	(double) Operating costs, \$/day
OUTPUTFILES	(bool) Flag to output files, 0:no output, 1:output to files
RUNNAME	(string) Name of the run, no spaces
WATERCOST	(double) Cost of handling water, \$/bbl
WELLCOST	(double) Cost of one well, \$/well

Nomenclature

a	constant, neural network node output
\mathbf{A}	population
A	string
A, B, C	outcome
b	constant
c	constant, polytope centroid
$CE()$	certainty equivalent
d	distance
$E()$	expected value
f	function value, fitness
\bar{f}	average fitness
\tilde{f}	random function
$g()$	sigmoid function
h	distance
H	schema
in	neural network node input
k	permeability
l	string size
L	lottery
$m()$	number of occurrence of schema
n	population size

N	number of
NPV	net present value
$o()$	order of schema, order of algorithm
p	probability
P	probability
Q	quality
r	risk aversion coefficient
$R()$	Arrow-Pratt measure of absolute risk aversion
t	generation
T	true output, computational time
U	utility value
$U()$	utility function
w	neural network connection weight, quality cell weight
x	point
Z	integer set
$Z()$	intrinsic random function

Symbols

α	reflection coefficient, neural network learning rate
β	expansion coefficient
$\delta()$	defining length of schema
γ	contraction coefficient
$\gamma()$	variogram function
λ	kriging weight
μ	Lagrange parameter
ϕ	porosity
σ	standard deviation
σ^2	variance

Superscripts

- * estimation
- t connection layer

Subscripts

- a alphabet
- b binary
- c polytope contraction, crossover, computer, cell
- cw cells c belonging to well w
- e polytope expansion
- i individual
- $event$ event
- E estimation
- i coordinate, event, string, node
- j coordinate, node
- m mutation
- max maximum
- min minimum
- r polytope reflection
- s survival
- t total
- w well
- wc well-cell

Bibliography

Aanonsen, S. I., Eide, A. L., Holden, L., and Aasen, J. O. (1995). Optimizing Reservoir Performance Under Uncertainty with Application to Well Location, paper SPE 30710 presented at the SPE Annual Technical Conference & Exhibition, Dallas, Texas, October 22-25.

Anderson, J. A. (1995). *An Introduction to Neural Networks*. MIT Press, Cambridge, MA.

Bäck, T. (1992). Self-Adaptation in Genetic Algorithms. *In Proceedings of the First European Conference on Artificial Life*, pp 263–271.

Barker, J. W., Cuypers, M., and Holden, L. (2000). Quantifying Uncertainty in Production Forecasts: Another Look at the PUNQ-S3 Problem, paper SPE 62925 presented at the SPE Annual Technical Conference and Exhibition, Dallas, Texas, October 1-4.

Beckner, B. L. and Song, X. (1995). Field Development Planning Using Simulated Annealing - Optimal Economic Well Scheduling and Placement, paper SPE 30650 presented at the SPE Annual Technical Conference & Exhibition, Dallas, Texas, October 22-25.

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

Centilmen, A., Ertekin, T., and Grader, A. S. (1999). Applications of Neural Networks

- in Multiwell Field Development, paper SPE 56433 presented at the SPE Annual Technical Conference & Exhibition, Houston, Texas, October 3-6.
- Chacko, G. K. (1993). *Operations Research/Management Science: Case Studies in Decision Making Under Structured Uncertainty*. McGraw-Hill, New York, USA.
- Chawathe, A. (1999). DeepLook: Toward the Common Goal of Reservoir Characterization Through Collaboration. *Paper OTC 10785, in Proceedings of the Offshore Technology Conference, Houston, Texas, May 3-6*.
- da Cruz, P. S., Horne, R. N., and Deutsch, C. V. (1999). The Quality Map: A Tool for Reservoir Uncertainty Quantification and Decision Making, paper SPE 56578 presented at the SPE Annual Technical Conference and Exhibition, Houston, Texas, October 3-6.
- Davis, L. (1989). Adapting Operator Probabilities in Genetic Algorithms. *In Proceedings of the Third International Conference on Genetic Algorithms*, pp 61–69.
- Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.
- De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. thesis, University of Michigan. *Dissertation Abstracts International* 36(10), 514B.
- DeGroot, M. (1970). *Optimal Statistical Decisions*. McGraw-Hill, New York, USA.
- Erdogan, M., Mudford, B., Chenoweth, G., Holeywell, R., and Jakubson, J. (2001). Optimization of Decision Tree and Simulation Portfolios: A Comparison, paper SPE 68575 presented at the SPE Hydrocarbon Economics and Evaluation Symposium, Dallas, Texas, April 2-3.
- Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution*. John Wiley, Chichester, UK.

- Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical Optimization*. Academic Press, London, UK.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.
- Goldberg, D. E., Deb, K., and Korb, B. (1991). Do Not Worry, Be Messy. *In Proceedings of the Fourth International Conference on Genetic Algorithms*, pp 154–30.
- Goovaerts, P. (1997). *Geostatistics for Natural Resources Evaluation*. Oxford University Press, Oxford.
- Grefenstette, J., Gopal, R., Rosmaita, R., and Gucht, D. (1985). Genetic Algorithms for the Traveling Salesman Problem. *In Proceedings of the Second International Conference on Genetic Algorithms*, pp 160–168.
- Güyağüler, B. and Gümrah, F. (1999). Comparison of Genetic Algorithm with Linear Programming for the Optimization of an Underground Storage Field. *IN SITU*, 23(2), pp 131–150.
- Güyağüler, 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 SPE Annual Technical Conference and Exhibition, Dallas, Texas, October 1-4. to appear in SPEREE 2002.
- Hinterding, R., Michalewicz, Z., and Eiben, A. E. (1997). Adaptation in Evolutionary Computation: A Survey. *In Proceedings of the Fourth IEEE Conference on Evolutionary Computation*, pp 65–69.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Harbor.
- Holloway, C. A. (1979). *Decision Making Under Uncertainty, Models and Choices*. Prentice-Hall, New Jersey, USA.

- Johnson, V. M. and Rogers, L. L. (2001). Applying Soft Computing Methods to Improve the Computational Tractability of a Subsurface Simulation-Optimization Problem. *Journal of Petroleum Science and Engineering*, special issue on Soft Computing, pp 153–175.
- Jonkman, R. M., Bos, C. F. M., Breunese, J. N., Morgan, D. T. K., Spencer, J. A., and Sondena, E. (2000). Best Practices and Methods in Hydrocarbon Resource Estimation, Production and Emissions Forecasting, Uncertainty Evaluation and Decision Making, paper SPE 65144 presented at the SPE European Petroleum Conference, Paris, France, October 24-25.
- Koza, J. R. (1992). *Genetic Programming*. MIT Press, Cambridge MA.
- Matheron, G. (1965). *Les Variables Régionalisées et leur Estimation, une Application de la Theorie de Fonctions Aleatoires aux Sciences de la Nature*. Paris, Masson et Cie.
- Mathias, K. E. and Whitley, D. (1994). Transforming the Search Space with Gray Coding. *In Proceedings of the IEEE Conference on Evolutionary Computation*, pp 513–518.
- Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21, pp 1087–1092.
- Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag Berlin Heidelberg.
- Nelder, J. and Mead, R. (1965). A Simplex Method for Function Minimization. *Computer Journal*, 7, pp 308–313.
- Pan, Y. and Horne, R. N. (1998). Improved Methods for Multivariate Optimization of Field Development Scheduling and Well Placement Design, paper SPE 49055 presented at the SPE Annual Technical Conference & Exhibition, New Orleans, Texas, September 27-30. (Also JPT, December 1998).

- Rechenberg, I. (1973). *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Hoolzboog Verlag, Stuttgart.
- Rian, D. T. and Hage, A. (1994). Automatic Optimization of Well Locations in a North Sea Fractured Chalk Reservoir Using a Front Tracking Reservoir Simulator, paper SPE 28716 presented at the SPE International Petroleum & Exhibition of Mexico, Veracruz, Mexico, October 10-13.
- Rogers, L. L. and Dowla, F. U. (1994). Optimal Groundwater Remediation Using Artificial Neural Networks with Parallel Solute Transport. *Water Resources Research*, 30(2), pp 457-481.
- Rubinstain, M. F. (1975). *Patterns of Problem Solving*. Prentice-Hall, New Jersey, USA.
- Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence, A Modern Approach*. Prentice Hall, New Jersey.
- Sarich, M. D. (2001). Using Genetic Algorithms to Improve Investment Decision Making, paper SPE 68725 presented at the SPE Asia Pacific Oil and Gas Conference and Exhibition, Jakarta, Indonesia, April 17-19.
- Simpson, G. S., Lamb, F. E., Finch, J. H., and Dinnie, N. C. (2000). The Application of Probabilistic and Qualitative Methods to Asset Management Decision Making, paper SPE 59455 presented at the SPE Asia Pacific Conference on Integrated Modeling for Asset Management, Yokohama, Japan, April 25-26.
- Stoisits, R. F., Crawford, K. D., MacAllister, D. J., Lawal, A. S., and Ogbe, D. O. (1999). Production Optimization at the Kuparuk River Field Utilizing Neural Networks and Genetic Algorithms, paper SPE 52177 presented at the Mid-Continent Operations Symposium, Oklahoma City, Oklahoma, March 28-31.
- Thankur, G. C. (1995). The Role of Technology and Decision Analysis in Reservoir Management, paper SPE 29775 presented at the SPE Middle East Oil Show, Bahrain, March 11-14.

- von Neumann, J. and Morgenstern, O. (1947). *Theory of Games and Economic Behavior*. Princeton University Press, New Jersey, USA.
- Wackernagel, H. (1998). *Multivariate Geostatistics*. Springer-Verlag, Berlin Heidelberg.
- Whitley, D. (1989). The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. *In Proceedings of the Third International Conference on Genetic Algorithms*, pp 116–121.