Optimizing Hydrocarbon Field Development Using a Genetic Algorithm Based Approach

Antonio Carlos Bittencourt de Andrade Filho

March 4, 1997

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
Antonio Carlos Bittencourt de Andrade Filho
March, 1997

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 Horne
(Principal Adviser)

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. Clayton Deutsch

Approved for the University Committee on Graduate Studies:

Acknowledgements

I would like to show my appreciation and gratitude to my research advisor Dr. Roland Horne for his advice, guidance, and encouragement during the course of this research.

I appreciate Arild Nystad and Fred Glover for the material supplied at the beginning of this work. I am also thankful to Yan Pan for adapting some optimization methods to my needs.

I would like to thank Petrobras for allowing me to take part of the company's training program providing the required financial support towards the doctorate degree. I also express my gratitude to the Stanford University Research Consortium for Innovation in Well Test Analysis (SUPRI-D) for providing computational resources.

I also grateful to Milton Gabrieli, Francisco Couri, Lideniro Alegre and Marcia Amitay for the effort and confidence that they invested on my success.

Finally, I would like to express my sincere gratitude to my family for their constant support and to Shoshanna Lurie for her assistance and friendship.

Abstract

The main task of a reservoir engineer is to develop a scheme to produce as much hydrocarbon as possible within economic and physical limits. The solution of this kind of problem encompasses two main entities: the field production system and the geological reservoir. Each of these entities presents a wide set of decision variables and the choice of their values is an optimization problem. In view of the large number of decision variables it is infeasible to try to enumerate all possible combinations. Analysis tools encoded in computer programs can spend hours or days of processing for a single run, depending on their sophistication and features. Also, it can be costly to prepare the input data if many hypotheses are going to be considered and if it is desirable to allow the parameters to vary.

A typical reservoir development problem involves many variables that affect the operational schedule involved in its management. These variables are usually used as input to a reservoir simulator that generates a forecast of the production profile. Using this forecast, the production engineer has to consider several hypotheses to achieve the best strategy for the field development problem. Also, each hypothesis can generate other ones, and so, the overall process is one of generating a hypothesis tree. More and more data is generated and analyzed. From this analysis more questions and hypotheses arise, often causing the engineer a sense of something left out when concluding the work. The solution of this problem requires the effort of several people as well as computer work and physical time.

An optimization procedure requires the characterization of the function to be optimized (minimized or maximized), known as the objective function, as well as the choice of a proper optimizing method. The complexity of predicting hydrocarbon

production profiles requires the use of reservoir simulators. So, the simulator must be part of the evaluation of the objective function.

This work concerns the optimization of characteristic petroleum problems considering economic factors. A hybrid algorithm based on direct methods such as Genetic Algorithm, polytope search method, Tabu Search and memory strategy is presented. Hybrid techniques were found to improve the overall method. The objective function consists of a cash flow analysis for production profiles obtained from simulation runs considering a particular set of parameter values. The optimizing procedure is able to interface with commercial simulators (generating its input data and retrieving the results) that work as data generators for the objective function evaluation.

These hybrid mathematical approaches were found to be successful in obtaining the optimal solution with less time and work than existing techniques. These approaches can speed up the study of a hydrocarbon reservoir development plan and allow consideration of a wider range of hypotheses. The engineer can also keep track of economics during the study, allowing better project decisions.

A real project was optimized using two approaches: the first one had the proposed solution inserted in the initial population and the second one did not. This second approach was able to find a better solution spending less function evaluations than the first one, which suffered from premature convergence.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Frequently the operational solution to a petroleum engineering problem is not the optimal solution but is merely a viable result that satisfies the operational restrictions at a specific moment. This lack of optimality may result from oversimplification of a problem and can cause us to neglect important aspects in the project that may be costly to the analysis.

The main task of a reservoir engineer is to develop a scheme to produce as much hydrocarbon as possible within economic and physical limits. The solution of this kind of problem encompasses two main entities: the well and the geological reservoir. Each of these entities presents a wide set of decision variables and the choice of their values is an optimization problem. Depending on the number of decision variables it is infeasible to try to enumerate all possible combinations. Analysis tools encoded as computer programs can spend hours or days of processing for a single run, depending on their sophistication and features. The challenge for this work was to develop more efficient and more effective algorithms for this task than those currently in use.

1.1 Problem Statement

Hydrocarbon field development planning involves several variables that affect the operational schedule involved in the management of the field. These variables are usually

used as input to a reservoir simulator that generates a forecast of the production profile. Using this forecast, the production engineer has to consider several hypotheses to achieve the best strategy for the field development problem. Also, each hypothesis can generate others, and so, the overall process is one of generating a hypothesis tree. More and more data are generated and analyzed.

Often, in the industry, there is insufficient time or personnel to perform such a task in a manner that includes all the requirements for the field development.

Looking at this as an optimization problem, we can improve the overall result and reduce the time spent and the human and machine effort. In an optimization, the search for the best solution navigates from one hypothesis to another in an interconnected process. As a result, we can find the best option among the several hypotheses under consideration.

An optimization procedure requires the characterization of the function to be optimized (minimized or maximized), known as the objective function, as well as the choice of a proper optimizing method. There are two traditional types of optimizing methods: derivative-based methods, that often assure a faster convergence but require evaluations of the objective function derivatives, and direct methods, that require only function evaluations but usually have slower convergence.

The complexity of predicting hydrocarbon production profiles requires the use of reservoir simulators. So, the simulator must be a part of the evaluation of the objective function. Simulation may become a limiting factor for the derivative-based methods because of the additional cost of evaluating the derivatives of the objective function using the simulator. Direct methods do not require derivatives and therefore may have an advantage when simulation is involved. The intent was to develop a hybrid algorithm that combined the advantageous aspects of the direct method approaches. The objective function represents an evaluation based on an economic analysis that considers the production data retrieved from a simulator. This objective function is maximized in a nonlinear optimization. The use of a simulator to generate the objective function allows us to treat problems of increased size and complexity. As a consequence of this work new techniques will be available to the engineer to perform better reservoir developments in which economic profit is maximized.

We focused our attention on characterizing the objective function, the process of interfacing the optimization algorithm with the reservoir simulator and the choice of the most adequate optimizing method. The test problems included field examples of practical significance, such as the scheduling of incremental developments to maximize production of a heterogeneous reservoir.

1.2 Literature Review

Previous authors have discussed methods to optimize hydrocarbon field production related problems. The common point among all works is the problem simplification, small to moderate problem sizes, none or few simulation runs, and few simplified explanatory variables. Sometimes economic analysis is considered but in a very simplified way. The main goal is to capture the most significant input factors and to get a response suitable for optimization. The works can be classified in the following categories:

- two explanatory variable methods;
- three or more explanatory variable methods;
- data integration based methods;

Two explanatory variable methods are represented by the work of Nystad. Nystad [1985, 1987, 1988] investigated this topic in a way related to the approach proposed in this work.

Nystad [1985a] analyzed the optimization of the exploitation of an oil or gas reservoir in terms of:

1. Total depletion rate (production capacity, including the aspects of Maximum Efficient Rate)
2. Geographical distribution of total production capacity (well density, number of platforms, etc.)

In Nystad's approach, a fine-scale network of production profiles is generated from a limited number of simulation runs. The income is computed with a reservoir cost module and the optimal solution is sought on the net present value surface. A simplification is made by incurring the capital costs in the first period and the net income due to the production in a second period. It is then assumed that the amount of extracted petroleum is a concave function of factor inputs, i.e., capital and operating costs. Factors like the choice of the production capacity and rate-sensitivity are disregarded.

Nystad says that the oil company will make an economic adjustment in order to maximize the net present value (the profit π):

$$\max_E \pi = \max_E(-WE + \frac{PF(E)}{1+r}) \tag{1.1}$$

where:

E = factor inputs

P = price of petroleum

$F(E)$ = extracted amount of petroleum (resource recovery)

W = price of factor input

r = discount factor

The first condition for the maximum occurrence is that of the first derivative of Equation 1.1 be zero, or:

$$F'(E) = \frac{W}{P}(1+r) \tag{1.2}$$

The second condition is satisfied by assuming the function to be concave.

The adjustments required to maximize the profit and to find the optimal recovery factor are then analyzed.

The depletion rate strategy must maximize the net present value given by:

$$\max_{\vec{x}} NPV = \max_{\vec{x}} \int_0^{t_c} [p(t)q(t,\vec{x}) - I(t,\vec{x}) - b(t,\vec{x})]e^{-rt_c}dt \tag{1.3}$$

where:

$\vec{x}$ = decision variables

$q(t,\vec{x})$ = petroleum production rate

$I(t, \vec{x})$ = capital costs

$b(t, \vec{x})$ = operating costs

$p(t)$ = price of petroleum

t_c = production cut-off time

r = discount factor

Nystad [1985a] performed this kind of analysis for four different types of reservoirs, attempting to cover the range of the most common types. The optimization was performed considering two main decision variables: depletion rate (x_1) and geographical distribution of total production capacity (x_2). Both variables were allowed to vary at the same time. The net present value surface to be optimized was then expressed in terms of these variables. For a specific reservoir this surface must be estimated with simulation runs.

The model produces values for optimal economic recoverable reserves, production capacity and number of platforms against tax share of investment for assumed tax share of revenue and operating cost.

Three or more explanatory variable methods have been described by Damsleth et al. [1992] as well as by Beckner and Song [1995]. Damsleth et al. [1992] applied a statistical design of experiments to maximize the information obtained from a specified number of simulation runs. The intent was to reduce the number of simulation runs required when varying one parameter at a time during the performance of a sensitivity analysis. Interactions among the various input parameters were identified and estimated with a more elaborate design. Considering the recovery factor as being the response variable y and the explanatory variables being x_1, x_2, x_3, their functional relationship defined by the reservoir simulator is:

$$y = f(x_1, x_2, x_3,) \tag{1.4}$$

Using response-surface analysis and experimental design, the function y above can be approximated by the following model:

$$y = \sum_{i=0}^{n} b_i x_i + \sum i = 0, j = 0^n b_{ij} x_i x_j + e \tag{1.5}$$

where b's are unknown coefficients and e is an error term. The model may consider or not the cross terms depending on how complex it should be. After estimating the unknown coefficients from a number of experiments the deviation between the true function (reservoir simulator) and the assumed model is minimized. Experimental design sets the procedure of how to choose a limited number of parameters in a way to estimate the polynomial approximation optimally .

The base case was a North Sea reservoir where, after a one-parameter-at-a-time sensitivity analysis, the chosen input parameters were: the original oil in place (N), relative permeabilities (k_r), vertical permeability (k_v), lateral segmentation of the reservoir (n) and the well skin factor (s). After applying the proposed model, the input parameters that affected N_p were:

- k_v and s were considered weak and excluded from the final model
- N was the most important explanatory variable
- k_r did not have a very strong effect on N_p
- n had a very weak effect on N_p but interacted with both N and k_r

The final model accounted for about 90% of the variability in the experiments. This technique is applicable to smooth functions and extrapolation may be dangerous for input values outside the range actually used.

Beckner and Song [1995] presented a method to optimize the net present value of a full field development by varying the placement and sequence of production wells. Beckner and Song used simulated annealing as the optimization engine and expressed the well scheduling and placement as a classical "traveling salesman" problem where the total traveled distance is optimized. In this analogy the well sites correspond to cities and scheduling is the traveling sequence. For a given scheduling (sequence of locations) the simulation output is used to calculate the project NPV (total distance).

The base problem was how to place twelve horizontal wells in order to develop a field. Seven different optimization cases were investigated. These cases ranged from optimizing a uniform reservoir, uniform well cost development to a case that optimizes

development of a reservoir with variable permeability, variable initial pressures and variable well costs. The average drilling and completion time of three months per well was the constraint for the scheduling plan.

The results showed that a nonuniform well spacing is optimal for development under primary production. In optimizing the project NPV, variable well costs influenced the choice of well locations as much as variations in reservoir properties. An optimality test was performed and a new optimum was found by hand but with the previous knowledge of the optimum suggested by the simulated annealing method. Even so, the NPV was improved by only 0.07%.

Data integration and interactive methods can be exemplified by Arnondin and Ding and Startzman works. Arnondin [1995] used a program called Production Analyst integrated with an Excel spreadsheet. Using the regression tools available in Excel along with the production data retrieved from Production Analyst, Arnondin was able to predict well production profiles constrained to production facility limitations. No simulation runs or other simpler flow model were used and no economic analysis was performed as well.

Ding and Startzman [1994] used an interactive optimizing program to solve the problem of placement and selection of production facilities in oil field development project. A sensitivity analysis can be obtained by interactively supplying the input data using an X-Motif based interface. A generalized model accommodates the objective function that can be submitted to two optimizing methods: 0-1 integer programming and Lagrangian Relaxation. This application can be used to minimize investments for a given scenario and to perform sensitivity analysis.

1.3 Preliminary Solution Analysis

In this section we analyze some of the methods described in the previous section.

Most of the solutions presented in the past used statistical, gradient methods or linear programming as an optimization strategy. The problems were also simplified to avoid complex simulation runs. Some efforts were made to overcome this limitation by running few simulations and using statistical methods to capture trends and

interpolated values. However other limitations arose such as uncertainties regarding the range of parameter validity, confidence intervals and average values. Also, only moderate size problems could be considered to save computer time.

Other researchers tried to reduce the parameters to a two dimensional domain. In this way, NPV was the third dimension and the NPV surface could be built easily using interpolation techniques in two dimensions. This strategy allowed easy visualization of the performance of gradient based methods. This is clearly a limiting strategy because it forces us to look at all problems and attempt to identify the two most sensitive parameters and then express the problem in terms of them. We have to oversimplify the problem to fit in this strategy. Again, statistical methods were applied to select the most sensitive parameters and new limitations arose.

Solutions based on a few evaluated function values were used to derive realizations according to statistical methods to obtain other scenarios. The approach depends on finding correlations between the different cases. A degree of uncertainty is a part of this kind of treatment once the extrapolations are based on a combination of parameters, their ranges and sensitivities.

Some methods used simplified simulators to speed up the overall run. The result obtained was used as an indicator for a more detailed run that would refine the solution. As the optimal value was estimated by simplified simulation runs it could deviate the solution when a more sophisticated calculation was performed.

None of the methods mentioned so far used a full field case with a full simulator generating the production data followed by detailed economic analysis. Such was the target of this work.

Chapter 2

The Mathematical Model

Optimization concerns the optimal solution determination using an oriented search towards the best possible value. Algorithms used in the optimization procedure are problem sensitive, so it is necessary to investigate the strengths, weaknesses and ranges of applicability of each.

2.1 Optimization Algorithms

In this section we describe the concepts and main features of several direct optimization methods. Selected concepts are described in more detail and others are just referred to by their central idea. As stated before, all these methods require only function evaluations and are useful when the calculation of the derivatives is not attractive.

2.1.1 Polytope Search Method

The simplex, the simplest polytope, is defined in the multidimensional analytic geometry as being a convex set of $n+1$ linearly independent points in an n-dimensional space. In this way, the simplex is a geometric figure bounded by hyperplanes. Each boundary of the simplex is the intersection of two and only two hyperplanes [Sommerville, 1929, Borsuk, 1969]. The simplex's dimensionality must be preserved

encompassing a finite n-dimensional volume, otherwise the simplex is degenerated.

This method, attributed to Nelder and Mead [1965], is based on the movement of a set of n decision variable values represented by the simplex with $n + 1$ vertices in the n-dimensional search space. We will refer to this method as the Polytope method to avoid confusion with Dantzig's simplex method for linear programming problems.

The polytope movement consists of a series of reflections of the worst point about the centroid of the remaining n nodes. In this way, the polytope flips about the centroid towards the optimum (in this section the minimum value is considered to be the best one). Figure 2.1 shows an example of a polytope search with two decision variables x_1 and x_2.

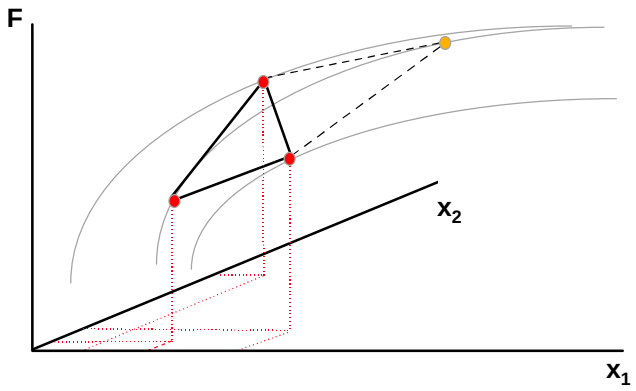

Figure 2.1: An example of a two-dimensional polytope search

A set of $n + 1$ points together with their function values is required to start the method. For each iteration, after sorting the function values F in a descending order so that,

$$F_{n+1} > F_n >, ..., F_2 > F_1 \tag{2.1}$$

the worst point x_{n+1} is taken as being the initial starting point P_o.

The worst point x_{n+1} is then replaced with a new point that is the reflection of x_{n+1} about the centroid c of the remaining n points [Gill et al., 1992]. This movement can be represented mathematically as:

$$c = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.2}$$

$$x_{new} = c + \alpha(c - x_{n+1}) \tag{2.3}$$

where α is called the reflection coefficient, which is always positive. This step is constructed to conserve the volume of the polytope ($\alpha = 1$), avoiding its degeneracy.

Three cases can occur when evaluating the new point:

1. The new point has the function value between the best and the worst points:

$$F_n > F_{new} > F_1 \tag{2.4}$$

   So, accept the new point and go to the next iteration.

2. The new point is the new worst point:

$$F_{new} > F_n \tag{2.5}$$

   Try a contraction factor $\alpha < 1$ and guess a new point. If successful, accept the contraction factor, i.e., the new volume, and go to the next iteration. Otherwise, try a smaller α.

3. The new point is the new best point:

$$F_{new} < F_1 \tag{2.6}$$

   Try to expand the polytope using an expansion factor $\alpha > 1$. If successful, accept the expansion factor. Otherwise, use the original x_{new} for the next iteration.

The sequence in Figure 2.2 illustrates these three cases. After verifying that the new point x_2 is worse than x_1, a reflection is performed about x_1 obtaining x_3. As an improvement in the objective function was obtained, the reflection is expanded further obtaining x_5. This expansion could not improve the objective function value or *evaluation* and so a contraction is tried obtaining x_4. Again, no improvement was obtained and x_3 becomes the new best point (graphs a and b). A reflection about x_3

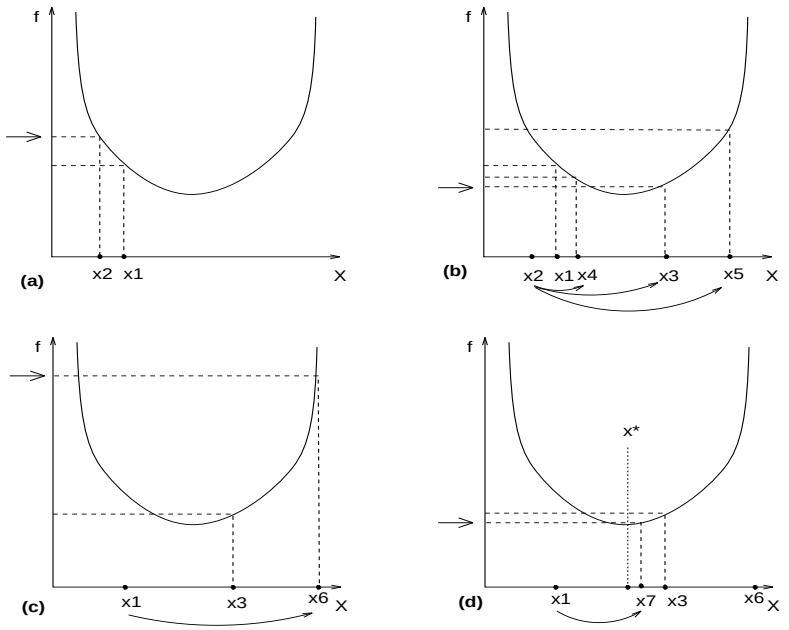

Figure 2.2: Polytope movements in two dimensions

brings x_1 to a worse point x_6 (graph c), a contraction is then performed obtaining x_7 with the best evaluation so far (graph d). This point is accepted as being the new x_1 and this procedure is repeated until x^* is found within a tolerance.

In a multidimensional problem we cannot merely set a tolerance for a single independent variable. However, the objective function can provide the mechanism that relates all variables. The termination criteria suggested in *Numerical Recipes* [Press et al., 1992] is given by:

$$\frac{|F_{best} - F_{worst}|}{\frac{|F_{best}|+|F_{worst}|}{2}} < \varepsilon \tag{2.7}$$

where F_{best} and F_{worst} are the function values for the $best$ and $worst$ vertices, respectively, and ε is a given tolerance.

Limitations

The main limitation of this method is its sensitivity to the initial size and location of the polytope [Bittencourt, 1994]. Tests have shown that a convergence to a local optimum may occur during the search. As the optimal location is not known in advance it may be necessary to make several optimization runs with different initial polytope sizes and locations.

2.1.2 Genetic Algorithm

Genetic Algorithm (GA) is a robust search method based on analogies to biology and genetics. Survival of the fittest among a population of individuals, selection criteria, and reproduction strategies are concepts copied from the natural life and used as operators in this artificial environment [Holland, 1975]. Although this analogy limited GA's appeal for some time new applications have been found in business, engineering and science [Rogers, 1992]. The GA has four advantageous features:

1. GA begins the search with a population of parameter realizations, rather than a single realization as most of the conventional optimization methods might. Each set of possible configuration of the decision variables is referred as one realization or member of the population. In this way, the search domain is covered in a random distribution;

2. The realizations are perturbed by probabilistic rules rather than deterministic ones;

3. The parameter itself is not manipulated directly by the GA operators. GA would alter the *chromosome* (or *individual* or *string*) that is a pattern of zeros and ones representing the whole set of parameters put all together in one binary entity. For binary alphabets, the smaller piece of a chromosome is called *bit* and can have the values zero or one. In this way, the *chromosome* is the set of all parameters' bits.

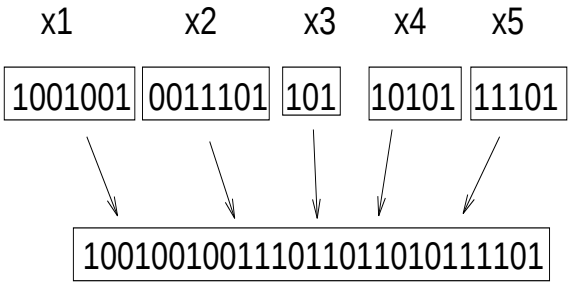

Figure 2.3: A chromosome in a genetic algorithm

4. Only function evaluations are used rather than derivatives or other secondary descriptors. The *fitness* is defined as $\frac{f_i}{\bar{f}}$ or $\frac{f_i}{\sum_i f_i}$, where f_i is the function evaluation of chromosome i and $\bar{f}$ is the average evaluation of the entire population. So, the fitness is a measure associated with each chromosome. These characteristics are particularly handy for production optimization application as the value of the objective function is known (see Figure 2.4).

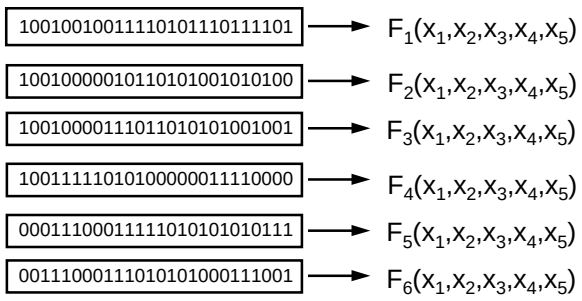

Figure 2.4: The first generation of the new population

Genetic Algorithms are a subset of reproductive population algorithms. This search algorithm starts from a population of parameter realizations and repeatedly

performs the following cycle of operations until the stopping criteria are reached [Rawlins, 1994]:

1. *Evaluation:* evaluate the fitness of each chromosome.
2. *Selection:* depending on each chromosome fitness, select the ones to build the *reproducing set* of a population.
3. *Reproduction:* from each reproducing set, apply reproductive strategies to generate a number of new chromosomes.
4. *Replacement:* replace some or all of the original population with new chromosomes.

Selection can use one of the four common selection schema [Goldberg and Deb, 1994]:

1. Proportionate reproduction;
2. Ranking selection;
3. Tournament selection;
4. Genitor (or "steady state") selection;

We present here a subset of the first selection scheme which was the approach used in the simple GA, base for this work. Further information for the other schemes can be found in the reference [Goldberg and Deb, 1994].

We adopted the model where all strings are able to mate with one another (*random mating*). The selection strategy of strings for reproduction is based on their fitness values and different methods can be used. The *stochastic sampling with replacement* (or *roulette wheel selection*) [De Jong, 1975] method uses the ratio of the evaluation of each string to the total evaluation of all strings to define its probability of selection, i.e.,

$$P_{selection}(i) = \frac{F_i(\vec{x})}{\sum_{i=1}^{all} F_i(\vec{x})} \tag{2.8}$$

The *stochastic remainder selection* [Booker, 1982, Brindle, 1981] uses the ratio of the evaluation of each string to the average population evaluation which is compared to unity, i.e.,

$$\alpha_i = \frac{F_i(\vec{x})}{F(\vec{x})} \diamond 1.0 \tag{2.9}$$

where $\diamond$ can be one of the following operators: equal to, less than, or greater than. If α_i is greater than 1.0, the integer portion indicates how many copies of the string i are placed directly into the intermediate population. All strings (including those with α_i less than 1.0) then place additional copies into the intermediate population with a probability corresponding to the fractional portion of α_i [Whitley, 1996].

Another more efficient way to implement the *stochastic remainder selection* is called *Stochastic Universal Sampling* [Baker, 1985]. The population is randomly distributed over a pie where each individual has its slice proportional to its contribution to the total fitness. An outer roulette composed of pointers equally spaced according to the population size is spun just one time. All individuals are selected at once according to the position of the fixed pointers relative to the pie. Baker [1985] showed that this strategy is unbiased.

Possible reproductive strategies are combinations of:

- Reproduction
- Mating or Crossover
- Mutation

Reproduction is an operator where parameter strings are selected using one of the procedures described above. An *intermediate population* or reproducing set, that is submitted to the other genetic operations, is filled with copies of the selected strings. This is often referred to by biologists as the fitness function in analogy to the Darwinian survival of the fittest. Goldberg [1989] uses the analogy of parameter strings being string creatures (chromosomes) and instead of predators or pestilence, the objective function is used as the final arbiter of whether the string creature will survive. In other words, the string creatures, or strings of parameter values, have a certain amount of information which may or may not be passed on to the next generation. The value of the objective function evaluated for that particular parameter realization will be important in deciding whether the information will survive.

Crossover is a process in which pairs of string creatures are selected from the intermediate population and their strings are cut at a random location and joined to the corresponding piece of the partner string. There are several ways to implement such an operator. The *n-point crossover* determines random crossover sites on the entire length of the chromosome. The crossover sites become the boundary points at which all the information on the first string is exchanged with all the information on the second string within the same boundary. This exchange occurs alternately from boundary to boundary until all crossover sites are exhausted. The *uniform crossover* generates one bit at a time. Each bit is inherited from one of the parents according to the crossover probability. Each one of these crossover operators has its advantages and disadvantages. *Disruption* is one of the effects caused by crossover. It causes the loss of a specific pattern of bits, called *schemata*, that can degrade the search when the bit value of the fixed position is changed *allele loss* causing the disruption of the schema. De Jong [1975] showed that two-point crossover is less disruptive than single-point crossover. However operators that use higher order *n-point* crossover are much more disruptive and should be avoided. Uniform crossover is highly disruptive because alleles not found in the parents can be produced. Although these results can be shown analytically, several researchers have suggested that uniform crossover works better in some cases. Other crossover operators can be defined according to the characteristics of the problem under investigation.

Mutation is surrounded by some controversy in genetics. Mutation is designed to avoid the loss of valuable genetic material, which theoretically may result from reproduction and crossover [Holland, 1975]. Although mutation is one of the most familiar terms in genetics, its role in the GA is sometimes misconceived. Its function is not to generate new structures or patterns; the crossover operator does that in a very efficient way. Mutation's primary and unique role is to create a mechanism by which information or small segments of parameter strings can be reintroduced into a population. In the simple GA, mutation is a random alteration of a variable in a string. However, one might randomly choose a mutation site bit once every second or third generation. Because of its nature, mutation is highly disruptive and the assumption of very low values is recommended. Some researchers suggest that

a strategy with no crossover and high mutation rate produces a fairly robust search [Schaffer et al., 1989, Fogel and Atmar, 1990]. Others state that mutation alone is not enough [Schaffer and Eshelman, 1991].

One of the easiest ways to apply the reproduction operator is using the *roulette wheel selection*. In this procedure the evaluations of a population are summed and then the contribution of each string creature is calculated (see Figure 2.5). This fitness of a specific string creature will determine how many times the creature gets into the *mating pool* or reproducing set, an intermediate step to the next generation.

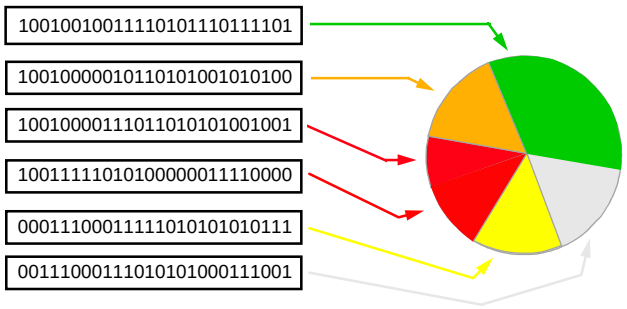

Figure 2.5: Contribution to the fitness: "the roulette"

In this way an individual vector of parameters could be analogous to an individual string creature (chromosome). We assign a specific length for each of the parameters compatible with their original domain and the desired resolution. The parameters are then mapped into the binary domain to compose the chromosome. After the three operations reproduction, crossover and mutation have taken place, the new individuals are mapped back to their domain (real or integer) and then used as input to the objective function. The measure of evaluation for the population will be the highest value of the objective function obtained in this generation.

For the next generation a couple of mates is selected after spinning the roulette twice (see Figure 2.6). The crossover site (if single-point crossover is used) is determined probabilistically and then two new individuals (children) are generated (see Figure 2.7).

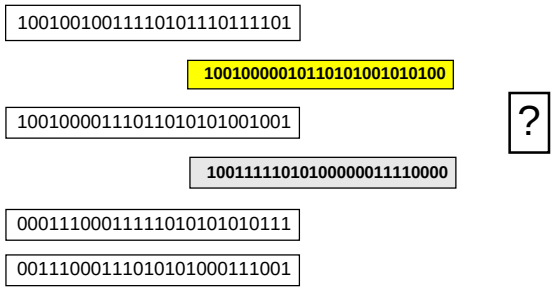

Figure 2.6: Selection for crossover: pick any two

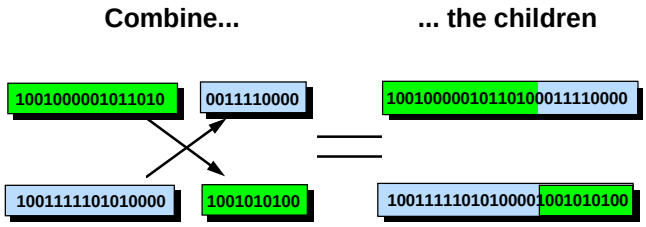

Figure 2.7: Children generation from crossover

The algorithm finally evaluates the objective function for each member of the population.

The GA will continue to cycle through the three procedures of reproduction, crossover and mutation in its search for the best patterns (see Figure 2.9), until some convergence or stopping criterion is reached.

Limitations

One of GA's features is the way that the initial population is started, covering the domain randomly. As any random process, this initial distribution can cover both good

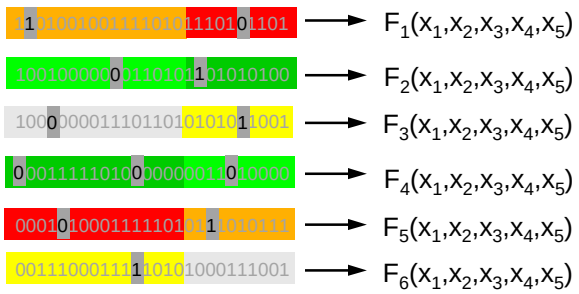

Figure 2.8: The new generation

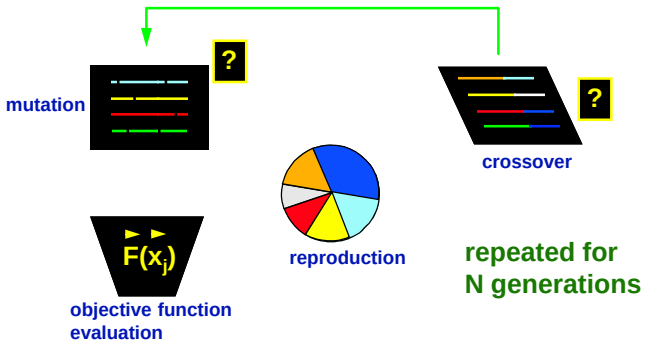

Figure 2.9: The GA cycle

and bad regions. The process of generating populations is also random and depends on the values of the initial random seed as well as on the crossover and mutation probabilities. The specific variants of the genetic operations (reproduction, crossover and mutation) play an important role in the overall procedure. Several GA runs may be required to test a method's convergence. Premature convergence, that means convergence behavior without guarantee of optimality [Goldberg, 1989], has been the object of study of some researchers [Ackley, 1987] [Eshelman et al., 1989, Syswerda, 1989] and some strategies like *incest prevention*,*niching* and *increasing mutation rate* have been suggested [Goldberg, 1989, Eshelman and Schaffer, 1991]. These strategies force

the production of string sequences not found in the parents, or in other words, new regions in the domain are visited.

Other techniques have been suggested. We can establish a threshold heuristic so that the best individual is always saved from generation to generation (*elitism*) as is used in the *Genitor* model [Whitley, 1989]. This approach assures that evaluation values will never decrease from one generation to the next and assures that crossover and mutation do not lead to a degradation. Another strategy can be to consider the best individual to be a permanent mate and the other to be selected on spinning the roulette. This can force the convergence to a specific region in the domain and may be not appropriate.

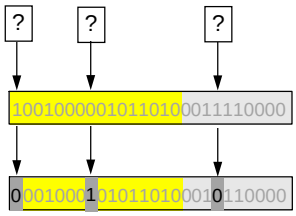

Figure 2.10: Mutations may happen

Along with mutation the other two "best individual" strategies encourage persistence of patterns judged to be "most fit" as the population of patterns evolve from one generation to the next.

The dependency of the method on so many genetic parameters requires extensive research to determine the most appropriate choice of strategies to the function to be optimized.

2.1.3 Tabu Search

Glover [1994] defines the Tabu Search (TS) as being a *metaheuristic* that guides a local heuristic search procedure to explore the solution space beyond local optimality. The algorithms described so far can converge to a local optimum neglecting other regions within the domain that may contain a better function value. TS provides a mechanism that forces the search into other regions far away from the current one by forbidding the current region to be revisited for some finite number of iterations.

Each parameter set is stored in a short or long-term memory. When a parameter happens to be repeated from one iteration to another, it is forbidden (tabu) and a new parameter must be generated to fill this slot. This forces the new parameters to try other regions for however many iterations or cycles the tabu restriction exists. The original region can be visited again only from new locations or after there is no more tabu for an existing parameter.

An *aspiration criterion* works as a threshold heuristic where an existing parameter can be accepted, even if it is tabu, if no improvement in the function value is obtained by visiting new regions.

Limitations

Because of the characteristics of the TS method, the algorithm must be adapted to every different problem. Also, the Tabu search may force the search to go unnecessarily to other regions spending function evaluations instead of refining better regions.

The idea behind this method is strongly related to *niching* in the GA search.

2.1.4 Multivariate Interpolation

Since it would not be feasible to exhaustively generate the entire n-dimensional surface to be optimized, it may be useful to use a method to represent this surface from a reduced number of points using an interpolation procedure. This approach would use the function value of a starting set of parameters to build a representation of the surface. Then intermediate values required by any of the optimization methods can be estimated by interpolation [Pan, 1995].

Limitations

A limitation of this method is the level of detail that the interpolated surface must have to allow the correct evaluation of the optimal value. The determination of the acceptable minimum number of points required to represent the surface satisfactorily would be difficult to establish for a given problem.

2.1.5 Fang Algorithm

The Fang algorithm [Fang, 1980] may be used to generate a starting distribution for GA. Although it is not an algorithm to be used during an iterative optimization, the Fang algorithm is very useful to start the set of parameters for the first iteration. The algorithm assures an even coverage of the search domain by populating a given search space as uniformly as possible for a specific number of points [Pan, 1995]. It works quickly for levels (population size) smaller than 100 but for those greater than this value it is recommended that the level be a prime number or a prime number minus one.

The following text was based on the discussion presented in *Application of Least Squares and Kriging in Multivariate Optimizations of Field Development Scheduling and Well Placement Design* (Pan [1995]).

Uniform Design

In a multidimensional problem with s *parameters*, or $factors$, and q *discretized possible values of parameters*, or $levels$, it is desirable to have the initial domain coverage to be as uniform as possible. Considering $tests$, or $individuals$, with equal levels of each factor, the number of possible combinations is q^s. Uniform design [Fang, 1980], that is based on number theory, can reduce this number to q.

For a distribution of q tests the following principles are applied to uniform design:

1. q is an odd number q_o or $q_o - 1$.

Select positive integers $a_1, a_2, \ldots, a_s$ satisfying $(a_i, q) = 1,\ i = 1, \ldots, s.$ where (a, b) is the maximum common factor of a and b. Then the uniformly distributed points

are:

$$P_q(k) = (ka_1, ka_2, \ldots, ka_s)(mod\ q_o),\ k = 1, \ldots, q_o. \tag{2.10}$$

2. q is a prime number p or $p - 1$.

Select a positive integer a such that $a < p$, then:

$$P_q(k) = (k, ka, ka^2, \ldots, ka^{s-1})(mod\ p),\ k = 1, \ldots, p. \tag{2.11}$$

Note that 2 is a special case of 1, so just one theorem to specify $a_1, a_2, \ldots, a_s$ is sufficient.

The following theorem is known as $Uniform\ Theorem$: the vector $\vec{a} = (a_1, a_2, \ldots, a_s)$ or $(1, a, a^2, \ldots, a^{s-1})$ is the one that minimizes $\xi(q, \vec{a})$ given by:

$$\xi(q, \vec{a}) = \frac{1}{q} \sum_{k=1}^{q} \prod_{i=1}^{s} (1 - \frac{2}{\pi} ln(2 \sin \pi \frac{a_{ik}}{q+1})) \tag{2.12}$$

where $a_{ik} \equiv ka_i(mod\ q)$ and $a_{iq} \equiv q(mod\ q)$.

2.2 Economic Analysis

Although economic considerations alone are not enough to justify decisions in real projects, economic analysis can be used to assist the process of making rational decisions [Fleischer, 1994]. Each alternative generate heterogeneous consequences that must be compared to help the decision process. A *common unit of measurement* makes the consequences commensurable. This single dimension is generally represented by money units and must also consider the time where it is applied and the problem of comparing capitals between different times.

The simplified methodology described below performs a *cash flow*, that refers to receipt or payment of an amount of money, in terms of actual receipts or actual expenditures. In order to compare the several values that occur in different times all the amounts must be corrected to a time reference using a compound interest calculation. Then, one can directly compare the equivalent values generated by such adjustments. *Interest* is a monetary value and *interest rate* is the ratio of interest

charged during an *interest period* to the amount of money at the beginning of the interest period. The length of the period is usually taken as the project life. The interest rate is generally written as a percentage and applies to a specific period length (month, year, quarter etc...).

The typical cash flow holds the relationships among interest, interest rate, and interest period as:

$$i = \frac{A_1 - A_0}{A_0} \tag{2.13}$$

or

$$A_1 = A_0(1 + i) \tag{2.14}$$

where A_0 is the amount at the start of the period, A_1 is the amount at the end of the period, i is the interest rate and $A_0 \cdot i$ is the interest. If the adjustment is to be performed over j periods of same length then Equation 2.14 becomes (Figure 2.11):

$$A_1 = A_0(1 + i)^j \tag{2.15}$$

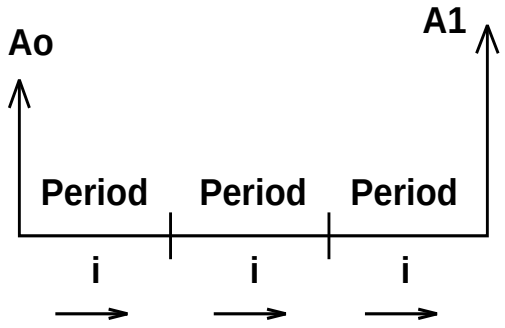

Figure 2.11: Typical cash flow for j periods

Usually A_0 is referred as *initial investment* P and A_1 as *future value* F or *repayment* and Equation 2.15 assumes the notation:

$$F = P(1+i)^j \tag{2.16}$$

In the same way, if we know F, i and j then P can be evaluated as:

$$P = F(1+i)^{-j} \tag{2.17}$$

Although there are four standard methods in the economic analysis of courses of action with regard to investment [Steiner, 1992], we considered only the *net present value* method. For selecting a project, the net present value criterion states that the present worth of benefits must equal or exceed the present worth of costs. So, the project may be accepted or not and there is nothing in between. In mathematical terms this analysis can be represented as:

$$\sum_{j=1}^{N}(B_j - C_j)(1+i)^{-j} \geq 0 \tag{2.18}$$

where B_j stands for benefits at the end of period j, C_j for costs at the end of the same period, and $(1+i)^{-j}$ the single payment present worth factor for period j at interest rate i. In this way, the costs are subtracted from the benefits at any period and then the result is adjusted to reference period. This is done for all periods in the problem where the result for all the periods under consideration were added. The attractiveness of the project will depend on the positiveness of the result as well as on its magnitude.

Chapter 3

The Objective Function

The economics related to reservoir development is a problem of current interest. Because of the large amount of money involved, any small variation in cost or income can represent millions of dollars. While developing a field the engineer faces a problem with multiple decision variables. Previous experience is required to make a good estimate of the sensitivity to these decision variables, which can change significantly from reservoir to reservoir. Sensitivity studies are commonly performed to know how reservoir and production parameters interact with each other.

Well placement is often one of the challenges. Some works have shown that a symmetric or an evenly-spaced well placement strategy may not be a good choice for heterogeneous reservoirs [Beckner and Song, 1995]. All reservoirs are naturally heterogeneous. It is desirable to have a method that can compute an optimal strategy considering not only the reservoir as it occurs in nature but also the economics related to the operations required to develop the field.

The present work investigated the optimization problem related to the development of reservoirs. One of the main issues was to determine how well placement can affect the profitability of the development project. The methods needed to handle any number of parameters to allow, for example, free choice of number of wells for a given rig capacity. Also, the reservoir model should account for any type of geometry and driving mechanism.

We first started by investigating a test set of mathematical functions to select the

best values for the optimization parameters.

This strategy allowed us to investigate the methods in preparation for a general case. Finally, a real case was considered: a full field reservoir analysis as commonly used in simulation studies. In this real problem the wells were allowed to be vertical or horizontal. The choice was addressed by the optimization method and is part of the solution. Drilling costs, production facilities costs, workover operation costs and other operation costs required to achieve the maximum recovery factor governed the best well placement and scheduling as well as the platform location. Wells were allowed to be placed anywhere in the reservoir domain, subjected to the real project constraints such as well scheduling and production capacity.

3.1 Economic Analysis

The optimization procedure requires that the function to be optimized be properly defined, considering the constraints and adequately representing the real problem.

In order to compare a heterogeneous group of parameters, we had to choose a reference applicable to different kinds of entities regardless of their nature. The cash flow (expressed as a net present value in US$) of the operations required to develop the reservoir was chosen. Hence, we were evaluating a problem concerning the minimization of the costs or maximization of the profits.

During the search, we considered that the maximum profit must be obtained at the end of the production time constrained by recovery factors. The cash flow analysis provided the value of the objective function for a specific combination of the decision parameter values considered for the optimization task.

So, each set of decision parameters $\vec{p} = (p_1, p_2, ..., p_n)$ was associated with a cash flow value C, or:

$$C = C(\vec{p}) \tag{3.1}$$

As we did not have a closed-form equation for this function due to the complexity of the problem, the function value was evaluated numerically by first generating the production data using the simulator and then subjecting this production to the cash flow analysis.

Relevant costs such as rig rental, drilling costs, facility costs, drilling platform cost and production cost represent together the local expenditure and the oil production sales represent the local income. After computing these values in the proper time scale, we correct each of them to the present value (Net Present Value), then we add them up to obtain the net cash flow value.

Figure 3.1 shows the expenditures (down arrows) and incomes (up arrows), expressed in US$, distributed along the *time* axis. To evaluate the net present value they must be corrected to the same reference time n considering the discount rate i.

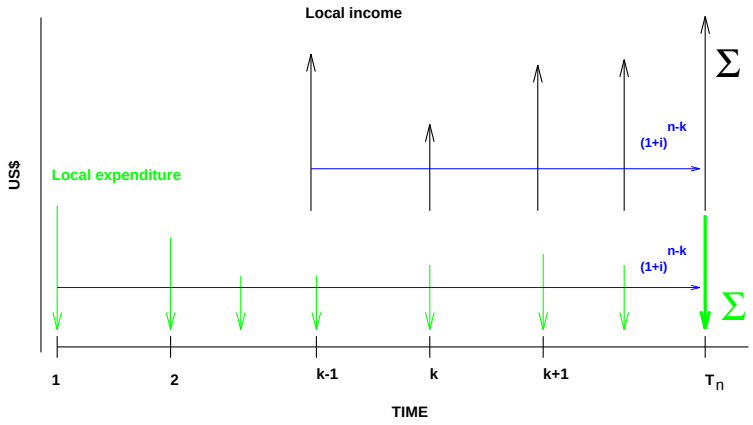

Figure 3.1: Cash flow analysis

From industry input we can approximate most of the unitary cost values as well as the oil price [Mannarino, 1991] as summarized in Table 3.1.

The values in Table 3.1 represent typical costs for a certain production level. The production level can vary and the values in Table 3.1 may become wrong estimates of the costs and prices for the problem. So, the economic model required further refinement.

Description	Value	Units
Rig Capacity	100	days/well
Interest per Year	10.00	percent
Fluids Handling Cost	1.00	US$/BBL
Oil Price	15.00	US$/BBL
Platform Cost	100,000,000.00	US$/15 wells
Rig Day Rate	70,000.00	US$/day
Rig Mobilization Cost	1,000,000.00	US$
Offshore Well Drilling Cost	1,400,000.00	US$/well/month
Facilities Initial Cost	400.00	US$/BPD
Insurance Cost	12,500.00	per month
Operational Cost per Well	5,000.00	US$/month

Table 3.1: Unitary Cost and Other Values (Typical)

3.2 Considered Costs

During the simulation each different set of parameter values generates different production profiles. We can expect that the maximum production rates of oil, gas and water will be different from set to set. Platforms or site preparation, production facilities, fluid transport and other installations are dimensioned based on the production level to be achieved. If this level changes from case to case then we must expect that the costs related to these entities will vary as well.

Kennedy [1993] demonstrated how these costs could be evaluated as well as their behavior according to several project parameters.

1. Onshore fields

   Costs = Camp + Development wells + Production Facilities + Transportation

   (a) Camp costs include:

   Base Cost = field production offices + warehouses + maintenance + buildings + helipads + living areas

   where:

Base Cost is given in millions of US dollars and depends on the existing infrastructure and accessibility to the production area (remoteness).

Required input data:

- maximum oil production rate
- remoteness

(b) Development well costs C_w include:

C_w = *(1 +* f_1 *+* f_2*)*(well site + intangible + tangible drilling costs)*

where:

f_1 = 0.20 if there is helicopter support or 0.0 otherwise,

f_2 = 0.15 (recommended)

well site depends of the terrain type and the reservoir depth

intangible drilling costs are equal to $n_w * n_d * (r_d + support + supervision)$

where:

n_w : number of wells;

n_d : number of drilling days per well;

r_d : rig day rate;

tangible drilling costs = casing + tubing costs

Required input data:

- terrain type
- reservoir depth
- country
- remoteness
- well depth

(c) Production Facilities Costs C_p include:

C_p = *Base cost ** f_1 *** f_2 *** f_3 *** f_4 *+ others*

where:

base cost is the facility costs for oil and gas production

f_i *factors* are defined as:

i. *Remoteness/Logistics Factor* f_1: this cost adjustment factor ranges from 1.0 to 2.0. It adjusts the project's proximity to existing infrastructure and transportation availability;

ii. *Construction Difficulty Factor* f_2: this factor ranges from 1.0 to 2.0. It calibrates for the difficulty confronted in facilities construction operations for various terrains;

iii. *Equipment Complexity Factor* f_3: this factor ranges from 1.0 to 1.75. Standards vary as to the level of complexity required for equipment specifications, instrumentation, data gathering and optional remote control. These standards may fluctuate from general industry guidelines and local regulations to a major oil company employing an Engineering/Procurement/Construction firm to assist in facilities design;

iv. *Enclosing Equipment Factor* f_4: this factor ranges from 1.5 to 2.0. This factor is necessary to adjust the costs of the facilities for the construction of enclosures to shelter the equipment from a harsh environment.

others include the disposition of natural gas, water treatment and water injection costs

Required input data:

- oil maximum flow rate
- gas maximum flow rate

(d) Transportation Costs C_t

C_t = *pipelines for oil and gas*

Required input data:

- region type
- oil maximum flow rate
- gas maximum flow rate

- pipeline length

2. Offshore fields

   Costs = Production systems + Development wells + Production Facilities + Transportation

   (a) Production system costs include:

      i. platforms: platform jacket, piles, module support frame (deck). Installation costs for all of the above and production module.

      ii. Submersible based floating production system (FPS), floating production storage and off-loading system (FPSO) and jackup production unit.

      iii. Satellite systems: auxiliaries for existing producing fields.

   Required input data:

   - system type
   - environment type
   - Jack payload

   (b) Development well costs C_w include:

   C_w = *intangible drilling costs + tangible drilling costs*

   where:

   intangible drilling costs are equal to $n_w * n_d * (r_d + support + supervision)$

   where:

   n_w : number of wells;

   n_d : number of drilling days per well;

   r_d : rig day rate;

   tangible drilling costs are the casing and tubing costs

   Required input data:

   - reservoir depth
   - rig type

- country
- remoteness
- exploration or development area

(c) Production Facilities Cost C_p

C_p = *Base cost* * f_1 * f_2 * f_3 + *other costs*

where:

base cost is the facility costs for oil and gas production

f_i *factors* are defined as:

i. *Equipment Specification Factor* f_1: this factor ranges from 1.0 to 1.85 and takes into account module enclosure option, location severity, instrument complexity, and optional electric pumps and compressors;

ii. *Construction Premium Factor* f_2: this factor ranges from 1.0 to 1.75 and accounts for the disparity in labor costs in the various regions involved in offshore construction;

iii. *Project Management Factor* f_3: this factor ranges from 1.0 to 1.15 and accounts for the fact that exists a project management premium for projects undertaken in various regions of the world.

others include treatment and compression of natural gas, sea water treatment for injection, living quarters and module hookup

Required input data:

- oil maximum flow rate
- gas maximum flow rate

(d) Transportation Cost C_t

C_t = *pipelines for oil and gas*

Required input data:

- region type
- oil maximum flow rate

- gas maximum flow rate
- pipeline length

Figures 3.2 to 3.7 show the average international costs estimated by Kennedy [1993] for the offshore case.

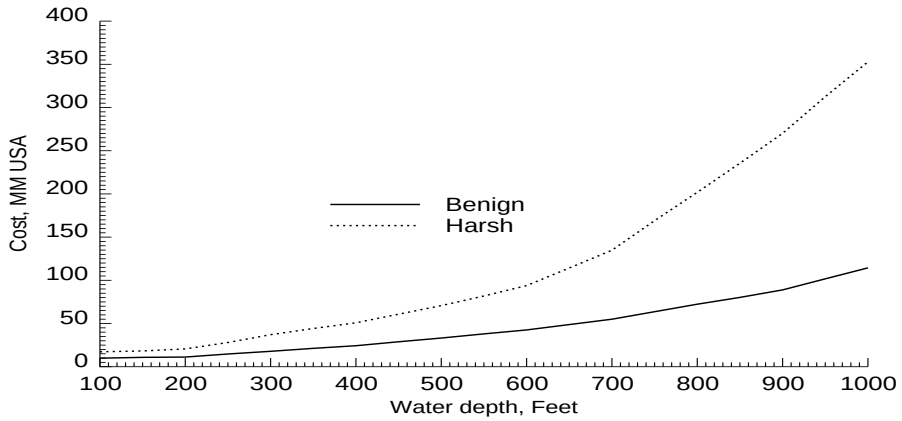

Figure 3.2: Platform cost (Jacket payload of 5000 short ton)

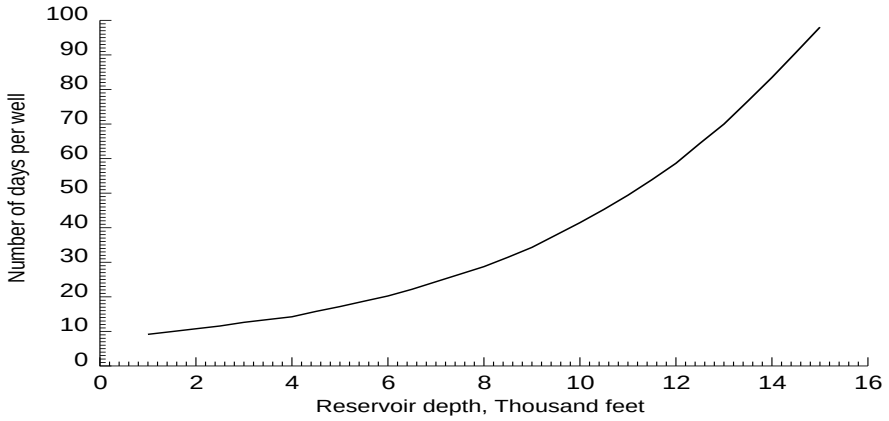

Figure 3.3: Drill days curve

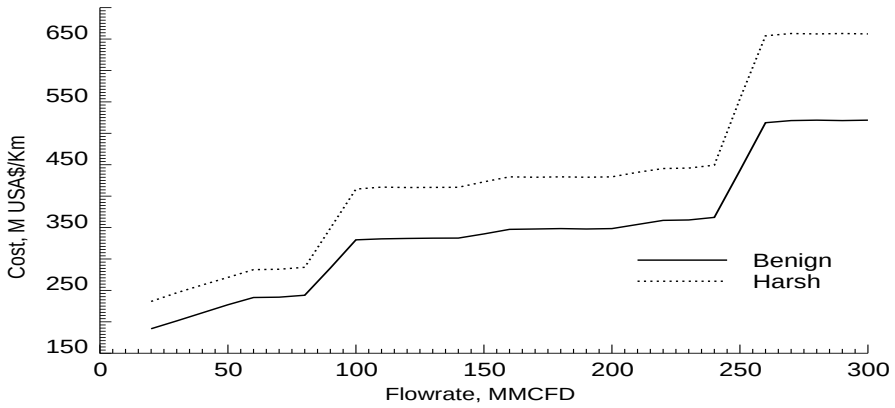

Figure 3.4: Gas pipeline cost

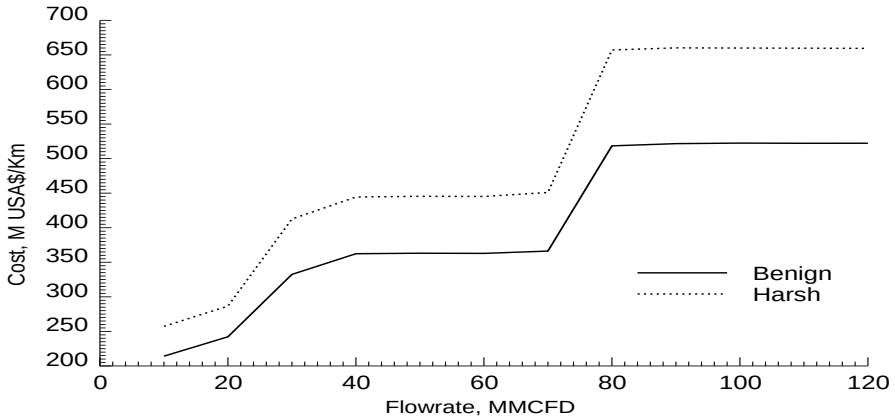

Figure 3.5: Oil pipeline cost

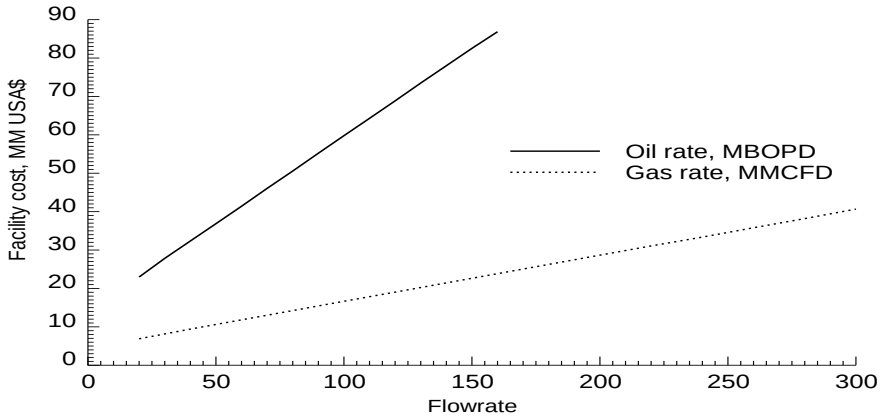

Figure 3.6: Oil and gas production module

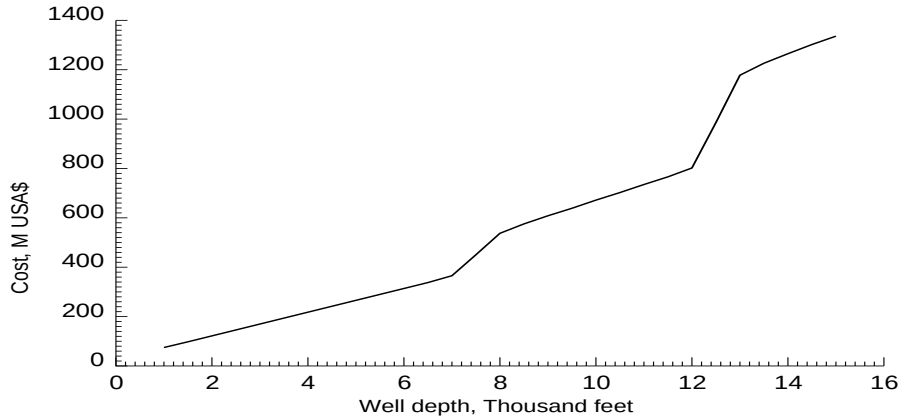

Figure 3.7: Well tubulars cost

3.3 The Simulator as a Data Generator

We investigated the solution of this problem using a commercial simulator as a data generator for the objective function. The production profile generated by the simulator was used to perform a cash flow analysis that returned the Net Present Value to the optimizing procedure.

Using a commercial simulator avoided the need to restrict problem size and complexity. It also allowed the objective function to take advantage of any resource offered by the software to better represent the requirements of the development plan. Some of these resources are a drilling queue of wells, rig scheduling and several different production controls.

The optimizing algorithms interfaced with the simulator generating the appropriate input data and retrieving the simulation results from temporary files. An economic analysis was then performed to calculate the cash flow for each set of parameters.

From the simulation we obtained the number of wells and the oil production during a given time interval. The objective function computation then set the rig allocation and the well placement, applied associated costs and handled time requirements.

Rig allocation required us to keep track of the already allocated rigs and to perform an economic analysis to check whether it was cheaper to drill one well immediately after another or to pay the rig mobilization cost for each well.

This approach required that the objective function evaluation algorithm interact with the simulator, generating the proper input file, dispatching the simulator run and recovering its results when the run ended. In this way, the values obtained from the simulation run could be considered in evaluating costs and placing them in the correct time scale. A more detailed description is presented in Chapter 6.

3.3.1 Data Transition to the Simulation Run

In order to use a commercial simulator the parameters passed from the optimization to the simulator must match the required format. Types such as integer and float were not enough to handle this problem. More types and special features were created to attend to special needs.

The main requirements were format and placement.

The *Format requirement* tells us how the parameter must be coded in the input deck. As in any program, parameter formats assume integer, float or string values. There are several ways of handling such a problem. The most convenient way is to have these proper formats and values handled by the optimizer code. In this way, the interface is freed from exhaustive translation work, reducing errors.

The *Placement requirement* tells us where the parameters must be placed. Many parameters depend on the value of others, e.g., well completions. One well completion parameter must depend on the well head location or on the previous completion, otherwise the well would be spread all over the reservoir. Other parameters are values applied to physical locations that are not parameters themselves but depend on the value of a real parameter. We named such false parameters as *carriers* since they carried the true parameter value to the proper location. For example, consider the case of a horizontal well location, with fixed length, completed along 20 grid blocks, assuming fixed direction. Each completion can be open or shut, these are the real parameters, but the block to be completed depends on the horizontal well head location. The block location is not a parameter, in this case, but the well head location is.

A template of the input deck was used in order to satisfy such requirements. This template carried all the data required by the simulator, excluding that of interest for the optimizer, as well as all the keywords to be used by the optimizer to parse the data into the final format.

Chapter 4

The Optimization Algorithms

In this chapter we describe the principal optimization issues considered in this work.

The algorithm investigated to solve the type of problem of interest works according to the following concepts:

The main approach is a GA environment in which all of the other methods were implemented. Together they help each other governed by genetic rules.

The optimizing procedure is responsible for generating the set of parameters and composing the string creatures (chromosomes) for the population to be used by the GA. The first population can be started either randomly or using Fang's algorithm to assure a good initial domain coverage. The chromosome is then decoded into the specified parameters and their values are passed to the simulator. An interface procedure generates the input data to the simulation run by reading a problem-specific template. Here, special keywords signal where to place the parameters. In this way, the procedure does not know which program is going to generate the data, freeing the procedure from any specific commercial code. The optimizer dispatches the run using a script file (in the UNIX system) that will handle all options to run the simulator properly. The retrieval procedure transfers the results in the proper format to the economic analysis module. This economic analysis module then returns the calculated function value to the optimizer. This cycle is repeated for every set of parameters being considered.

The polytope method may be used to generate new individuals in selected generations of the GA. These individuals can be accepted or not, depending on whether they improve or degrade the objective function value when compared against the evaluation of the other vertices. Tabu search and memory strategy approaches may also be applied to these new individuals, during their generation, forcing or not other domain regions to be visited.

The main idea was to develop a hybrid optimizing procedure, based on the procedures already described, but adapted to overcome their particular limitations. Any modified or hybrid algorithms need to be not only robust and provide fast convergence towards the optimum, but must also be able to fit a range of problems.

4.1 Implementing a Hybrid Algorithm

The main hybridization was the combination of a Genetic Algorithm search with the polytope method. Along the generations many function evaluations were required and it must be remembered that we are dealing with functions that are costly to evaluate. Each of the hybrid procedure components contributes to the overall method in one or more of the following ways:

- applying gradient effects to the search;
- reducing the number of function evaluations;
- avoiding premature convergence;

The *Polytope* search was implemented as a primary option for the selection procedure followed by the crossover option. An individual is generated by the polytope and if rejected by the acceptance criterium then a new individual was generated according to the GA selection and crossover procedures. Mutation may or may not be applied to the polytope generated individual. Both options were analyzed in this work. The polytope was allowed to move on the genetic (strings represented) grid only in order not to disturb the genetic environment substantially, although the algorithm encompasses a general case.

In an $n - dimensional$ search the polytope requires $n + 1$ vertices as described in Chapter 2. We can pick $n + 1$ vertices randomly, but we do not want to pick a polytope that does not improve the search. The number of possible polytopes to be chosen from a population of size N individuals in an $n - dimensional$ problem is given by:

$$P = C_{n+1}^{N} = \frac{N!}{(n+1)!(N-n-1)!} \tag{4.1}$$

This equation requires that the population size be greater than the dimension of the search space. So, for a population size exceeding the dimension of the search space by one we have just one single polytope to choose from. The number of possible polytopes increases drastically for larger population sizes. If the polytope succeeds in generating a new individual according to the criteria described below, the individual is placed in the new population. Otherwise, if the polytope fails, the pure GA is allowed to generate the new individual according to the genetic rules.

Two methods were tested to select the polytope vertices within a population. First, a function value ordered list without repeated individuals is built from the actual genetic population. Then the two methods work in different ways. *Method 1* determines n fixed vertices by taking the n best individuals from the sorted list. The $n + 1$ vertex is the one that varies whenever a polytope is to be tried. It is picked also from the ordered list, in the decreasing function value direction. *Method 2* picks the $n+1$ vertices from the ordered list, in the decreasing function value direction. As there are no fixed vertices, all the $n + 1$ vertices are taken sequentially from the list. For both methods the ordered list is considered circular, e.g., if the list reaches the end, then the selection restarts from the beginning of the list. Repeated polytopes will be discarded naturally by the algorithm.

These two methods try to reduce the number of polytopes to be considered in the search applying a *preferred direction* guideline. Because the list includes all the function evaluations, even the worst ones, the low fitness regions are still considered for the polytope formation in both methods because we never know if a bad location will be catapulted to a very good region, unreached by apparently better-formed polytopes. Many other methods could be considered in a future work.

Another issue that had to be addressed is the polytope existence and polytope improvement capacity. The first concerns the polytope degeneracy. To avoid degeneracy, all the polytopes were tested and only the fully $n+1$ dimensional ones accepted. The test of existence is based on the evaluation of the polytope's content given by:

$$\frac{1}{n!}\begin{vmatrix} x_{11} & x_{12} & \dots & x_{1n} & x_{1,n+1} \\ x_{21} & x_{22} & \dots & x_{2n} & x_{2,n+1} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nn} & x_{n,n+1} \\ 1 & 1 & \dots & 1 & 1 \end{vmatrix} \tag{4.2}$$

where x_{ij} are the coordinates i of vertex j [Kendall, 1961]. The effect of a degenerate polytope is to perform the search in a dimension lower than the one used for the original problem. It can throw the search to undesirable regions, spending time and function evaluations. The second issue concerns the polytope's efficiency. This criterion rejects the polytopes that cannot improve their own maximum function value (*internal efficiency*). Although a function evaluation was wasted here, it was felt to be better than wasting room in the population with weak individuals. In this way, the polytope was also submitted to genetic concepts and the pure GA was allowed to try to generate a stronger individual.

Finally, the matter that concerns the number of polytope movements is now discussed. The polytope is known to be a walker on an n-dimensional surface. The question now is: how long should the polytope to be allowed to walk in a hybrid algorithm? To answer this question we must go back to the principles of the hybrid algorithm. The main algorithm is a GA search that carries genetic material along the generations. The polytope's main role is to add gradient effects to increase the search efficiency. We do not want the GA to become a polytope launching base and then allow the several polytopes to find the optimum. Why not? Because the polytope can miss regions of the search space and nothing assures that the same region is not being visited by different polytopes at different times. The polytope should not destroy the GA strategy. The polytope already disturbs the genetic material working much like a strong mutation operator. The difference here is that mutation can generate unpredictable parameter values and the polytope tries to apply some sense to the search

through the gradient effect. The maximum number of movements beyond which the polytope would overcome the GA material is still unknown and it clearly depends on the type of the function to be optimized. To keep things simple we allowed just one polytope movement with no further expansion or contraction. So, the polytope has just one chance to improve its own maximum function value.

Memory Strategy is a technique introduced in this work that uses a long term memory where each coordinate, or individual (chromosome) for GA, has a neighborhood of variable n-dimensional radius. Any new individual placed within this neighborhood assumed the already evaluated function value. When the number of individuals coexisting in the neighborhood reached a specified limit the radius was decreased and the function was evaluated for the individuals that fell outside the reduced neighborhood. This technique allowed us to save function evaluations and to detail the domain only when it was required, shaping the multidimensional surface gradually along the generations. The initial set of coordinates (initial population for GA) can be chosen according to the Fang algorithm to better cover the entire feasible domain.

The *Tabu Search* was implemented through the use of counters associated to each memory cell. Every time that a location (affected by its radius) was revisited, its counter was incremented. If there was a tabu then the new individual was refused and another created, pushing the search towards regions not yet visited.

The hybrid algorithm attempts to use the best feature of each method to speed up the search and to free the procedure from limitations of the individual approaches.

4.1.1 Disruption by the Polytope

The *Schema Theorem* to be presented in this section states how disruption can destroy the evolution of the genetic material. The main concern of this theorem is to describe how the algorithm efficiency depends on the preservation of the string patterns of high fitness.

A schema (plural, *schemata*) [Holland, 1975] is a similarity template describing a subset of strings with similarities at certain string positions. If we consider the

strings formed by the elements of the binary alphabet {0,1} then each position (*bit*) can have the allele (bit value) 0 or 1. We assign the symbol * or *don't care* symbol to this particular situation. So, based in the ternary alphabet {0,1,*}, we can form strings with fixed bit values (0 or 1) mixed with the *don't care* value *. These strings are pattern matching devices or schemata. Now, consider the strings and schemata of length three. The schema **1 matches the strings 001, 011, 101 and 111. The schema 0*1 matches the strings 001 and 011. So, a schema matches a particular string if at every location in the schema a 1 matches a 1 in the string, a 0 matches a 0, or a * matches either [Goldberg, 1989]. For an alphabet of cardinality (number of alphabet characters) k the total number of schemata with length l is $(k+1)^l$.

The order of a schema H, denoted by $o(H)$, is the number of fixed positions (positions with only 0 or 1). The defining length of a schema H, denoted by $\delta(H)$, is the distance between the first and last fixed positions. For example, the schema *0***10*110** has order six and defining length nine. And the schema **1* has order one and defining length zero.

Disruption is related to the fact that after the crossover and mutation operations the children produced may not be members of the same schema as their parents. The survival of a schema will depend on how disruptive an operator can be.

Holland [1995] showed that the expected number of individuals allocated to a particular schema H of order *o(H)* at time (or generation) *t+1* is given by:

$$m(H, t+1) \geq m(H, t) \cdot \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1} - o(H) p_m \right] \tag{4.3}$$

where $f(H)$ is the average function value of the current individuals allocated to H, $\bar{f}$ is the average population function value, $1 - p_c \frac{\delta(H)}{l-1}$ is the crossover survival probability and $o(H)p_m$ is the probability of surviving mutation. The main assumptions for Equation 4.3 are that the individuals are represented by fixed-length binary strings, the crossover points can occur with equal probability between any two adjacent bits, there are enough individuals to provide reliable estimates of schema fitness, and that the effects of crossover and mutation are not too disruptive [Spears and De Jong, 1994]. Equation 4.3 is known as the *Schema Theorem* and states that short, low-order, above-average schemata (called *building blocks*) receive exponentially increasing trials in the

subsequent generations. All this goes in parallel with no special memory than the population of n strings. In this way, the genetic algorithm can investigate several individuals that satisfy a specific schema H. The number of schemata that are usefully processed in each generation is of the order of n^3, which compares favorably with the number of function evaluations n. This processing leverage is called *implicit parallelism*.

When disruption occurs it affects all the points discussed so far in this section. Now, we can understand why the analysis of disruption avoidance is so important in the genetic algorithm, even though disruption itself is not forbidden.

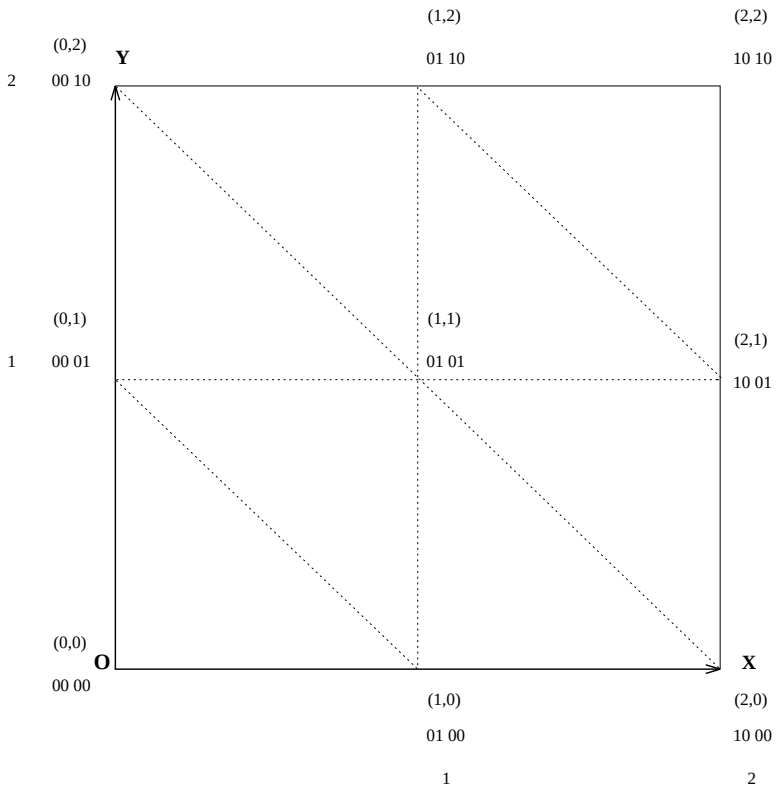

Figure 4.1: Integer and genetic grid

Each polytope vertex is considered to be a genetic individual, composed of a binary code of all parameters together, as explained previously. The polytope movement is performed in the real plane where the vertex with lowest function value (L) flips about the centroid obtained from the highest function value vertex (H) and the near highest

one (NH). Consider the two-parameter problem in the integer domain [0,2]. The grid has nodes (x, y) as shown in Figure 4.1. If we express the nodes considering the genetic coding then we get strings of the form $(b_x b_y)$ where b_x and b_y are the binary coding of x and y put side by side, we can call these grid points *chromosome grid nodes*. The polytope moves to a new or resulting point (R) in the real plane according to Equation 2.3. If we consider only reflection (no expansion or contraction) we can easily see that this real plane operation, for integer values, is equivalent to the binary operator

$$R = H + NH - L \tag{4.4}$$

performed in the binary plane, where + and - are binary addition and subtraction operations, respectively. This operation holds true for any three chromosome grid nodes.

Consider now the three chromosome grid nodes 01101011, 10111010 and 1001 as being the highest (H), near highest (NH) and lowest (L) polytope vertices, respectively. The resulting node (R), after the binary operation given by Equation 4.4, is 100011100 (Figure 4.2). As we are interested in the above-average schemata, the polytope is expected to improve the function value of the nodes H and NH whose schema is **1*101. But the resulting node R is not a member of such a schema. The polytope turned out to be a highly disruptive operator in the genetic environment. How should we see this fact in the context of the *Schema Theorem*? How could the polytope move increase the search efficiency? We must first recall that the polytope applies a gradient effect to the search and what the polytope is showing is that another schema can provide better function values than the current parents' schemata. The polytope directs the search switching to better performance schemata. If we decide to keep the search in the parents' schema we can force the resulting string R to become a member of such a schema. This is shown in Figure 4.2 where the resulting string R is now 100111010. If we accept this disruption in favor of better performance we can consider the polytope move as being a disruptive mutation operator with directed results although it acts in the selection and reproduction procedures. If the resulting string R does not improve the parents' function values (the resulting

schema has lower performance) then it is not worthy to disrupt the current schema. In this case the resulting string R is refused and the classical GA is allowed to generate new individuals. The polytope search is submitted to the genetic rules and both algorithms coexist to help each other.

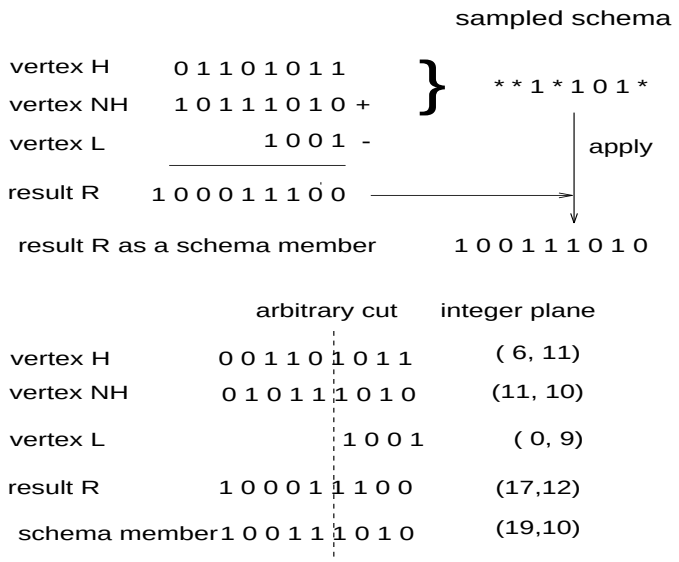

Figure 4.2: Binary and integer operations compared

The probability p_p that the polytope will be disruptive depends on the schemata involved and on the binary operations themselves. Each string can satisfy several schemata but a combination of two strings reduce this number substantially. All this points to a highly disruptive procedure, but the advantage of this is the fitness improvement. The schema survival probability under the polytope generation is $(1 - p_p)$ and can equal 1.0 if we force the resulting string to be a member of the parents' schema. Let p_i be the probability that the polytope will improve the function values of polytope's vertices (parents). As the polytope operator is clearly independent of

the reproduction, crossover and mutation operations, and considering that the above-average individuals are the best vertices in the polytope, the *Schema Theorem* can be estimated as

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\bar{f}} \left\{ \left[1 - p_c \frac{\delta(H)}{l-1} - o(H)p_m \right] (1-p_i) + (1-p_p)p_i \right\} \quad (4.5)$$

This shows a small increase in the lower bound on the right hand side of the equation decreasing the number of individuals satisfying schema H at time $t+1$ but with expected better performance. In the case of fitness improvement being obtained by switching to a schema other than the parents' one, a new relation based on polytope implementation of the selection operator must be developed.

Considering now the chromosome grid nodes H, NH, L and R we can make an arbitrary cut in the strings and map the two substrings in the integer plane (Figure 4.2). The coordinates obtained in the integer plane match the ones obtained if Equation 2.3 was used, as expected. It is also shown that the coordinates of the individual is a member of the parents' schema.

4.1.2 The Single-point Crossover Operator

The single-point crossover was implemented by Goldberg [1989] using the following procedure:

1. Sample the crossover probability to determine if the chromosome is to be submitted to the crossover operation;

2. If yes, determine the crossover point randomly taking into account the length of chromosome minus one. This is done to assure that, once the crossover must happen, the crossover point not be equal to the chromosome length, preserving it, instead of cutting it. Proceed with the crossover operation. In this way, the probability of a crossover point to fall in any chromosome location is:

$$\frac{p_{cross}}{(chromosome\ length) - 1} \quad (4.6)$$

3. If not, the chromosome is preserved and the children are perfect copies of the parents.

We also implemented this operator in a different way, which we called *fixed crossover rate*. The implementation is as follows:

1. Determine the crossover point randomly taking into account the full length of chromosome. In this way, the probability of a crossover point to fall in any chromosome location is:

$$\frac{1}{chromosome\ length} \tag{4.7}$$

   In this way, the chromosome can be preserved or not based on this single step;

2. Proceed with the crossover operation, if required.

This second method is the same as considering the crossover probability equal to 1.0 but taking into account the full chromosome length. In this way, the crossover operator may or may not preserve the entire chromosome in a single step. As the crossover probability approaches unity both methods tend toward the same value for all chromosome lengths. They differ substantially for probabilities less than 0.1. The second method applies higher rates to the crossover process.

4.2 Interfacing With the Simulator

Production data required for the cash flow analysis is generated by simulation. Each candidate set of parameter values must be provided as input to the simulator. This process may become a limiting factor depending on the type of parameters. The computation can become dependent on a specific reservoir simulator and also can become nonfunctional if a new version of the simulator changes format of its input.

The parameters considered in an optimization step are, somehow, constructed as an input file to be passed to the simulator. After the simulation run the results are retrieved into a table which includes such variables as: $time$, Q_o, Q_g, Q_w, N_p, G_p

and W_p. From this information the optimization algorithm can infer the values for decision variables such as *Number of producing wells* and *Total number of wells* at any time. Then the algorithm can set the rig allocation to drill such wells and specify the rig schedule considering economic factors as described in the next section.

At the end of the series of simulations, we developed a set of solutions containing production data, well and rig schedules and all incomes and outcomes properly placed in time. The cash flow analysis module was then used to compute the final objective function for each case.

4.3 The Stopping Criteria

In a multidimensional problem it can be very complex to define a stopping criterion. In a hybrid algorithm handling multidimensional problem this complexity is increased significantly.

To simplify this task, we defined a set of stopping criteria. The search stops when at least one of them is satisfied:

- The maximum allowable number of generations is reached;
- The fitness reaches a target value within a specified tolerance;
- The maximum number of function evaluations is reached;
- No more improvement in the function value is obtained after a specified number of generations.

Chapter 5

Results

In Chapter 2 we described several methods used to solve different optimization problems. Each of them uses different simplifications. Based on results obtained by earlier authors, for example, Nystad [1985b] it was noted that it is generally not a good strategy to rely on only one method.

A previous work [Bittencourt, 1994] focused on the polytope method alone and found that the search is very sensitive to the initial polytope size and location. These two parameters can determine the degree of success achieved by the method. As the optimal region is not known, the polytope could never reach the optimal if launched from a bad region.

It seemed to be more reasonable to take advantage of the main features of each algorithm and allow the solution to emerge naturally from the one most suitable for a specific problem.

As we did not know the behavior of the hybrid optimization algorithm on a multidimensional function in advance we started analyzing simpler two-dimensional test functions.

5.1 Test Functions

The test functions were purely mathematical functions that represent both easy and difficult problems in two-dimensional space. The main parameters investigated in the

hybrid algorithms were the following:

Maximum number of generations;

Population size. Same as chromosome length [Goldberg, 1989]. Due to the polytope performance this population size was reasonable although a conventional GA could require a population size of [Goldberg et al., 1992]:

$$n = O[\frac{l}{k}X^k] \tag{5.1}$$

where $O[\ ]$ is the *order of* function, l is the chromosome length and k is the schema size, X is the number of characters in the GA alphabet (2, for binary coding);

Initial population type. The initial population can be generated randomly (0) or using Fang's algorithm (1);

Selection type. The *Roulette Wheel* (0) or *Stochastic Universal Sampling* (1) can be used to select a pair of mates;

Crossover type can be fixed (-1), uniform (0) or single-point crossover (1);

The Elitism parameter represents whether the best individual of the previous generation is to be saved in the next generation (1) or not (0);

Tabu flags the use of TS off (0) or on (1)

Polytope activity determines if the polytope technique is not active (0), is always active (1) or is active every other generation (-2);

Polytope type chooses between method 1 or method 2 as described in Chapter 4.

Polytope movements sets the maximum number of movements;

Polytope existence flags if the polytope must have a content (0) or can be degenerate (1);

Polytope acceptance flags if the polytope is to be accepted only if it improves its own function values (0) or anyway (1);

Polytope mutation flags if the individual generated by the polytope must be submitted to mutation (1) or not (0);

Crossover probability. Varies between 0 and 1.

Mutation probability. Varies between 0 and 1.

From the results obtained for the mathematical test functions we were able to select the most appropriate values for the hybrid parameters to be applied to multidimensional petroleum engineering problems. These are field examples of practical significance where the relevant difficulties of real problems were well represented.

5.1.1 Mathematical Functions

Three different two-dimensional functions were used to test the hybrid algorithms. They represent different degrees of difficulty.

Each *experiment* was a complete optimization, using a specific combination of the variable hybrid parameters. Because the GA process is based on a random seed value each experiment was run 50 times for different random values of the random seed. So, each experiment encompassed 50 *subexperiments*. The maximum allowed number of generations was 200, the maximum allowed number of function evaluations was 5000 and the tolerance to consider the optimal value found was specified for each case. The optimal known value is one that occurs in the *genetic grid*. Because of the binary encoding the real domain becomes discrete when mapped onto the genetic grid domain. The hybrid parameter *parameter string length* determines the resolution of the genetic grid. In these test functions we chose the resolution first and then calculated the parameter string length required to satisfy such a resolution.

A subexperiment was considered successful if it reached the optimal value satisfying the number of generations and function evaluation restrictions. An *index of measure of performance* was associated with each experiment representing the number of successful subexperiments divided by the total number of subexperiments, 50

in our case. So, this index varied between 0 and 1, with 0 being an absolute failure and 1 an absolute success. Other important information such as the number of the generation in which the optimal value was reached, the number of function evaluations consumed to reach the optimal value, the value of the optimum found, and the variation of the function values in the optimal population were saved. The memory feature was activated in all tests using a radius that matched the genetic grid. In this way, we avoided the chance of the polytope not landing on a grid node. If this happened the polytope vertex location was shifted to the closest genetic grid node. This means that only new locations were counted in the number of function evaluations.

Because of the large number of possible combinations of the hybrid parameters, we tried to reduce the computer work by fixing or limiting some of them. Table 5.1 shows the fixed value for selected parameters. These values were chosen based on results obtained from preliminary tests.

Hybrid Parameter	Value
Crossover types	-1, 0, 1
Selection type	0
Polytope movements	1
Polytope existence	0

Table 5.1: Fixed hybrid parameters

Table 5.2 shows selected values for other hybrid parameters. This selection was based on discussions found in Goldberg [1989]. After the best results were obtained, we then tried to improve them, allowing selected hybrid parameters to vary. This was the case for the Tabu parameter. It was left out in the primary runs to reduce computer work, but was used later in the improvement process.

So, the hybrid parameters assumed the values shown in Table 5.3 during the experiments.

The total number of combinations produced 3740 experiments of 50 subexperiments each. The pure GA ran 220 experiments and the hybrid GA, 3520.

The crossover probability associated with the fixed type crossover (-1) is shown

Hybrid Parameter	Values
Crossover probability	0, 0.15, 0.35, 0.60, 1.0
Mutation probability	0, 0.02, 0.10, 0.25, 0.5

Table 5.2: Selected values for hybrid parameters

Hybrid Parameter	Possible values	Total
Initial Population	Random or Fang	2
Crossover type	Random, Uniform or 1-p	3
Elitism	Yes (1) or No (0)	2
Polytope activity	Off (0), Continuous (1) or Alt (-2)	3
Polytope type	1 or 2	2
Polytope internal effic	0 or 1	2
Polytope mutation	Yes (1) or No (0)	2
Crossover probability	0, 0.15, 0.35, 0.60, 1.0	5
Mutation probability	0, 0.02, 0.10, 0.25, 0.5	5

Table 5.3: Possible values for the hybrid parameters

in the tables as being equal to *1/chromosome length*, but the effective rate is 1.0. Although the fixed type crossover should equal the single-point when the rate is 1.0 we noted a different performance between them.

Agnesi Function

This unimodal function is described by the equation:

$$f(x,y) = \frac{64a^6}{(x^2+4a^2)(y^2+4a^2)} \tag{5.2}$$

The function has its optimum at $(x, y) = (0,0)$ with an optimal value equal to 100 (Figure 5.1). The tolerance for this case was 1%. The feasible domain was [-100, 100] for both independent variables and a resolution of 1.0 required a parameter string length of eight bits to cover the entire domain.

A population size equal to the chromosome length of 16 was assumed. Figure

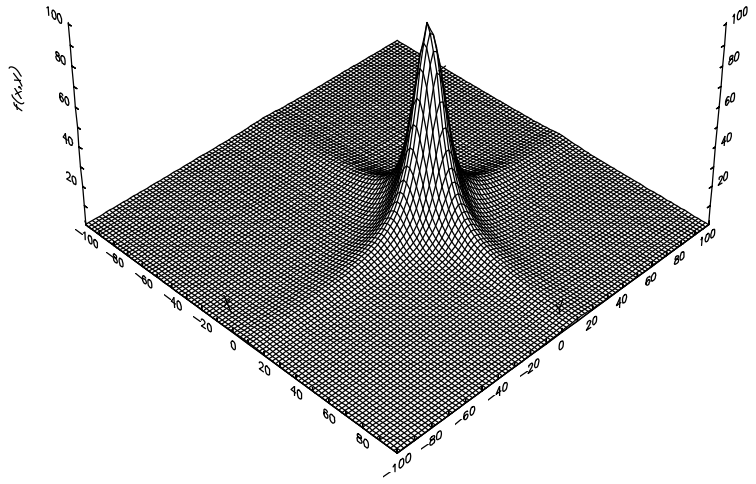

Figure 5.1: Agnesi function

5.2 shows the results obtained for the pure GA case. The top left plot shows all experiments for the pure GA algorithm. The remaining plots in the figure were obtained filtering out all experiments that did not produce the highest performance index, 1.0 in this case. All these plots must be considered simultaneously with the other values of measure shown in Table 5.4.

Out of the 220 experiments, 57 produced a performance index equal to 1.00 (26%), and 27 failed absolutely (12%). The range [0.5, 1.0] was achieved by 63 experiments (29%). Other values of measure are also shown in Table 5.4. Observing the top left plot in Figure 5.2 along with the information available in the database generated by all experiments (not shown) we observed that the low performance index of some experiments was due mainly to premature convergence where even high mutation rates were not effective. The mutation rate of 0.0 caused the lowest performance indices (below 0.15) while the rate of 0.50 produced those of 0.15 to 0.35. The midrange performance of 0.40 to 0.80 was generated mainly by mutation rates of 0.10 and 0.25, with a predominant uniform crossover and a kind of high-low balance between the crossover and mutation probabilities. The upper portion, from 0.97 to 1.00, was produced mainly by mutation rates of 0.02 and 0.10, with a predominance of fixed and single-point crossover. So, for this function, the low mutation rates associated

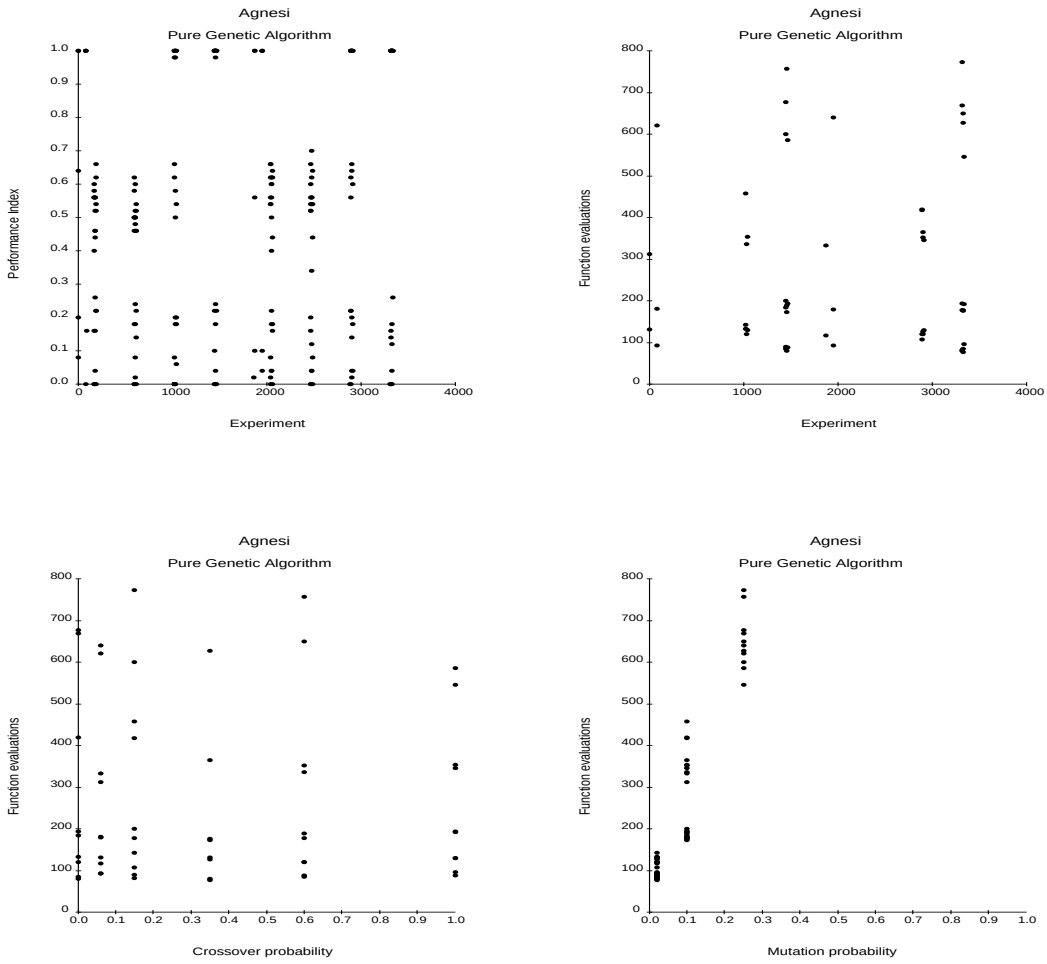

Figure 5.2: Pure GA performance (Agnesi)

with fixed and single-point crossover worked better. The other hybrid parameter influences varied substantially depending on the values of the hybrid parameters in the combination set.

Looking at the top right plot in Figure 5.2 we observe that the most efficient experiment consumed 78 function evaluations (on average) to find the optimum. The major number of best performers used single-point crossover as we can observe in Table 5.4. The bottom left plot shows no strong influence of the crossover probability. The bottom right plot of Figure 5.2 shows that the lowest mutation rate (excluding

Exper #	Initial pop	Xover type	Save best	Xover prob	Mut prob	Opt gen	Func eval	Perf index
2	0	-1	0	0.063	0.02	32	132	1.00
3	0	-1	0	0.063	0.10	24	312	1.00
87	0	-1	1	0.063	0.02	17	94	1.00
88	0	-1	1	0.063	0.10	13	182	1.00
89	0	-1	1	0.063	0.25	41	622	1.00
1022	0	1	0	0.0	0.02	44	133	1.00
1027	0	1	0	0.1	0.02	44	143	1.00
1028	0	1	0	0.1	0.10	41	459	1.00
1032	0	1	0	0.3	0.02	36	132	1.00
1037	0	1	0	0.6	0.02	30	120	1.00
1038	0	1	0	0.6	0.10	27	336	1.00
1042	0	1	0	1.0	0.02	29	131	1.00
1043	0	1	0	1.0	0.10	29	354	1.00
1447	0	1	1	0.0	0.02	22	86	1.00
1448	0	1	1	0.0	0.10	15	185	1.00
1449	0	1	1	0.0	0.25	45	678	1.00
1452	0	1	1	0.1	0.02	23	91	1.00
1453	0	1	1	0.1	0.10	16	200	1.00
1454	0	1	1	0.1	0.25	39	600	1.00
1457	0	1	1	0.3	0.02	18	81	1.00
1458	0	1	1	0.3	0.10	14	174	1.00
1462	0	1	1	0.6	0.02	18	89	1.00
1463	0	1	1	0.6	0.10	15	189	1.00
1464	0	1	1	0.6	0.25	50	757	1.00
1467	0	1	1	1.0	0.02	15	88	1.00
1468	0	1	1	1.0	0.10	14	194	1.00
1469	0	1	1	1.0	0.25	38	587	1.00
1872	1	-1	0	0.063	0.02	22	118	1.00
1873	1	-1	0	0.063	0.10	26	333	1.00

Table 5.4: Best performers for GA (Agnesi)(to be continued)

zero) was the one that required least computer work to find the optimum. Elitism, Fang's algorithm and single-point crossover along with small mutation rate and moderate crossover rate were the hybrid parameters of major influence to obtain the best results.

Figure 5.3 shows the results obtained for the hybrid (HGA) case. The top left plot shows all experiments. The remaining plots in the figure were obtained filtering out all experiments with a performance index lower than the highest performance index

Exper #	Initial pop	Xover type	Save best	Xover prob	Mut prob	Opt gen	Func eval	Perf index
1957	1	-1	1	0.063	0.02	14	93	1.00
1958	1	-1	1	0.063	0.10	13	180	1.00
1959	1	-1	1	0.063	0.25	42	640	1.00
2892	1	1	0	0.0	0.02	36	120	1.00
2893	1	1	0	0.0	0.10	37	420	1.00
2897	1	1	0	0.1	0.02	28	108	1.00
2898	1	1	0	0.1	0.10	37	419	1.00
2902	1	1	0	0.3	0.02	31	127	1.00
2903	1	1	0	0.3	0.10	30	365	1.00
2907	1	1	0	0.6	0.02	27	120	1.00
2908	1	1	0	0.6	0.10	29	352	1.00
2912	1	1	0	1.0	0.02	26	131	1.00
2913	1	1	0	1.0	0.10	28	346	1.00
3317	1	1	1	0.0	0.02	21	81	1.00
3318	1	1	1	0.0	0.10	16	194	1.00
3319	1	1	1	0.0	0.25	44	670	1.00
3322	1	1	1	0.1	0.02	19	83	1.00
3323	1	1	1	0.1	0.10	14	179	1.00
3324	1	1	1	0.1	0.25	52	773	1.00
3327	1	1	1	0.3	0.02	16	78	1.00
3328	1	1	1	0.3	0.10	14	177	1.00
3329	1	1	1	0.3	0.25	41	628	1.00
3332	1	1	1	0.6	0.02	15	85	1.00
3333	1	1	1	0.6	0.10	14	179	1.00
3334	1	1	1	0.6	0.25	42	650	1.00
3337	1	1	1	1.00	0.02	15	96	1.00
3338	1	1	1	1.00	0.10	14	193	1.00
3339	1	1	1	1.0	0.25	35	546	1.00

Table 5.4: (continued) Best performers for GA (Agnesi)

obtained by the pure GA, e.g., 1.0. Unlike the pure GA, we observed that the HGA produced performance indices covering the full range [0,1]. Out of 3520 experiments, 1135 succeeded absolutely (32%) and 207 failed absolutely (6%). The range [0.5, 1.0] was achieved by 815 experiments (23%). These numbers reveal that many different combinations performed absolutely well. Also, in the performance index range [0.7, 0.97] the pure GA could not produce one single result.

Selecting the experiments that consumed less than 78 function evaluations from the top right plot in Figure 5.3, we obtained the five that are shown in Table 5.5.

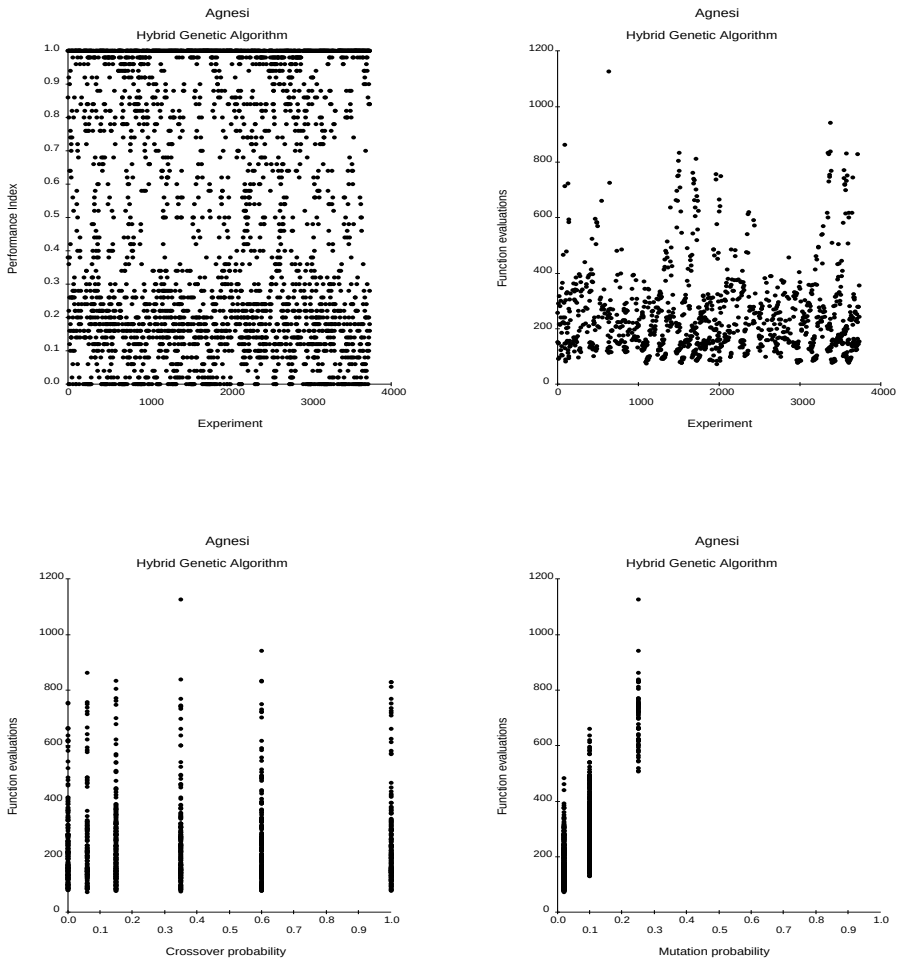

Figure 5.3: HGA performance (Agnesi)

We noted that, for this type of function, the polytope activity constant over all generations applies a strong influence in the search process. The best performance was achieved with Fang's algorithm, elitism, fixed crossover, polytope active along all the generations, polytope type 1, accepting any individual generated by the polytope without submitting it to the mutation operator and small crossover and mutation rates. The top two plots in Figure 5.4 show the performance index against the mutation rate for both GA and HGA algorithms. We noted an improvement to the search process for every value of mutation rate. The bottom two plots in the

Exper #	Init pop	Xover type	Save best	Poly actv	Poly type	Poly Effc	Poly mut	Xover prob	Mut prob	Opt gen	Func eval
1107	0	1	0	1	1	1	0	0.35	0.02	14	76
1537	0	1	1	1	1	1	0	0.60	0.02	17	77
1972	1	-1	1	1	1	1	0	0.06	0.02	17	74
3397	1	1	1	1	1	1	0	0.15	0.02	19	77
3607	1	1	1	-2	1	1	0	0.60	0.02	14	77

Table 5.5: Best performers for HGA (Agnesi)

same figure show the performances achieved when the individual generated by the polytope was not submitted to the mutation operator (bottom left) and when it was (bottom right).

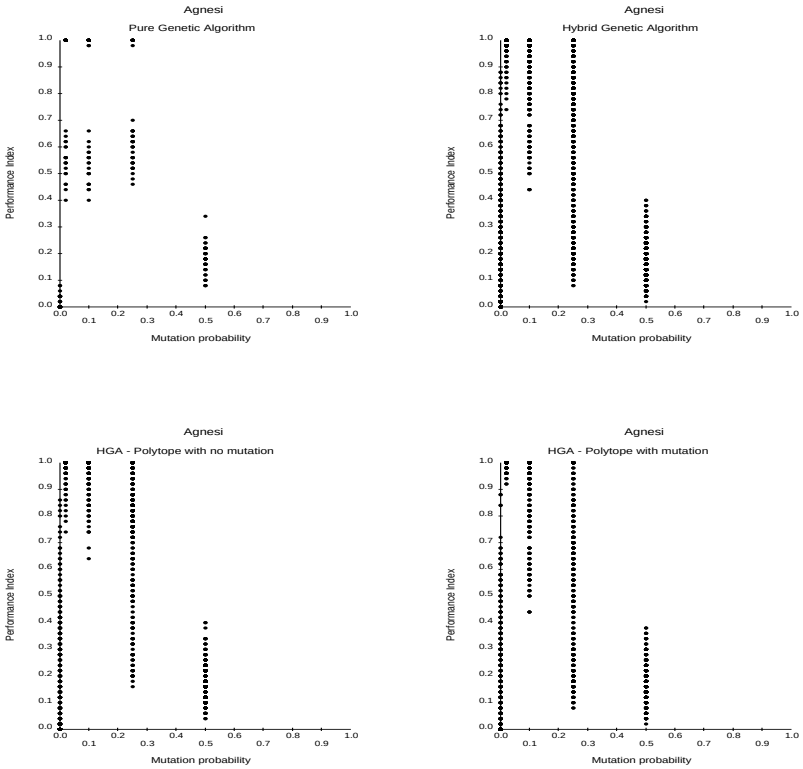

Figure 5.4: GA and HGA compared (Agnesi)

The performance indices, for both GA and HGA algorithms, were analyzed according the variation of the various genetic parameters, as shown in Figures 5.5 to 5.15 and discussed below. From this analysis we could obtain the most significant genetic parameter values for this type of function.

For the pure GA algorithm Figure 5.5 shows that the average number of function evaluations spent to generate successful experiments varied from 220.25 to 305.5 and so the influence of the crossover probability was shown to be very weak. Figure 5.6 shows a better performance when the Fang's algorithm and fixed or single-point crossover methods were used. Elitism performed better according to the plots in Figure 5.7. The crossover probability did not play an important role as is shown in Figure 5.8. The mutation probabilities that performed better were 0.02 and 0.10 while the values of 0.0 and 0.5 could not produce a single successful experiment (Figure 5.9).

For the hybrid HGA algorithm Figure 5.10 shows that the average number of function evaluations spent to generate successful experiments varied from 237.36 to 259.84. Although this range is narrower than the one for the pure GA approach it does not show a strong influence of the crossover probability. Figure 5.11 does not show a significant performance improvement when Fang's algorithm was used. The uniform crossover method had a significant performance improvement achieving almost the same performance as fixed and single-point crossover methods. This resulted in a weak influence of the crossover type. Elitism performed better according to the two top plots in Figure 5.12. The remaining plots in this figure show no influence of the type of polytope activity but the polytope type 1 performed better than the type 2. Figure 5.13 shows a slightly better performance when the individual generated by the polytope is accepted only if it improves the polytope fitness and when no mutation is applied to this new individual. The crossover probability does not show a significant influence according to the Figure 5.14. However, the mutation probability played an important role when successful experiments were generated. Low values such as 0.02 and 0.10 produced the best performance while the values of 0.0 and 0.5 could not produce a single successful experiment (Figure 5.15).

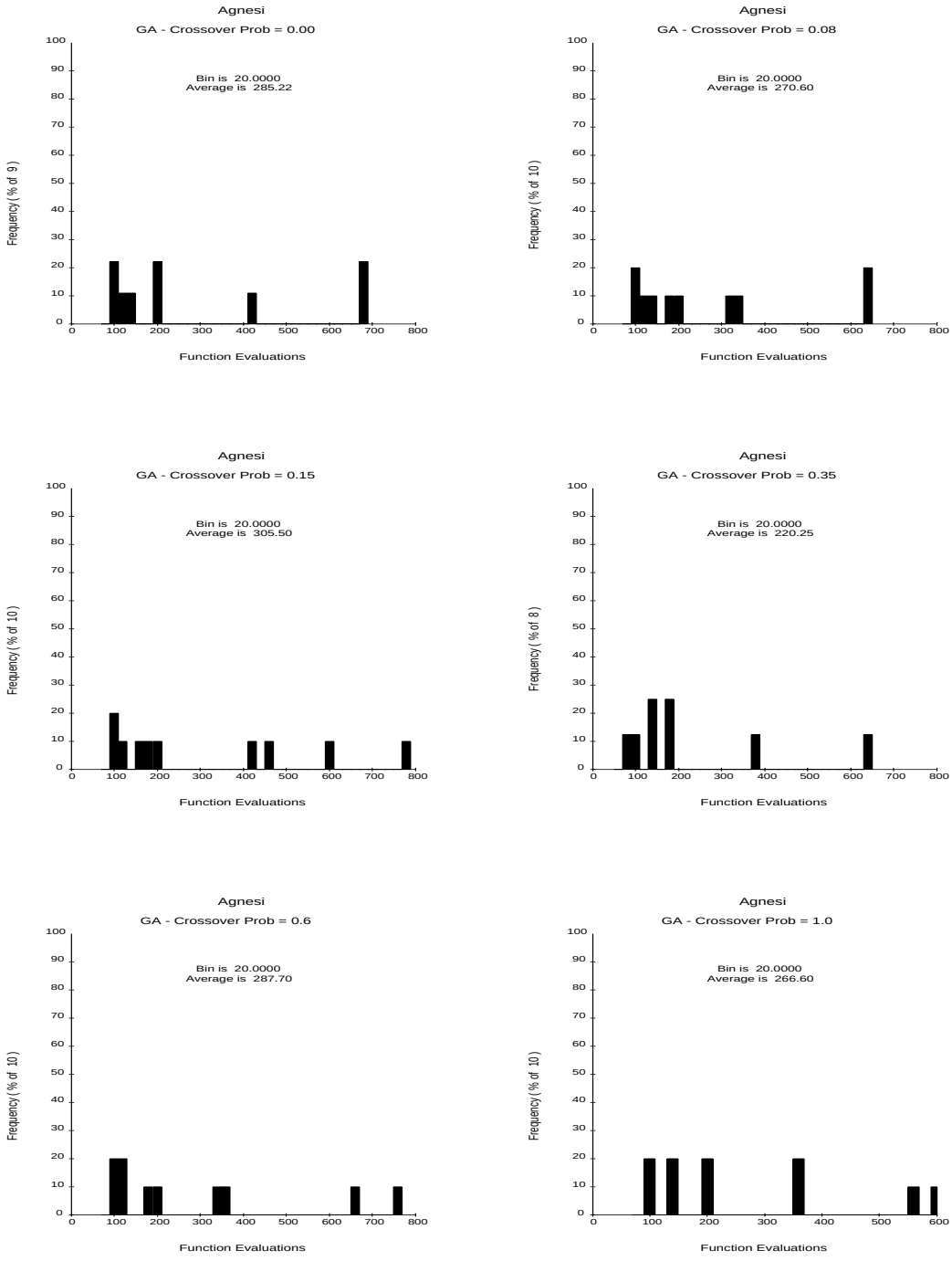

Figure 5.5: Crossover probabilities for successful experiments (Agnesi)

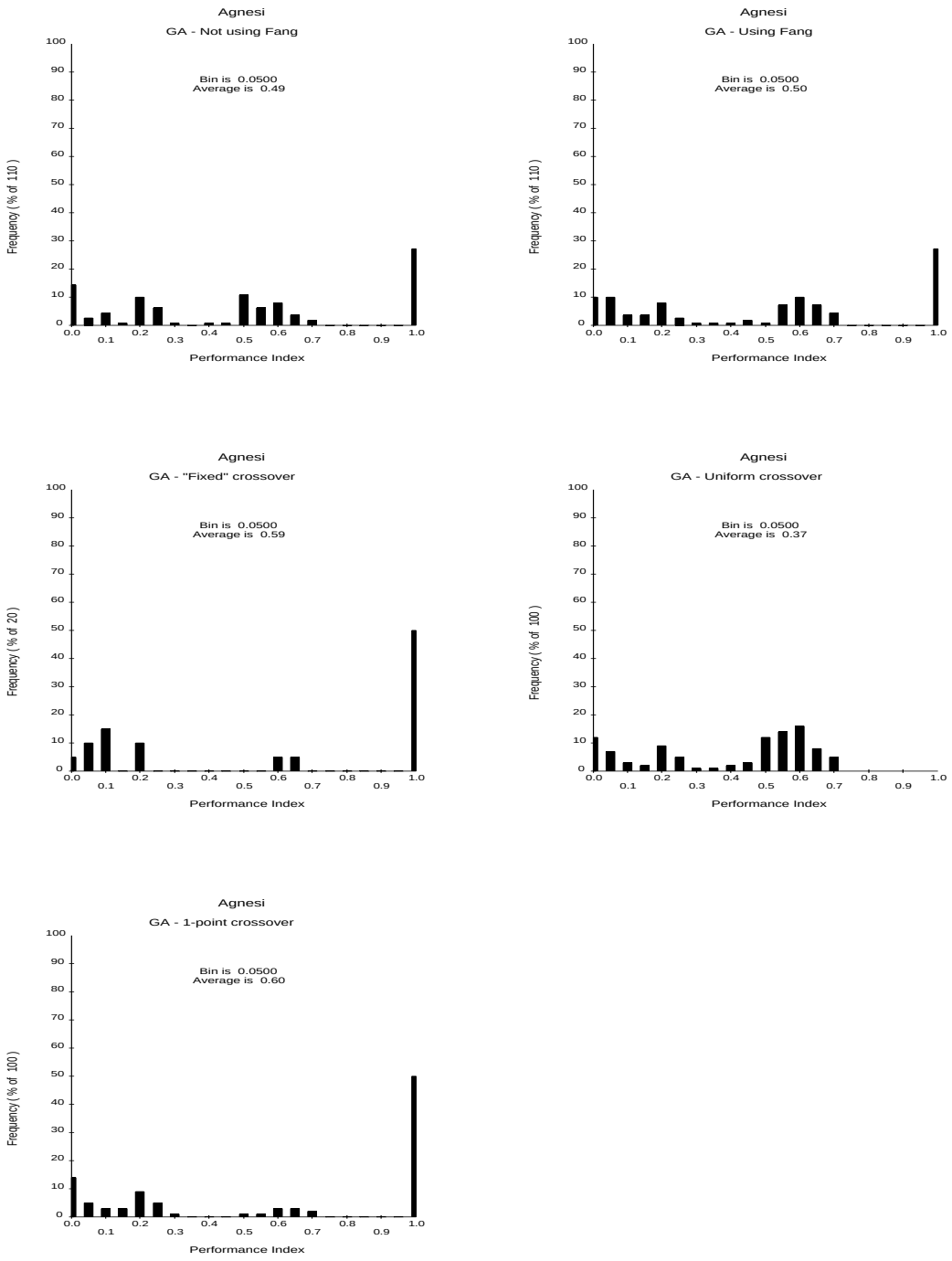

Figure 5.6: Performance index analysis (Agnesi)

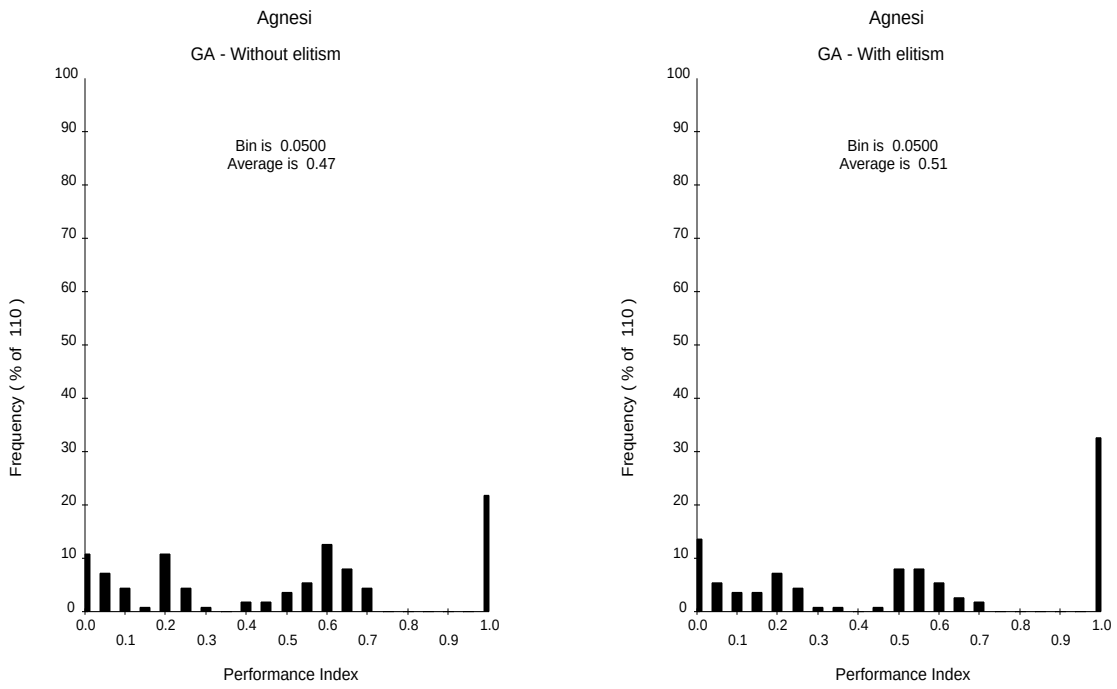

Figure 5.7: Performance index analysis (Agnesi)

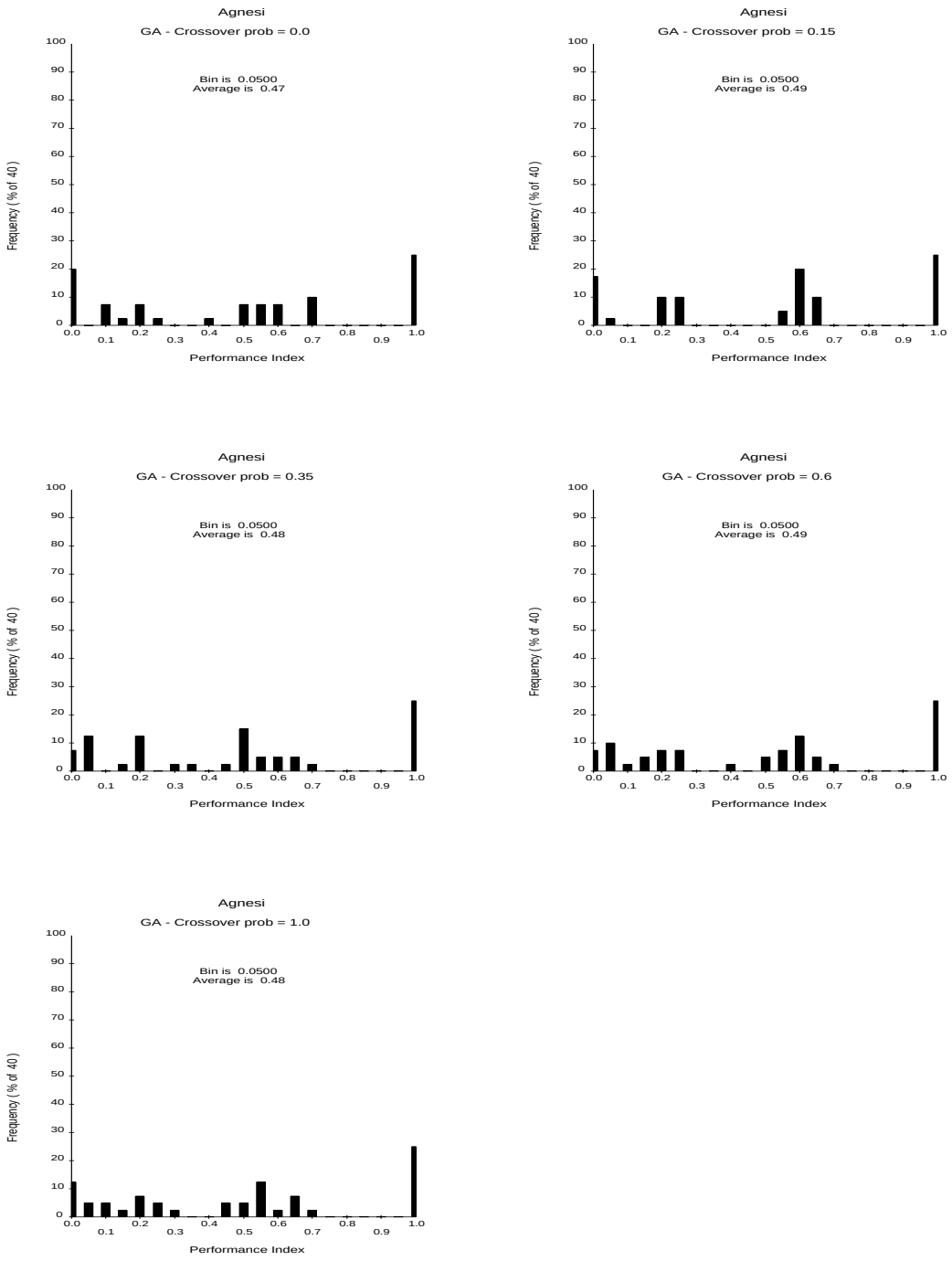

Figure 5.8: Performance index analysis (Agnesi)

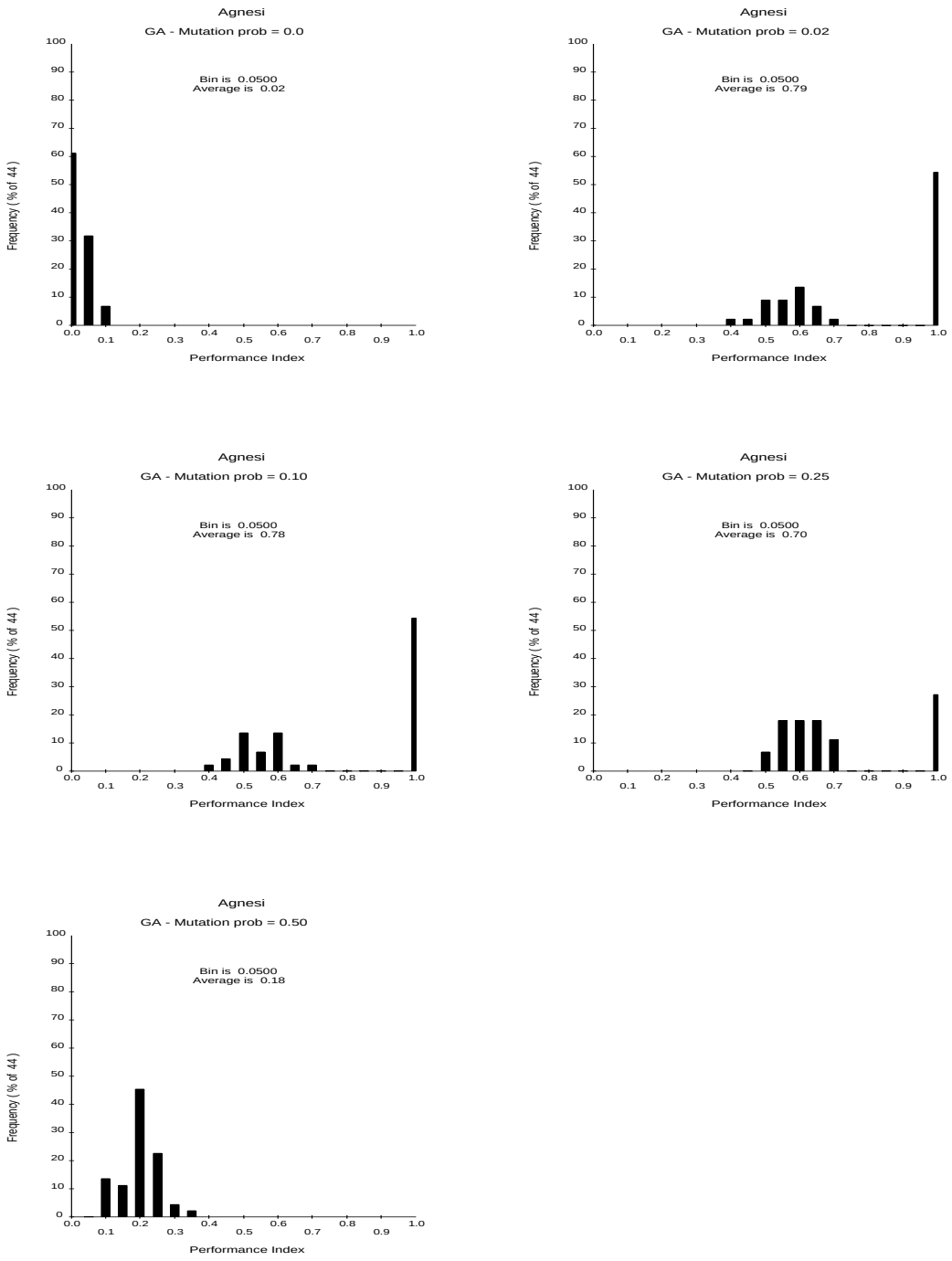

Figure 5.9: Performance index analysis (Agnesi)

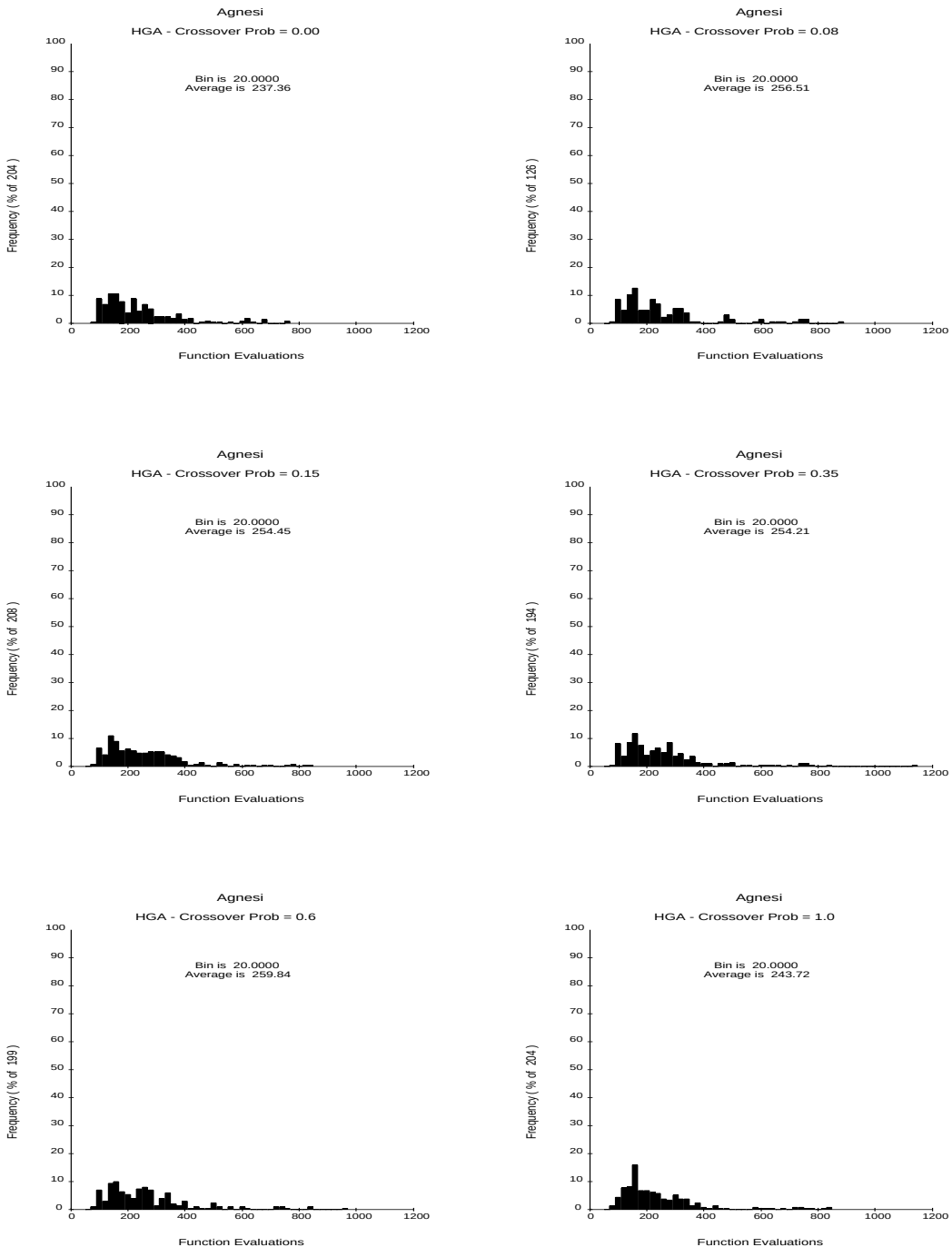

Figure 5.10: Crossover probabilities for successful experiments (Agnesi)

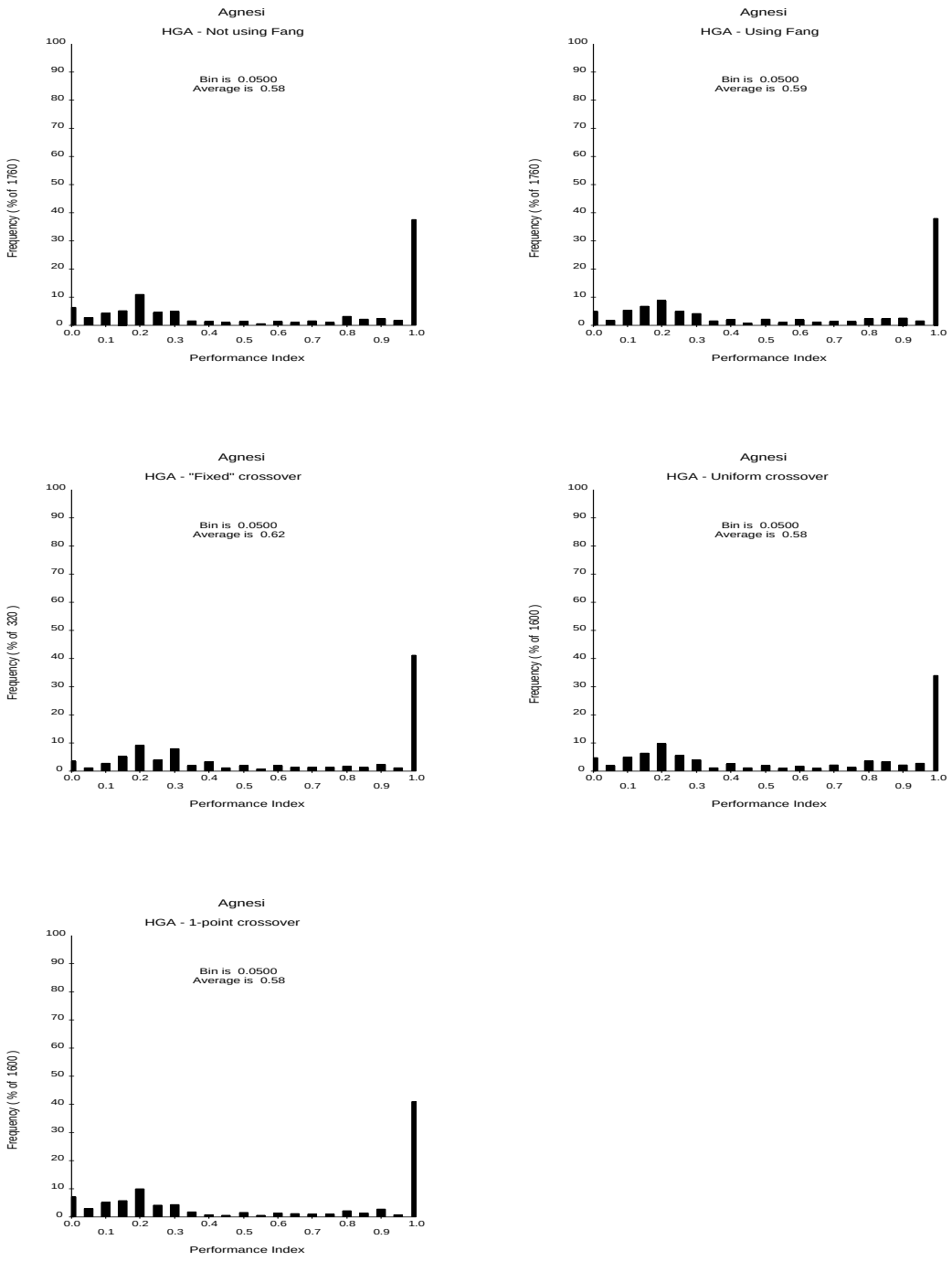

Figure 5.11: Performance index analysis (Agnesi)

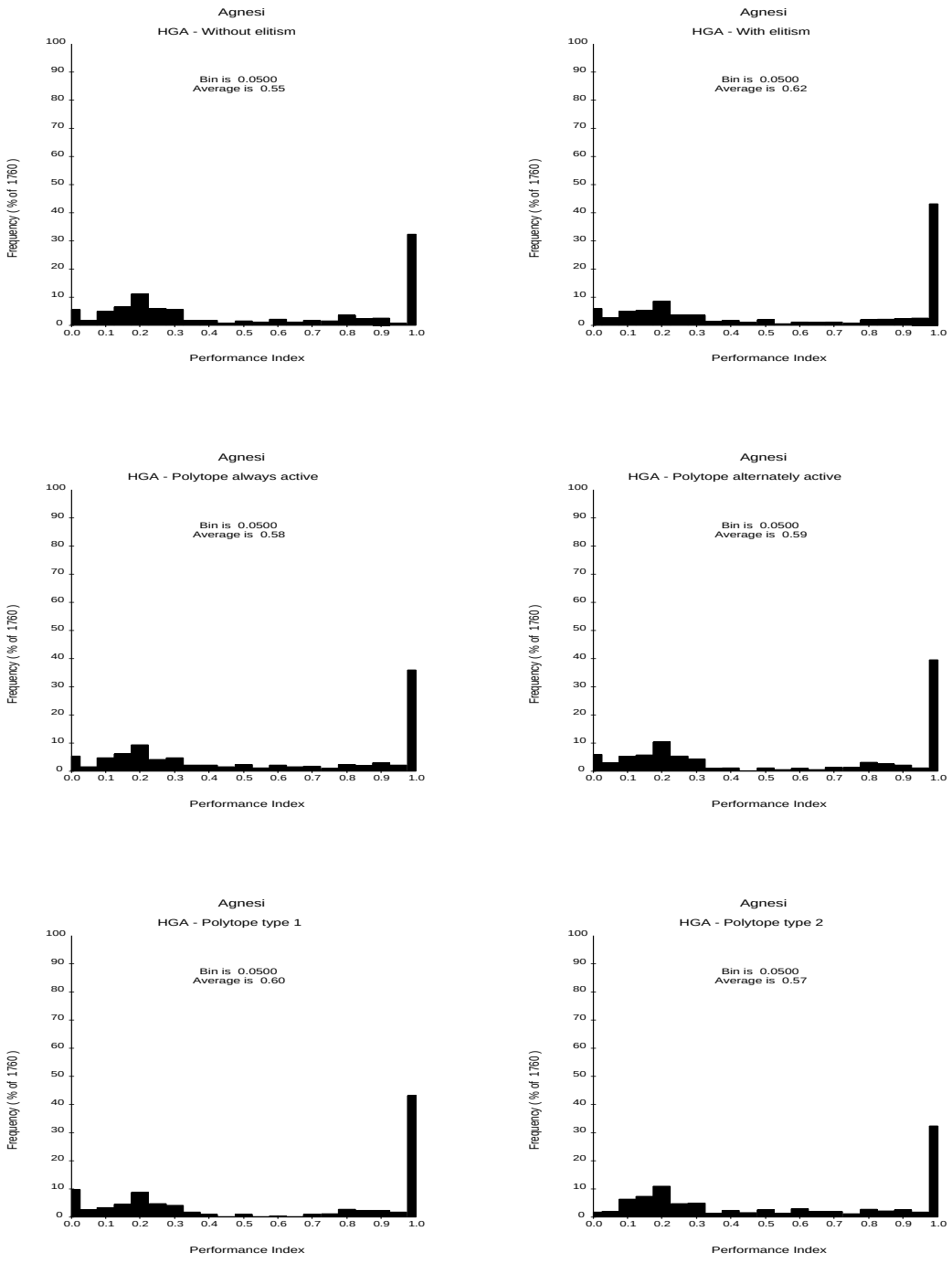

Figure 5.12: Performance index analysis (Agnesi)

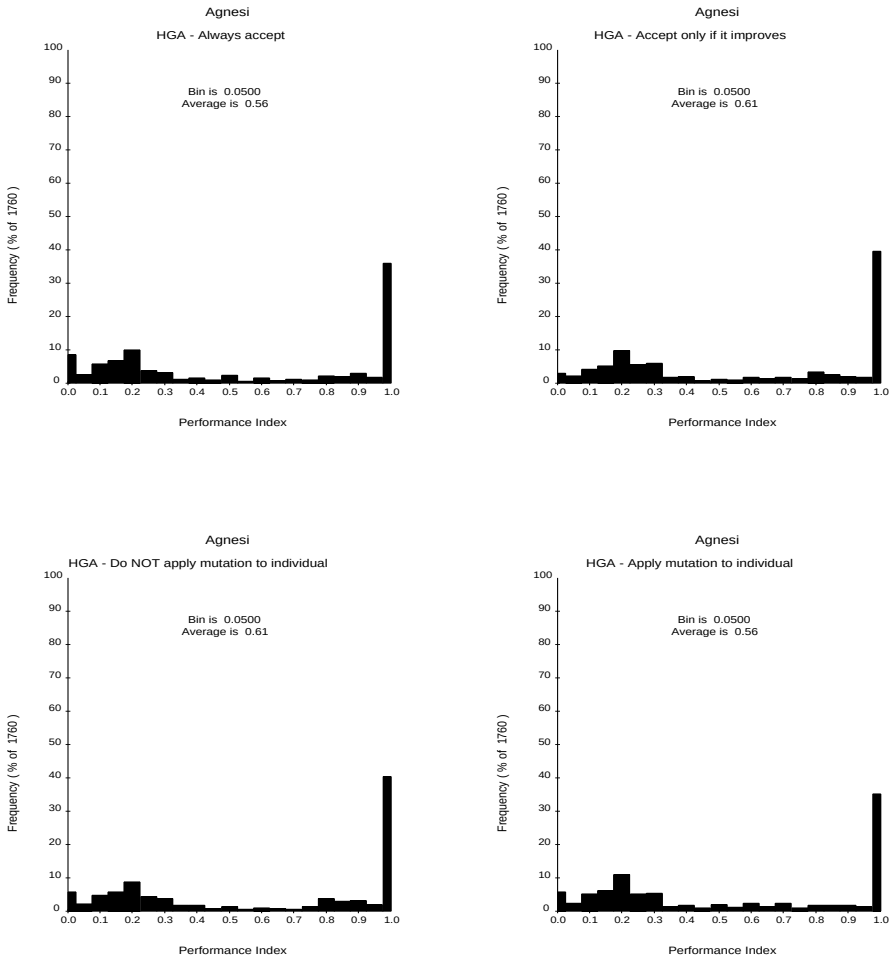

Figure 5.13: Performance index analysis (Agnesi)

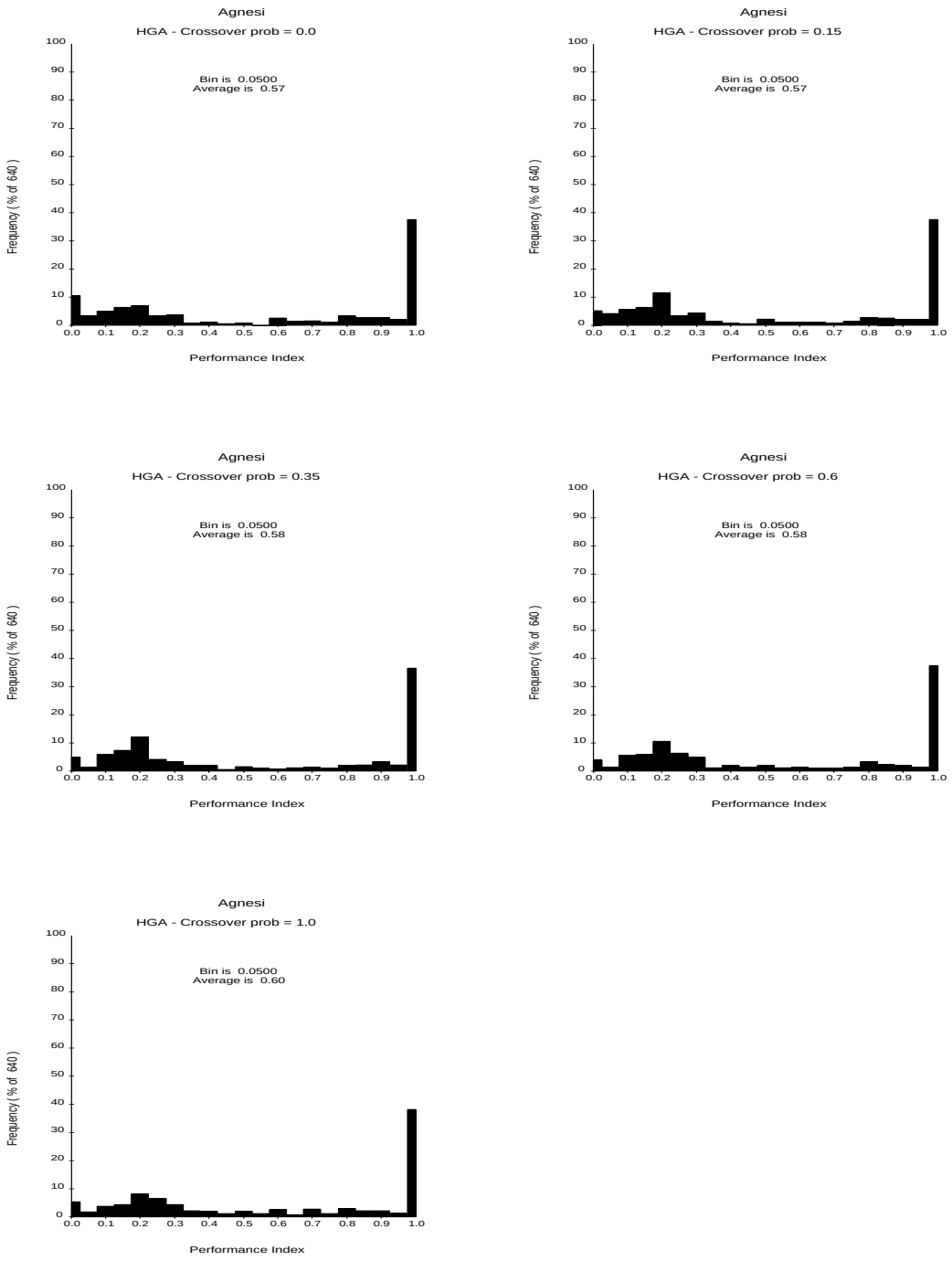

Figure 5.14: Performance index analysis (Agnesi)

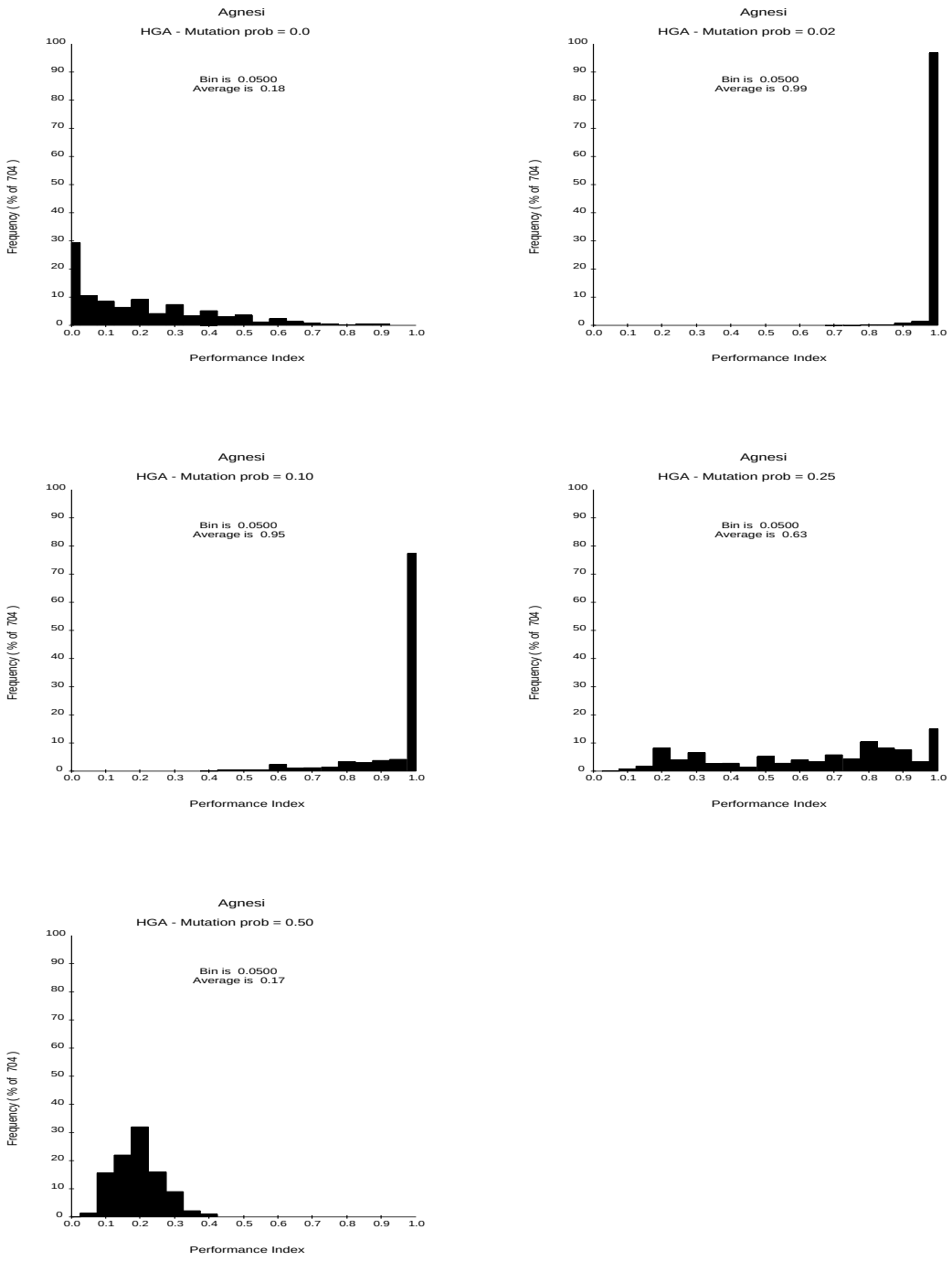

Figure 5.15: Performance index analysis (Agnesi)

Shubert Function

This multimodal function is described by the equation[Cvijovic and Klinowski, 1995]:

$$f(x,y) = -\left\{\sum_{i=1}^{5} i\cos((i+1)x+i)\right\}\cdot\left\{\sum_{i=1}^{5} i\cos((i+1)y+i)\right\} \tag{5.3}$$

The function has, in the feasible domain [-10, 10] for both independent variables, 760 local maxima, 18 of which are global with a value equal to 186.7309 (Figure 5.16). The tolerance considered for this case was 0.538%. A resolution of 0.5 required a parameter string length of six bits for each independent variable to cover the entire domain.

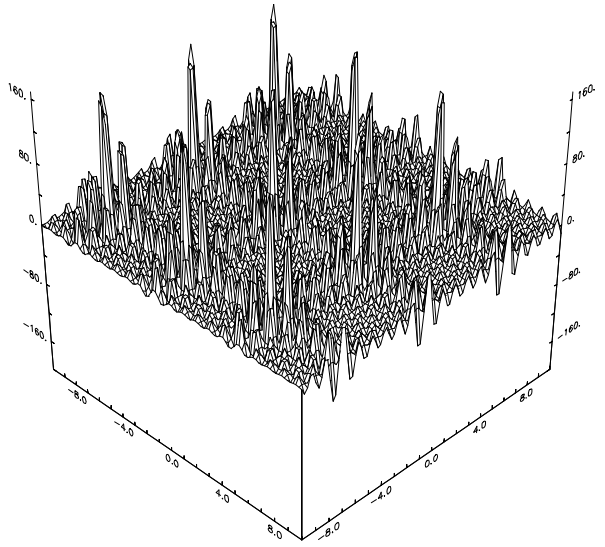

Figure 5.16: Shubert function

A population size equal to the chromosome length of 12 was assumed. Figure 5.17 shows the results obtained for the pure GA case. The top left plot shows all experiments for the pure GA algorithm. The remaining plots in the figure were obtained filtering out all experiments that did not produce the highest performance index, 1.0 in this case.

Out of the 220 experiments, six produced a performance index equal to 1.00 (3%), and 113 failed absolutely (51%). The performance index range [0.5, 1.0] was hit by 66

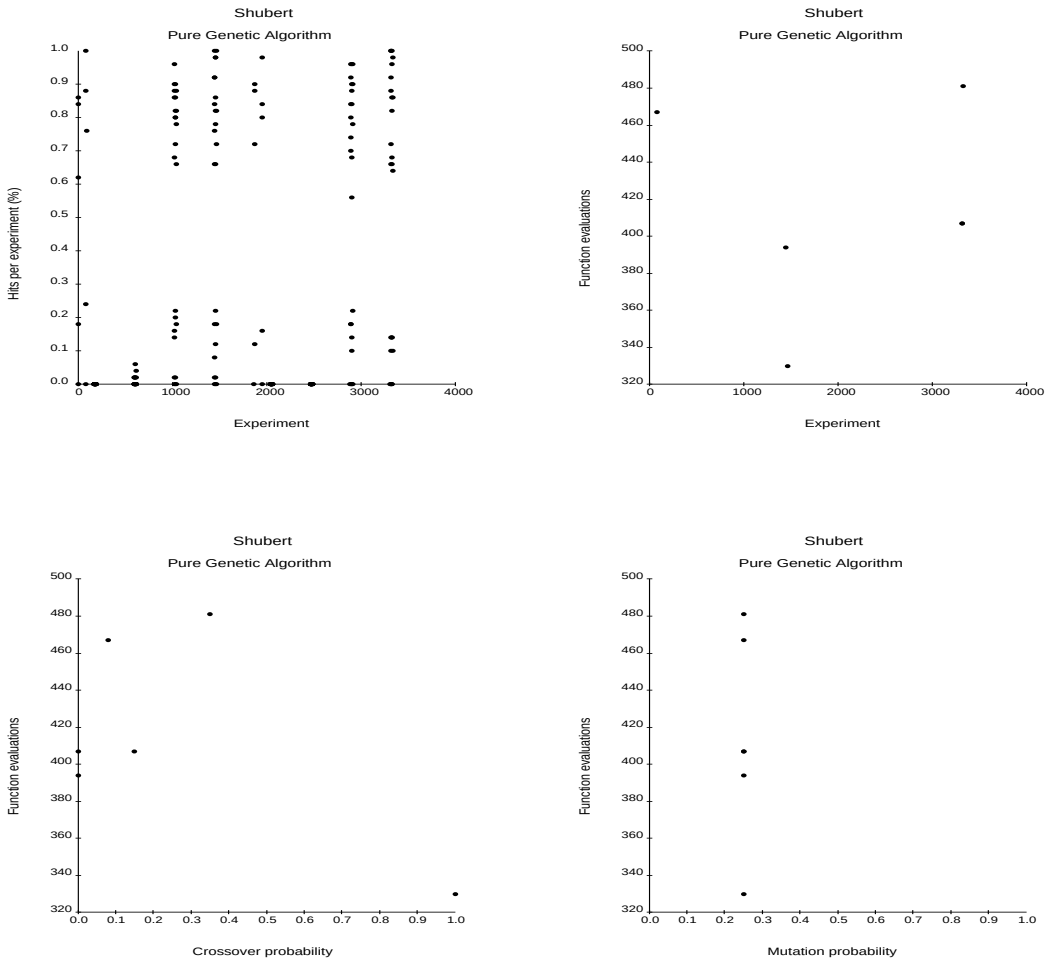

Figure 5.17: Pure GA performance (Shubert)

experiments (30%). Other values of measure are shown in Table 5.6. Observing the top left plot in Figure 5.17 and the information available in the database generated by all experiments (not shown) we observed that the low performance index of some experiments was mainly due to premature convergence. The lowest performance indices (including 0.0, absolute failure) were generated mostly by experiments using zero mutation rate or uniform crossover. The mutation rate of 0.02 associated with the single-point crossover generated performance indices in the range [0.1,0.3]. When the mutation rate was raised to 0.50, keeping the crossover type, the range [0.55, 0.8]

was obtained. The mutation rates of 0.10 and 0.25 were responsible for generating the highest scores, from 0.9 to 1.0. The best indices were generated when elitism, single-point crossover and a mutation rate of 0.25 were used. The crossover probability of 1.00 generated the best result. So, for this function, elitism, a moderate mutation rate associated with single-point crossover and high crossover rate worked better.

Exper #	Initial pop	Xover type	Save best	Xover prob	Mut prob	Opt gen	Func eval	Perf index
89	0	-1	1	0.08	0.25	59	467	1.00
1449	0	1	1	0.00	0.25	49	394	1.00
1469	0	1	1	1.00	0.25	37	330	1.00
3319	1	1	1	0.00	0.25	49	407	1.00
3324	1	1	1	0.15	0.25	48	407	1.00
3329	1	1	1	0.35	0.25	60	481	1.00

Table 5.6: Best performers for GA (Shubert)

Looking at the top right plot in Figure 5.17 we observed that the most efficient experiment consumed 330 function evaluations (on average) to find the optimum. The major number of best performers used single-point crossover as we can observe in Table 5.6. The bottom left plot shows no strong influence of the crossover probability. The bottom right plot of Figure 5.17 shows that the choice of a moderate mutation rate was the only one able to produce good results. Elitism and single-point crossover were the parameters of major influence to obtain the best results.

Figure 5.18 shows the results obtained for the hybrid (HGA) case. The top left plot shows all experiments. The remaining plots in the figure were obtained filtering out all experiments with a performance index lower than the highest performance index obtained by the pure GA, e.g., 1.0. Unlike the pure GA, we observed that the HGA produced performance indices covering the full range [0,1]. Out of 3520 experiments 40 succeeded absolutely (1%) and 1296 failed absolutely (37%). The range [0.5, 1.0] was achieved by 1040 experiments (30%). The results obtained by the pure GA, based on the performance index, were superior in percentage to the hybrid GA but when we look at number of function evaluations consumed to reach the optimum then

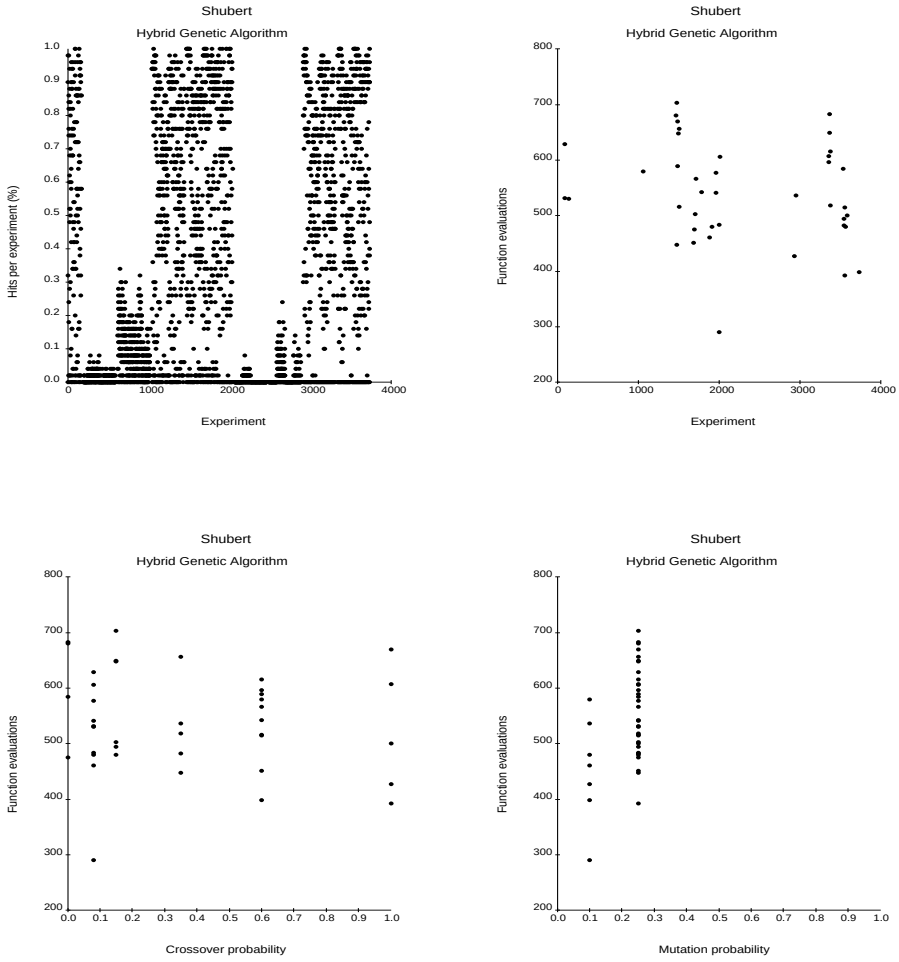

Figure 5.18: HGA performance (Shubert)

the hybrid GA performed better. The best experiment using pure GA consumed 330 function evaluations compared to 290 consumed by the best one using hybrid GA. For this test only one hybrid experiment performed better than the pure GA search. The hybrid parameters associated with this experiment were: fixed crossover, elitism, polytope type 1 activated every other generation, individuals generated by the polytope accepted only if they improved the polytope's function values with mutation being applied to the generated individual, and, finally, a mutation rate of 0.10. So, 40 function evaluations were saved by the hybrid GA with this combination of parameters

for this type of function.

We noted that, for this type of function, the polytope activity applies a strong influence on the search process. The top two plots in Figure 5.19 show the performance index against the mutation rate for both GA and HGA algorithms. We noted an improvement to the search process for all values of the mutation rate. The bottom two plots in the same figure show the performance achieved when the individual generated by the polytope was not submitted to the mutation operator (bottom left) and when it was (bottom right).

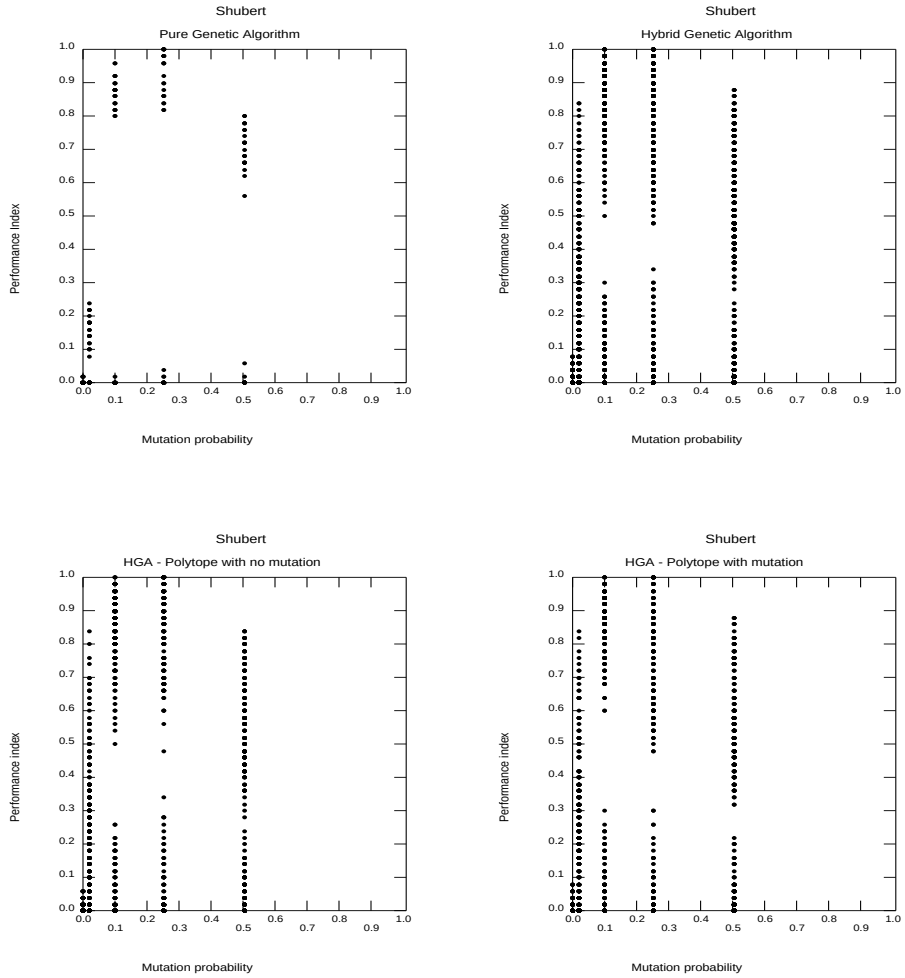

Figure 5.19: GA and HGA compared (Shubert)

The performance indices, for both GA and HGA algorithms, were analyzed according to the variation of the various genetic parameters, as shown in Figures 5.20 to 5.30 and discussed below. From this analysis we could evaluate the most significant genetic parameter values for this type of function.

For the pure GA algorithm Figure 5.20 shows that the average number of function evaluations spent to generate successful experiments varied from 330.00 to 481.0 and so the influence of the crossover probability was shown to be weak although the probability of 0.6 did not generate a single successful experiment. Figure 5.21 shows no improvement when the Fang algorithm was used. Fixed or single-point crossover performed well while the uniform crossover had a poor performance. Elitism performed slightly better according to the plots in Figure 5.22. The crossover probability did not play an important role as is shown in Figure 5.23. The mutation probabilities that performed better were 0.10 and 0.25 while the values of 0.0, 0.02 and 0.5 could not produce a single successful experiment (Figure 5.24).

For the hybrid HGA algorithm Figure 5.25 shows that the average number of function evaluations spent to generate successful experiments varied from 513.00 to 605.75. This range is higher than the one for the pure GA approach showing that, for this function, the hybrid HGA spent more function evaluations to produce successful experiments than the pure GA. Again, HGA did not show a strong influence of the crossover probability for this problem. Figure 5.26 does not show a significant performance improvement when the Fang algorithm was used and no improvement was observed in the crossover type performance. This result suggests the either the fixed or the single-point crossover for this type of function. Elitism performed slightly better according to the two top plots in Figure 5.27. The remaining plots in this figure show no influence of the type of polytope activity but the polytope type 1 performed better than the type 2 when the successful experiments are considered. Figure 5.28 shows a better performance when the individual generated by the polytope is accepted only if it improves the polytope fitness and slightly better when no mutation is applied to this new individual. The crossover probability does not show a significant influence according to the Figure 5.29. However, the mutation probability played an important role when successful experiments were generated. Moderate values as 0.10 and 0.25

produced the best performance while the values of 0.0, 0.02 and 0.5 could not produce a single successful experiment (Figure 5.30).

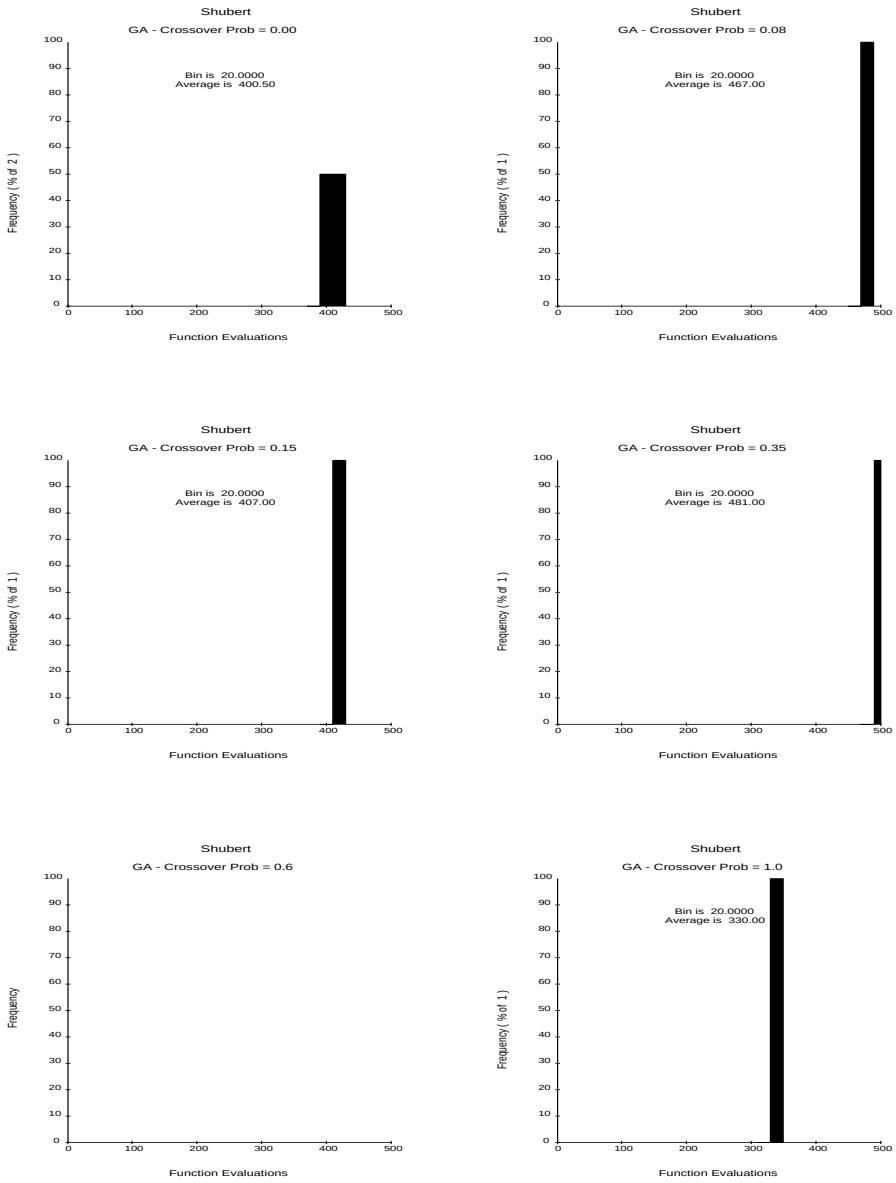

Figure 5.20: Crossover probabilities for successful experiments (Shubert)

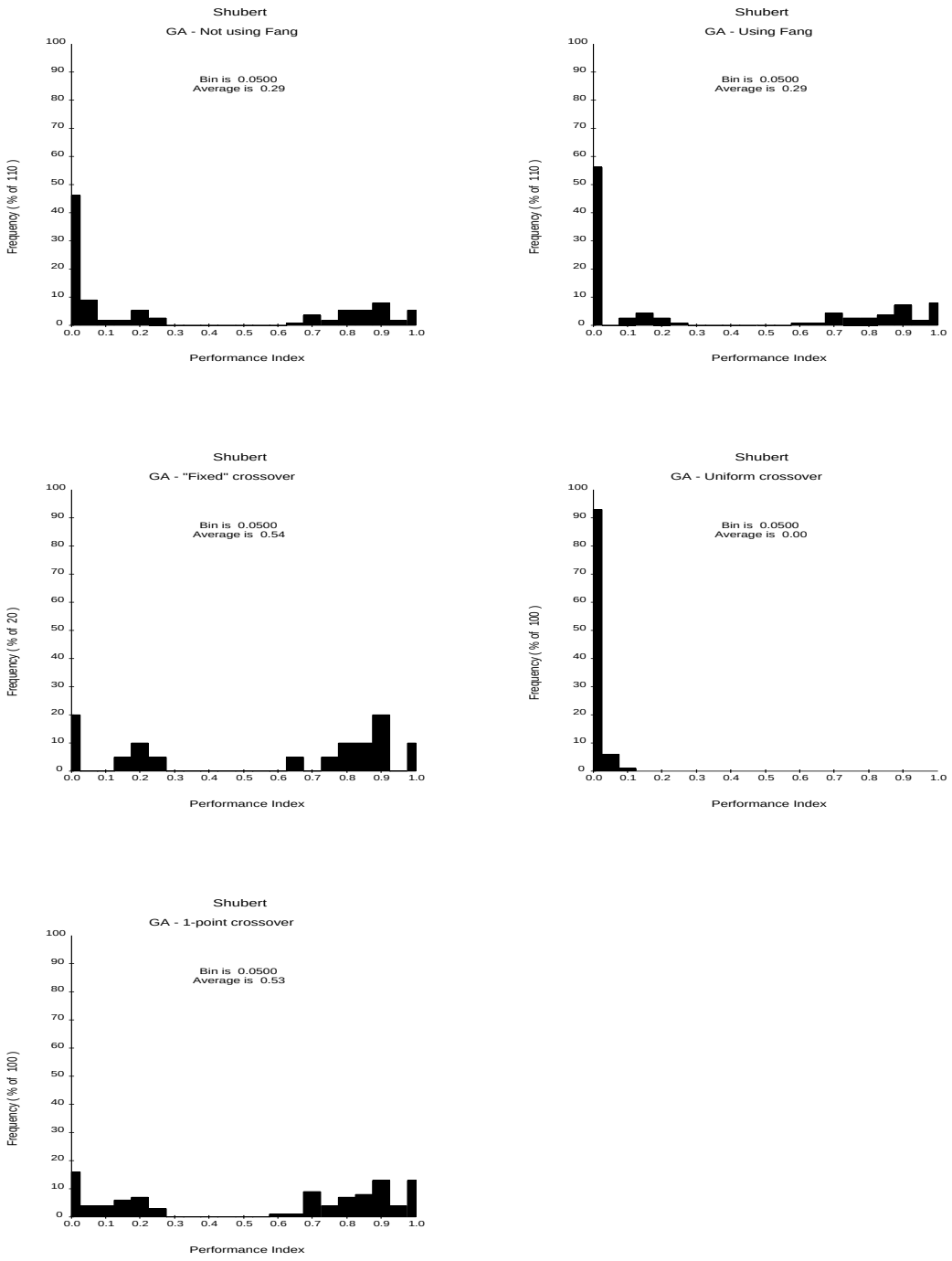

Figure 5.21: Performance index analysis (Shubert)

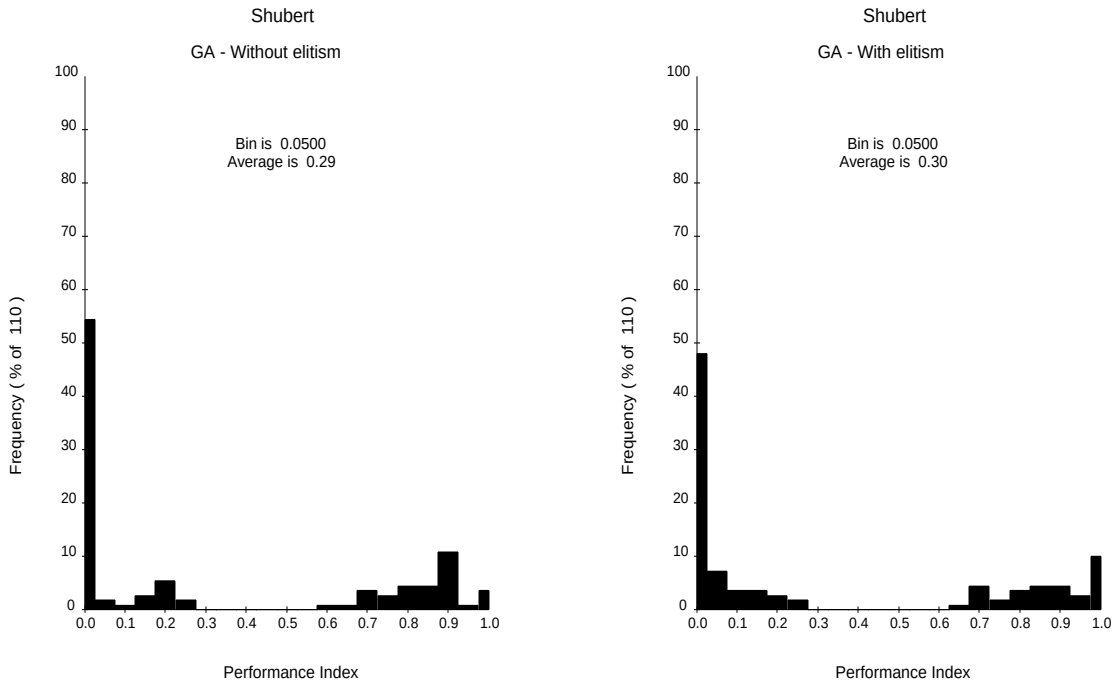

Figure 5.22: Performance index analysis (Shubert)

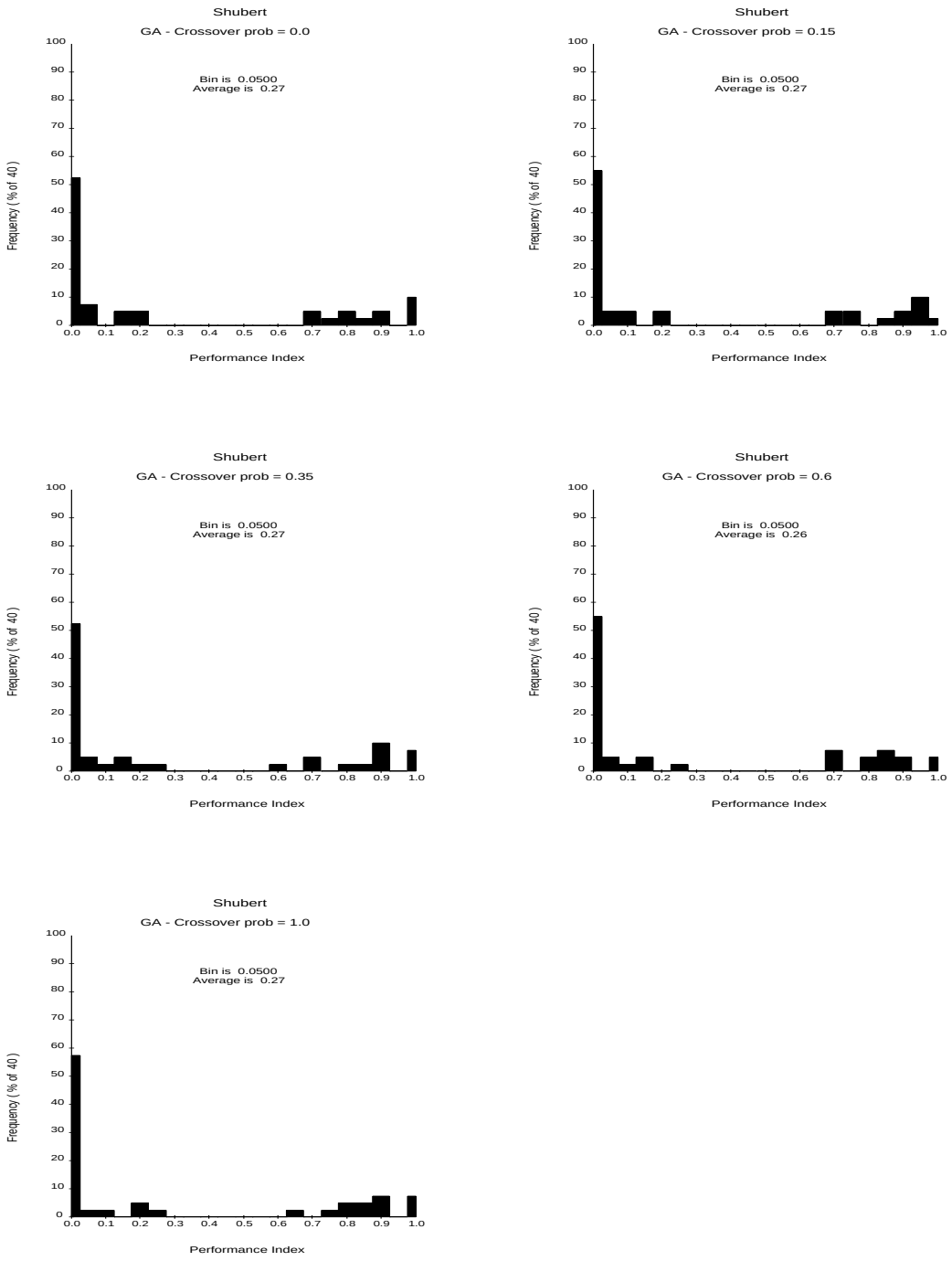

Figure 5.23: Performance index analysis (Shubert)

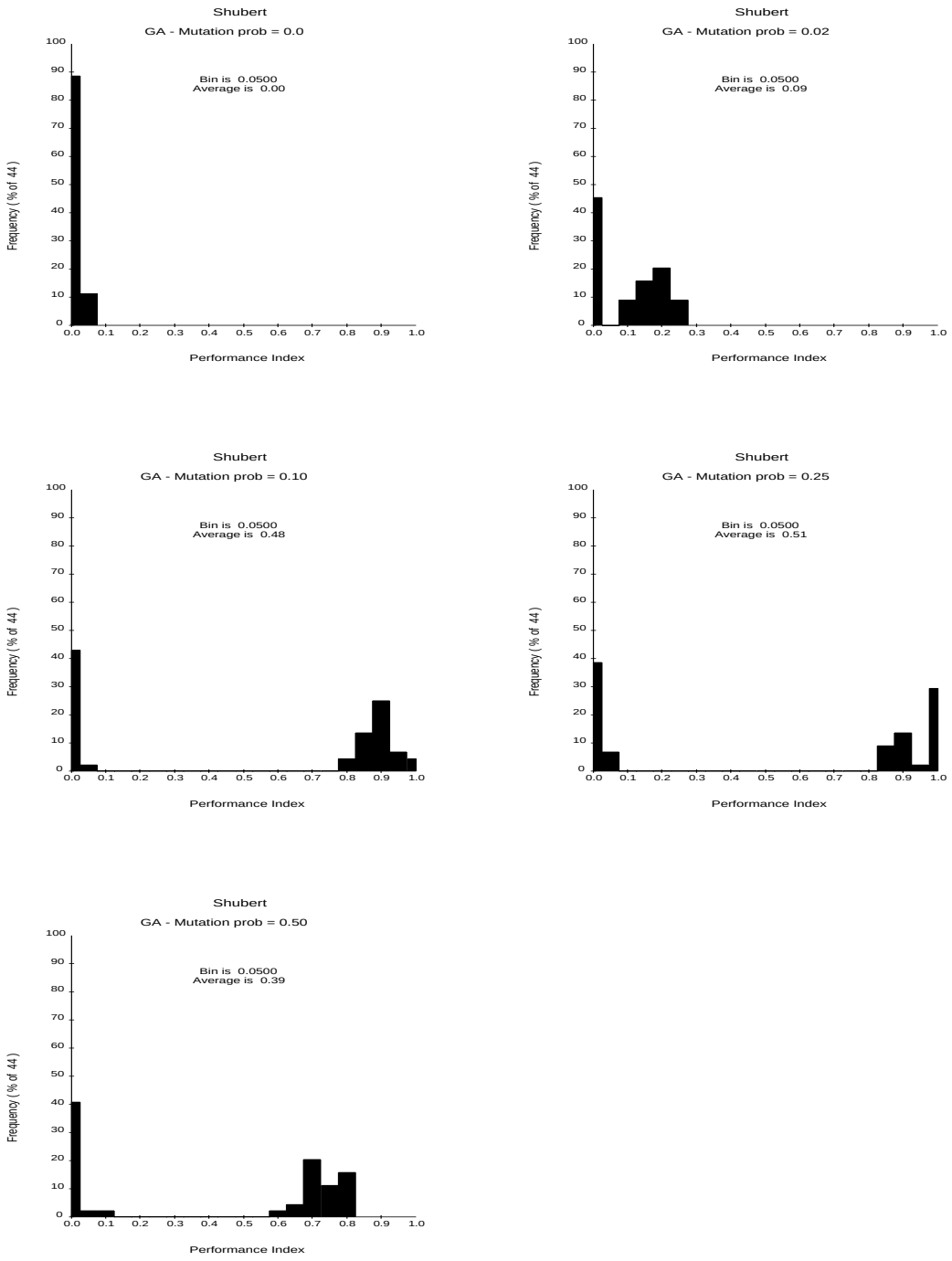

Figure 5.24: Performance index analysis (Shubert)

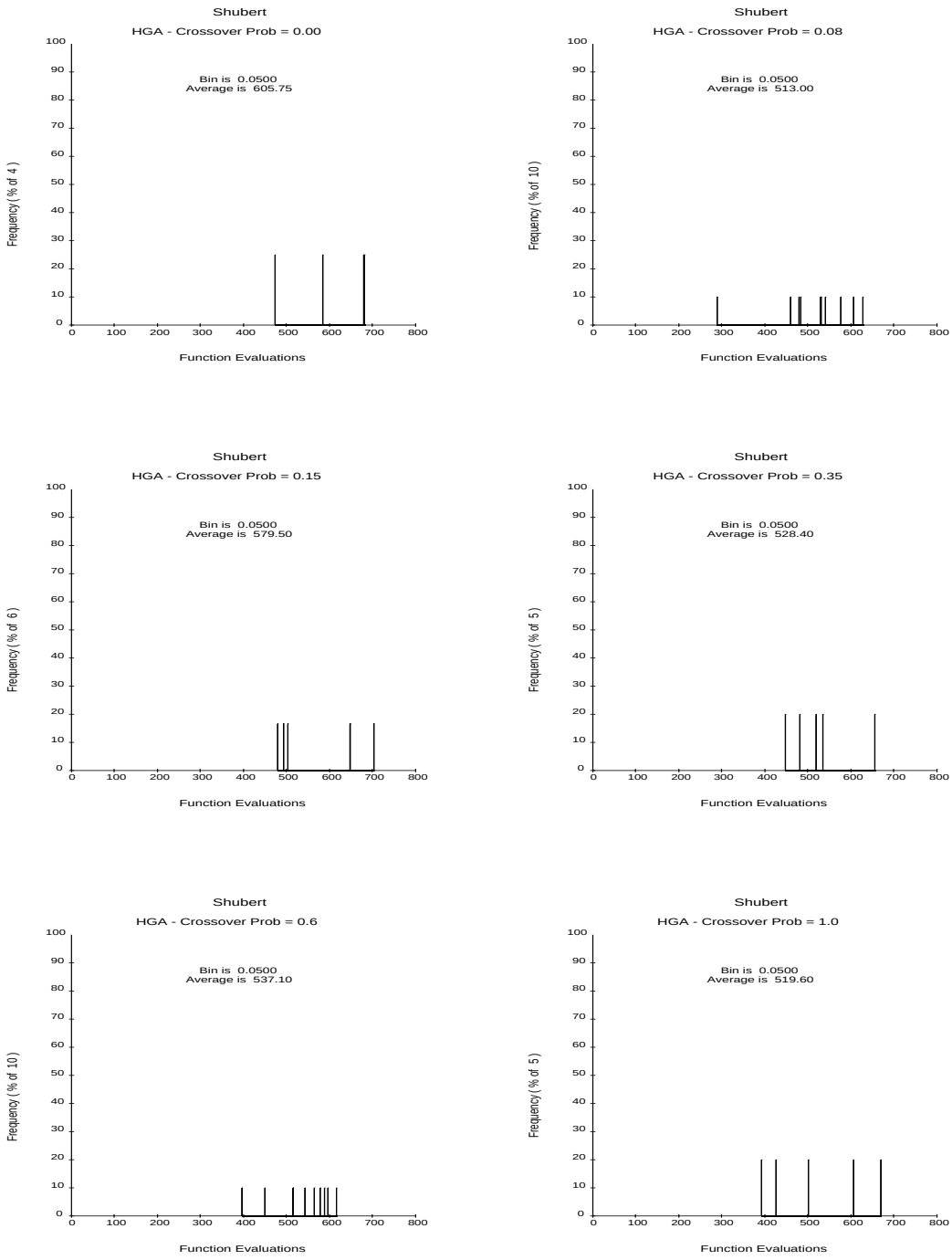

Figure 5.25: Crossover probabilities for successful experiments (Shubert)

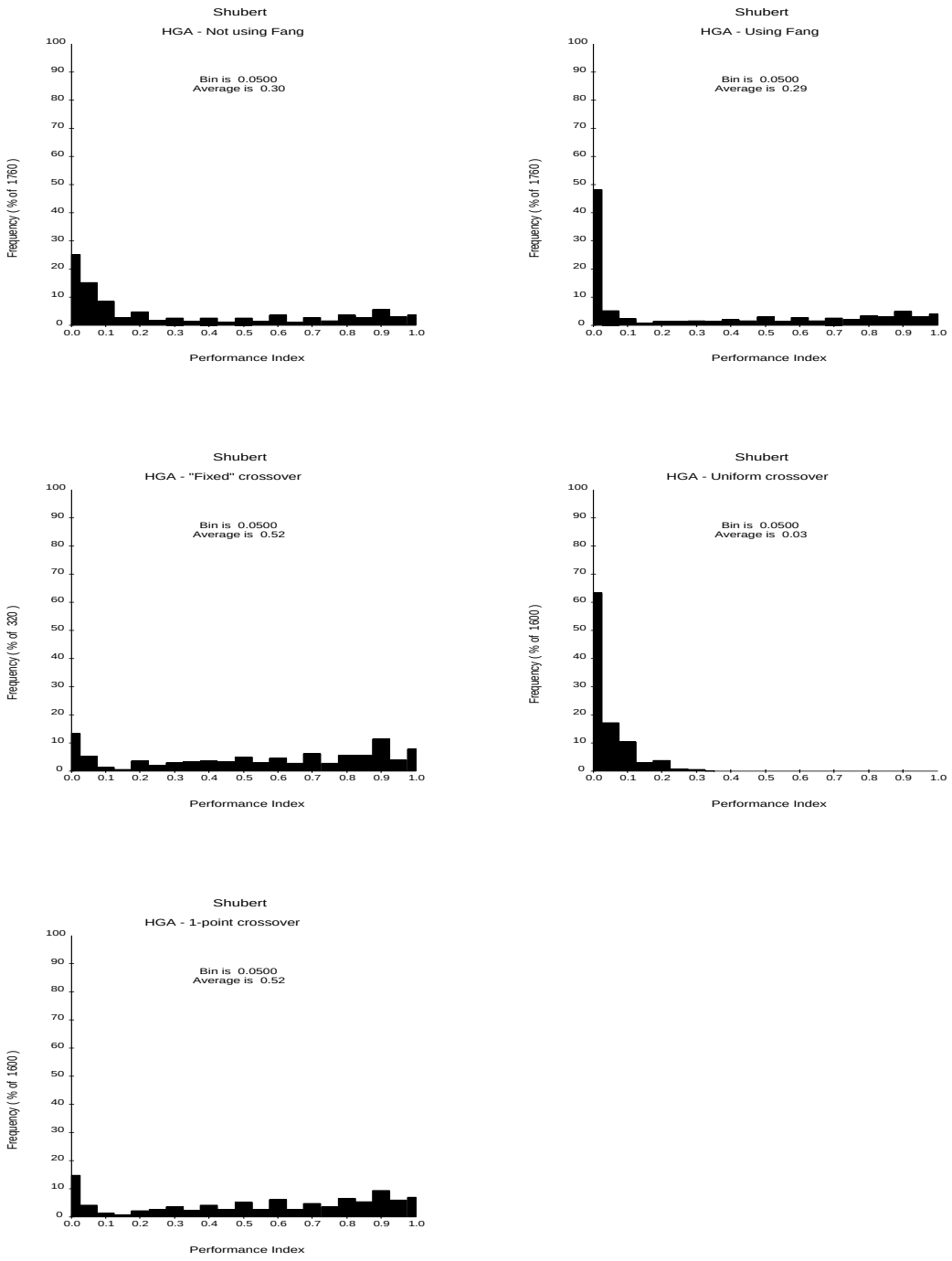

Figure 5.26: Performance index analysis (Shubert)

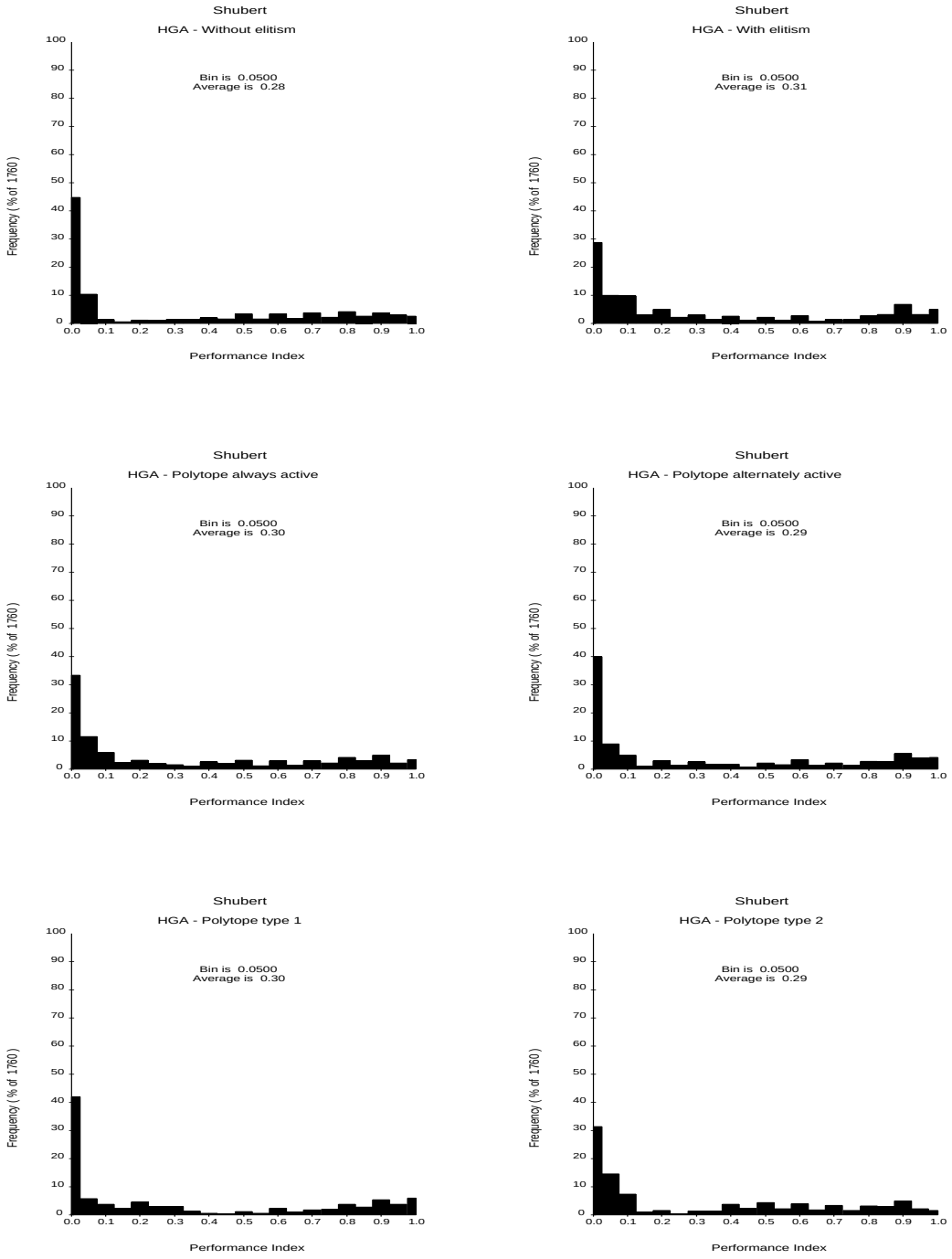

Figure 5.27: Performance index analysis (Shubert)

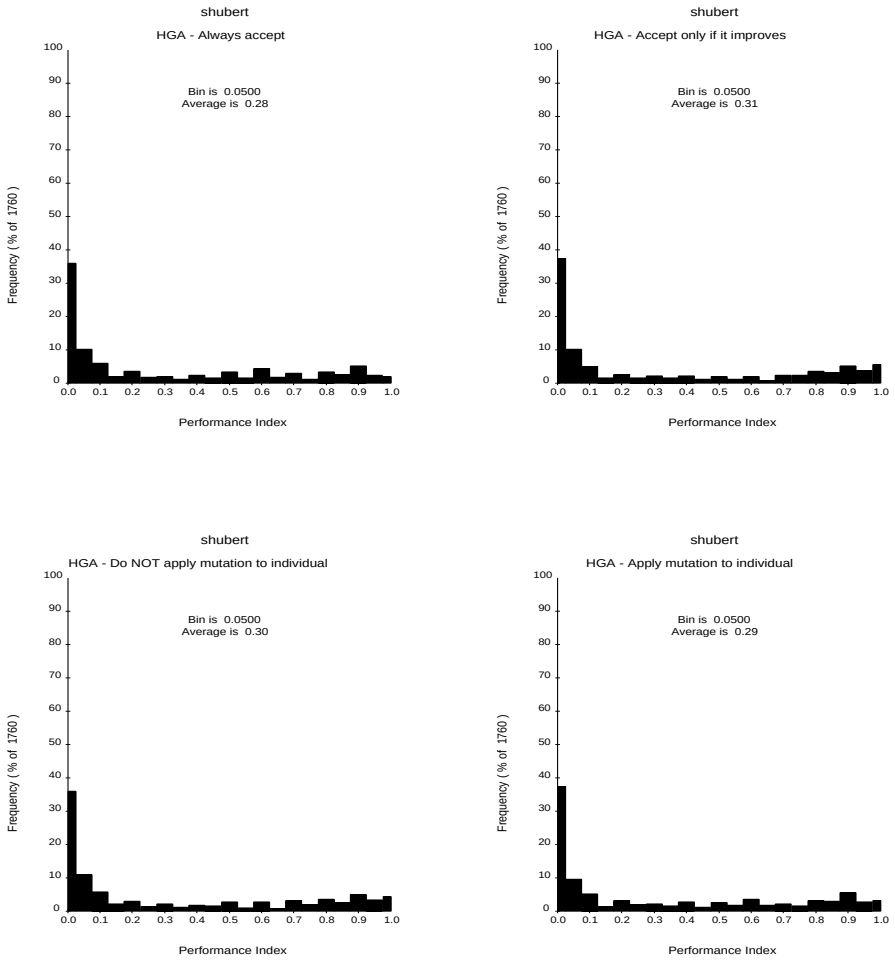

Figure 5.28: Performance index analysis (Shubert)

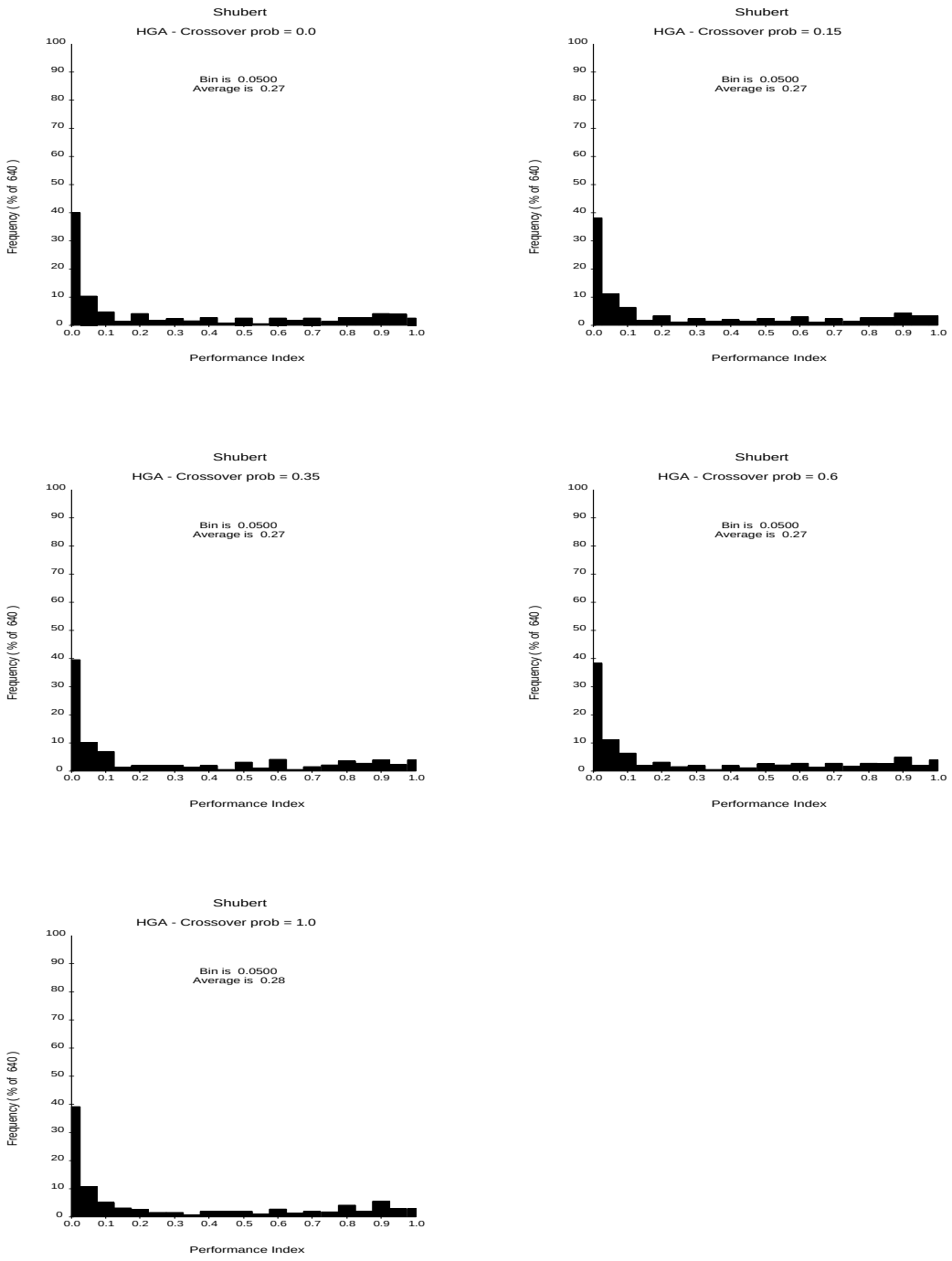

Figure 5.29: Performance index analysis (Shubert)

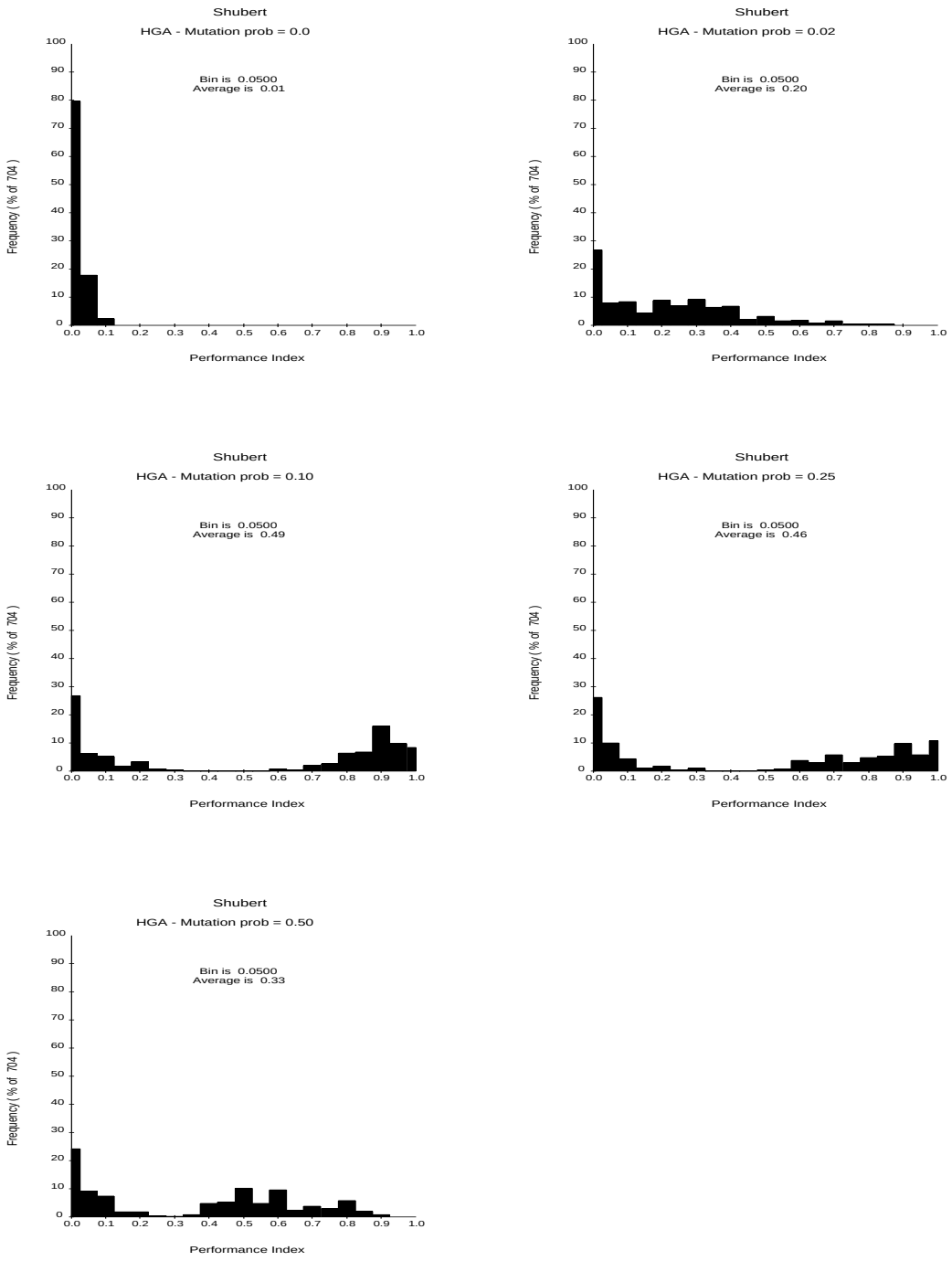

Figure 5.30: Performance index analysis (Shubert)

Composed Function

This multimodal function is described by the equation:

$$f(x, y) = (1 - x) \cdot \sin(x) \cdot (1 - y) \cdot \sin(y) \tag{5.4}$$

In the feasible domain [-18.5, 18.5] for both independent variables this function has 169 stationary points, 85 of which are local maxima. The global maximum at (-17.333, -17.333) has a value equal to 335.11 (Figure 5.31). The optimum value occurring in the genetic grid is 325.77 at (-17.5, -17.5). The tolerance considered for this case was 0.2985%. A resolution of 1.0 required a parameter string length of six bits for each independent variable to cover the entire domain.

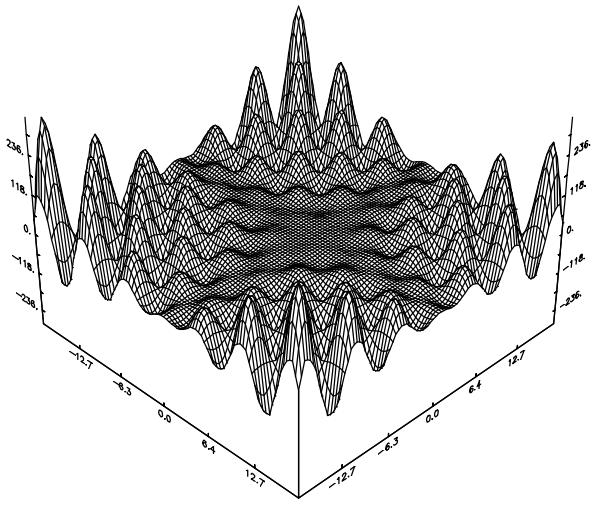

Figure 5.31: Composed function

A population size equal to the chromosome length of 12 was used. Figure 5.32 shows the results obtained for the pure GA case. The top left plot shows all experiments for the pure GA algorithm. The remaining plots in the figure were obtained filtering out all experiments that did not produce the highest performance index, 0.86 in this case.

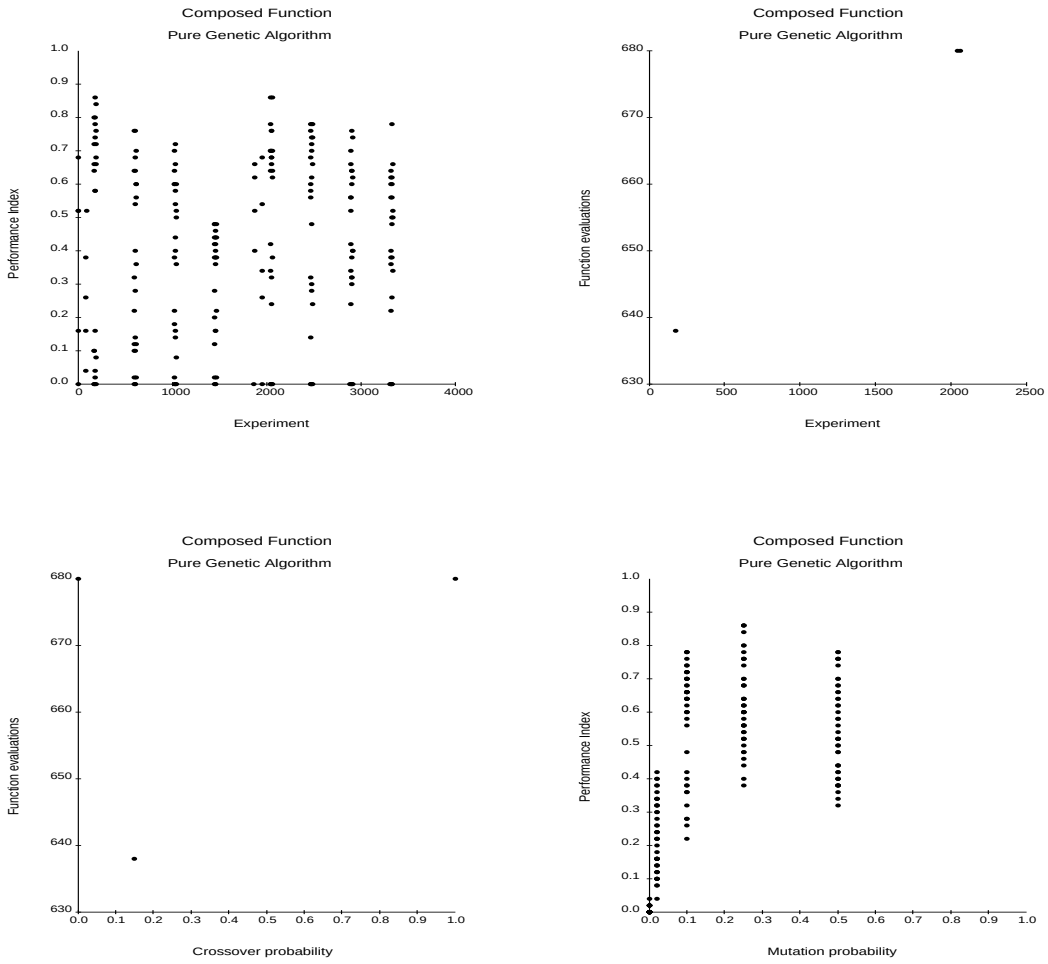

Figure 5.32: Pure GA performance (Composed)

Out of the 220 experiments none succeeded absolutely and 37 failed absolutely (17%). The highest performance index of 0.86 was produced by three experiments (1%). The performance index range [0.5, 1.0] was achieved by 94 experiments (43%). Other values of measure are shown in Table 5.7. Observing the top left plot in Figure 5.32 and the information available in the database generated by all experiments (not shown) we observed that the low performance index of some experiments was due mainly to premature convergence. The lowest performance index of 0.0 (absolute failure) was generated mainly by experiments using zero mutation rate. The mutation

rate of 0.02 associated with the uniform and single-point crossover generated performance indices in the range [0.0, 2.0]. When the mutation rate was raised to 0.10, 0.25 and 0.50, and depending on the crossover type associated with the crossover probability, the performance index range [2.00, 0.80] was generated. The mutation rate of 0.25, along with uniform crossover and no elitism, were responsible for generating the highest score of 0.86.

Exper #	Initial pop	Xover type	Save best	Xover prob	Mut prob	Opt gen	Func eval	Perf index
179	0	0	0	0.15	0.25	74	638	0.86
2044	1	0	0	0.00	0.25	82	680	0.86
2064	1	0	0	1.00	0.25	81	680	0.86

Table 5.7: Best performers for GA (Composed)

Looking at the top right plot in Figure 5.32, we observed that the most efficient experiment consumed 638 function evaluations (on average) to find the optimum. All the best performers used uniform crossover as we can observe in Table 5.7. The bottom left plot shows no strong influence of the crossover probability. The bottom right plot of Figure 5.32 shows the clearly better performance for the moderate mutation rate of 0.25. No elitism and uniform crossover were the other two hybrid parameters of major influence to obtain the best results.

Figure 5.33 shows the results obtained for the hybrid (HGA) case. The top left plot shows all experiments. The remaining plots in the figure were obtained filtering out all experiments with a performance index lower than the highest performance index obtained by the pure GA, e.g., 0.86. Unlike pure GA, we observed that the HGA produced performance indices covering the full range [0,1]. Out of 3520 experiments 68 succeeded absolutely (2%) and 475 failed absolutely (14%). The range [0.5, 1.0] was hit by 1684 experiments (48%), 494 of which (14%) produced a performance index above 0.86. The results obtained by the pure GA, based on the performance index, were significantly below the hybrid GA for this type of function. The best

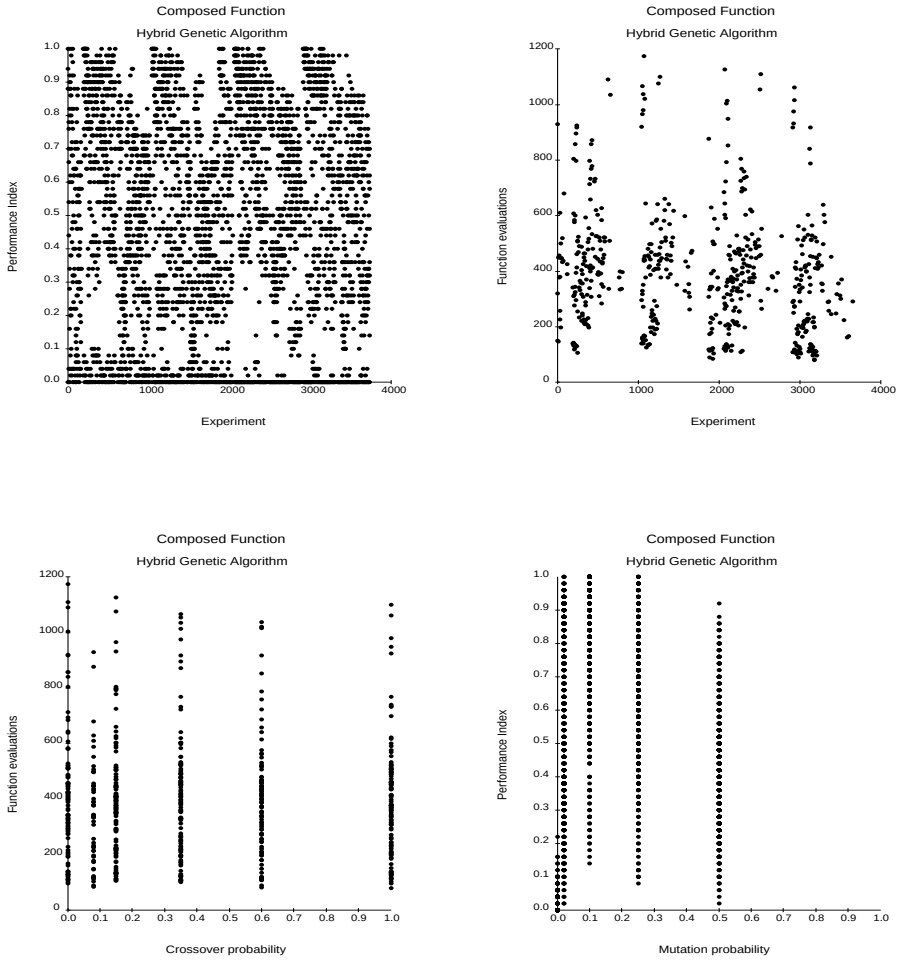

Figure 5.33: HGA performance (Composed)

experiment using pure GA consumed 638 function evaluations compared to 508 experiments (14%), consuming less than 638 function evaluations when the hybrid GA was used. Table 5.8 shows the best performers that consumed less than 117 function evaluations along with the other hybrid parameter values and measure indices. The best experiment considering the lowest number of function evaluations achieved a performance index of 0.92 with 80 function evaluations. Experiments with an index of 0.86 consumed about 110 function evaluations. The absolute success was reached with 107 function evaluations and the most balanced experiment produced an index

of 0.98 with 83 function evaluations.

Exper #	Init pop	Xover type	Save best	Poly actv	Poly type	Poly Effc	Poly mut	Xover prob	Mut prob	Opt gen	Func eval	Perfer Index
257	0	0	0	1	1	1	0	0.35	0.02	71	107	0.92
1882	1	-1	0	1	1	0	1	0.08	0.02	33	115	0.96
1887	1	-1	0	1	1	1	0	0.08	0.02	48	91	0.92
1917	1	-1	0	-2	1	0	0	0.08	0.02	53	116	0.96
1927	1	-1	0	-2	1	1	0	0.08	0.02	41	86	0.88
1932	1	-1	0	-2	1	1	1	0.08	0.02	45	105	0.86
2087	1	0	0	1	1	0	0	1.00	0.02	50	114	0.92
2157	1	0	0	1	1	1	1	0.60	0.02	56	114	0.88
2277	1	0	0	-2	1	0	0	0.35	0.02	63	110	0.86
2297	1	0	0	-2	1	0	1	0.15	0.02	64	114	0.86
2917	1	1	0	1	1	0	0	0.00	0.02	36	109	1.00
2922	1	1	0	1	1	0	0	0.15	0.02	37	110	0.96
2927	1	1	0	1	1	0	0	0.35	0.02	35	108	0.96
2932	1	1	0	1	1	0	0	0.60	0.02	31	110	0.94
2942	1	1	0	1	1	0	1	0.00	0.02	36	113	1.00
2952	1	1	0	1	1	0	1	0.35	0.02	34	107	1.00
2957	1	1	0	1	1	0	1	0.60	0.02	32	115	0.98
2967	1	1	0	1	1	1	0	0.00	0.02	59	102	0.94
2982	1	1	0	1	1	1	0	0.60	0.02	47	89	0.92
2987	1	1	0	1	1	1	0	1.00	0.02	72	115	0.90
3007	1	1	0	1	1	1	1	0.60	0.02	46	109	0.96
3012	1	1	0	1	1	1	1	1.00	0.02	45	105	0.96
3117	1	1	0	-2	1	0	0	0.00	0.02	53	112	0.94
3132	1	1	0	-2	1	0	0	0.60	0.02	54	115	0.88
3157	1	1	0	-2	1	0	1	0.60	0.02	50	117	0.94
3167	1	1	0	-2	1	1	0	0.00	0.02	55	98	0.86
3172	1	1	0	-2	1	1	0	0.15	0.02	65	107	0.86
3177	1	1	0	-2	1	1	0	0.35	0.02	60	101	0.90
3182	1	1	0	-2	1	1	0	0.60	0.02	41	83	0.98
3187	1	1	0	-2	1	1	0	1.00	0.02	38	80	0.92
3197	1	1	0	-2	1	1	1	0.15	0.02	57	114	0.92
3202	1	1	0	-2	1	1	1	0.35	0.02	49	111	0.86
3207	1	1	0	-2	1	1	1	0.60	0.02	46	111	0.90
3212	1	1	0	-2	1	1	1	1.00	0.02	37	98	0.92

Table 5.8: Best performers for HGA (Composed)

The hybrid parameters that played a major role in these results were: single-point crossover, no elitism, use of Fang's algorithm, polytope type 1 and mutation rate of 0.02. The highest index of 1.0 added: acceptance of the individual generated by

the polytope only if it improved the polytope's function values with mutation being applied to the accepted individual and the polytope being active all generations. So, more than 520 function evaluations were saved by the hybrid GA for this type of function.

We noted that, for this type of function, the polytope activity applies a strong influence in the search process. The top two plots in Figure 5.34 show the performance index against the mutation rate for both GA and HGA algorithms. We noted an improvement in the search process for every value of the mutation rate. The bottom two plots in the same figure show the performances achieved when the individual generated by the polytope was not submitted to the mutation operator (bottom left) and when it was (bottom right).

The performance indices, for both GA and HGA algorithms, were analyzed according to the variation of the various genetic parameters, as shown in Figures 5.35 to 5.45 and discussed below. From this analysis we could obtain the most significant parameter values for this type of function.

Although there were no completely successful experiments for the pure GA, we are going to consider the highest performance in the following discussion. For the pure GA algorithm Figure 5.35 shows that the average number of function evaluations spent to generate successful experiments varied from 638.0 to 680.0 for the crossover probabilities of 0.00, 0.15 and 1.0. Crossover probabilities of 0.08, 0.35 and 0.60 could not generate a single high performance experiment. Figure 5.36 does not show a performance improvement when the Fang algorithm was used. The uniform crossover method had the best performance. Avoiding elitism performed better according to the plots in Figure 5.37. The crossover probability did not play an important role as is shown in Figure 5.38. The mutation probability that performed better was 0.25 followed by 0.10 and 0.50 while the values of 0.0 and 0.02 had a poor performance (Figure 5.39).

For the hybrid HGA algorithm Figure 5.40 shows that the average number of function evaluations spent to generate successful experiments varied from 345.86 to 375.64. This range is lower than the one for the pure GA approach and shows that the hybrid GA spent less function evaluations to generate successful experiments

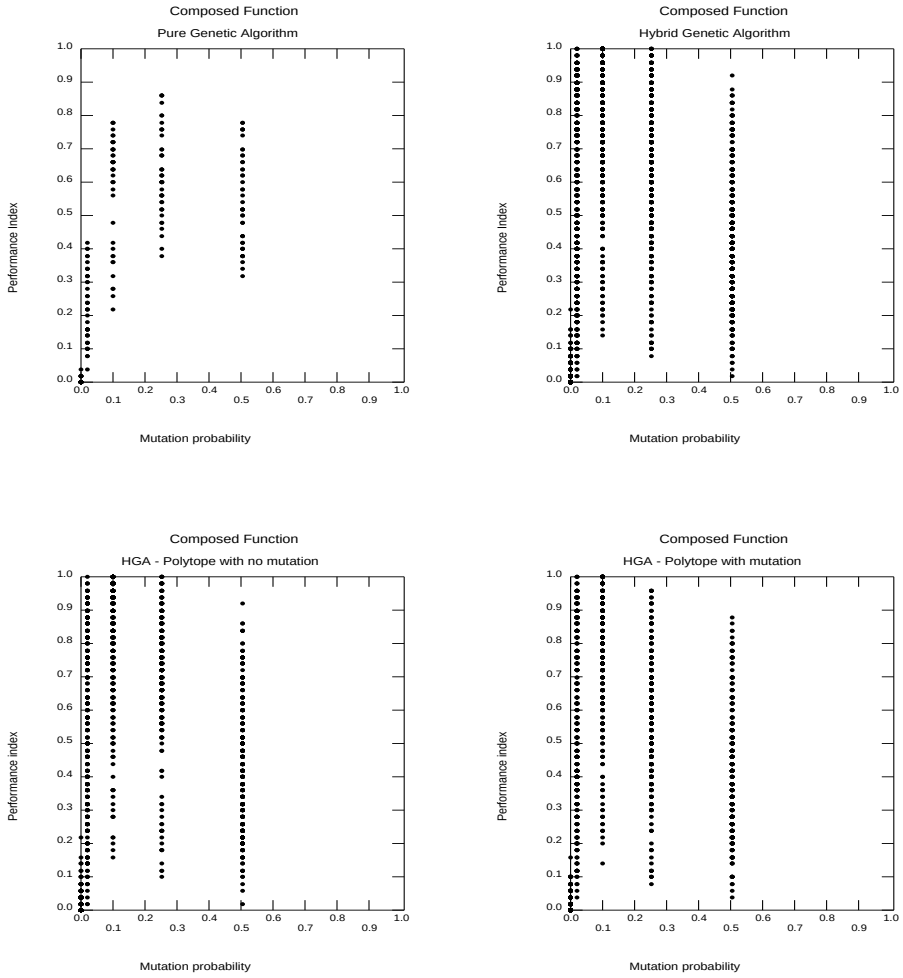

Figure 5.34: GA and HGA compared (Composed)

than the pure GA spent without generating a single successful one. Again, we could not observe any specific influence of the crossover probability. Figure 5.41 shows a slightly significant performance improvement when the Fang algorithm was used. The crossover type did not play an important role for this function. Elitism performed poorer according to the two top plots in Figure 5.42. The remaining plots in this figure show slightly better performance when the polytope is always active as well as a better performance for polytope type 1. Figure 5.43 shows a better performance when the individual generated by the polytope is accepted only if it improves the polytope

fitness and when no mutation is applied to this new individual. The crossover probability does not show a significant influence according to the Figure 5.44. However, the mutation probability played an important role when successful experiments were generated. Mutation rate of 0.10 produced the best experiments followed by the rates of 0.25 and 0.02. The mutation rates of 0.0 and 0.5 could not produce any successful experiment (Figure 5.45).

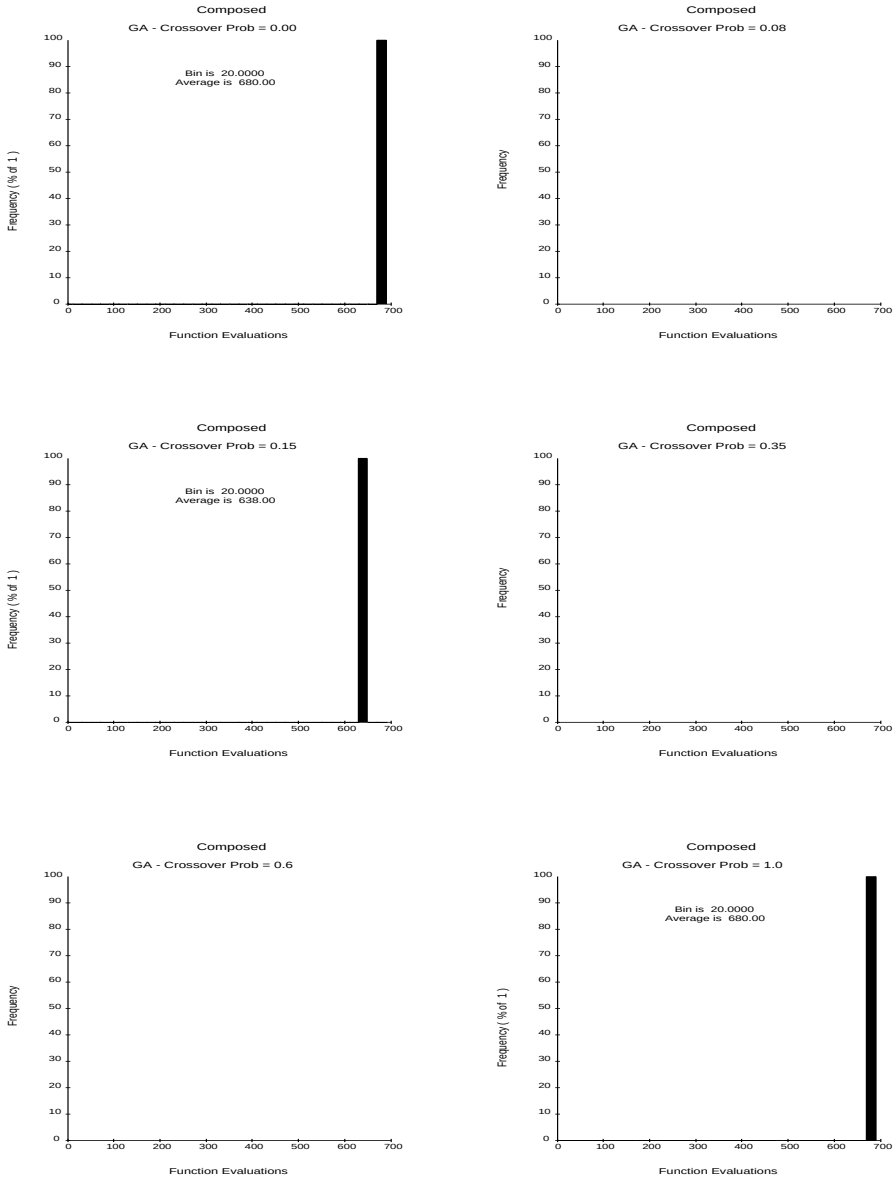

Figure 5.35: Crossover probabilities for high performance experiments (Composed)

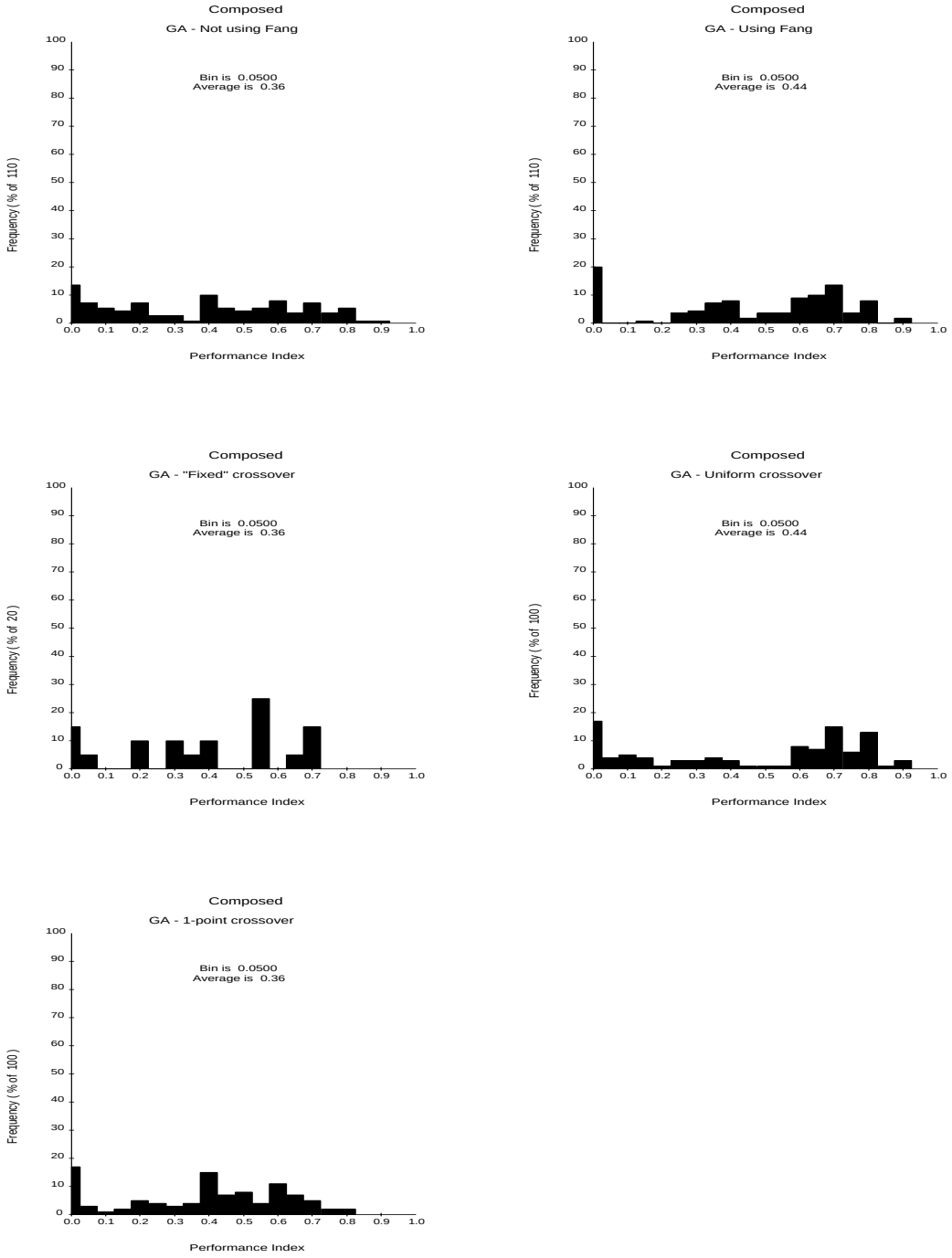

Figure 5.36: Performance index analysis (Composed)

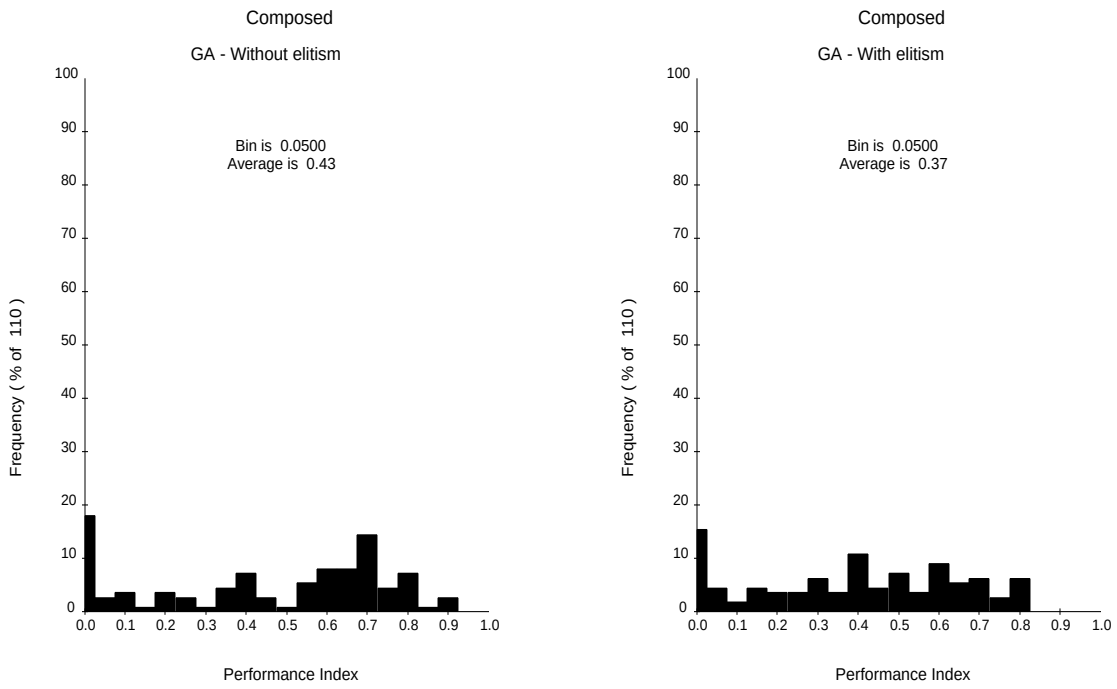

Figure 5.37: Performance index analysis (Composed)

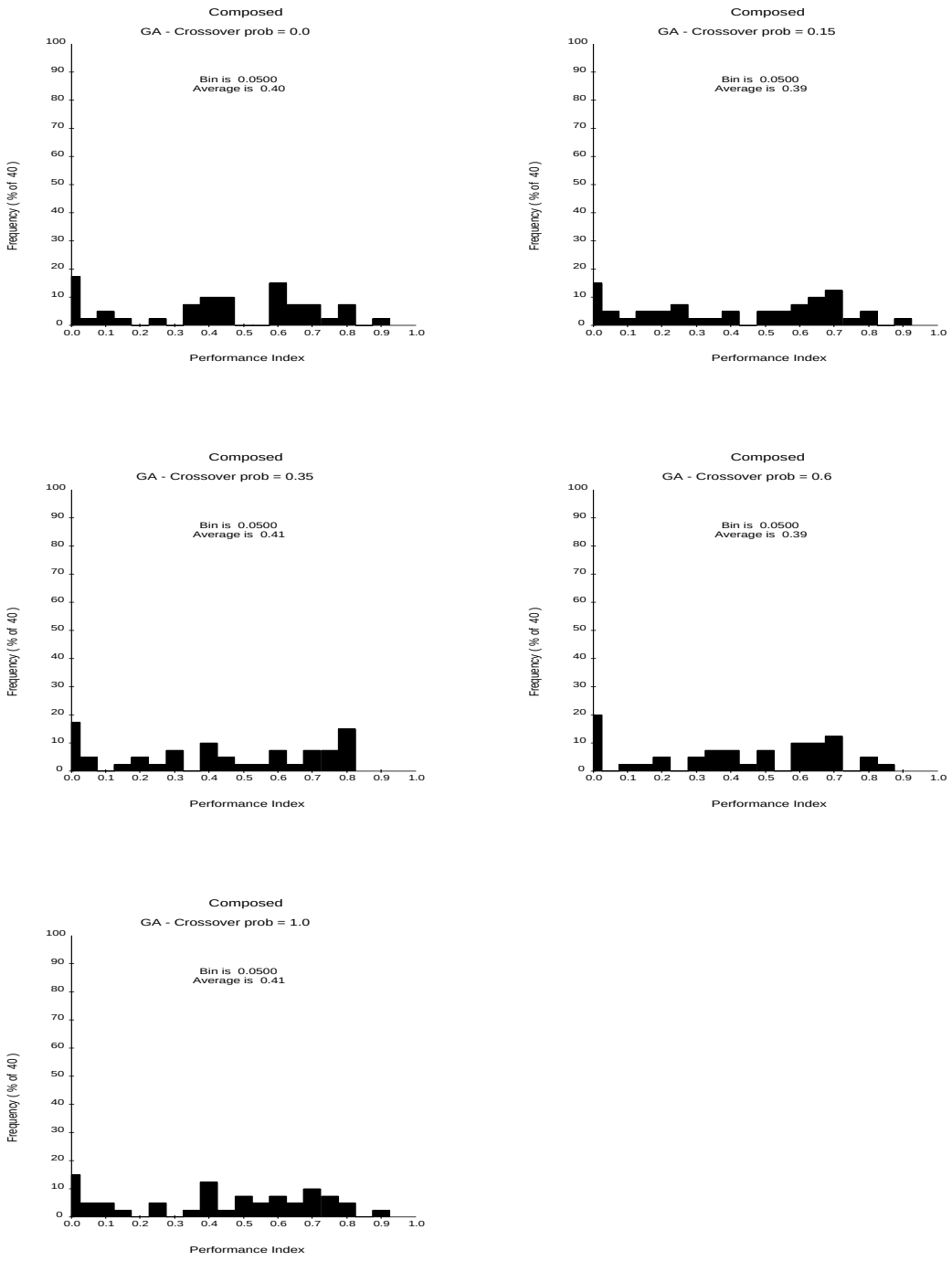

Figure 5.38: Performance index analysis (Composed)

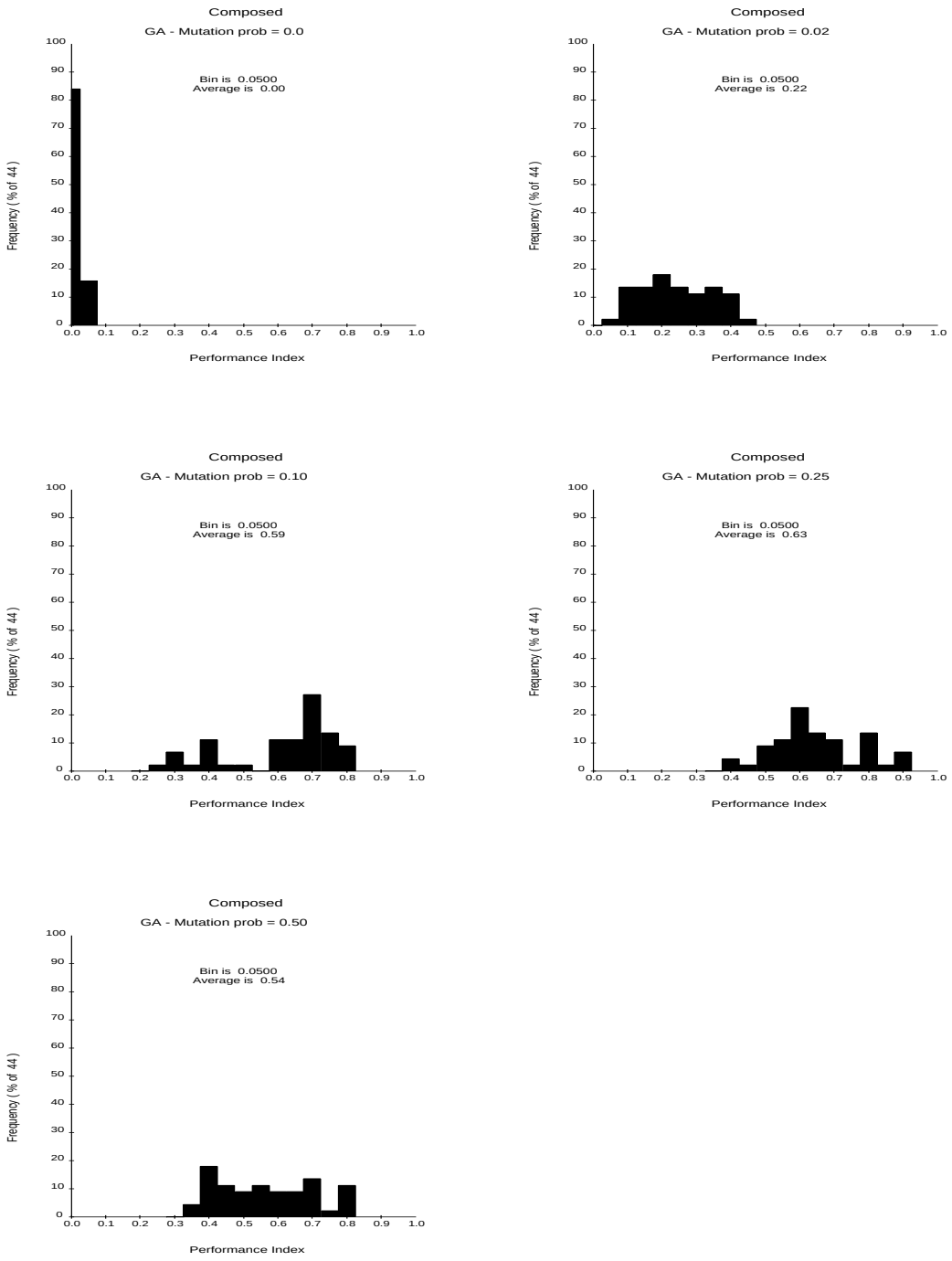

Figure 5.39: Performance index analysis (Composed)

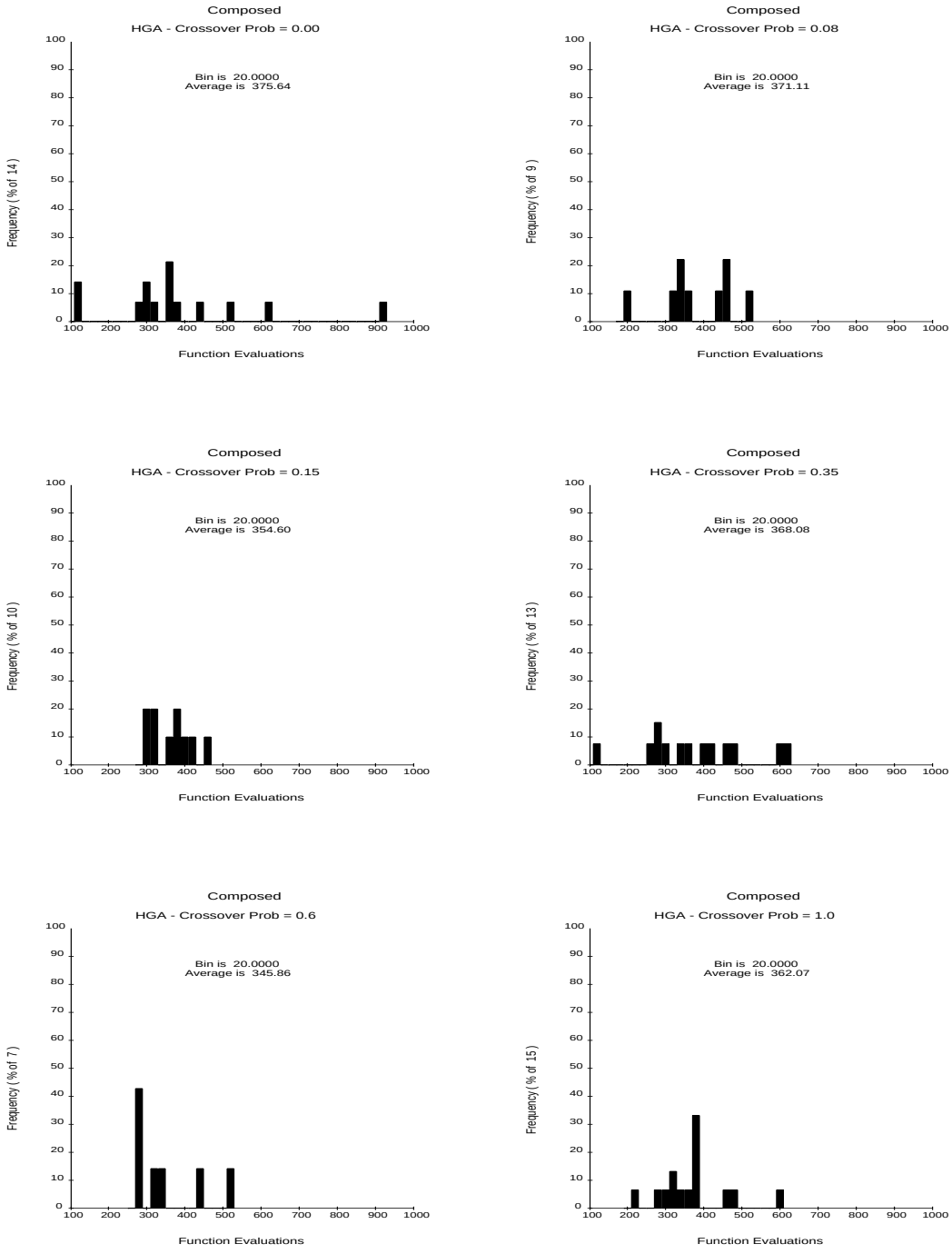

Figure 5.40: Crossover probabilities for successful experiments

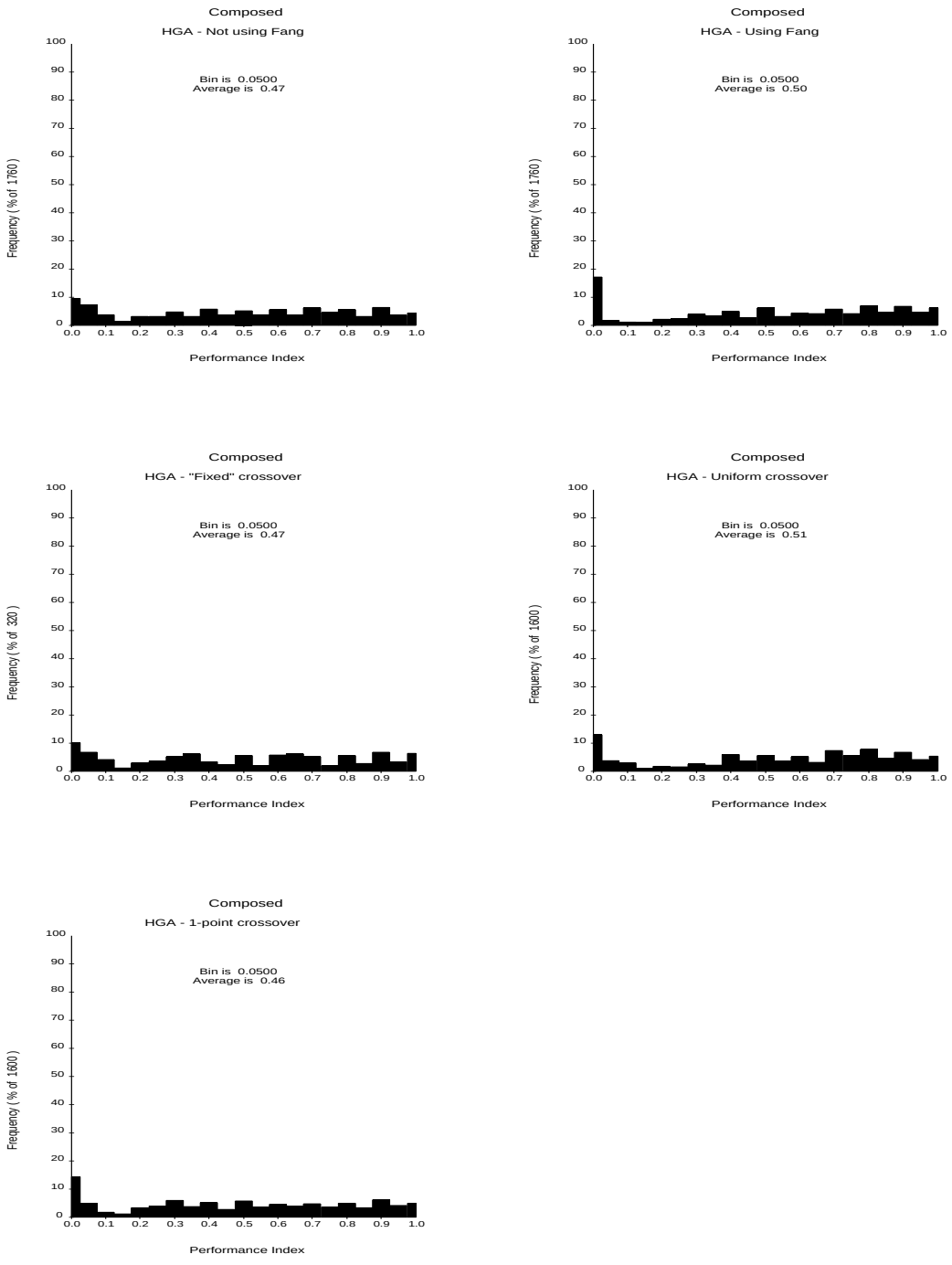

Figure 5.41: Performance index analysis (Composed)

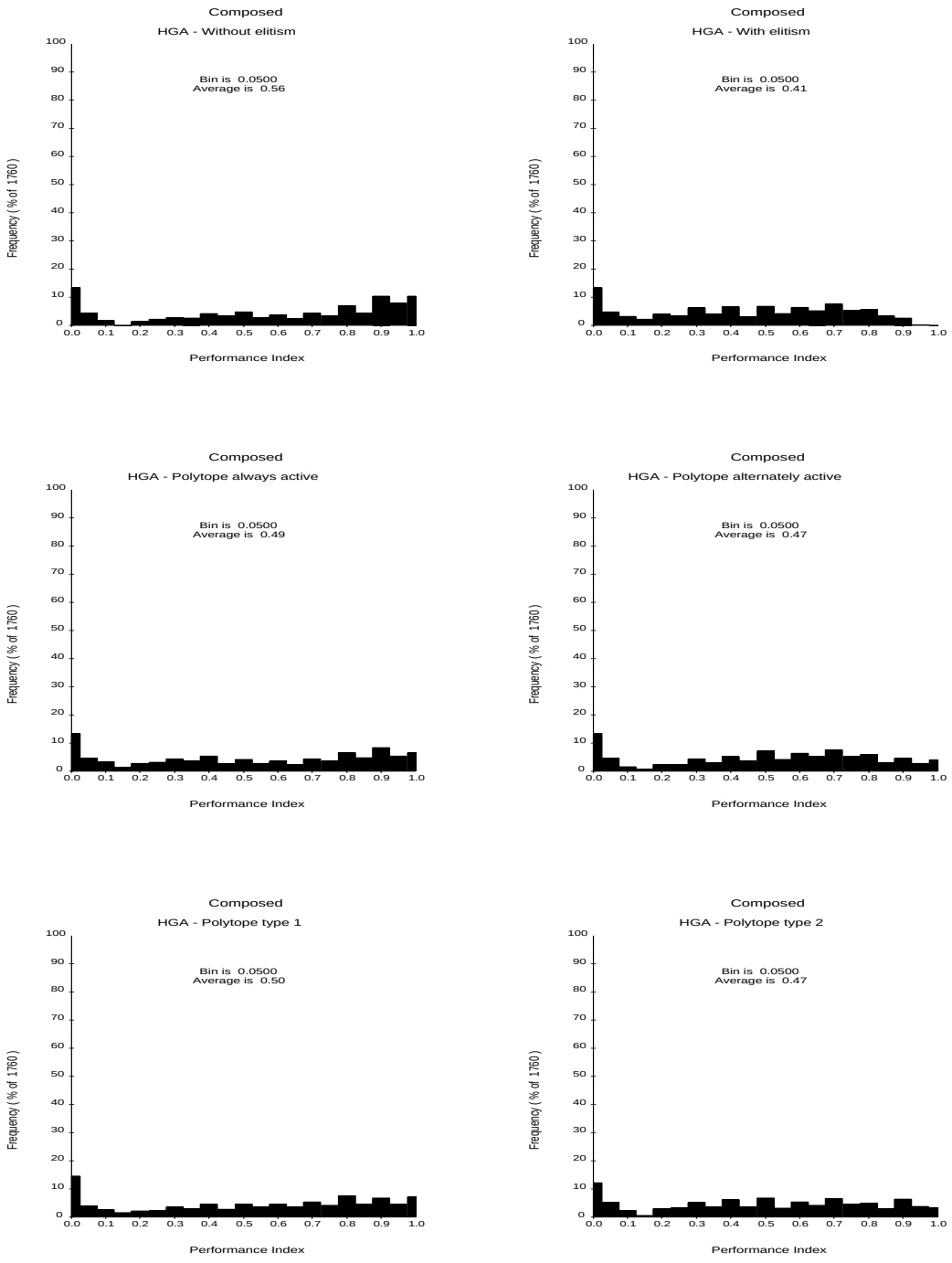

Figure 5.42: Performance index analysis (Composed)

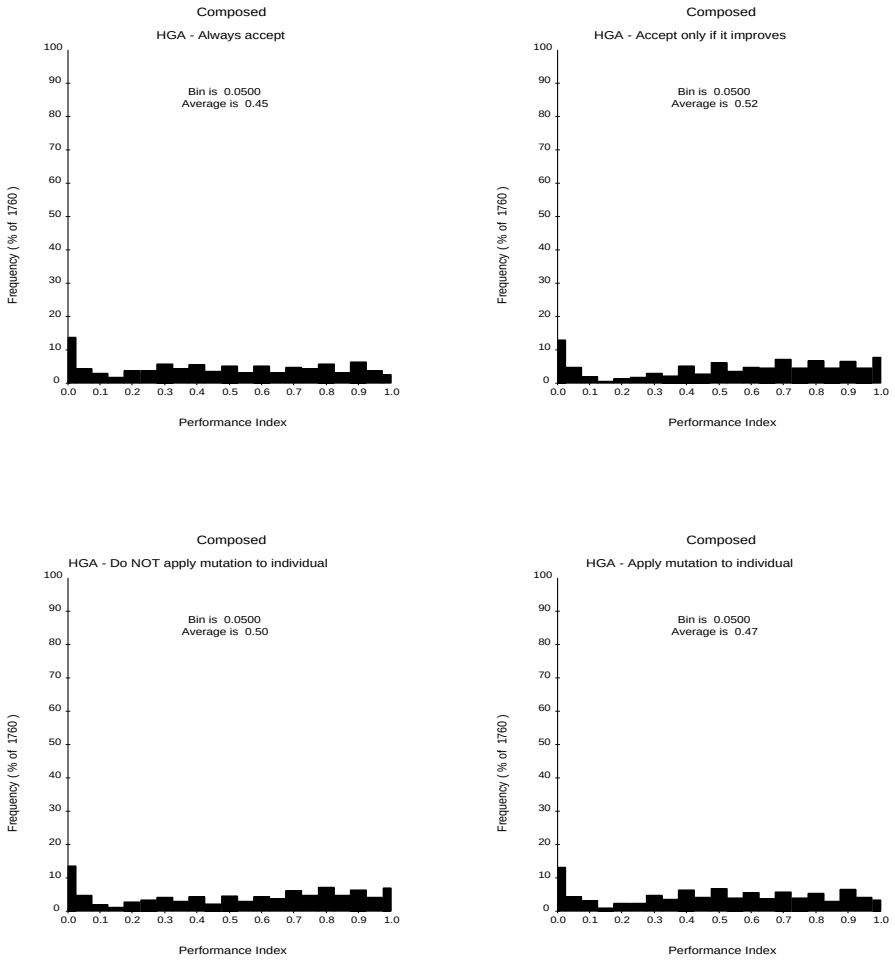

Figure 5.43: Performance index analysis (Composed)

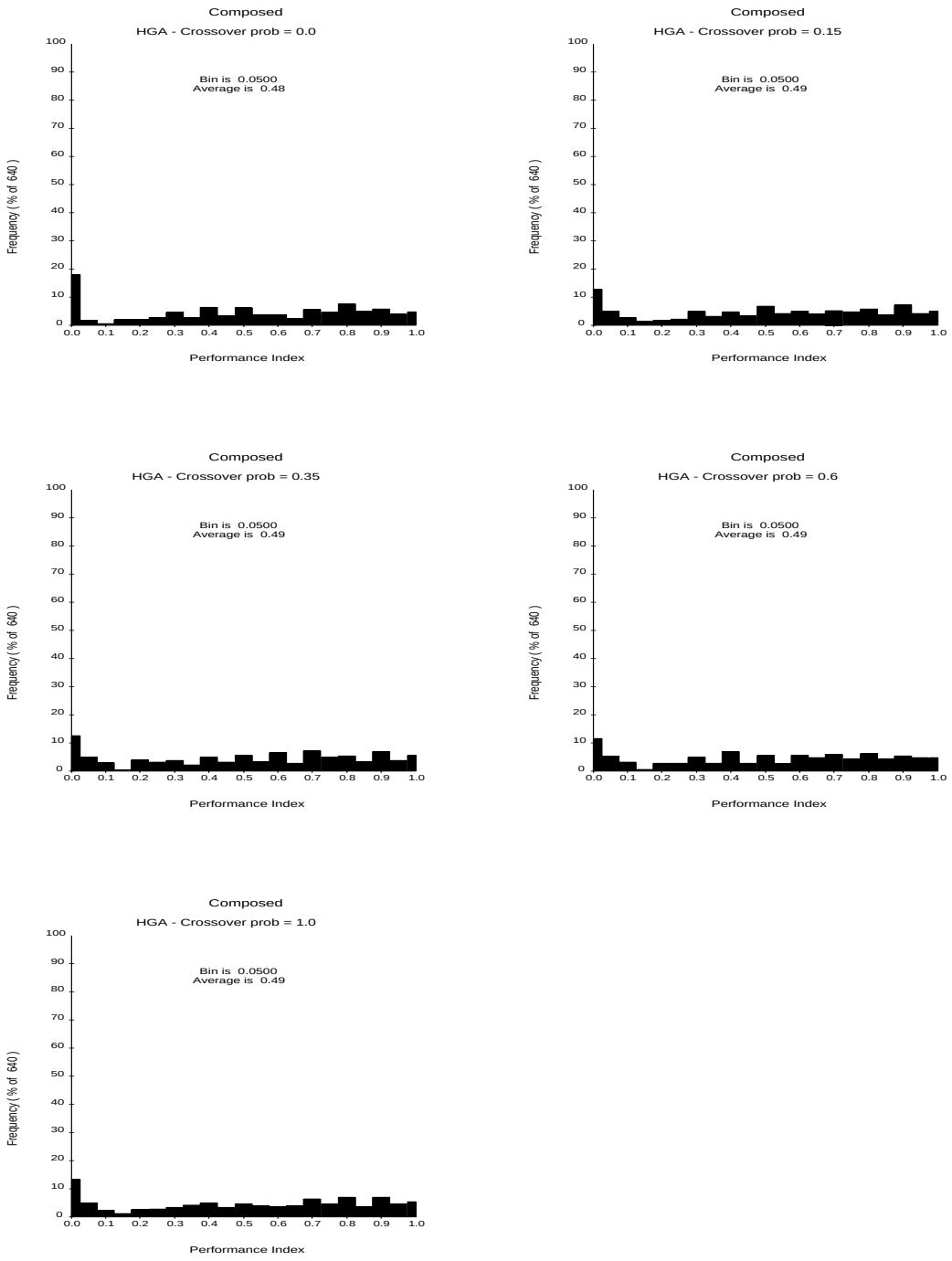

Figure 5.44: Performance index analysis (Composed)

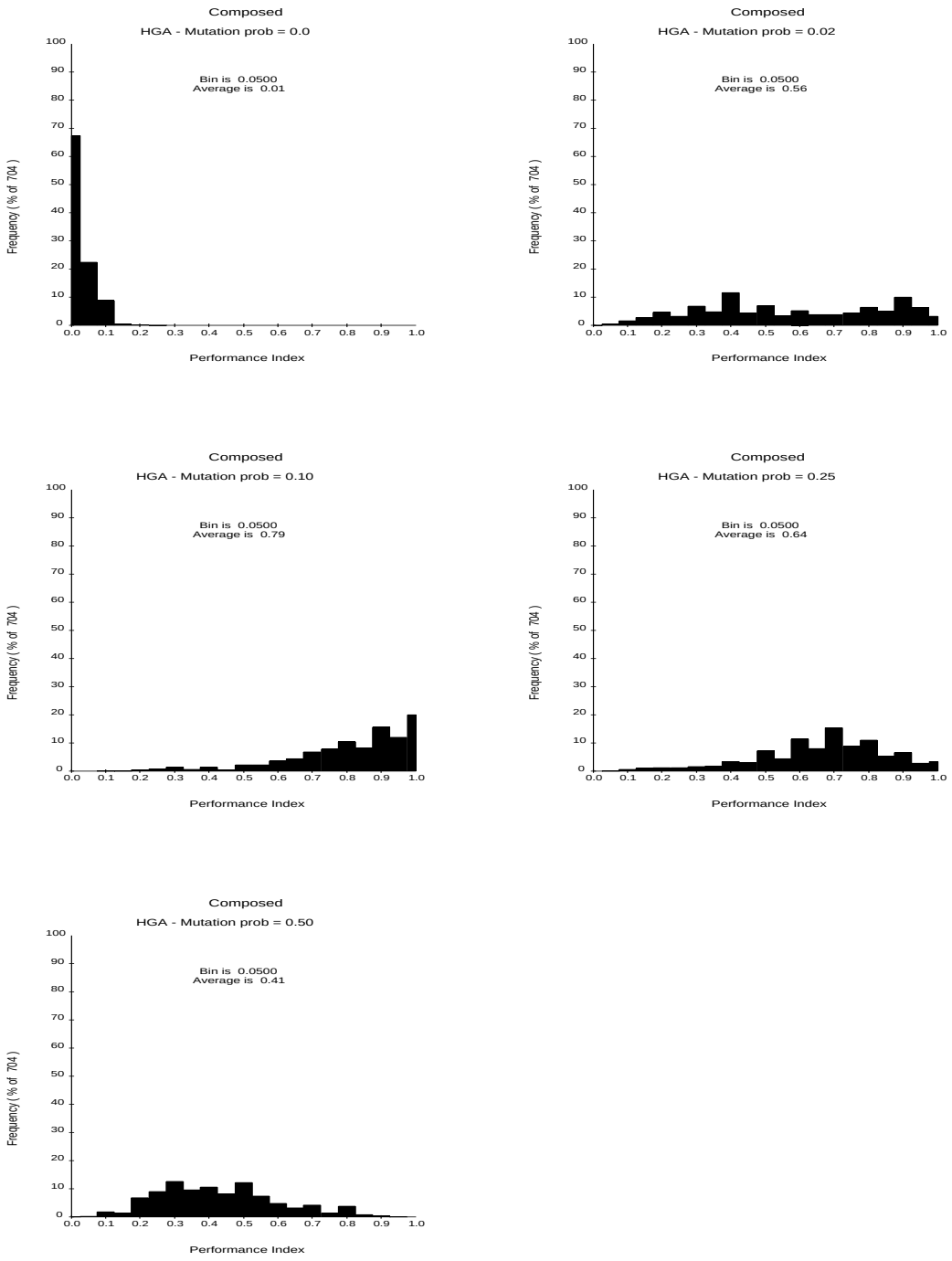

Figure 5.45: Performance index analysis (Composed)

Partial Conclusion

We could see in the three test problems how the hybrid approach improved the pure GA. The polytope method working as a primary selection and crossover mechanism was usually able to produce better results than the GA alone with greater improvement for smoother functions than for rougher ones. The combination of the genetic and hybrid parameters of the best experiments for pure GA and hybrid GA are shown in Tables 5.9 and 5.10, respectively. The top performers for the hybrid GA are summarized in Table 5.11 showing the values of the hybrid parameters.

Func name	Initial pop	Xover type	Save best	Xover prob	Mut prob	Opt gen	Func eval	Perf index
Agnesi	1	1	1	0.3	0.02	16	78	1.00
Shubert	0	1	1	1.00	0.25	37	330	1.00
Composed	0	0	0	0.15	0.25	74	638	0.86

Table 5.9: Best performers for GA (All functions)

Func name	Init pop	Xover type	Save best	Poly actv	Poly type	Poly Effc	Poly mut	Xover prob	Mut prob	Opt gen	Func eval	Perf index
Agnesi	1	-1	1	1	1	1	0	0.06	0.02	17	74	1.0
Shubert	1	-1	1	-2	1	0	1	0.08	0.10	53	290	1.00
Composed	1	1	0	-2	1	1	0	0.60	0.02	41	83	0.98
Composed	1	1	0	1	1	0	1	0.35	0.02	34	107	1.00

Table 5.10: Best performers for HGA (All functions)

Fang's algorithm and polytope type 1 are solid improvements that should be permanently considered in hybrid GA runs. Low mutation rates should also be used as is shown in Table 5.10.

Func name	Init pop	Xover type	Save best	Poly actv	Poly type	Poly Effc	Poly mut	Xover prob	Mut prob	Opt gen	Func eval	Perf index
Agnesi	0	1	0	1	1	1	0	0.35	0.02	14	76	1.00
Agnesi	0	1	1	1	1	1	0	0.00	0.02	17	80	1.00
Agnesi	0	1	1	1	1	1	0	0.60	0.02	17	77	1.00
Agnesi	0	1	1	1	1	1	0	1.00	0.02	16	79	1.00
Agnesi	1	1	0	1	1	1	0	0.15	0.02	20	79	1.00
Agnesi	1	1	0	1	1	1	0	1.00	0.02	18	80	1.00
Agnesi	1	1	1	1	1	1	0	0.15	0.02	19	77	1.00
Agnesi	1	1	1	1	1	1	0	0.60	0.02	20	81	1.00
Agnesi	1	1	1	1	1	1	0	1.00	0.02	19	80	1.00
Agnesi	1	1	1	-2	1	1	0	0.60	0.02	14	77	1.00
Shubert	0	1	1	1	1	0	0	0.35	0.25	31	448	1.00
Shubert	0	1	1	-2	1	0	0	0.60	0.25	40	451	1.00
Shubert	0	1	1	-2	1	0	1	0.00	0.25	44	475	1.00
Shubert	1	-1	0	1	1	0	1	0.08	0.10	48	461	1.00
Shubert	1	-1	1	-2	1	0	1	0.08	0.10	53	290	1.00
Shubert	1	1	0	1	1	0	0	1.00	0.10	44	427	1.00
Shubert	1	1	1	-2	1	0	0	0.35	0.25	43	482	1.00
Shubert	1	1	1	-2	1	0	0	1.00	0.25	34	393	1.00
Shubert	1	1	1	-2	2	1	1	0.60	0.10	68	398	1.00
Composed	1	1	0	1	1	0	0	0.00	0.02	36	109	1.00
Composed	1	1	0	1	1	0	1	0.00	0.02	36	113	1.00
Composed	1	0	0	1	1	0	0	0.60	0.02	48	120	0.98
Composed	1	1	0	1	1	0	0	1.00	0.02	39	118	0.98
Composed	1	1	0	1	1	0	1	0.60	0.02	32	115	0.98
Composed	1	1	0	-2	1	0	1	1.00	0.02	56	128	0.98
Composed	1	1	0	-2	1	1	0	0.60	0.02	41	83	0.98

Table 5.11: Top performers for HGA (All functions)

Chapter 6

Petroleum Production Optimization

Based on the results obtained using the test functions, we designed a hybrid algorithm for application to a petroleum engineering optimization problem. We choose the hybrid parameter values that produced the best performance for the three test function as shown in Table 5.10.

6.1 Problem Statement

A real case of an offshore oil reservoir development project was optimized. The optimizing algorithm was allowed to run considering two cases: in the first one the solution proposed by the project design team was not included in the initial HGA population and in second one the proposed solution was considered to be known. The restrictions applied in the real project were also considered in the optimizing runs. Injector and producer wells had the same domain and the optimization algorithm determined their best location. The solution found had no restrictions concerning well spacing and injector/producer patterns.

The reservoir has three layers: the top one with gas, the middle one with oil and the bottom one with water. The three parts of the reservoir are connected to each other in the eastern part of the field and were modeled using a 40x80x3 grid with

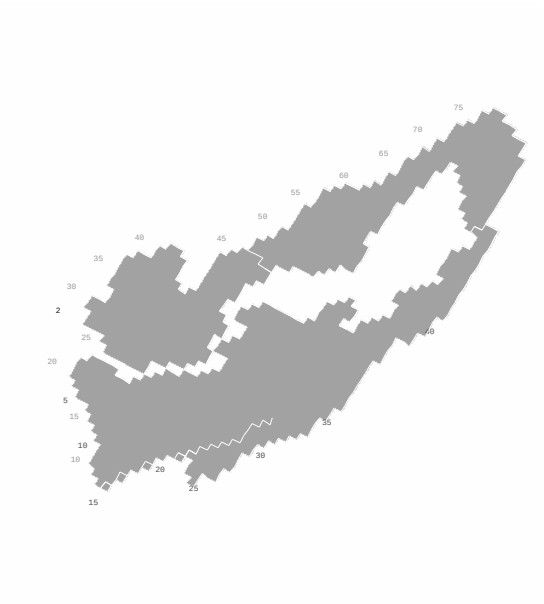

Figure 6.1: Sealing faults in the oil layer

a commercial three-dimensional, three-phase flow simulator (ECLIPSE) to generate the production profile. Because of existing sealing faults in the oil layer (Figure 6.1) two independent accumulations, AREA-1, the larger one, and AREA-2, were the only areas considered for development. The region of interest is shown in Figure 6.2 although the entire system was considered in the simulation runs. As the wells will produce to the same production facilities installed on a single platform the wells of both areas were submitted to the same field constraints. AREA-1 can accommodate 14 oil producing wells and 12 water injectors and AREA-2 can accommodate three oil producers and four water injectors (Table 6.1). The well layout proposed by the project design team is shown in Figure 6.3.

The oil layer is saturated with 20% connate water and no gas. The reservoir body is a sandstone with permeabilities varying from 1000 md to 2000 md, and porosities varying from 0.20 to 0.25. The oil has a variable API gravity which affects the calculation of the pressure loss in the multiphase flow in pipelines. The real project development plan proposed the location of 33 wells in addition to the two existing wells, LOC-1A and LOC-2H, which were not considered as parameters for optimization although they were specified in the simulation runs. The well LOC-1A

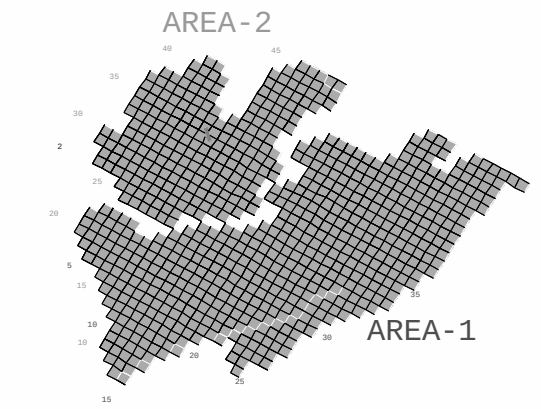

Figure 6.2: Region of interest in the oil layer

Type	Producers		Injectors		Total
Direction	vertical	horizontal	vertical	horizontal	
AREA-1	11	3	5	7	26
AREA-2	0	3	0	4	7
subtotal	11	6	5	11	33
Total	17		16		33

Table 6.1: Proposed well distribution

started producing 36 months before the other wells and is going to be connected to the planned permanent facilities as soon as the development plan is implemented. As the sea bottom lines connecting the sea bottom wellheads to the platform are expensive, the platform location was also optimized in a secondary optimization task. This secondary optimization also considered that the bottom sea flowlines should approach the platform avoiding the anchor lines. A simplified way to solve this problem was to evaluate the flowline length using a Bezier curve drawn from the bottom sea wellhead to the platform location.

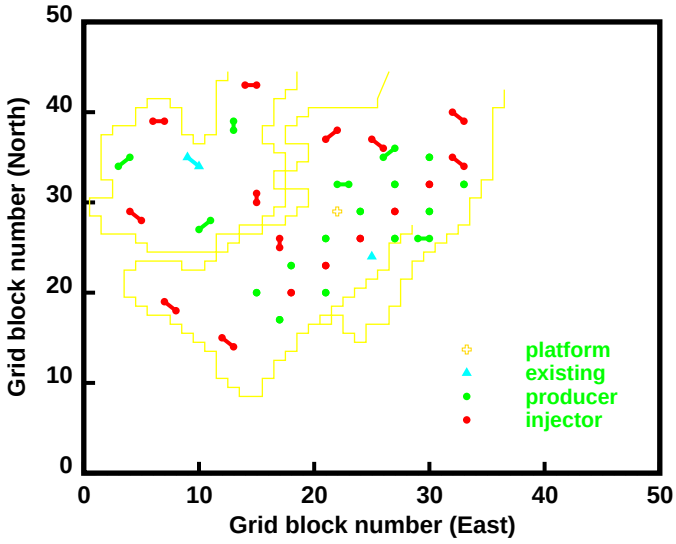

Figure 6.3: Proposed well locations

The objective of the optimization work was to better locate the wells and the platform subject to the given schedule in the two different accumulations. The objective function had to handle some simulation issues because the available simulators were not prepared for optimization tasks. Some features that the simulators would require to be more suitable for optimization work can be listed as:

1. Well properties and constraints assigned by geological region;
2. Well location specified by sequence domain. This domain is specified by geological region;
3. Generation of return codes reporting the ending status of the simulator;
4. Verticalization/shrinkage of horizontal wells when completions occur in the same block;
5. Wells defined by head location, orientation, length and/or parametric curve (two-dimensional/three-dimensional);

6. User-defined action when a predictable misuse occurs, e.g., a well is completed in some wrong way (out of the grid, same location as another);
7. Injection/producer specification by geological region;
8. Production/economic criteria defined by geological region;
9. Time specification of some items not associated to the schedule data structure;
10. Flexible drilling queue allowing changes in the sequence of the queue;
11. Restart file should have a "save report" feature because the data do not accumulate information from the previous steps;

Well locations given by a pair of (i, j) coordinates were not suitable decision variables for the optimization problem because the optimizer could place wells even outside the reservoir. To handle this issue the feasible domain for well placement was considered to be a vector with the indices of the active cells only. An additional one grid block strip was taken out all around the two accumulations to avoid having horizontal wells with their head in the reservoir and the tail outside. In this way, the first domain represented AREA-1 with 369 grid blocks, the second one represented AREA-2 with 140 grid blocks and the platform domain was considered to be the union of both plus the intermediate cells with a total of 785 grid blocks. The well scheduling was the same as that in the real project because it could not be changed due to rig availability. So, the optimization had to determine the best well completion sequence for the given well schedule. Another restriction imposed by real limitations was the horizontal well length, of about 820 feet, requiring two adjacent grid blocks. Also, all wells were completed in the oil layer only, even the injector wells because the water layer does not communicate with the oil layer in the western part of the reservoir.

In the optimization task any well was allowed to be vertical or horizontal and, if horizontal, to have any orientation in the oil layer. From the GA point of view we used three parameters per well: well location, well direction (vertical or horizontal) and

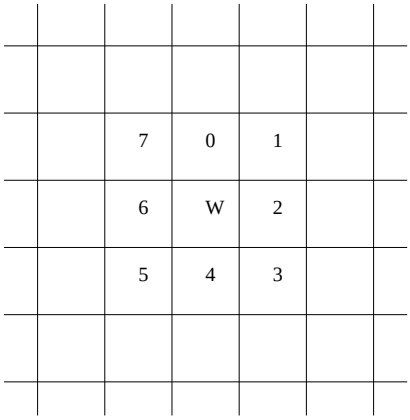

Figure 6.4: Codes for horizontal well orientation

horizontal well orientation (Figure 6.4). To represent the parameter values of AREA-1 wells in the binary form 13 bits were required: nine bits for the well location, one for the direction and three for the orientation. For AREA-2 wells 12 bits were required because the range of the first parameter is smaller for this area (Table 6.2).

Parameter	Range	required bits
AREA-1 well location	1-369	9
well direction	0-1	1
horizontal well orientation	0-7	3
Total for one well in AREA-1		13
AREA-2 well location	1-140	8
well direction	0-1	1
horizontal well orientation	0-7	3
Total for one well in AREA-2		12

Table 6.2: Bit consumption per well

A chromosome length of 422 was required to represent the 33 wells as summarized in Table 6.1. Using the results obtained from the test functions the following genetic

parameters were adopted: initial population generated by Fang's algorithm, single-point crossover, crossover probability of 0.6, mutation probability of 0.05, elitism, memory used for sets of parameters with exact match (same as zero radius), polytope type 1 activated every other generation with one movement allowed and no mutation applied to the generated individual. The Tabu Search strategy was not used. An important matter was the population size to be used. If we had adopted the same strategy used in the test function runs then a population size equal to the chromosome length would be used, namely 422. On the other hand, the polytope requires that the minimum population size must exceed the number of parameters by one, which is 34. Considering that each simulation run was taking about 2 minutes on a DEC ALPHAstation 500 (400 MHz) then it would take about 14 hours to run one single generation, or less than 2 generations per day. As this example had to optimize a real project it could take weeks to find a better well distribution. However as the polytope was expected to improve the search, we were able to work with smaller population sizes, which was advantageous for such costly function evaluation problems. We tried a population less than the half of the chromosome length and adjusted it to the Fang algorithm requirement for populations greater than 100. A population size of 198 was used to handle the problem. The fact of having the well domain as a vector with the indices of the active cells caused Fang's algorithm not to provide an evenly spaced distribution concerning the well placement on the three-dimensional grid. Even so, Fang's algorithm produced a better initial generation than the random approach.

In order to prepare the input data to the simulation run the hybrid parameters were translated into the proper format required by the simulator. At that time, when the well locations were mapped from the sequentially ordered vector into the (i, j) grid coordinates, the platform location optimization took place by calculating the minimal total sea bottom line length connecting each well to the platform. This minimal total length would be considered later in the economic analysis.

How the simulation results were retrieved by the objective function and how the economic analysis was performed is described next. An interface module was required to analyze how the simulation run ended because this job was run in a network of UNIX workstations where the programs are installed in different computers and the

Single valued economic parameters		
remoteness		mature area
production system type		platform
environment type		benign
factor 1		1.00
factor 2		1.00
factor 3		1.00
interest per year	%	10.00
platform cost	MM US$	250.00
production facilities cost	MM US$	70.00
pipeline cost	MM US$	45.00
rig mobilization cost	US$	1,000,000.00

Unitary valued economic parameters		
oil price	US$/BBL	18.00
gas price	US$/MSCF	2.00
water price	US$/BBL	1.00
oil handling cost	US$/BBL	1.00
rig day rate	$/day	70,000.00
oil production operational cost	US$/well/month	5,000.00
water injection operational cost	US$/well/month	1,500.00
intangible drilling cost	MM US$/well	7.50
tubing cost	M US$/well	255.00

Table 6.3: Economic factors for a typical configuration

simulator does not issue any ending status code. So, after each simulation run the interface module analyzes the files generated by the simulator and signals to the optimizer if the simulator had performed well. If any problem occurred with the simulation run then the job was stopped. If not, another program retrieved the results file generated by the simulator and translated the data into a proper format required by the objective function. This was the production profile for oil, gas and water for the entire run. Because the well schedule for this case was fixed it was supplied in a separate file although the interface program was able to detect some information about well scheduling. A more sophisticated interface program could fully describe well activities but an additional feature within the simulator could provide

this information with minimal effort.

The economic analysis was based on the tables provided by Kennedy [1993]. The production system type (platform) cost was evaluated considering the available infrastructure for the area to be developed and the kind of environment (benign or harsh). In our case, the area has a good infrastructure and the environment is benign. For this situation the platform cost was evaluated as US$ 250 million. This value along with the production facilities cost and sea bottom pipeline cost composed the *base cost*. When the production profile was retrieved by the objective function the highest production levels for oil, gas and water were detected. These values were used as input to obtain the production facilities cost from the proper table for each phase. The total production facilities $TPFC$ cost was then evaluated by:

$$TPFC = (Oil\ Facilities\ Cost) + (Water\ Facilities\ Cost) + (Gas\ Facilities\ Cost) \tag{6.1}$$

The cost for the sea bottom flowlines was obtained from the pipelines cost table by setting the optimally evaluated total length of sea bottom flowlines. The total base cost TBC is given by:

$$TBC = (base\ cost) + (sea\ bottom\ pipeline\ cost) \tag{6.2}$$

Using a project interest rate of 10% per year the *total base cost* payment was considered to be paid over two years before the production started according to the following payment plan:

- 30% of the *total base cost* at the beginning of the payment period
- remaining 70% distributed along 23 monthly installments using the rate:

$$interest\ per\ month = (interest\ per\ year + 1)^{\frac{1}{12}} - 1 \tag{6.3}$$

The production facilities followed the same payment plan but over one year period.

The next step was to evaluate the drilling and production cost and income due to production for each time step of production activity. The production rates for oil, water and gas for the current time step were obtained from the production profile

table generated by the simulator. The costs associated to this production level was then computed as:

$$period's\ cost = (oil\ production\ cost) + (gas\ production\ cost) + (water\ production\ cost) \tag{6.4}$$

where:

oil production cost is equal to:

$$(oil\ handling\ cost) \cdot (time\ interval) \cdot (oil\ rate) \cdot (1 + overhead\ factor) + (insurance\ cost) + isl \tag{6.5}$$

In this equation the *overhead factor* accounts for supervision and administrative costs, *insurance cost* accounts for the facilities insurance for the period and *isl* accounts for royalties and governmental taxes (local) and is given by:

$$isl = 0.05 \cdot (oil\ price) \cdot (1 - f) \cdot (time\ interval) \cdot (oil\ rate) \tag{6.6}$$

where f is a factor that accounts for average transportation cost (local).

gas production cost. As the handling cost for gas is low we accounted only for the royalties and governmental taxes. The gas production cost is given by:

$$0.05 \cdot (gasprice) \cdot (1 - f) \cdot (time\ interval) \cdot (gas\ rate) \tag{6.7}$$

water production cost. The handling cost for water was considered to be one tenth of the same cost for oil. The water production cost is given by:

$$\frac{(oil\ handling\ cost)}{10} \cdot (time\ interval) \cdot (water\ rate) \cdot (1 + overhead\ factor) \tag{6.8}$$

The three costs above composed the production cost and the income generated by the production in each time step was then computed by:

$$income = \{(oil\ rate) \cdot (oil\ price) + (gas\ rate) \cdot (gas\ price)\} \cdot time\ interval \tag{6.9}$$

The well drilling costs were evaluated based on the number of vertical and horizontal wells set by the hybrid parameter set. Horizontal wells were considered to be 50% more expensive to drill than the vertical ones. This is an intermediate value in the normal range of 1.3 to 1.8 and because the horizontal well length did not vary significantly this assumption seemed reasonable. Because we assumed this fixed factor the cost for drilling wells was calculated by:

$$(number\ of\ wells) \cdot (intangible\ drilling\ cost + completion\ cost) \cdot facw \tag{6.10}$$

where:

intangible drilling cost is equal to:

$$rig\ capacity \cdot (rig\ day\ rate + support\ and\ supervision) \tag{6.11}$$

where *rig capacity* was obtained from the proper table after setting the reservoir depth, *rig day rate* was user supplied and *support and supervision* depend on the *remoteness*;

completion cost was obtained from the proper table after setting the average well depth;

facw is the factor that increases the unitary drilling well cost due to horizontal wells and is given by:

$$\frac{(number\ of\ vertical\ wells) + 1.5 \cdot (number\ of\ horizontal\ wells)}{(number\ of\ vertical\ wells) + (number\ of\ horizontal\ wells)} \tag{6.12}$$

The last factor to be accounted was the minimum cost per well. This costs is applied even if the well is not producing or injecting. For producing wells it is given by:

$(operational\ cost\ per\ well) \cdot (time\ interval) \cdot$

$$(number\ of\ producers) \cdot (1 + overhead\ factor) \tag{6.13}$$

Injecting well operational cost was considered to be 30% of the operational cost for producing wells.

After having all these economic factors calculated and located in time the cash flow was calculated correcting the values to the reference time. This was the objective function value passed to the hybrid GA algorithm.

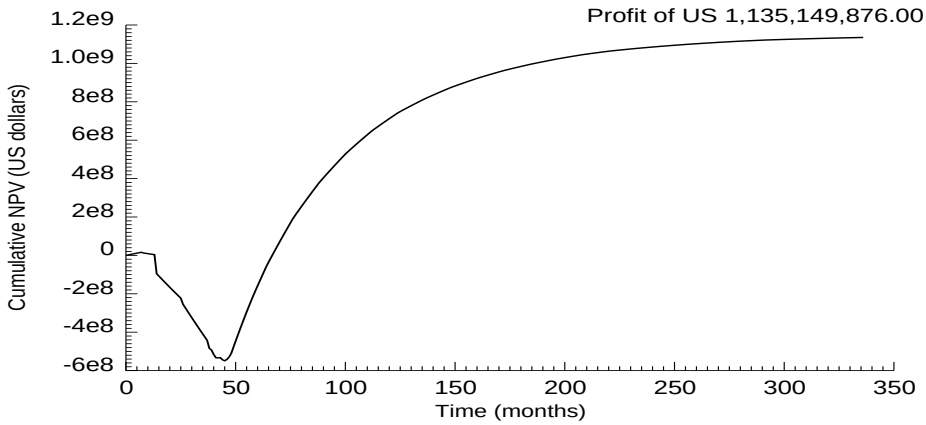

Figure 6.5: Cash flow for the proposed solution

Applying this methodology to the proposed solution we obtained the reference for the optimization task as being equal to the profit of US$1,135,149,876.00 at the end of the project life of 336 months (Figure 6.5). At this time the cumulative production of oil, gas and water were 269,851 x 10^3 STB, 119,833 x 10^6 SCF and 238,029 x 10^3 STB, respectively.

Using the optimization approach that did not initially include the proposed solution the algorithm generated a profit of US$1,189,921,627.00 with the well distribution as shown in Figure 6.6 and summarized in Table 6.4.

Type	Producers		Injectors		Total
Direction	vertical	horizontal	vertical	horizontal	
AREA-1	8	6	6	6	26
AREA-2	2	1	2	2	7
sub total	10	7	8	8	33
Total	17		16		33

Table 6.4: Optimized well distribution (the proposed solution was not known)

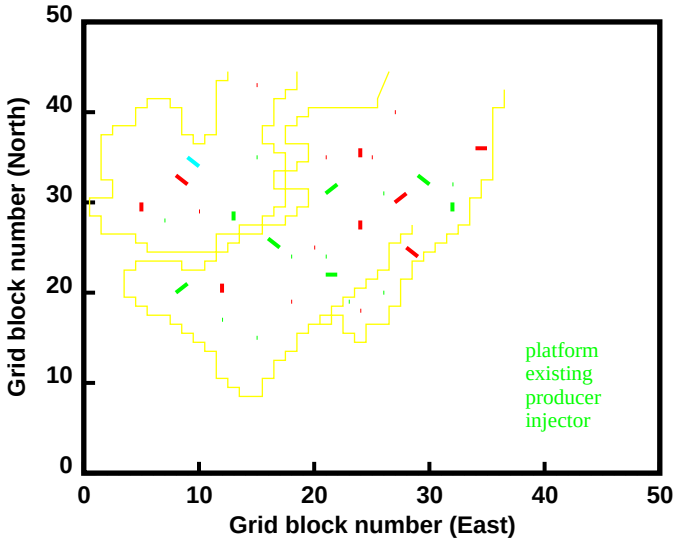

Figure 6.6: Optimized well locations (the proposed solution was not known)

The net present value for this approach is compared to the net present value of the proposed solution in Figure 6.7. Total incremental improvement was US$54.8 million (4.82%). The cumulative oil production for both strategies is shown in Figure 6.8.

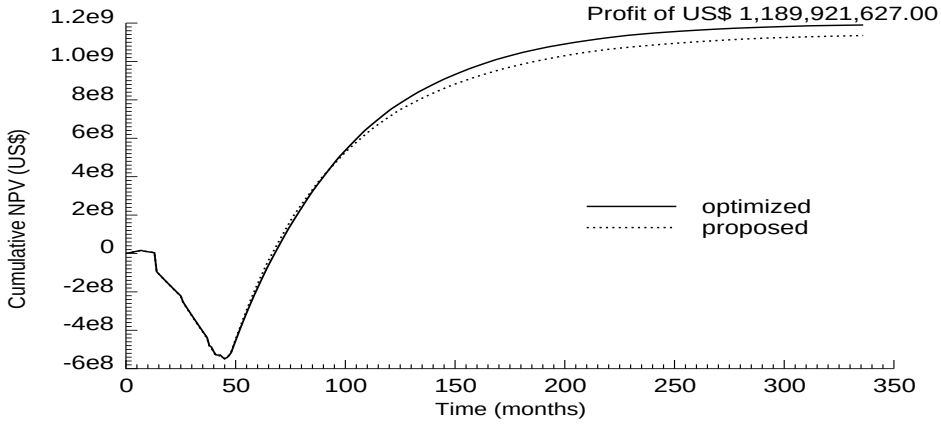

Figure 6.7: Optimized cash flow (the proposed solution was not known)

The next ten best trials (individuals) made by the algorithm to achieve the current optimal solution can be observed in Figures 6.9 through 6.18. In these poorer attempts

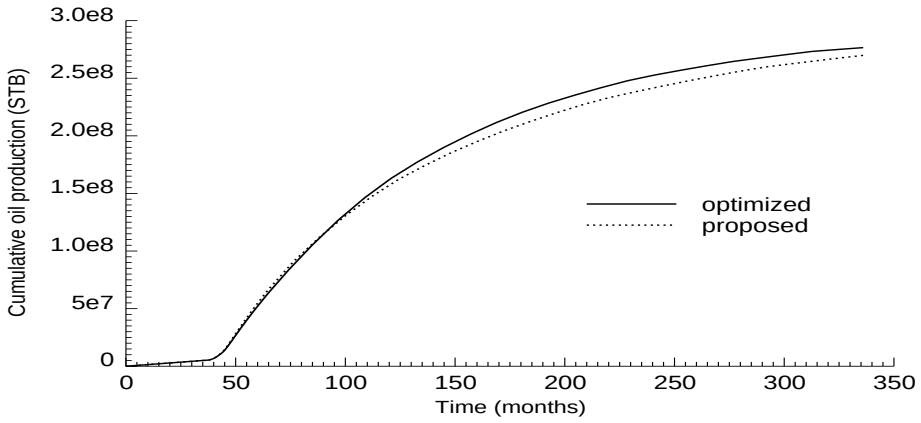

Figure 6.8: Optimized cumulative oil production (the proposed solution was not known)

we could observe wells completed very close to the faults or in the same location or with a inappropriate distribution. The other 187 trials are not shown.

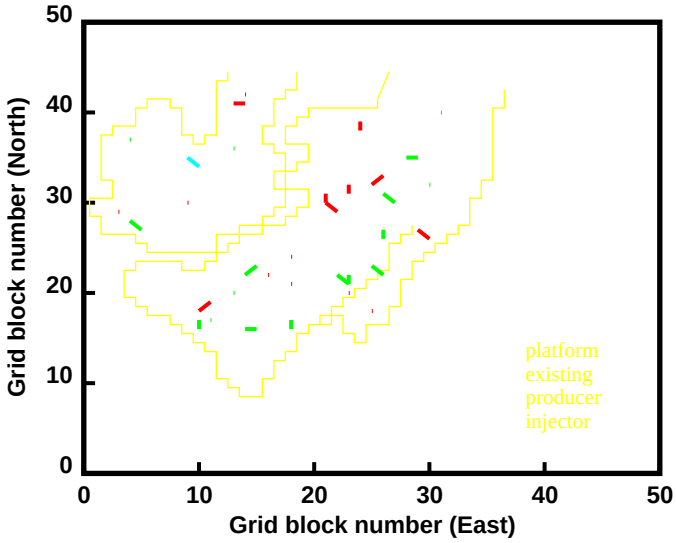

Figure 6.9: Well distribution generating a profit of 1.147 x 10^9

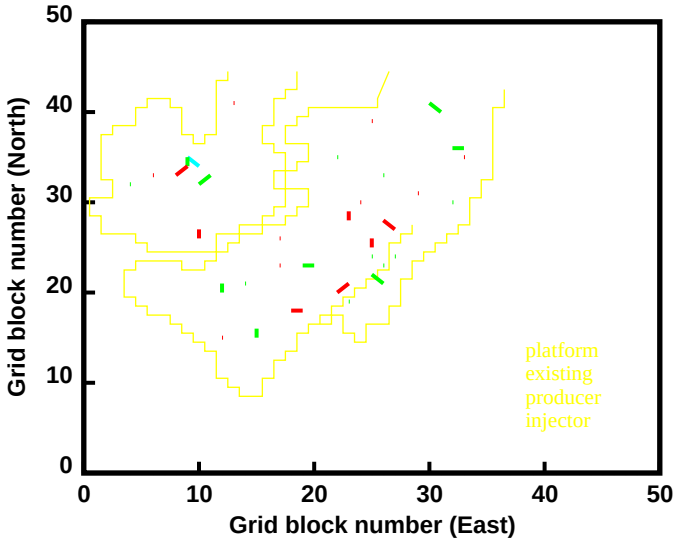

Figure 6.10: Well distribution generating a profit of 1.112 x 10^9

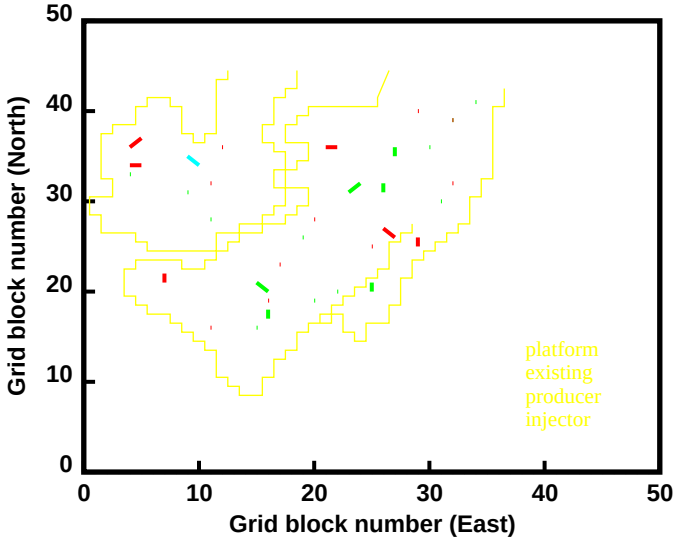

Figure 6.11: Well distribution generating a profit of 1.088 x 10^9

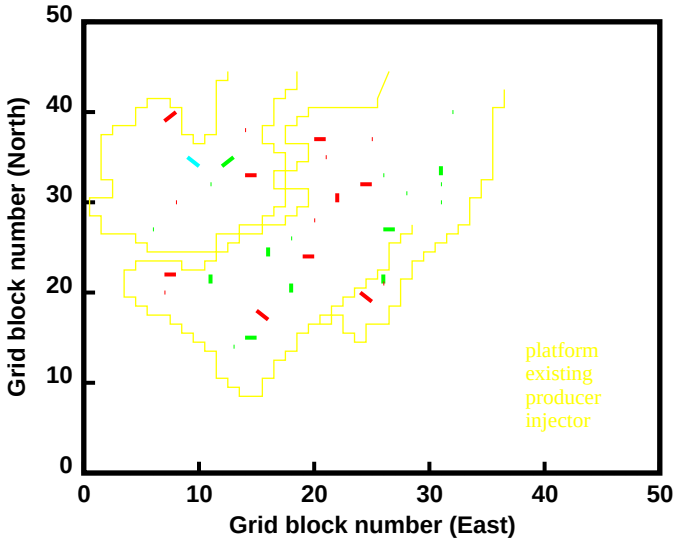

Figure 6.12: Well distribution generating a profit of 1.048 x 10^9

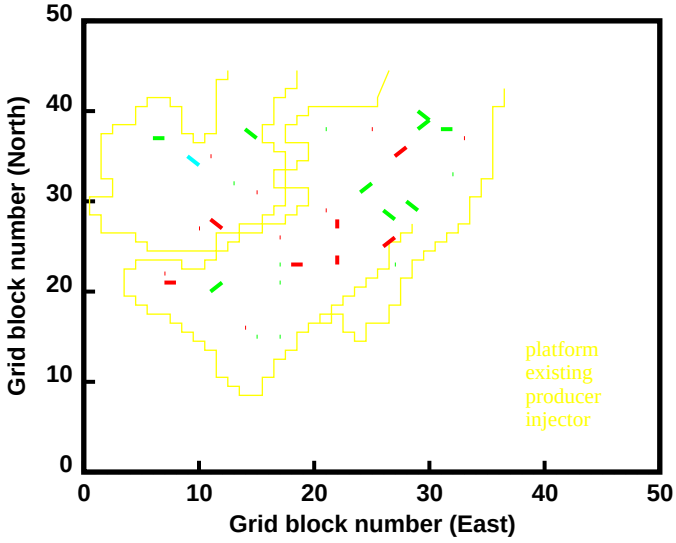

Figure 6.13: Well distribution generating a profit of 1.040 x 10^9

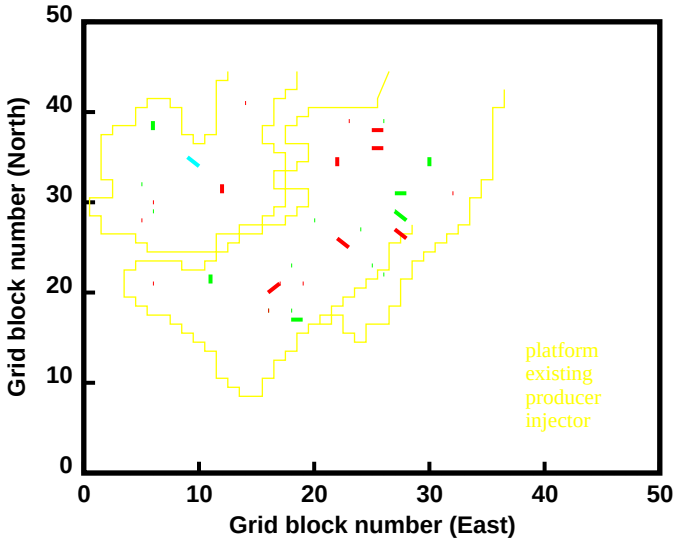

Figure 6.14: Well distribution generating a profit of 1.038 x 10^9

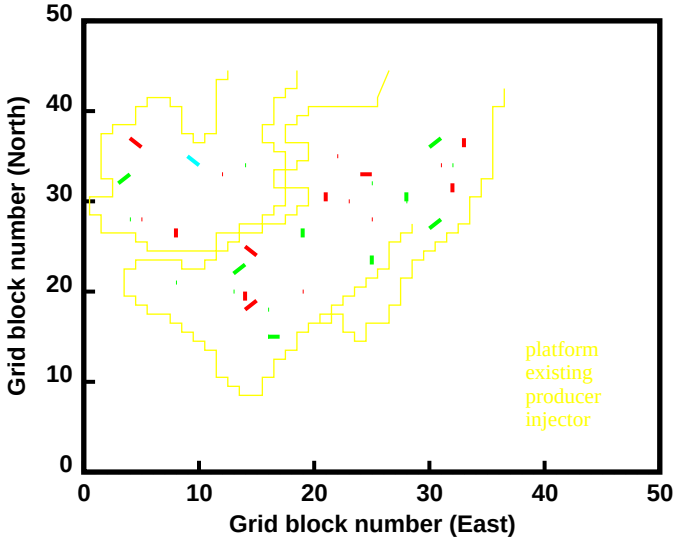

Figure 6.15: Well distribution generating a profit of 1.035 x 10^9

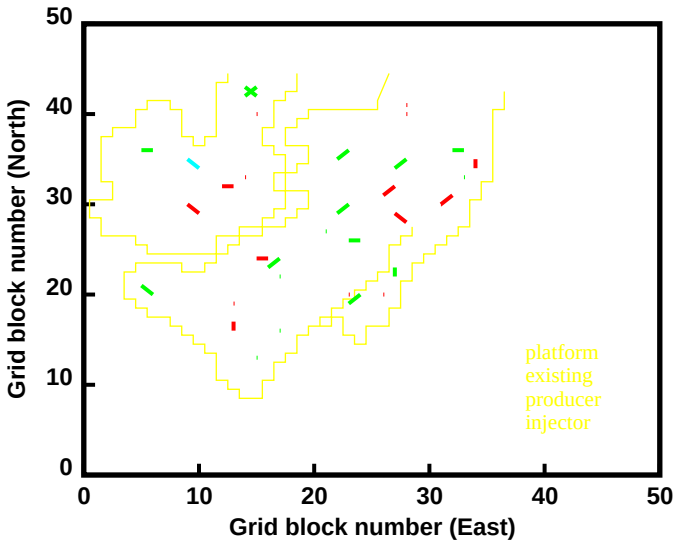

Figure 6.16: Well distribution generating a profit of 1.034 x 10^9

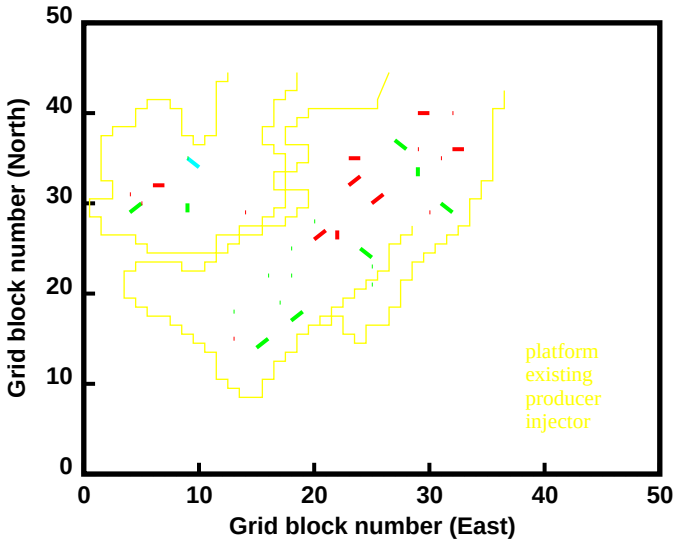

Figure 6.17: Well distribution generating a profit of 1.019 x 10^9

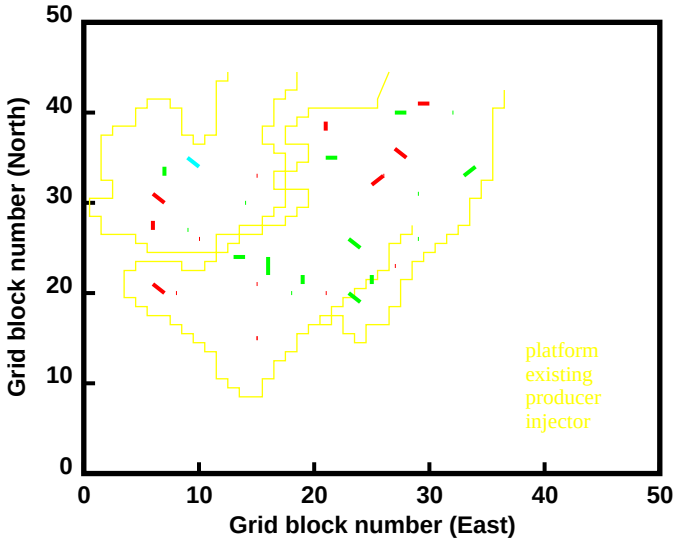

Figure 6.18: Well distribution generating a profit of 1.014 x 10^9

In the second approach the proposed solution was inserted as an individual in the initial population of the hybrid GA. We investigated how the algorithm would behave when such information was provided. This approach generated a profit of US$1,150.96 million at generation 6 and became steady until generation 17 with the well distribution as shown in Figure 6.19 and summarized in Table 6.5.

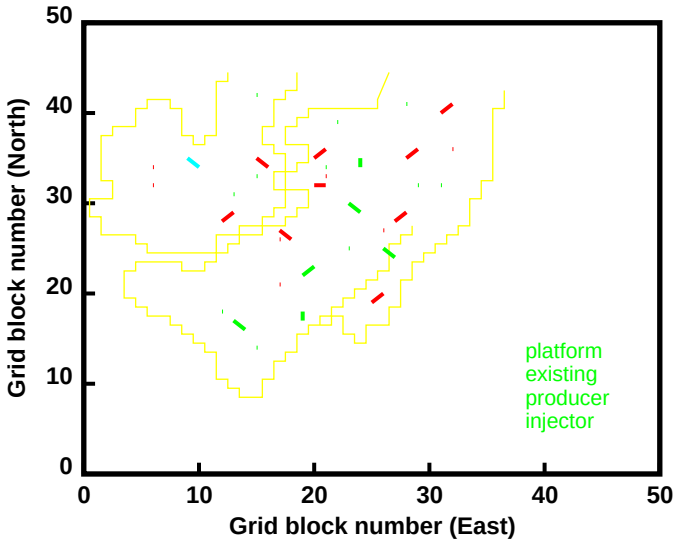

Figure 6.19: Optimized well locations (the proposed solution was known)

Type	Producers		Injectors		Total
Direction	vertical	horizontal	vertical	horizontal	
AREA-1	8	6	5	7	26
AREA-2	3	0	2	2	7
sub total	11	6	7	9	33
Total	17		16		33

Table 6.5: Optimized well distribution (generation 6 to 17)

From generation 18 until generation 28 the well distribution shown in Figure 6.20 and summarized in Table 6.6 generated a profit of US$1,171.90 million.

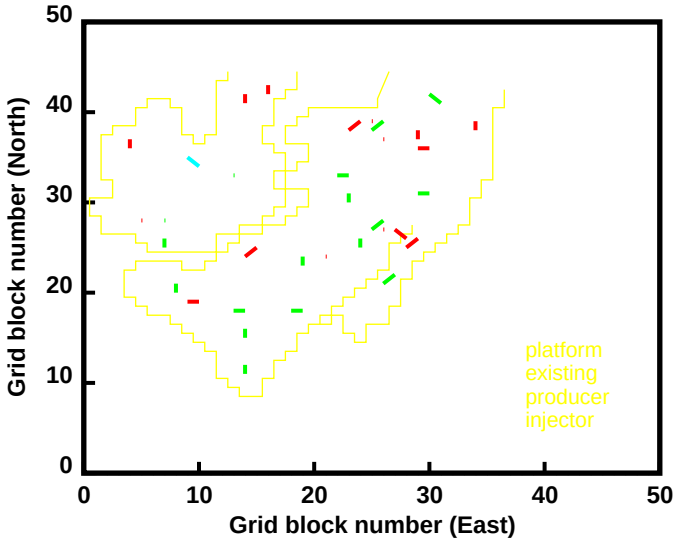

Figure 6.20: Optimized well locations (generation 18 to 28)

Type	Producers		Injectors		Total
Direction	vertical	horizontal	vertical	horizontal	
AREA-1	0	14	4	8	26
AREA-2	2	1	3	1	7
sub total	2	15	7	9	33
Total	17		16		33

Table 6.6: Optimized well distribution (generation 18 to 28)

Generation 29 introduced the well distribution shown in Figure 6.21, and summarized in Table 6.7, with a profit of US$1,203.83 million.

The net present value for this approach is compared to the net present value of the proposed solution in Figure 6.22. Total incremental improvement was US$68 million (6.08%). The cumulative oil production for both strategies is shown in Figure 6.23.

The variation of the number of function evaluations and objective function value along the generations are shown in Figures 6.24 and 6.25 for both strategies.

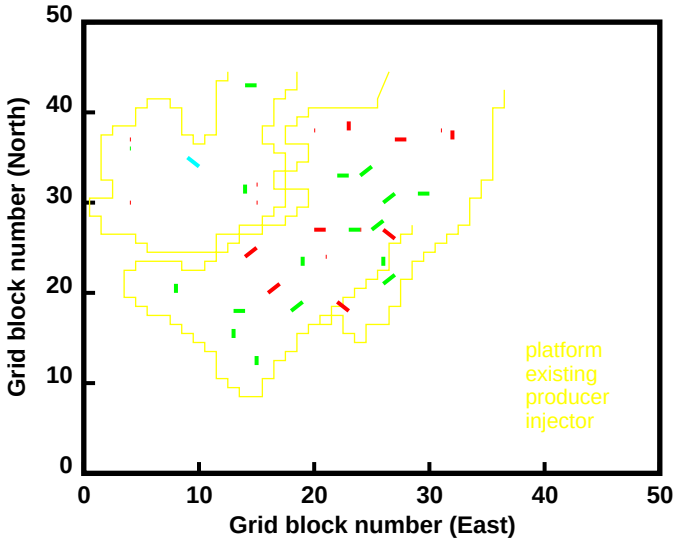

Figure 6.21: Optimized well locations (generation 29)

Type	Producers		Injectors		Total
Direction	vertical	horizontal	vertical	horizontal	
AREA-1	0	14	4	8	26
AREA-2	1	2	4	0	7
sub total	1	16	8	8	33
Total	17		16		33

Table 6.7: Optimized well distribution (generation 29)

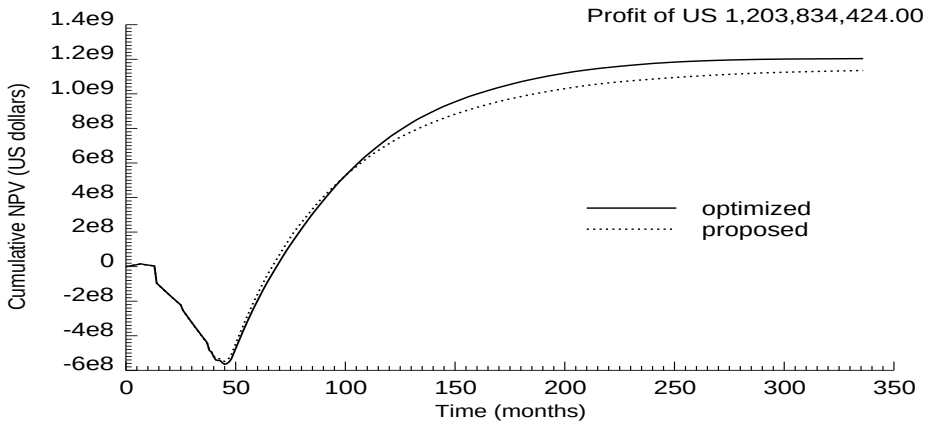

Figure 6.22: Optimized cash flow (the proposed solution was known)

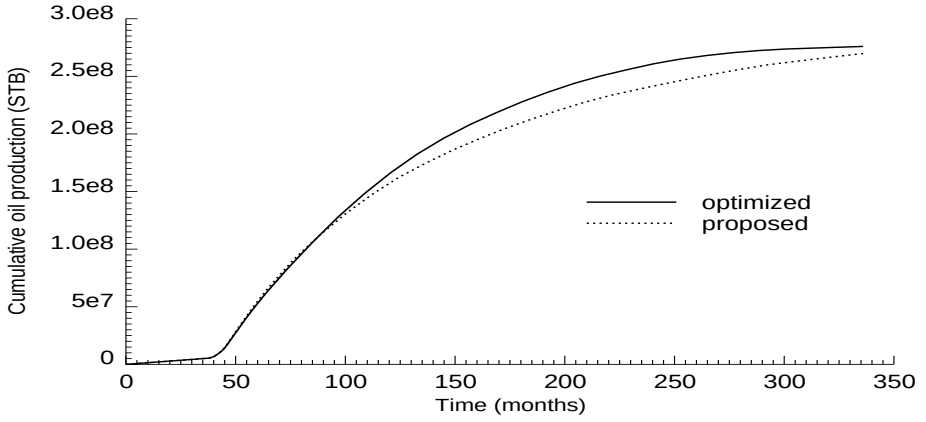

Figure 6.23: Optimized cumulative oil production (the proposed solution was known)

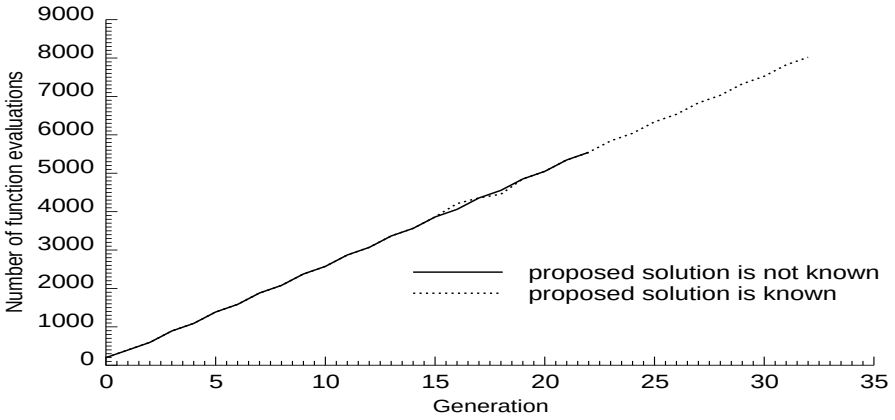

Figure 6.24: Function evaluation consumption

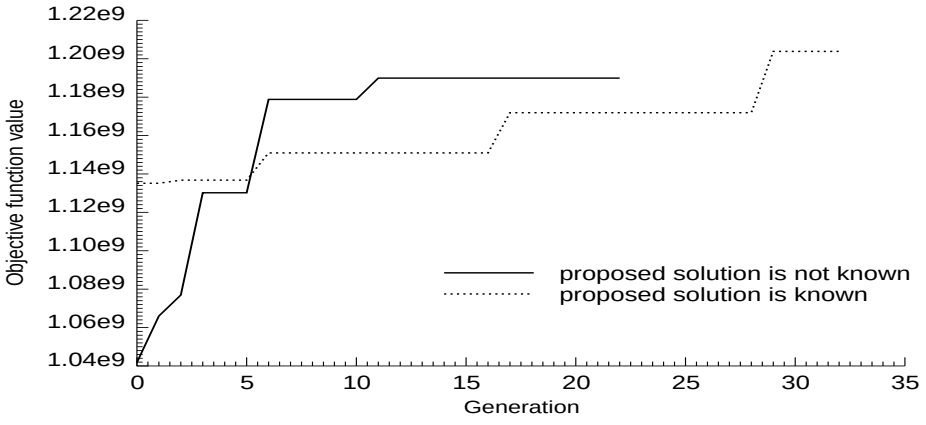

Figure 6.25: Objective function behavior

In both the cases considered in the real optimization problem an improved solution could be obtained. The improvement of 4.82% for the case that did not know the proposed solution represents US$54 million distributed along the project's life. For the case that knew the proposed solution an improvement of 6.08% was obtained which represents US$68 million.

Figure 6.25 shows how the insertion of the proposed solution affected the optimization behavior. The fitness associated with the individual that carries this knowledge was about three times greater than the worst fitness in the population. This produced a tendency towards premature convergence since predominance of this individual over the others and a relatively smooth function mean that a significant improvement were difficult to obtain. As can be observed in Tables 6.5 through 6.7 the main effect of the optimization was the one of horizontalization of wells causing more oil to be produced earlier and so, making money earlier although the improvement in the final cumulative production was not significant. The *location* parameters were harder to optimize because the knowledge of the proposed solution produced a strong individual concerning the *location*. Basically the only move that the algorithm could try was the direction and orientation of wells. Probably a fitness scaling mechanism would be helpful to assure a better regulation of the number of copies in the new populations. The case where the proposed function was not known had more individuals with high fitnesses, contributing to a better regulated generation of new individuals.

Chapter 7

Conclusions

A hybrid GA algorithm was developed and an improvement in finding the optimal value was demonstrated. The approach of having each specific optimization algorithm adding to the overall procedure instead of trying to predominate during the search allowed the use of the best features when they were required.

Purely mathematical functions with different behavior were tested with the new algorithms demonstrating their efficiency when compared to conventional methods. Some function evaluations were spent to support the decision of which method should be used but the improvement in the fitness was the reward for such a strategy. The best value of the hybrid parameters could be determined for the three different types of functions and then applied to the real petroleum engineering problem. The success in the optimization of the real problem supports the values suggested for the hybrid parameters from the test problems.

The hybrid parameter values shown in Table 5.10 were used to investigate test functions of different types: smooth, rough, unimodal and mutimodal. The values that would better optimize functions of similar behavior but with different degree of smoothness, for example, may be different.

The genetic algorithm is very sensitive to parameters like crossover and mutation probability, population size and selection method. The population size used in the petroleum engineering problem was about of 10% smaller than the one that should be tried according to common population sizing rules. Even so, the algorithm produced

good results. A greater population size would provide a greater diversity increasing the chances of better regulation of individuals. Different mutation and crossover rates could also help in the optimization task but the values that should be tried are still not well determined.

The petroleum engineering optimization problem was not constrained to any reservoir engineering concept regarding well location. Injectors and producers had the same domain and the optimization task had to determine their best location based on the economic point of view. The solution found had no restrictions concerning well spacing and injector/producer patterns as well. All this was determined by the optimization algorithm using the flow simulator to generate the production profile only. Since the close placement of wells has a negative impact on overall recovery efficiency, poorer location schemes were automatically given lower weight in that they produced smaller profit. In Figure 6.6 about five wells were placed very close to injectors, faults or even to other producers. This distribution was generated to accomodate the fixed number of wells to be placed but it points to the fact that the optimal number of wells may be lower than the proposed one.

The real problem optimization involved several issues to overcome some limitations of the flow simulator. A more detailed economical model is the natural step for this kind of problem. It was shown how standard simulators can be made suitable for optimization tasks, separating the objective function evaluation from the data manipulation.

The hybrid algorithm was found to be effective and the results obtained in the real problem indicate more profitable strategies for the field development.

A significant optimization of a real project is not an easy job. However, if we consider that the solution found by the algorithm can be an initial configuration for the engineering problem then reservoir engineers could try other schemes based on that solution and restart the optimization task.

7.1 Future work

Future work may focus three different topics: optimization algorithm, simulator features and economic model.

The optimization algorithm does not know anything about the objective function. It requires the function value for the specific set of parameters (individual) only. In this way, the algorithm can be implemented with more advanced features available for GA such as: selection and crossover operators, variable length individuals (messy GA), fitness scaling function, and a better understanding about the already implemented methods to better identify their interaction with the HGA.

The simulator itself can be implemented with features to help the optimization task such as the ones already described in Chapter 6:

1. Well properties and constraints assigned by geological region;
2. Well location specified by sequence domain. This domain is specified by geological region;
3. Generation of return codes reporting the ending status of the simulator;
4. Verticalization/shrinkage of horizontal wells when completions occur in the same block;
5. Wells defined by head location, orientation, length and/or parametric curve (two-dimensional/three-dimensional);
6. User-defined action when a predictable misuse occurs, e.g., a well is completed in some wrong way (out of the grid, same location as another);
7. Injection/producer specification by geological region;
8. Production/economic criteria defined by geological region;
9. Time specification of some items not associated to the schedule data structure;
10. Flexible drilling queue allowing changes in the sequence of the queue;

11. Restart file should have a "save report" feature because the data do not accumulate information from the previous steps;

The economic model must be more detailed. This would allow a more accurate economic analysis to support the overall confidence in the optimized solution. Several other modules could be attached such as a completion database, facilities and flowlines design, etc.

It is important to note that the integration of different technical disciplines provided by this kind of optimization work will require that people from different areas in the company will be more concerned with other people's work. Often, a field development project is made without interaction throughout the entire project life. The interface between areas is poor, limited and usually only considers some revision to be done. This work presents an overall approach considering all the areas involved in the project at the same time.

Chapter 8

Nomenclature

8.1 Letters

C: objective function general representation;

C_w: development well costs;

C_p: production facilities costs;

C_t: transportation costs;

F: objective function value;

GA: Genetic Algorithm;

G_p: gas cumulative production;

N_p: oil cumulative production;

NPV: Net Present Value;

Q: rate at standard conditions;

SA: Simulated Annealing;

TS: Tabu Search;

W_p: water cumulative production;

f: fitness;

l: length of the binary encoding;

n_p: number of parameters;

N *or* n: population size;

p_c: crossover probability;

p_m: mutation probability

q: rate at reservoir conditions;

t: time;

$\delta(H)$: defining length of a schema H.

$o(H)$: order of hyperplane H;

θ: highest order of hyperplane represented in a population of size N by at least ϕ copies. $\theta = log(N/\phi)$;

8.2 Subscripts

c: crossover;

i: index of an individual;

g: gas;

m: mutation;

o: oil;

p: cumulative;

w: water;

8.3 Glossary

The following definitions are presented to complement the ones introduced in the text.

Allele: bit or position value;

Binary coding[**Carroll, 1996a**]**:** "the parameters are discretized into a number of possibilities, but the chromosome length is based on the total number of possibilities in a binary format, e.g., 32 possibilities would be represented by a string of five zeros and ones, whereas 16 possibilities would be represented by a string of four zeros and ones. During crossover with binary coding, the crossover point may occur in the middle of one of the parameter strings; this allows the child to have a parameter string that is a mix of the parents parameter strings and, consequently, the child may have an allele (parameter value) between the two alleles of the parents. Thus, in binary coding, more alleles (possible values of the parameters) are preserved as new generations are created." The precision of the binary coding is given by [Janikow and Michalewicz, 1991]:

$$\frac{UB - LB}{2^n - 1} \tag{8.1}$$

where UB and LB are the upper and lower domain bounds and n is the parameter length.

*Building blocks***:** are schemata of above-average fitnesses and short defining length;

Canonical GA[**Rawlins, 1994**]**:** assumes that: "(a) all the strings are of the same length (b) populations are of constant size (c) mutation is probabilistic and is usually independent of the string value (d) reproductive sets are of size two, and members are chosen probabilistically, with probabilities weighted by string values (e) mating is usually restricted to crossover. In *general crossover* the i^{th} symbol of an offspring is the symbol i^{th} of one of the members of the reproductive set. Most GAs limit crossover such that symbols in positions $1...i$ of the new string come from one parent and those in positions $i+1...l$ come from the other parent, where i is chosen probabilistically";

Creep mutation[**Carroll, 1996a, Carroll, 1996b**]: one or more of the child parameters will be mutated by a single increment up or down that affects the chromosome (bit) keeping the parameter within its domain. The value usually taken is $prob = 1/N$ (sometimes $2/N$). Overall probability of a creep mutation for an individual i is: $(p_{creep})_i = 1 - (1 - p_{creep})^L$

Chromosome: alphabet encoded string (no *don't care* character *);

Crossover: genetic operation responsible for the generation of offsprings;

Elitism: the best of the actual generation is copied into the next one;

Expected value selection[**Carroll, 1996a**]: "the fitness of all of the individuals in the population is summed, and then the expected probability of selection is based on the fitness of the individual divided by the total fitness of the population, i.e., $p_j = \frac{f_j}{\sum f_j}$. The expected number of parents with chromosome set j for the new generation is simply $n * p_j$. This procedure will fill most of the parent slots, but there will be a fractional remainder of slots that are filled using the stochastic remainder sampling without replacement method Random pairs of mates are then chosen from this population of fit parents. Then, each pair of mates creates two children, e.g., one child could end up with chromosome set abcDE and the other with ABCde";

Family size: each pair of parents can generate one, two or more children. Two children (larger family) adds more stability and less variability to the evolution of the population;

Fitness of an individual: is defined as $f_i/\bar{f}$, where f_i is the function evaluation of string i and $\bar{f}$ is the average evaluation of the entire population;

Fixed position: is a position in the chromosome string with an alphabet value other than the *don't care* character *;

Floating point coding[**Carroll, 1996a**]: "the parameter is discretized into a number of possibilities, and there is one chromosome for each parameter. The value of

the chromosome is stored as a floating point number. For floating point coding, each parameter is represented by a single chromosome in the chromosome string; therefore, for floating point coding, the parameter length is unity. In floating point coding, the child must have a mix of the parents' alleles but cannot have alleles which are not present in the parents chromosome strings";

Function evaluation: is defined as a measure of performance of a string representing a set of parameters and it is independent of the evaluation of any other string;

Genitor GA (steady state class)[**Whitley, 1996**]: assumes that: "(a) reproduction produces one offspring at a time (b) the produced offspring replaces some relatively less fit member of the population, so, the best points are maintained in the population (c) fitness is assigned according to rank rather than by fitness proportionate reproduction";

Intermediate popopulation: the population created for the selection step, after this recombination (crossover) and mutation are applied;

Jump mutation[**Carroll, 1996a, Carroll, 1996b**]: each bit is tested for mutation with $prob = 1/N$ and the resulting chromosome may not be inherited from either parent. The overall probability of a jump mutation for an individual i is:

$(p_{jump})_i = 1 - (1 - p_{jump})^{np}$

for $jump = creep$ let $p_{creep} = \frac{L}{np} p_{jump}$

Loose ordering: the bits of each parameter are separated from on another such that the defining length of the building block is long;

Mutation: genetic operation that introduces information or small segments of parameter strings into a population;

Niching (sharing)[**Deb and Goldberg, 1991**]: the individuals are grouped according to their proximity to strong individuals. In a multimodal function it assures the convergence to the different optima without loosing points due to the concentration about just one optimum. the niching is done based on a moderated

fitness that is calculated by taking the potential fitness and dividing through by the accumulated number of shares. One example of shares is the triangular sharing function (based on the distance between individuals);

Offspring: string generated from the parent strings, child;

Premature convergence: when there is no more change in the bit values the GA algorithm converged prematurely to a non satisfactory solution;

Remainder stochatic sampling[**Whitley, 1996**]: "for each string where $\frac{F_i}{F_avg}$ is greater than 1.0, the integer portion indicates how many copies of that string are directly placed in the intermediate population. All strings then place additional copies in the intermediate population with a probability corresponding to the fractional portion";

Schema[**Holland, 1975**]: is a hyperplane in the search space. Pattern matching device containing the *don't care* character * (plural *schemata*) ;

Single-point crossover: one crossover point is randomly chosen;

Stochastic sampling with replacement[**Whitley, 1996**]: "the population is mapped onto a roulette wheel, where each individual is represented by a space that proportionally corresponds to its fitness";

Tight ordering: the bits of each parameter are adjacent to one another. The building block is the shortest;

Tournament selection[**Carroll, 1996a**]: "random pairs are selected from the population and the stronger (most fit) of each pair is allowed to mate. Each pair of mates create one child, which has some mix of the two parents chromosomes according the method of crossover";

Two-point crossover: there are two crossover points and the string between these two points is changed;

Uniform crossover: each bit is inherited from one of the parents. Tends to preserve alleles better than single-point crossover. Tends to destroy building blocks indiscriminately. Not suitable for deceptive problems;

References

[Ackley, 1987] Ackley, D.: *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Boston, USA (1987).

[Arnondin, 1995] Arnondin, M.: "Integration of Production Analyst and Microsoft Excel's Solver for Production Forecasts and Optimization", *SPE Computer Aplications*, pages 118–121 (1995).

[Baker, 1985] Baker, J.: "Adaptive Selection Methods for Genetic Algorithms", Proceedings of the International Conference on Genetic Algorithms and their Applications. Lawrence Erlbaum Associates, Publishers. J. Grefenstettte (1985).

[Beckner and Song, 1995] Beckner, B. L. and Song, X.: "Field Development Planning Using Simulated Annealing - Optimal Economic Well Schedulling and Placement", 1995 SPE Annual Technical Conference Exhibition. Dallas, USA. Paper SPE 30650 (1995).

[Bittencourt, 1994] Bittencourt, A. C.: "Optimal Scheduling of Development in an Oil Field", Master's thesis, Stanford University (1994).

[Booker, 1982] Booker, L.: *Intelligent Behavior as an Adaptation to the Task Environment*, PhD thesis, Universtiy of Michigan. Technical Report No. 243 (1982).

[Borsuk, 1969] Borsuk, K.: *Multidimensional Analytic Geometry*. PWN - Polish Scientific Publishers, Warszawa, Poland (1969).

[Brindle, 1981] Brindle, A.: *Genetic Algorithms for Function Optimization*, PhD thesis, Universtiy of Michigan. Technical Report TR81-2 (1981).

[Carroll, 1996a] Carroll, D. L.: "Chemical Laser Modeling with Genetic Algorithms", *AIAA Journal*, 34(2):338–346 (1996).

[Carroll, 1996b] Carroll, D. L.: "Genetic Algorithms and Optimizing Chemical Oxygen-Iodine Lasers", *Developments in Theoretical and Applied Mechanics*, XVIII (1996).

[Cvijovic and Klinowski, 1995] Cvijovic, D. and Klinowski, J.: "Taboo Search: An Approach to the Multiple Minima Problem", *Science*, 267:664–666 (1995).

[Damsleth et al., 1992] Damsleth, E., Hage, A., and Volden, R.: "Maximum Information at Minimum Cost: A North Sea Field Development Study With an Experimental Design", *Journal of Petroleum Technology*, pages 1350–1356 (1992).

[De Jong, 1975] De Jong, K.: *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD thesis, Universtiy of Michigan. University Microfilms No. 76-9381 (1975).

[Deb and Goldberg, 1991] Deb, K. and Goldberg, D.: "An Investigation of Niche and Species Formation in Genetic Function Optimization", Proceedings of the Third International Conference of Genetic Algorithms, pages 42–50. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1991).

[Ding and Startzman, 1994] Ding, Z. and Startzman, R. A.: "A Software fo Oilfield Facility Investment Minimization", 1994 SPE Petroleum Computer Conference. Dallas, USA. Paper SPE 28252 (1994).

[Eshelman and Schaffer, 1991] Eshelman, L. and Schaffer, J.: "Preventing Premature Convergence by Preventing Incest", Proceedings of the Fourth International Conference on Genetic Algorithms, pages 115–122. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1991).

[Eshelman et al., 1989] Eshelman, L., Schaffer, J., and Caruana, R.: "Biases in the Crossover Landscape", Proceedings of the Third International Conference on Genetic Algorithms, pages 10–19. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1989).

[Fang, 1980] Fang, K.: "Uniform Design: Application of Number Theory in Test Design", *ACTA Mathematicae Applicatae Sinica*, 3(4):363–372. in Mandarin (1980).

[Fleischer, 1994] Fleischer, G.: *Introduction to Engineering Economy*. PWS Publishing Company, Boston, USA (1994).

[Fogel and Atmar, 1990] Fogel, D. and Atmar, J.: "Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems", *Biological Cybernetics*, 63:111–114 (1990).

[Gill et al., 1992] Gill, P., Murray, W., and Wrigth, M.: *Practical Optimization*. Academic Press, California, USA (1992).

[Glover, 1994] Glover, F.: "Tabu Search Fundamentals and Uses". Technical report, University of Colorado. Notes for the Graduate School of Business (1994).

[Goldberg, 1989] Goldberg, D.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, USA (1989).

[Goldberg et al., 1992] Goldberg, D., Deb, K., and Clark, J. H.: "Genetic Algorithms, Noise, and the Sizing of Populations", *Complex Systems*, 6(19):333–362 (1992).

[Goldberg and Deb, 1994] Goldberg, D. E. and Deb, K.: "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms", *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1994).

[Holland, 1975] Holland, J.: *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, USA (1975).

[Janikow and Michalewicz, 1991] Janikow, C. Z. and Michalewicz, Z.: "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms", Proceedings of the Fourth International Conference on Genetic Algorithms, pages 31–36. Morgan Kaufmann Publishers, Inc., La Jolla, USA (1991).

[Kendall, 1961] Kendall, M.: *A Course in the Geometry of N Dimensions*. Hafner Publishing Company, New York, USA (1961).

[Kennedy, 1993] Kennedy, J.: "Cost Trends in International Petroleum Activities", SPE 1993 68th Annual Technical Conference and Exhibition. Houston, USA. Paper SPE 26410 (1993).

[Mannarino, 1991] Mannarino, R.: *Introdução À Engenharia Economica*. Editora Campus, Rio de Janeiro, Brazil (1991).

[Nelder and Mead, 1965] Nelder, J. A. and Mead, R.: "A Simplex Method for Function Minimization", *Computer Journal*, 7:308–313 (1965).

[Nystad, 1985a] Nystad, A. N.: "Petroleum Taxes and Optimal Resource Recovery", *Energy Policy*, pages 381–402 (1985).

[Nystad, 1985b] Nystad, A. N.: "Reservoir Economic Optimization", 1985 SPE Hydrocarbon Economics and Evaluation Symposium. Dallas, USA. Paper SPE 13775 (1985).

[Nystad, 1987] Nystad, A. N.: "Rate Sensitivity and the Optimal Choice of Production Capacity of Petroleum Reservoirs", *Energy Economics*, pages 37–45 (1987).

[Nystad, 1988a] Nystad, A. N.: "On the Economics of Improved Oil Recovery: The Optimal Recovery Factor from Oil and Gas Reservoirs", *The Energy Journal*, 9(4):49–61 (1988).

[Nystad, 1988b] Nystad, A. N.: "Petroleum Reservoir Management: A Reservoir Economic Approach", *Natural Resource Modeling*, 2(3):345–382 (1988).

[Pan, 1995] Pan, Y.: "Application of Least Squares and Kriging in Multivariate Optimizations of Field Development Scheduling and Well Placement Design", Master's thesis, Stanford University (1995).

[Press et al., 1992] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B.: *Numerical Recipes in C*. Cambridge University Press, New York, USA (1992).

[Rawlins, 1994] Rawlins, G. J.: *Foundations of Genetic Algorithms*. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1994).

[Rogers, 1992] Rogers, L.: *Optimal Groundwater Remediation Using Artificial Neural Networks and The Genetic Algorithm*, PhD thesis, Stanford University (1992).

[Schaffer et al., 1989] Schaffer, J., Caruana, R., Eshelman, L., and Das, R.: "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization", Proceedings of the Third International Conference on Genetic Algorithms, pages 51–60. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1989).

[Schaffer and Eshelman, 1991] Schaffer, J. and Eshelman, L.: "On Crossover as an Evolutionarily Viable Strategy", Proceedings of the Fourth International Conference on Genetic Algorithms, pages 61–68. Morgan Kaufmann Publishers, Inc., La Jolla, USA (1991).

[Sommerville, 1929] Sommerville, D.: *An Introduction to the Geometry of N Dimensions*. Methuen & Co. Ltd., London, England (1929).

[Spears and De Jong, 1994] Spears, W. M. and De Jong, K. A.: "An Analysis of Multi-Point Crossover", *Foundations of Genetic Algorithms*, pages 301–315. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1994).

[Steiner, 1992] Steiner, H.: *Engineering Economic Principles*. McGraw-Hill, Inc. (1992).

[Syswerda, 1989] Syswerda, G.: "Uniform Crossover in Genetic Algorithms", Proceedings of the Third International Conference on Genetic Algorithms, pages 2–9. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1989).

[Whitley, 1989] Whitley, D.: "The GENITOR Algorithm and Selective Pressure", Proceedings of the Third International Conference on Genetic Algorithms, pages 116–121. Morgan Kaufmann Publishers, Inc., San Mateo, USA (1989).

[Whitley, 1996] Whitley, D.: "A Genetic Algorithm Tutorial". Technical report, University of Colorado (1996).