New Methods in Genetic Search with Real-Valued Chromosomes

by

Adenike A. Adewuya

B.S. Mechanical Engineering
Mississippi State University, 1993

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology
June 1996

Signature of Author: ______________________
Department of Mechanical Engineering
May 29, 1996

Certified by: ______________________
Mark Jakiela
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by: ______________________
Ain A. Sonin
Professor of Mechanical Engineering
Chairman, Department Committee on Graduate Students

New Methods in Genetic Search with Real-Valued Chromosomes

by

Adenike A. Adewuya

Submitted to the Department of Mechanical Engineering
on May 29, 1996 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

A new crossover operator, quadratic crossover, for numerical optimization with genetic algorithms is presented. Quadratic crossover works with chromosomes that have real-valued or floating-point genes, making unnecessary the transformation of an optimization problem from its natural real space to a binary space. The performance of quadratic crossover on standard genetic-algorithm test problems was compared to the performance of existing crossover operators for real-valued chromosomes on the same set of test problems. Quadratic crossover was found to be more reliable, and in many cases, more efficient than the existing crossover operators. The effectiveness of the quadratic crossover in guiding genetic search to the global optimum for a wide variety of standard nonlinear programming test problems and real world design problems was investigated. In 75% of the test problems considered, the quadratic crossover either found the exact global optimum or a better optimum than the best known optimum. In the remaining 25%, the maximum percent deviation from the global optimum or the best known optimum is only 0.13%. Results show that the quadratic crossover is effective in guiding genetic search to the global optimum, or at least, the neighborhood of the global optimum, for all classes of linear and nonlinear problems.

Thesis Supervisor: Mark Jakiela
Title: Associate Professor of Mechanical Engineering

Dedication

To my two loves

OPEYEMI & OLUWASEYI

Acknowledgements

At last, I have the opportunity to give thanks to all those who made this endeavor possible. First of all, I would like to thank my heavenly father who continually gives me the desire and strength to accomplish many things.

I would like to thank my advisor, Professor Mark Jakiela, who patiently guided me throughout this project. I would like to acknowledge the fellowship from NSF and thank the MIT Mechanical Engineering department for fully supporting me during my first semester at MIT.

Thanks are due to my father, Dr. George Adebiyi, who has always inspired me to do great things, and also to my mother, Mrs. Mary Adebiyi, for affording me the opportunity to accomplish great things.

I would like to thank my mother from Starkville, Mrs. Iyabo Adebiyi, and my mother-in-law from Nigeria, Mrs. Anthonia Adewuya, who took time out of their busy schedules to help me take care of Oluwaseyi. Thanks are also due to their husbands, Dr. Adebiyi and Dr. Adewuya, who took care of the house while their wives were taking care of Oluwaseyi. I am especially indebted to my brother Debo who assisted me in taking care of Oluwaseyi, and did a great job, I might add, while I worked on this project.

Finally, I am grateful to my husband, Opeyemi, for willingly allowing me to leave home and come to MIT for graduate studies. I am thankful for his support and encouraging words throughout his endeavor.

Table of Contents

List of Figures

List of Tables

Chapter I

Introduction

1.1 Genetic Algorithms as Global Function Optimizers

The design of real world systems is usually presented as optimization problems, where only certain combinations of the design variables result in a feasible design. It is often desirable to find the best of the combinations of the design variables that result in a feasible design. Thus, design optimization problems become global optimization problems.

A global optimization problem can be stated as follows: Given a feasible region χ and an objective function $f: \chi \rightarrow \Re^1$, approximate the value

$$f^* = \min_{x \in \chi} f(x)$$

and the point $x^* \in \chi$ at which the minimum f^* is attained. This problem is the global minimization problem. The minimization problem can be transformed to a maximization problem by replacing $f(x)$ with $-f(x)$.

Weierstrass theorem guarantees the existence of a global optimum for an objective function that is continuous on a non-empty, closed, and bounded design domain [Arora, 1989]. In addition, if the design domain is convex, the local optimum is the same as the global optimum, and the local optimum is attained at the point where the gradient of the objective function is zero. However, many real world problems do not have objective functions that are everywhere continuous on the design domain, nor do they always have design domains that are convex. Perhaps for a nonconvex design domain, a convex subregion can be identified, and the global optimum for that convex subregion can be found. If the objective function is multimodal, several convex subregions may have to be identified. The reality is that global optimization problems are intractable; there is no local criterion that guarantees that a local optimum is global if the design domain is not *nice* and the objective function is not convex or, at least, quasi-convex.

Nevertheless, global optimization problems are tackled every day. There are numerous global optimization algorithms that attempt to construct a sequence $\{x_k\}$ of points in χ that converge to a point x^* in which the objective function f approximates or equals f^* [Zhigljavsky, 1991]. These optimization algorithms can be classified under two broad categories: deterministic and stochastic methods. Deterministic methods include local search techniques, covering methods, branch and bound methods, decomposition based approaches, approximation and integral representations, and interval analysis.

Stochastic methods include random search techniques, simulated annealing, clustering methods, and stochastic and axiomatic models [Floudas and Pardalos, 1992]. A comprehensive review of most of the optimization methods enumerated above can be found in [Zhigljavsky, 1991]. Genetic algorithms, when used as function optimizers, can be classified under stochastic methods.

Genetic algorithms are stochastic algorithms that simulate the process of natural evolution. They start with a population of randomly created individuals, and by means of natural selection and natural genetics, the individuals combine to produce new individuals. The fittest individuals survive with the progression of genetic simulation, and the not-so-fit individuals die off. The idea of the survival of the fittest is useful in optimization because the individuals in the population can represent solution vectors to an optimization problem so that the fitness of an individual is measured by the objective function. The global optimum can be found by keeping track of the best individual during the evolution process.

Genetic algorithms differ from other optimization and search techniques in that they do not work with the design variables directly; instead, they manipulate the design variable codings and use only the objective values to determine the direction of search. Also, genetic algorithms process multiple starting points in parallel, so that the chance of converging to a local sub-optimum is usually minimal. The underlying motivation for using genetic algorithms as function optimizers is robustness. Genetic algorithms have the ability to adapt to a wide variety of environments.

1.2 An Illustration of the Mechanics of a Simple Genetic Algorithm

The following optimization problem is considered in the illustration of the mechanics of a simple genetic algorithm:

$$\text{Maximize } f(x) = x^3$$

$$\text{where } \quad x \in [0, 7]$$

A simple genetic algorithm starts with an initial population of single-chromosome individuals or potential solutions to an optimization problem. In this case, the chromosomes would represent the variable x. The chromosomes are coded as strings defined over a finite-length alphabet, usually the binary alphabet {0, 1}. In the example problem, the variable x is an unsigned integer. Therefore, in binary notation, the variable x will be represented with three bits. Correspondingly, the structure of the chromosomes would be ###, where each # could be a 0 or 1. The initial population is created by randomly assigning a 0 or 1 to each # in the chromosome. For example, an initial population of four individuals could be

010 110 101 011

The next step is to determine the fitness of the individuals in the population. This is done by mapping the binary strings to unsigned integer numbers and evaluating the objective function for each individual. The evaluation of the individuals in the initial population given above is presented in Table 1.1.

Table 1.1: Fitness values of individuals in initial population

i	Individuals (Binary)	Individuals (Integer)	Fitness, $f_i = x^3$
1	010	2	8
2	110	3	27
3	101	5	125
4	011	6	216
Total			376
Average			94

The evolution process now starts. Evolution is composed of three basic steps: reproduction, crossover, and mutation. Reproduction involves the selection of a new population based on the probability distribution of the fitness values of the current population. The new population is selected by spinning a weighted roulette wheel with slots n times, where n is the population size. The roulette wheel (see Figure 1.1) is constructed from the probability of selection for each individual in the population, where the probability of selection of an individual is defined as the ratio of the fitness of that individual to the total fitness of the population. The probability of selection of the individuals for the example problem is shown in Table 1.2.

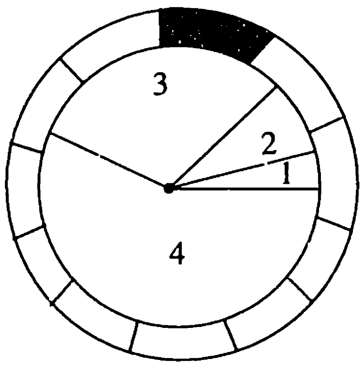


Figure 1.1: Sample roulette wheel

Selection using the roulette wheel is accomplished in two steps [Michalewicz, 1994]:

1. Spinning the Wheel: This step involves the generation of a random number r, where r is in the range [0...1].

2. Selecting the Individual: This step involves finding the individual that corresponds to the black slot shown in Figure 1.1. This is done as follows: if $r < q_1$, the first individual is selected; otherwise, the i^{th} individual such that $q_{i-1} < r \leq q_i$ for $i \in [2, n]$ is selected -- q_i is the cumulative probability of the i^{th} individual.

The cumulative probability, q_i, of an i^{th} individual is shown in Table 1.2. The roulette wheel selection is illustrated for the example problem.

Table 1.2: Cumulative probability of the individuals in the initial population

i	Individual (Binary)	Fitness, $f_i = x^3$	Probability of Selection, $p_i = f_i / \left(\sum_{i=1}^{n} f_i \right)$	Cumulative Probability, $q_i = \sum_{j=1}^{i} p_j$
1	010	8	0.021	0.021
2	110	27	0.072	0.093
3	101	125	0.332	0.425
4	011	216	0.575	1
Total		376	1	
Average		94		

Suppose that the following sequence of numbers are generated by spinning the roulette wheel 4 times:

0.655416   0.200995   0.893622   0.281887

The first number $r = 0.655416$ is greater than q_3 and smaller than q_4, so individual 4 gets selected for the new population; the second number $r = 0.200995$ is greater than q_2 and smaller than q_3, so individual 3 gets selected for the new population; the third number $r = 0.893622$ is greater than q_3 and smaller than q_4, so individual 4 is selected for the new population; the fourth number $r = 0.281887$ is greater than q_2 and smaller than q_3, so individual 3 gets selected for the new population. The new population consists of the individuals

011 101 011 101

Note that an individual can get selected more than once.

Now, it is time for the individuals to undergo genetic operation. Genetic operation is accomplished by the genetic operators -- crossover and mutation operators. Crossover involves the recombination of the individuals in the new population to produce new offspring. A simple crossover involves the swapping of alleles of the chromosomes. This is illustrated for the chromosomes in the new population:

Crossing of the First Two Chromosomes
Parents:

0|11
1|01

Offspring:

001
111

Crossing of the Last Two Chromosomes
Parents:

01|1
10|1

Offspring:

011
101

After crossover, the new population is

001 111 011 101

Mutation involves the altering of a single bit with a probability. For the binary strings, this is equivalent to changing a bit with a value of 0 to 1 or vice versa. Let the probability of mutation for this example problem be 0.001. Then for the initial population of four 3-bit individuals, 12*0.001 = 0.012 bits should undergo mutation during a single generation. Suppose that the actual simulation of the mutation process does not result in alteration of any of the strings, then the new population is still

001 111 011 101

This completes the first generation of the evolution process. This cycle of reproduction, crossover, and mutation is repeated until the individuals have converged to an optimum. In this case, the optimum is at $x = 7$ with an objective value of 343. The statistics of the new population is presented in Table 1.3.

Table 1.3: Fitness values of individuals in the new population

i	Individuals (Binary)	Individuals (Integer)	Fitness
1	001	4	64
2	111	7	343
3	011	6	216
4	101	5	125
Total			748
Average			187

Note that the average fitness of the new population has increased by almost 100% during a single evolution cycle. Also, the maximum fitness of the individuals has increased by more than 50% during that same period. This is because the highly-fit individuals in the initial population have combined to produce even more highly-fit individuals in the new population. The notion of combining highly-fit individuals to produce even more highly-fit individuals is the main goal of the genetic algorithm.

1.3 The Theory Behind the Canonical Genetic Algorithm

Genetic algorithms work with single-chromosome individuals, where the chromosomes are coded as finite-length strings that are defined over some finite alphabet. The traditional alphabets used are the binary alphabets $\{0,1\}$. So for a function

$$f(x, y) = x^2 + y^2 \quad \text{where} \quad 0 \le x, y \le 5 \text{ , and } x \text{ and } y \text{ are not necessarily integers}$$

a chromosome could have the structure ######, where each # could be 0 or 1. The first three positions or genes correspond to the parameter x and the last three genes correspond to the parameter y. A similarity template, called a schema, can be constructed for a subset of strings that have same alleles or gene values at certain positions in a string. The schema is generally limited to the ternary alphabet $\{0, 1, \#\}$, where # is a wild card. The similarity information carried by each schema will depend on the number of wild cards in the string. The fewer the number of wild cards, the more specific is the schema. The number of fixed positions in a schema is known as the *order* of the schema, and the distance between the first and last positions in a string is known as the *defining length* of the schema. There are 3^l schemata for a string of length l that are defined over the ternary alphabet {0, 1, #}. In general, a string of length l defined over an alphabet of k cardinality has $(k+1)^l$ schemata.

A simple genetic algorithm uses three genetic operators to advance from one generation to the next. These three genetic operators are reproduction, crossover, and mutation. Reproduction involves the selection of individuals for mating. The individuals are selected according to their objective values. Highly fit individuals have a better chance of getting selected to contribute to the offspring of the next generation. The crossover operator decides how the selected individuals mate to produce offspring. The mutation operator alters a single gene of a chromosome.

Assuming that reproduction, crossover, and mutation operations are independent, then the expected count of a schema h after reproduction is given by

$$m(h, t+1) \geq m(h, t) \cdot \frac{f(h)}{f_{ave}}$$

where $m(h,t)$ is the number of strings representing schema h at time t, $m(h,t+1)$ is the number of strings representing schema h at time $t+1$, $f(h)$ is average fitness of the strings representing schema h at time t, and f_{ave} is the average fitness of the population at time t. This equation shows that a schema is expected to grow in proportion to the ratio of average fitness of the schema to the average fitness of the population.

A schema survives after undergoing crossover if the crossover site falls outside the *defining length*, δ, of the schema. So, if simple crossover is performed with a probability p_{cross}, then the probability of the schema h surviving after crossover is given by

$$p_s \geq 1 - p_{cross} \cdot \frac{\delta(h)}{l-1}$$

where $\delta(h)$ is the *defining length* of the schema h and l is the length of the strings representing schema h.

For a schema to survive after undergoing mutation, all the fixed positions in the schema must remain unchanged. A single allele remains unaltered with the probability ($1-p_{mut}$), where p_{mut} is the probability of mutation. Therefore, the probability of a schema surviving after undergoing mutation is $(1-p_{mut})^{o(h)}$, where $o(h)$ is the number of fixed positions. For small values of p_{mut}, the probability of a schema surviving after undergoing mutation can be approximated by the expression:

$$p_s \geq 1 - o(h) \cdot p_{mut}$$

The combined effect of reproduction, crossover, and mutation on the expected growth rate of a schema h is expressed as

$$m(h, t+1) \geq m(h, t) \cdot \frac{f(h)}{f_{ave}} \left[1 - p_{cross} \frac{\delta(h)}{l-1} - o(h) p_{mut} \right]$$

where

$f(h)$	: fitness of the schema
f_{ave}	: average fitness of the population
$\delta(h)$	: defining length of the schema
$o(h)$	: order of the schema
p_{cross}	: probability of crossover
p_{mut}	: probability of mutation

The primary conclusion from the growth rate equation is that "short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations" [Goldberg, 1989]. This conclusion is known as the Schema Theorem or the Fundamental Theorem of Genetic Algorithms.

1.4 Chromosome Representation Issues

The schema theory was based on the binary alphabet representation. It should be noted that any other finite alphabet could also have been considered. However, two observations can immediately be made about binary representation with respect to numerical optimization problems: (1) binary representation requires that the design problem be mapped from its natural real space to binary space, and (2) multidimensional optimization problems that require high precision need a rather long binary string to represent the solution vector. Mapping the design problem from real space to binary space is not a big problem, except that opportunities for exploiting gradualness of functions with variables over continuous domains is forfeited. On the other hand, long binary strings result in rather large search spaces, and genetic algorithms slowly converge in such spaces. Real-coded genetic algorithms, on the other hand, quickly narrow searching to a much smaller region of the entire search space [Goldberg, 1990].

Thus it seems only natural to use real-number coding instead of binary coding for numerical optimization problems, and the number of researchers using real coding has been on the rise lately; even though the genetic algorithm theory suggests that alphabets of low cardinality have higher schema processing capabilities than alphabets of high cardinality, and hence, should be more effective in search [Goldberg, 1990]. This last statement requires further explanation. In Holland's schema notation, the maximum number of schemata for a string of length l defined over an alphabet of k cardinality is $(k+1)^l$ [Holland, 1975]. So it would be expected that an alphabet of high cardinality should have a higher number of schemata than an alphabet of low cardinality. However, an alphabet of high cardinality generally requires a much shorter string to represent variables than an alphabet of low cardinality, so that the maximum number of schemata for the alphabet of low cardinality is higher than the maximum number of schemata for the alphabet of high cardinality. Nonetheless, it appears that from the practical standpoint alphabets of high cardinality (in this case, real or floating-point alphabet) give better results for numerical optimization problems [Davis, 1991; Michalewicz, 1994].

Considering the success of real-coded genetic algorithms, the natural question is why does the genetic algorithm theory fail to predict success for real-coded genetic algorithms? Really, there are two issues to be addressed here: the first issue is that the real alphabet over which real-valued chromosomes are defined is not finite, making the schema theory irrelevant, and the second issue is the failure of the schema theory to predict success for alphabets of high cardinality. A discussion of these two issues follows:

The basic assumption in the development of the schema theory is that the alphabet over which the strings or chromosomes are defined is finite. This means that schemata analysis and schema theory make sense if the real alphabet over which the real-valued chromosomes are defined is indeed finite. Before making further comments, it is in order to formally state the definition of a real-valued chromosome.

A real-valued chromosome has genes that are coded as floating point numbers. The length of the real-valued chromosome is equal to the number of the independent variables in the objective function, where each gene in the chromosome represents a single independent variable.

A graphical representation of a real-valued chromosome for a function $f(\vec{x})$ is shown in Figure 1.2.

x_1	x_2	x_3	x_4	x_5	$\cdots$	x_{n-1}	x_n

Figure 1.2: Representation of a real-valued chromosome

where $L_j \leq x_j \leq U_j$ for $j = 1 \ldots n$.

Now, each x_j can assume infinitely many values in the interval $[L_j, U_j]$. This means that the real alphabet over which the real-valued chromosome is defined is not finite. However, suppose that the interval $[L_j, U_j]$ can be divided into subintervals $[\alpha_i, \beta_i]$ such that $[L_j, U_j] = \prod_{i=1}^{m} [\alpha_i, \beta_i]$; this is Wright's connected schemata [Wright, 1991]. Then each subinterval $[\alpha_i, \beta_i]$ can be thought of as a virtual character, and the collection of the virtual characters can be thought of as a virtual alphabet. In other words, the schemata that is relevant to the real-valued chromosome is the interval-schemata, as opposed to the traditional symbol schemata used in schemata analysis.

Now, the question remaining to be resolved is how to assign subintervals to the virtual characters. One definition of virtual characters can be found in Goldberg's report on "Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking," [1990]. Goldberg defines a virtual character as a non-empty intersection of a basin (above-average intervals attracted to a local optimum) and a one-dimensional slice of the objective function along each dimension [Goldberg, 1990], and a virtual alphabet as a collection of the virtual characters. There may be other ways to define the interval-schemata. The primary conclusion is that an alternate schemata notation, such as the interval-schemata, makes it possible to apply schemata analysis and schema theory to real-valued chromosomes.

The issue of the failure of schema theory to predict success for alphabets of high cardinality in genetic search can now be addressed. Intuition suggests that the more expressive a language is, the more effective it should be in search, but the fundamental theorem of genetic algorithm claims that alphabets of low cardinality are more effective in search than alphabets of high cardinality. Antonisse [1989] attributes the disparity in what intuition suggests and the claims of the schema theory to an incorrect interpretation of the schema notation.

For instance, consider the schema 000# that is defined over the ternary alphabet, where # is a wild card. In Holland's interpretation of schema notation, the schema 000# refers to the set of strings {0000, 0001, 0002}. However, Antonisse says that "a set of schemata should adequately express the quantification over values, because the wild card for a three-valued alphabet can be between 0 and 1, 0 and 2, 1 and 2, or 0, 1, and 2." Therefore, the schema 000# should refer to the set of strings {0000, 0001}, {0000, 0002}, {0001, 0002}, and {0000 0001 0002}. In other words, for a three-valued alphabet, four wild cards are needed: $\# = \{\#_{01}, \#_{02}, \#_{12}, \#_{012}\}$. With this analysis, a string of length l defined over an alphabet of k cardinality has $(2^k-1)^l$ schemata [Antonisse, 1989], as opposed to the $(k+1)^l$ schemata predicted by Holland.

The implication of Antonisse's analysis is best illustrated with an example. Let A be a binary string of length 8. A has 256 distinct states. Let B be a ternary string of length 5. B has 243 distinct states. Using Holland's interpretation of schema notation, the number of schemata defined for A is $(2+1)^8 = 6561$, and the number of schemata defined for B is $(3+1)^5 = 1024$. The number of schemata defined for A is much greater than the number of schemata defined for B. Now using Antonisse's interpretation of schema notation, the number of schemata defined for A is $(2^2-1)^8 = 6561$, and the number of schemata defined for B is $(2^3-1)^5 = 16{,}807$. A couple of comments can be made about the results from this example:

1. With Antonisse's interpretation of schemata, the maximum number of schemata for string B is about 16 times the maximum number of schemata predicted for string B by Holland.

2. The maximum number of schemata for string B is about 2.5 times the maximum number of schemata for string A. This would suggest that alphabets of high cardinality should be more effective in search since they have higher schema processing capabilities than alphabets of low

cardinality.

3. Antonisse's interpretation of schemata notation for binary strings is consistent with Holland's interpretation of schemata notation for binary strings. This happens because the binary representation is indifferent to the ambiguity between the wild card interpretation and an "interpretation based on the quantification over subsets of values at a string position" [Antonisse, 1989].

Antonisse's interpretation of schemata aligns the predictions of the schema theory with the suggestions of intuition. Now, Antonisse's interpretation of schema notation can be applied to real-valued chromosomes. In this case the wild card would be a set of intervals, say $\#_i = \{\#_{i,\ [\alpha_1, \beta_i]}, \ldots, \#_{i,\ [\alpha_m, \beta_m]}\}$ for the connected schemata.

Therefore, with the definition of virtual characters and alphabet and Antonisse's interpretation of schemata notation, it is probable that the apparent paradox in the schema theory and the success of real-coded genetic algorithms may be resolved.

On the other hand, the inadequacy of the schema theory to explain the real-coded genetic algorithm and some other variants of the genetic algorithm that have better performance than the canonical genetic algorithm may mean that it is time to seek an alternate theory for genetic algorithms. Encouragingly, a number of theorists are developing new theories for genetic algorithms. For example, Peck and Dhawan present genetic algorithms as global random search methods, and some other theorists model genetic algorithms using the Markov chain and simulated annealing theories [Peck and Dhawan, 1995].

Much of the work done in the area of real-valued genetic algorithms has been in the development of genetic operators, i.e. crossover and mutation operators, for real-valued chromosomes. Fortunately, the users of real-coded genetic algorithms can continue to improve these operators and enjoy their success without fully understanding the theory behind genetic algorithms. This does not mean that the users of real-coded genetic algorithm are absolved of the responsibility of seeking to understand why a real-coded genetic algorithm works. Particularly, genetic algorithm theory is pertinent to understanding why a real-coded genetic algorithm may excel for some classes of problems, and why it may not work for other classes of problems.

1.5 Problem Statement

Out of all the operators that are responsible for advancing genetic search from one generation to the next, only the genetic operators are directly involved with the manipulation of the variable coding. With the real-number variable coding, it becomes possible to develop specialized genetic operators that take advantage of local continuities in the problem domain.

Therefore, the goal of this thesis is to develop a minimal set of specialized genetic operators for real-coded genes that effectively finds a solution vector $\vec{x} = (x_1, x_2, \ldots, x_n)$

that globally minimizes (or maximizes) the objective function

$$f(\vec{x}) = f(x_1, x_2, \ldots, x_n)$$

subject to

$$h_i(\vec{x}) = h_i(x_1, x_2, \ldots, x_n) = 0 \;;\quad i = 1\ldots p \quad \text{and} \quad p \le n$$

$$g_j(\vec{x}) = g_j(x_1, x_2, \ldots, x_n) \le 0 \;;\quad j = 1\ldots m$$

in the domain $L_k \le x_k \le U_k$ where $L_k, U_k \in \Re$ and $k = 1\ldots n$. This problem statement is quite general. There is no restriction on the nature of the objective function and constraints, i.e. the objective function and constraints can be linear or nonlinear.

The effectiveness of the genetic operators will be measured by the realization of the global optimum, the number of function evaluations required to obtain the global optimum, and the time it takes to obtain the global optimum for a wide variety of standard numerical optimization problems.

Chapter II

Genetic Operators for Real-Valued Chromosomes: *A Literature Survey*

2.1 Overview

Any genetic simulation starts with an initial population of individuals that are usually randomly distributed over a given domain. The successful progression of the simulation depends heavily on the effectiveness of the genetic operators: the genetic operators are responsible for determining how the individuals in a current generation recombine to produce new offspring. In terms of function optimization, the new offspring are the new potential solutions to an optimization problem. Effective genetic operators should lead to the production of better potential solutions or offspring with the progression of the genetic simulation. Furthermore, genetic operators should guard against premature convergence and exploit local continuities in objective functions with continuous variables when possible.

Genetic operators can be divided into two groups: crossover operators and mutation operators. The crossover operators are primarily responsible for the recombination process of the individuals and the mutation operators ensure that new information is introduced into the search space. A literature survey of the current available crossover and mutation operators for real-valued chromosomes is presented in this chapter.

2.2 Crossover Operators

One-Point Crossover: This is analogous to the usual one-point crossover in binary representation. The permissible crossover site is between the genes of a given chromosome, where the crossover site is randomly selected. An illustration of a one-point crossover is shown in Figure 2.1. One-point crossover is seldom used alone, and when it is used alone, the mutation rate must be high, since mutation is the only means by which new alleles are introduced into the search space.

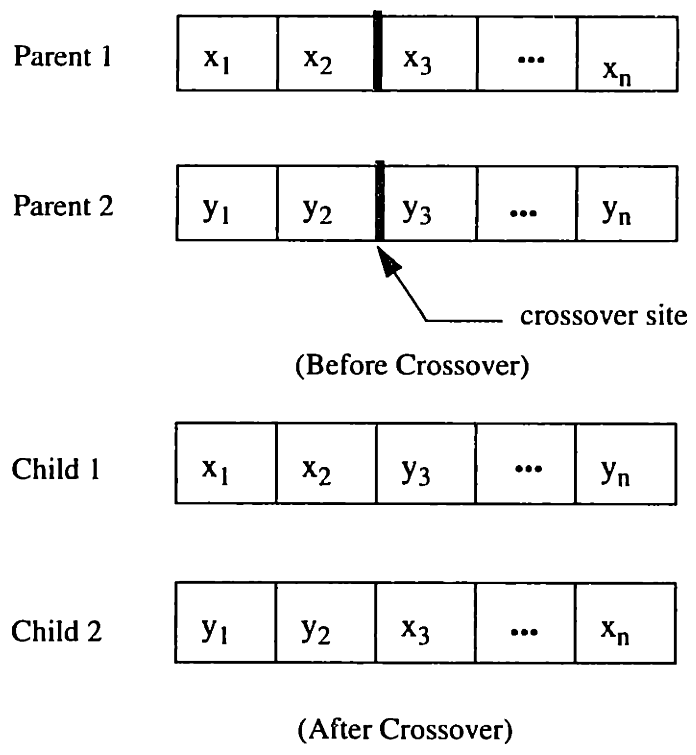

Parent 1
x_1 x_2 x_3 ••• x_n
Parent 2
y_1 y_2 y_3 ••• y_n
crossover site
(Before Crossover)
Child 1
x_1 x_2 y_3 ••• y_n
Child 2
y_1 y_2 x_3 ••• x_n
(After Crossover)

Figure 2.1: Illustration of one-point crossover

Two-Point Crossover: This is similar to the one-point crossover, except that there are two crossover sites. An illustration of the two-point crossover is shown in Figure 2.2.

Uniform Crossover: This is a multi-point crossover. Two offspring are produced from two parents. For every i^{th} gene of the offspring, a coin is flipped to determine which of the offspring gets the i^{th} gene of the first parent and which one gets the i^{th} gene of the second parent.

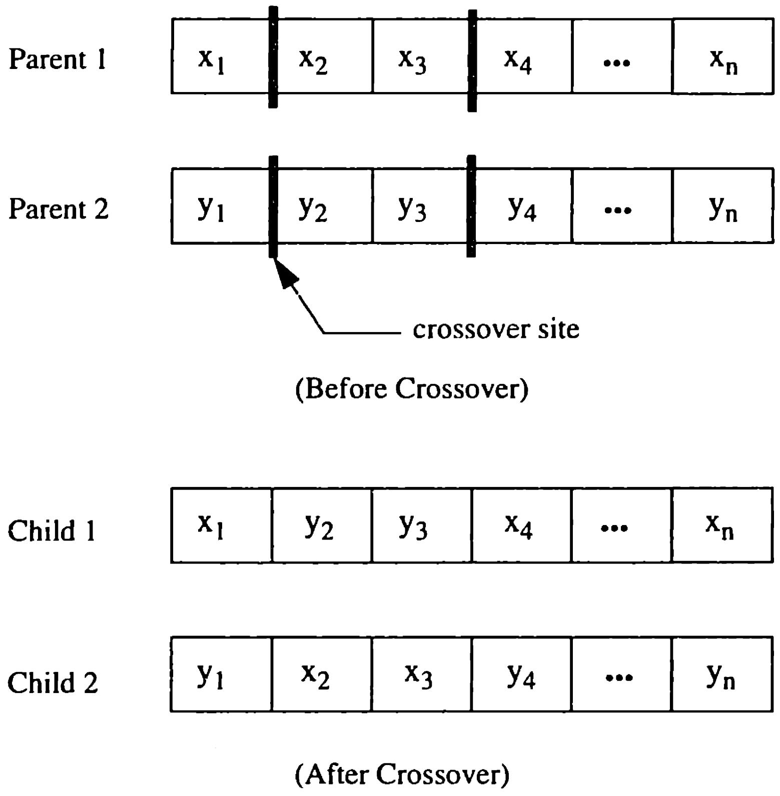
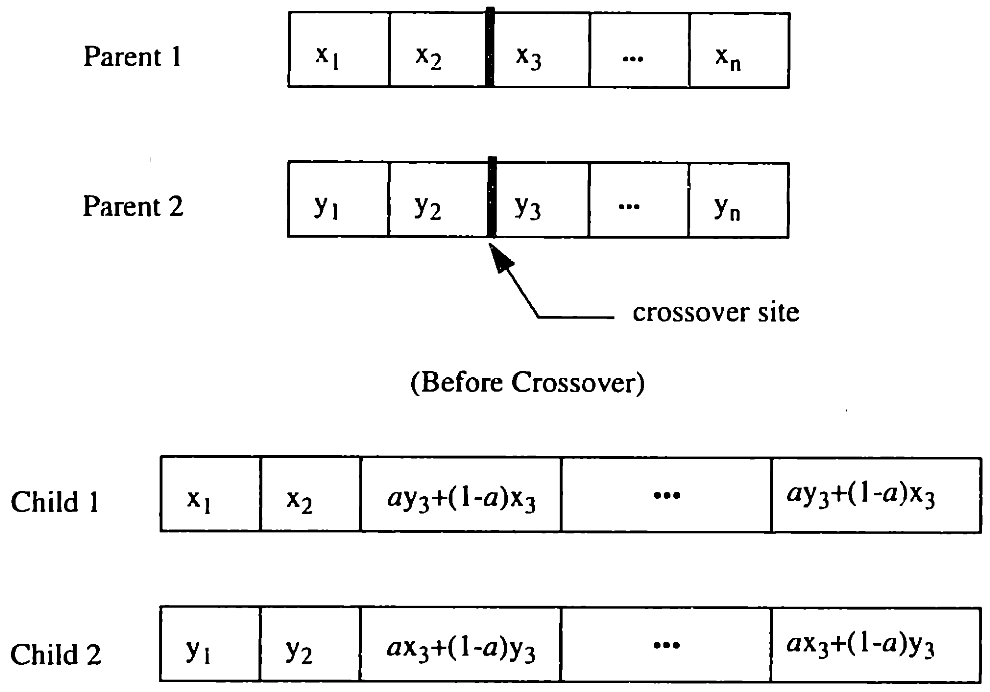

Parent 1
x_1 x_2 x_3 x_4 ••• x_n
Parent 2
y_1 y_2 y_3 y_4 ••• y_n
crossover site
(Before Crossover)
Child 1
x_1 y_2 y_3 x_4 ••• x_n
Child 2
y_1 x_2 x_3 y_4 ••• y_n
(After Crossover)

Figure 2.2: Illustration of two-point crossover

Simple Crossover [Michalewicz, 1994]: This is similar to one-point crossover. The major difference is that linear combinations of the alleles of the chromosomes are taken after the crossover site as opposed to just swapping alleles. Simple crossover is illustrated in Figure 2.3.

The parameter *a* (see Figure 2.3) starts at a value of 1 and if either Child 1 or Child 2 fails to satisfy the constraints, *a* is reduced by some constant 1/q. If after q attempts, Child 1 or Child 2 fails to satisfy the constraints, then Child 1 and Child 2 simply become copies of their parents. Note that when *a* is 1, the simple crossover is exactly the same as the one-point crossover.



(After Crossover)

Figure 2.3: Illustration of simple crossover

Average Crossover [Davis, 1991]: Given two parents $P_1 = \{v_{11}, v_{12}, ..., v_{1n}\}$ and $P_2 = \{v_{21}, v_{22}, ..., v_{2n}\}$, the i^{th} gene of the resulting offspring $C = \{c_1, c_2, ..., c_n\}$ has the form:

$$c_i = \frac{v_{1i} + v_{2i}}{2}$$

Linear Crossover [Wright, 1991]: Given two parents P_1 and P_2, the candidate offspring C_1, C_2, and C_3 have the form:

$$C_1 = 0.5P_1 + 0.5P_2 \ , \quad C_2 = 1.5P_1 - 0.5P_2 \ , \text{ and } \quad C_3 = -0.5P_1 + 1.5P_2$$

Note that the linear combinations of the parents are performed on a gene-by-gene basis. The best two of C_1, C_2, and C_3 are the new offspring.

Whole Arithmetic Crossover [Michalewicz, 1994]: Given two parents P_1 and P_2, the resulting offspring C_1 and C_2 have the form:

$$C_1 = aP_2 + (1-a)P_1$$

$$C_2 = aP_1 + (1-a)P_2$$

where $a \in [0, 1]$ is a randomly generated number, and the linear combinations of the parents is performed on a gene by gene basis.

BLX-α Crossover [Eshelman and Schaffer, 1993]: This is a blend crossover. It uniformly picks parameter values that lie between two points that contain two parents (see Figure 2.4). For instance, BLX-0.5 picks parameter values from points that lie on an interval that extends 0.5I on either side of the interval I between the parents. BLX-0.0, the flat crossover, picks parameter values from points between two parents [Radcliffe, 1990].

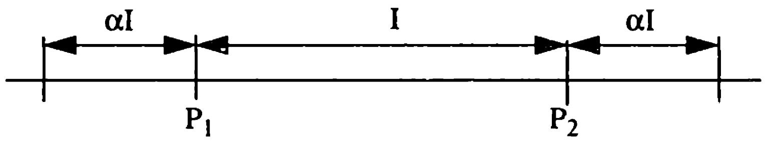


Figure 2.4: Illustration of BLX-α crossover

Heuristic Crossover [Michalewicz, 1994]: Given two parents $P_1 = \{v_{11}, v_{12}, ..., v_{1n}\}$ and $P_2 = \{v_{21}, v_{22}, ..., v_{2n}\}$, where the fitness of P_2 is higher than or equal to the fitness of P_1, the ith gene of the resulting offspring $C = \{c_1, c_2, ..., c_n\}$ has the form:

$$c_i = r(v_{2i} - v_{1i}) + v_{2i}$$

where $r \in [0...1]$. The parameter r is randomly generated.

It is possible that the resulting offspring can be infeasible with this operator. If the offspring is infeasible, another r will be generated and the heuristic crossover will be performed again. If the offspring is still infeasible after w attempts, the heuristic crossover will produce no offspring. The heuristic crossover has the added advantage that it uses the objective function trends to determine the direction of search.

2.3 Mutation Operators

Uniform Mutation: If $V^t = \langle v_1, v_2, \ldots, v_k, \ldots, v_n \rangle$ is a chromosome, then each gene v_k has exactly equal chance of undergoing mutation. The result of a mutative process is

$$V^{t+1} = \langle v_1, v_2, \ldots, v_k', \ldots, v_n \rangle$$

where $1 \le k \le n$ and v_k' is a randomly generated real number between the upper and lower bounds of v_k.

Boundary Mutation [Michalewicz, 1994]: If $V^t = \langle v_1, v_2, \ldots, v_k, \ldots, v_n \rangle$ is a chromosome, then each gene v_k has exactly equal chance of undergoing mutation. The result of a mutative process is

$$V^{t+1} = \langle v_1, v_2, \ldots, v_k', \ldots, v_n \rangle$$

where $1 \le k \le n$ and v_k' is either L_k or U_k where $L_k \le v_k' \le U_k$.

Non-Uniform Mutation [Michalewicz, 1994]: If $V^t = \langle v_1, v_2, \ldots, v_k, \ldots, v_n \rangle$ is a chromosome, then each gene v_k has exactly equal chance of undergoing mutation. The result of a non-uniform mutative process is

$$V^{t+1} = \langle v_1, v_2, \ldots, v_k', \ldots, v_n \rangle$$

where

$$v_k' = \begin{cases} v_k + \Delta(t, u_k - v_k) & \text{if random digit is 0} \\ v_k - \Delta(t, v_k - l_k) & \text{if random digit is 1} \end{cases}$$

u_k is the upper bound of v_k, l_k is the lower bound of v_k, and the delta function is defined as

$$\Delta(t, y) = y * r\left(1 - \frac{t}{T}\right)^b$$

where t is the generation number, T is the maximum number of generations, r is a randomly generated number between 0 and 1, and b is a system parameter determining non-uniformity. The delta function tends to zero as the number of generations tend to the maximum number of generations. This mutation operator perturbs the chromosome about its current position in the domain.

Gaussian Mutation: If $V^t = \langle v_1, v_2, \ldots, v_k, \ldots, v_n \rangle$ is a chromosome, then each v_k has exactly equal chance of undergoing mutation. The result of a mutative process is

$$V^{t+1} = \langle v_1, v_2, \ldots, v_k', \ldots, v_n \rangle$$

where v_k' is randomly generated from a Gaussian curve with a mean of v_k and a standard deviation of one.

2.4 A Note on the Performance of the Genetic Operators

The crossover operators can be categorized into two groups, namely blending crossovers and extrapolating crossovers. The blending crossovers are the average, arithmetic, BLX-0.0, and simple crossovers. The extrapolating crossover operators are the linear, heuristic, and BLX-α with $\alpha > 0.0$ crossovers. The point crossovers do not fit nicely under any of these two categories.

The blending and extrapolating crossovers have different strengths and weaknesses, hence it is common to find two or more crossover operators in a single real-coded genetic algorithm. For instance, the blending and point crossovers cannot sample parameter values that lie outside the extrema represented in the population. The implication is that if the optimum is not enclosed in the initial population, the blending and point crossovers will not be able to find the optimum. On the other hand, the extrapolating crossovers can sample parameter values that lie outside the extrema represented in the population. They can also produce offspring that are not in the design domain. The point crossovers have no way of sampling new parameter values other than the ones in the initial population if the probability of mutation is zero, neither do they have the ability to exploit local continuities in objective functions with continuous variables.

Michalewicz [1994] used two of the blending crossover operators, arithmetic and simple crossovers, and one of the extrapolating crossover operators, heuristic crossover, in his GENOCOP system. The combination of the operators achieved desired accuracies in results, but not without running the genetic algorithm for thousands of generations in some instances, where the bulk of the work done is in the local fine tuning of the optimum. It would be desirable to achieve desired accuracies in results in as few generations as possible, this is especially important for optimization problems with objective functions that are costly to evaluate.

A lot of work has not been done with the mutation operator, but it is clear that the desired characteristic of a mutation operator is perturbation about the current solution.

Random mutation has a tendency to produce infeasible offsprings for constrained numerical optimization problem. The boundary mutation is a nuisance if the optimum solution does not lie on the boundary of the domain.

Chapter III

New Genetic Operator for Real-Valued Chromosomes

3.1 Quadratic Crossover in Genetic Search

The beauty of real representation is that specialized genetic operators can be developed for numerical optimization problems. Therefore, the goal is to develop a single crossover operator that will effectively find the optimum to numerical optimization problems with the desired accuracy in as few generations as possible. The proposed crossover operator is quadratic crossover. Quadratic crossover combines the techniques of quadratic interpolation, a one-dimensional local search technique, and heuristic extrapolation, a pseudo-gradient local search technique, to produce a single individual from three individuals. Quadratic crossover is defined for maximization problems, since genetic algorithm problems are formulated as maximization problems. The description of the quadratic crossover is as follows:

Let $P_1 = \{v_{11}, v_{12}, v_{13}, \ldots, v_{1n}\}$, $P_2 = \{v_{21}, v_{22}, v_{23}, \ldots, v_{2n}\}$, and $P_3 = \{v_{31}, v_{32}, v_{33}, \ldots, v_{3n}\}$ be three parents selected for crossover, where the v_{ij}'s are real numbers. Also, let the associated fitness values of the parents be f_1, f_2, and f_3 respectively. Now, suppose that the fitness of a parent can be related to each gene of the parent such that $f_i = g_j(v_{ij})$, where i refers to the parent and j refers to the gene of the parent. Then if $g_j(v_{ij})$ is continuous on the given interval $[L_j, U_j]$, where $L_j \leq v_{ij} \leq U_j$, $g_j(v_{ij})$ can be approximated by a sufficiently high-order polynomial [Arora, 1989]. Since the idea is to find a general relationship between f_i and v_{ij}, A quadratic curve is sufficient to approximate $g_j(v_{ij})$.

In order to fit a quadratic curve to g_j, three points are needed at every j^{th} gene. The three points are obtained from the three parents. So for every j^{th} gene, the quadratic curve $h_j(\xi) = a_j\xi^2 + b_j\xi + c_j$ is fitted to the genes v_{1j}, v_{2j}, and v_{3j}. The coefficients of the quadratic curve are given by the following expressions:

$$a_j = \frac{1}{v_{3j} - v_{2j}}\left[\frac{f_3 - f_1}{v_{3j} - v_{1j}} - \frac{f_2 - f_1}{v_{2j} - v_{1j}}\right]$$

$$b_j = \frac{f_2 - f_1}{v_{2j} - v_{1j}} - a_j(v_{2j} + v_{1j})$$

$$c_j = f_1 - a_j v_{1j}^2 - b_j v_{1j}$$

The critical point of $h_j(\xi)$ is obtained from the following expression:

$$\frac{d}{d\xi} h_j(\xi) = b_j + 2a_j\xi = 0$$

The critical point is $\xi_j^* = -\frac{b_j}{2a_j}$. If $\frac{d^2}{d\xi^2} h_j(\xi) = 2a_j < 0$, then $\xi_j^* = -\frac{b_j}{2a_j}$ is the maximum point of $h_j(\xi)$.

Now, let the new individual be $D = \{d_{11}, d_{12}, d_{13}, \ldots, d_{1n}\}$. Then $d_i = \xi_j^*$ if ξ_j^* is the maximum point of $h_j(\xi)$ and $L_j \le \xi_j^* \le U_j$. This ends the quadratic interpolation part of the quadratic crossover.

What happens if $h_j(\xi)$ does not have a maximum point in the interval $[L_j, U_j]$ or if at least two of the jth genes have the same, or nearly the same, alleles, so that a_j, b_j, or c_j are undefined? This is where the heuristic extrapolation comes in. For every kth gene where the quadratic interpolation fails to provide a value for the gene of the new individual D, heuristic extrapolation will be invoked.

The heuristic extrapolation is implemented as follows: Let the best of the three parents be $M_1 = \{m_{11}, m_{12}, m_{13}, \ldots, m_{1n}\}$ and let the worst of the three parents be $M_2 = \{m_{21}, m_{22}, m_{23}, \ldots, m_{2n}\}$, then the kth gene of the new individual D that does not yet have a value will asume a value according to the expression:

$$d_k = r(m_{1k} - m_{2k}) + m_{1k}$$

where $r \in [0 \ldots 1]$ is a randomly generated number. If at least one d_k among all d_k's needed is not in the interval $[L_k, U_k]$, then r will be decreased by half, and the heuristic extrapolation will be attempted again. If after W attempts, at least one d_k among all d_k's needed is not in the interval $[L_k, U_k]$ then the heuristic extrapolation would have failed. If the heuristic extrapolation fails, then the empty k^{th} position of D wili be randomly assigned values from the k^{th} position of any of the three parents.

The structure of the quadratic crossover procedure is summarized in Figure 3.1.

```
procedure quadratic crossover

begin
  /* quadratic interpolation */
  $j \leftarrow 1$
  repeat
    compute $a_j$, $b_j$, $c_j$
    compute $\xi_j^*$
    if $L_j \le \xi_j^* \le U_j$
      $d_j = \xi_j^*$
      $index_j \leftarrow FULL$
    else $index_j \leftarrow EMPTY$
    $j \leftarrow j + 1$
  until (j > length of D)


  /* heuristic interpolation */
  $M_1$ = parent with highest fitness
  $M_2$ = parent with lowest fitness
  Generate a random number r in [0,1]
  $w \leftarrow 0$
  repeat
    feasible = TRUE
    $j \leftarrow 1$
    do
      if $index_j = EMPTY$
        $d_j = r*(m_{1j}-m_{2j}) + m_{1j}$
        if $d_j \notin [L_j, U_j]$
          feasible = FALSE
        $j \leftarrow j + 1$
    while (feasible = TRUE &
          j $\le$ length of D)
    r = r/2
    $w \leftarrow w + 1$
  until (w > W || feasible =
      TRUE)

  /* Random Assignment */
  if feasible = TRUE
    crossover is done
  else
    $j \leftarrow 1$
    do
      if $index_j = EMPTY$
        $d_j = v_{1j}$ or $v_{2j}$ or $v_{3j}$
        $j \leftarrow j + 1$
    while (j $\le$ length of D)
end
```

Figure 3.1: Quadratic crossover procedure

3.2 Performance Tests with Quadratic Crossover

The performance of the quadratic crossover is compared to the performances of five crossover operators for a selected number of standard genetic algorithm test problems with known optima. The crossover operators considered are linear, flat (BLX-0.0), BLX-0.5, average, and heuristic crossovers. These operators are described in Chapter II. The evaluation of the performance of the crossover operators are based on the number of times that the optimum is obtained in five independent runs and the average number of generations or function evaluations required to obtain the optimum, if the optimum is obtained at all.

The genetic algorithm used in optimization operates at steady-state with overlapping population. Steady-state with overlapping population means that only a specified percentage of the population is replaced every generation. Given a replacement ratio x and a population of size n, the percentage of the population to be replaced every generation is $n \cdot x$. So for every generation t+1, an intermediate population is created by inserting $n \cdot x$ newly created individuals into the population at generation t; then $n \cdot x$ worst individuals are deleted from the intermediate population to create the new population for generation t+1. The newly created individuals are created by crossover of individuals from the population at generation t. Individuals are selected for crossover by the roulette wheel selection scheme, where individuals with above-average fitness values have a better chance of getting selected for crossover. The control parameters for the steady-state genetic algorithm for all the test runs are listed below:

Population Size :	60
Probability of Crossover :	1.0
Probability of Mutation (per gene) :	0.001
Replacement Ratio :	0.25

The maximum number of generations depends on the difficulty of the problem but will not exceed 10,000. The mutation operator used in the optimization process is Gaussian mutator. A low mutation rate is selected for test runs so as to place emphasis on the performance of the crossover operators.

The performance of a real-coded genetic algorithm is also compared with the performance of a binary-coded genetic algorithm for the same set of test problems. The binary-coded genetic algorithm uses uniform crossover and flip-a-bit mutator with the same population size, crossover rate, and mutation rate as the real-coded genetic algorithm. The binary representation codes one variable with 16 bits, so a solution vector with n variables would be represented with a binary string of length $16 \cdot n$.

3.2.1 Optimization of Parabolic Function

The optimization problem for the parabolic function is stated as follows:

$$\text{Maximize} \quad f(x) = \sum_{i=1}^{3} x_i^2$$

subject to

$$-5.12 \le x_1, x_2, x_3 \le 5.12$$

This problem is from [DeJong, 1975]. A plot of the objective function in two-dimensions in shown in Figure 3.2. There are eight global optima with f* = 78.6432. The global optima are located at the boundary of the domain.

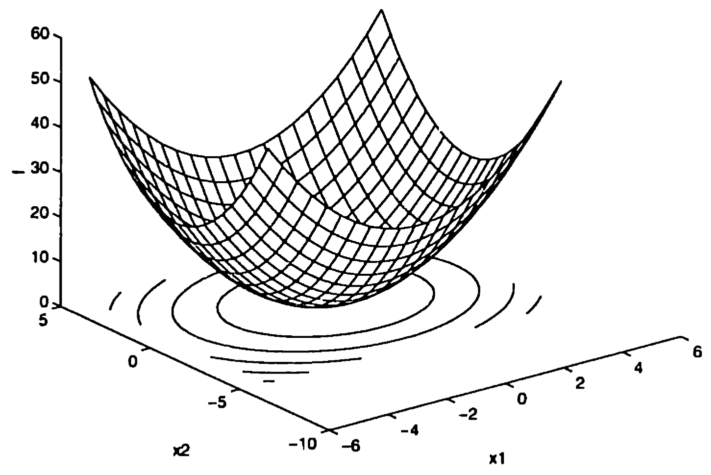


Figure 3.2: Plot of parabolic function in two-dimensions

The number of times that the optimum is found by the crossover operators and the binary implementation in five consecutive runs are listed in Table 3.1. The maximum number of generations for each run is 500. Convergence plots for the different crossover operators and the binary implementation are shown in Figures 3.3a through 3.3c.

The quadratic crossover found the optimum in all 5 runs, and so did the linear, heuristic, and BLX-0.5 crossovers. The binary-coded genetic algorithm also found the optimum in all 5 runs. The linear crossover found the optimum in the least number of generations. The performances of the blending crossovers, i.e. the average and flat crossovers, are not quite as good as the performances of the other crossover operators on this problem. The not-so-good performance of the average and flat crossovers is not surprising -- if the initial population does not envelope the global optimum, it would be impossible for the average and flat crossovers to attain the optimum (except, by mutation).

Table 3.1: Comparison of the performance of selected crossover operators for the parabolic function

Crossover Operator	% of time that exact optimum is found in 5 runs	Average number of generations to find optimum	Average number of function evaluations[a] to find optimum
Quadratic	100	45	735
Linear	100	1	84
Flat	0	▨	▨
BLX-0.5	100	170	2610
Average	40	470	7110
Heuristic	100	40	660
Uniform (Binary)	100	117 (best)	1815 (best)

a. Number of function evaluations = Replacement ratio * Population size * Number of generations + Population Size (for initialization)

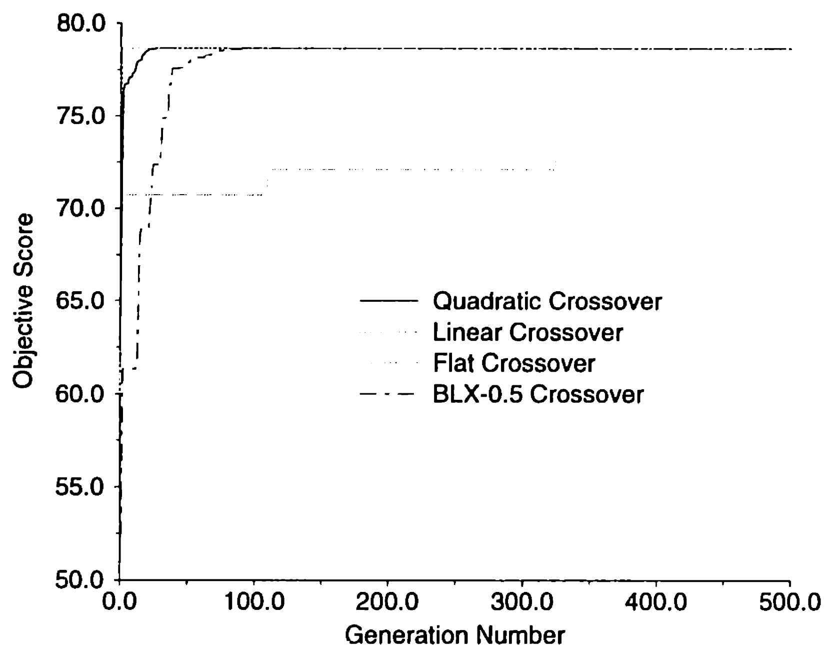


Figure 3.3a: Convergence plot for parabolic function (best performance)

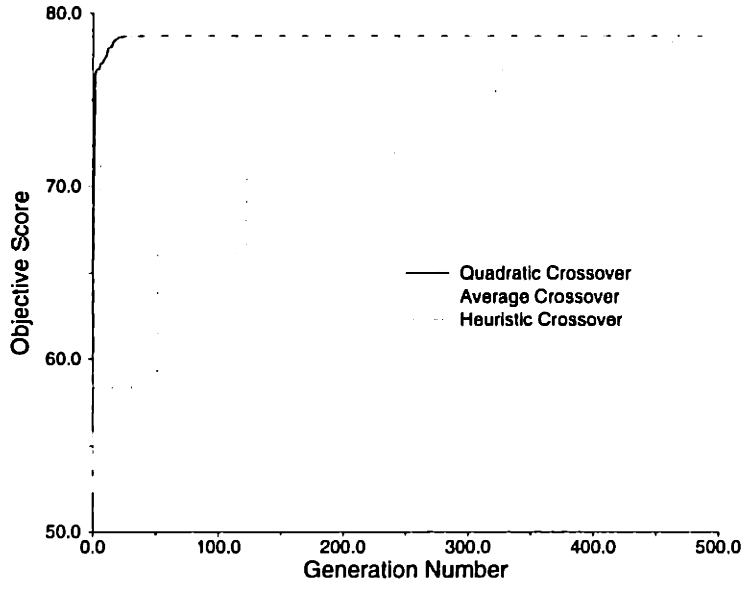


Figure 3.3b: Convergence plot for parabolic function (best performance)

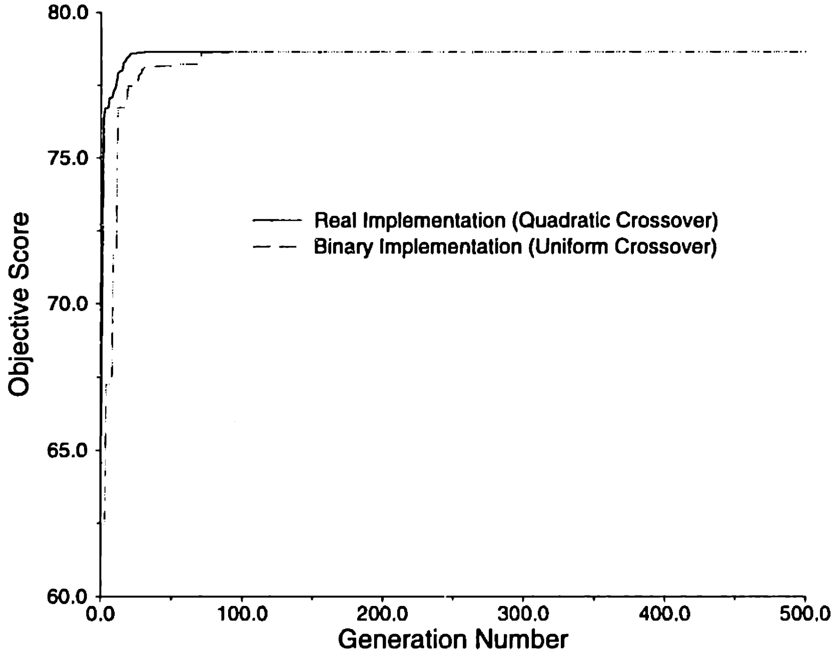


Figure 3.3c: Convergence plot for parabolic function (best performance)

3.2.2 Optimization of Rosenbrock Function

The optimization problem for the Rosenbrock function, or Rosenbrock's saddle, is stated as follows:

$$\text{Minimize } f(x) = 100\left(x_1^2 - x_2\right)^2 + \left(1 - x_1\right)^2$$

subject to

$$-2.048 \le x_i \le 2.048$$

This problem is from [DeJong, 1975]. A plot of the objective function is shown in Figure 3.4. The global optimum is f* = 0 at x* = (1,1).

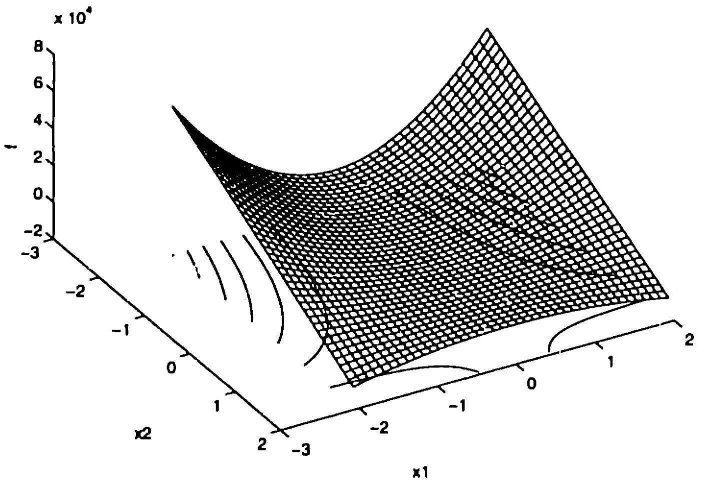


Figure 3.4: Plot of Rosenbrock's saddle

The number of times that the optimum is found by the crossover operators and the binary implementation in five consecutive runs are listed in Table 3.2. The maximum number of generations is 200. The convergence plots for the different crossover operators and the binary implementation are shown in Figures 3.5a through 3.5c. Note that the objective score in the convergence plots is -f.

The quadratic, linear, and heuristic crossovers found the exact optimum in all 5 runs. The binary-coded genetic algorithm found the optimum in 1 of the 5 runs. The average, flat, and BLX-0.5 crossovers did not find the optimum in any of the runs. The quadratic crossover required the least number of function evaluations to get the optimum.

Table 3.2: Comparison of the performance of selected crossover operators for Rosenbrock's saddle

Crossover	% time that optimum is found in 5 runs	Best optimum found	Average number of generations to find optimum	Average number of evaluations[a] to find optimum
Quadratic	100	0	57	915
Linear	100	0	92	2268
Flat	0	0.0411396	▨	▨
BLX-0.5	0	0.076034	▨	▨
Average	20	8.56772e-11	200	3060
Heuristic	100	0	103	1605
Uniform (Binary)	20	8.2217e-05	162	2490

a. Number of function evaluations = Replacement ratio * Population size * Number of generations + Population Size (for initialization)

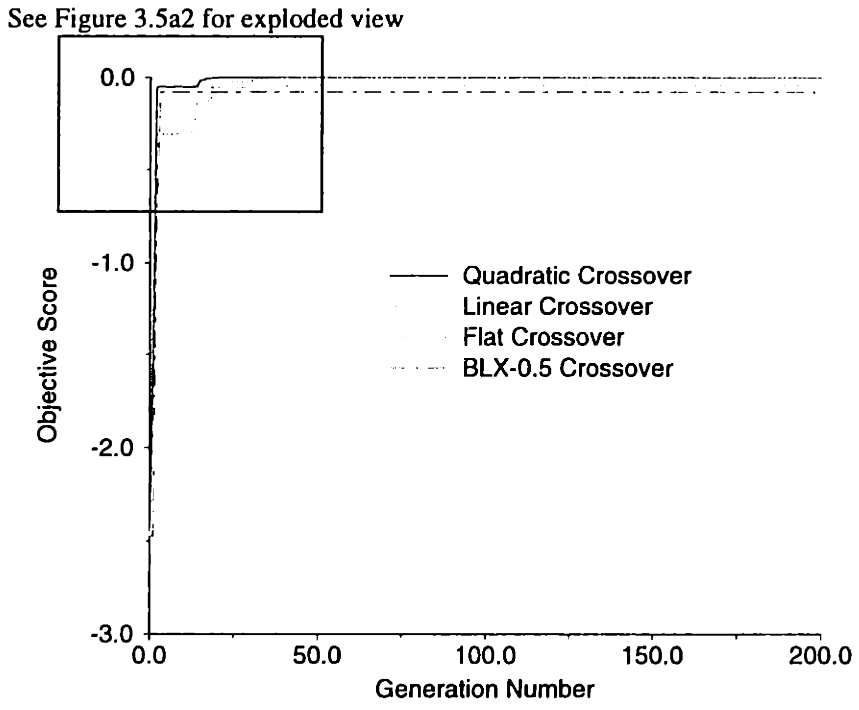


Figure 3.5a1: Convergence plot for Rosenbrock's saddle (best performance)

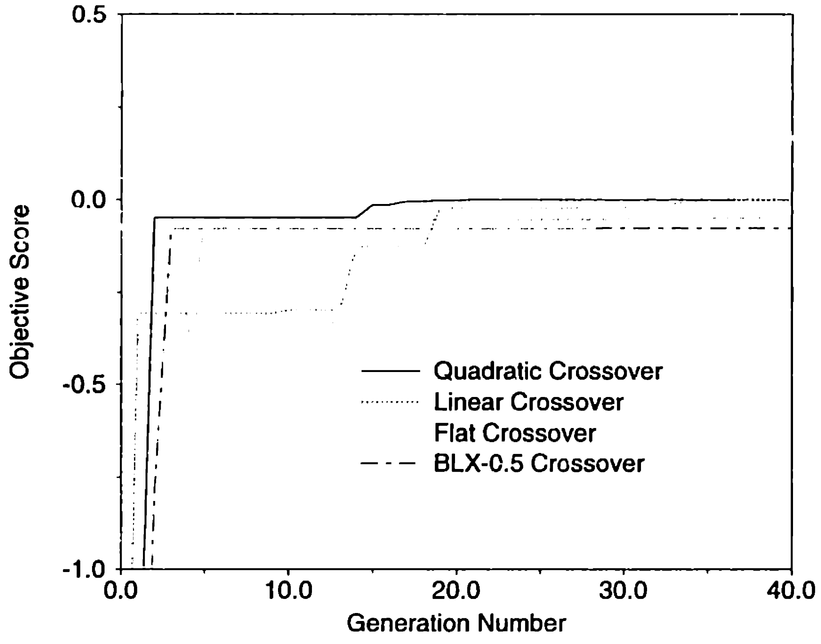


Figure 3.5a2: Exploded view of Figure 3.5a1

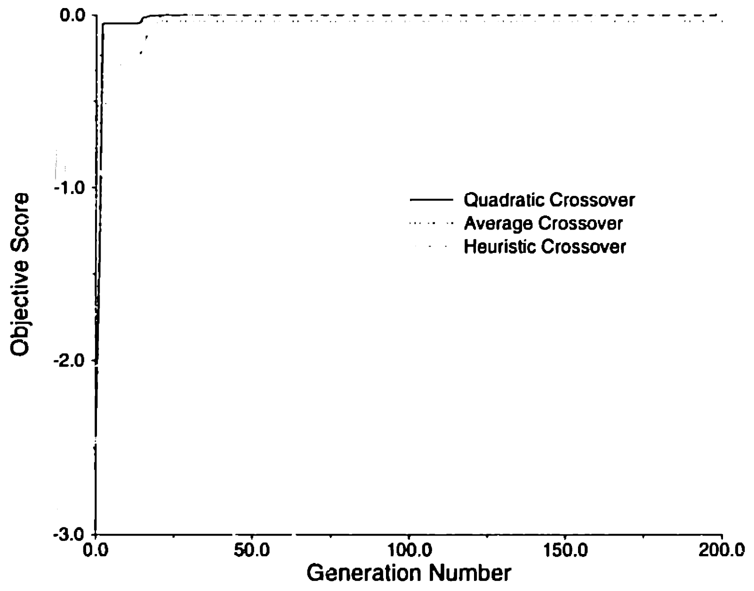


Figure 3.5b: Convergence plot for Rosenbrock's saddle (best performance)

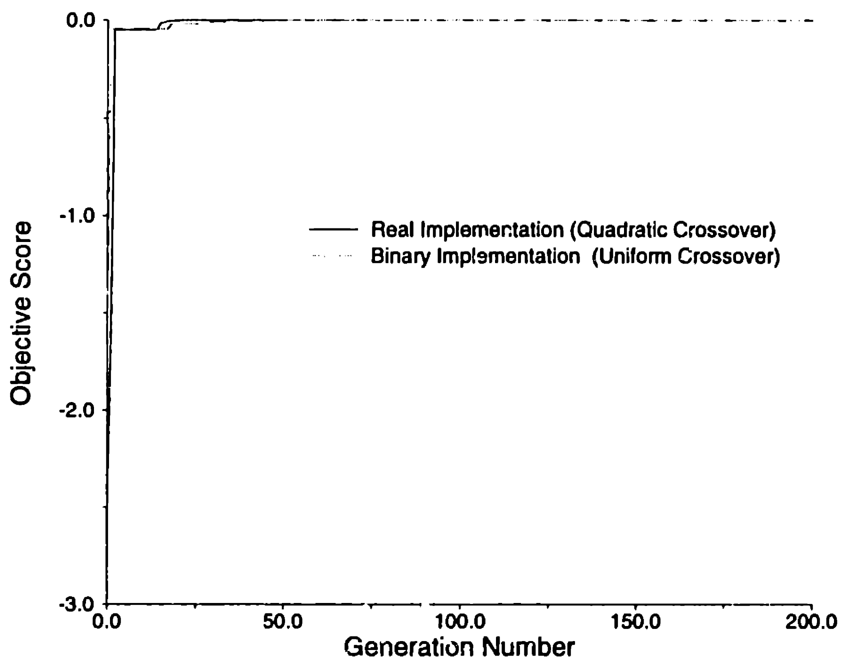


Figure 3.5c: Convergence plot for Rosenbrock's saddle (best performance)

3.2.3 Optimization of Stair-Steps Function

The optimization problem for the stair-steps function is stated as follows:

$$\text{Maximize } f(x) = \sum_{i=1}^{5} integer\left(x_i\right)$$

subject to

$$-5.12 \le x_i \le 5.12$$

This problem is from [DeJong, 1975]. The stair-steps function is discontinuous. A plot of the objective function in two-dimensions is shown in Figure 3.6. The global optimum is f* = 25 at x* = (5, 5, 5, 5, 5).

The number of times that the optimum is found by the crossover operators and the binary implementation in five consecutive runs are listed in Table 3.3. The maximum number of generations for all runs is 1,000. Convergence plots for the different crossover operators and the binary implementation are shown in Figures 3.7.

Again, the quadratic, linear, BLX-0.5, and heuristic crossovers found the optimum in all runs. The flat crossover found the optimum in 1 out of 5 runs. The average crossover did not find the optimum in any of the runs. The binary-coded genetic algorithm found the optimum in 4 out of 5 runs.

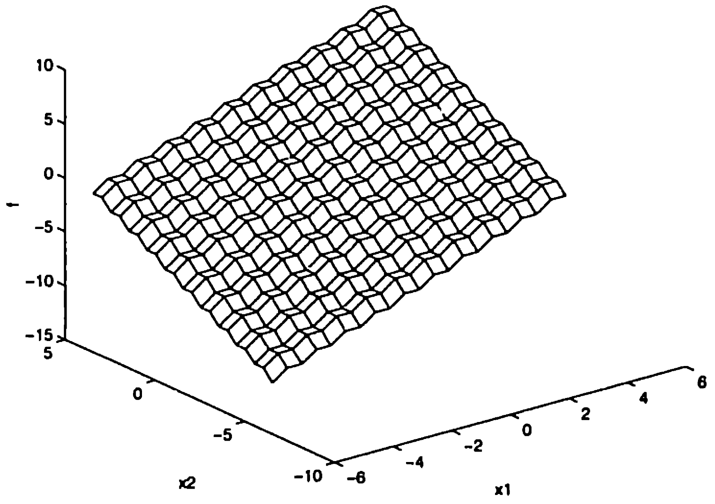


Figure 3.6: Plot of stair steps function in two-dimensions

Table 3.3: Comparison of the performance of selected crossover operators for the stair-steps function

Crossover	% time that optimum is found in 5 runs	Number of generations to obtain optimum (best)	Number of generations to obtain optimum (average)	Average number of evaluations[a] to find optimum
Quadratic	100	33	567	8565
Linear	100	9	21	564
Flat	20	958	▨	▨
BLX-0.5	100	378	378	5730
Average	0	▨	▨	▨
Heuristic	100	116	392	5940
Uniform (Binary)	80	28	28	480

a. Number of function evaluations = Replacement ratio * Population size * Number of generations + Population Size (for initialization)

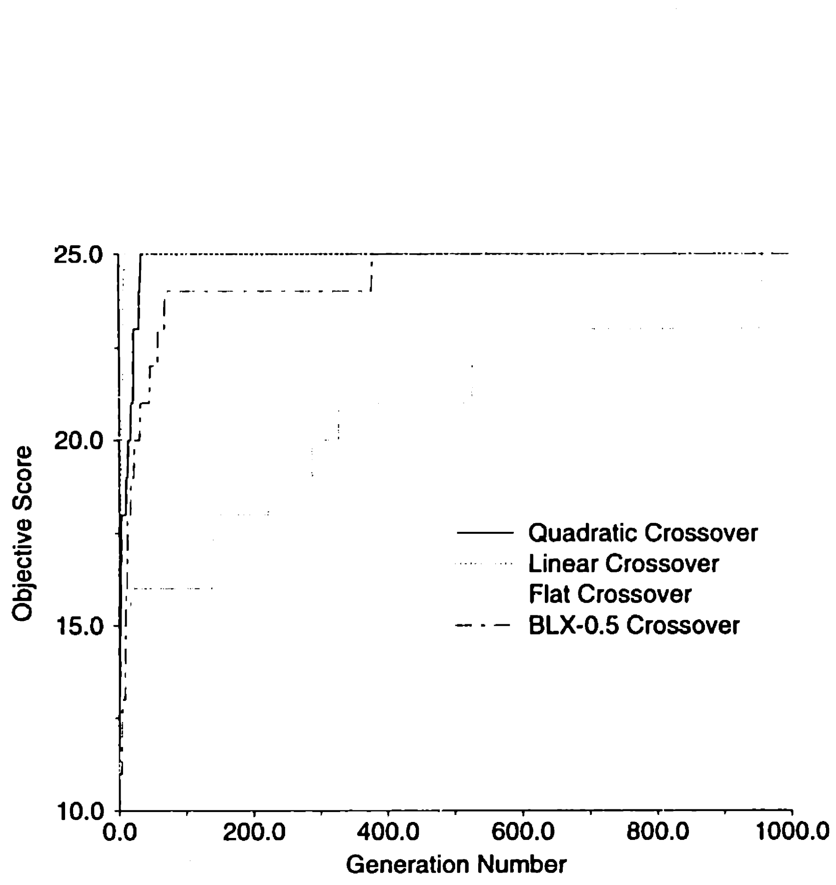


Figure 3.7a: Convergence plot for stair-steps (average performance)

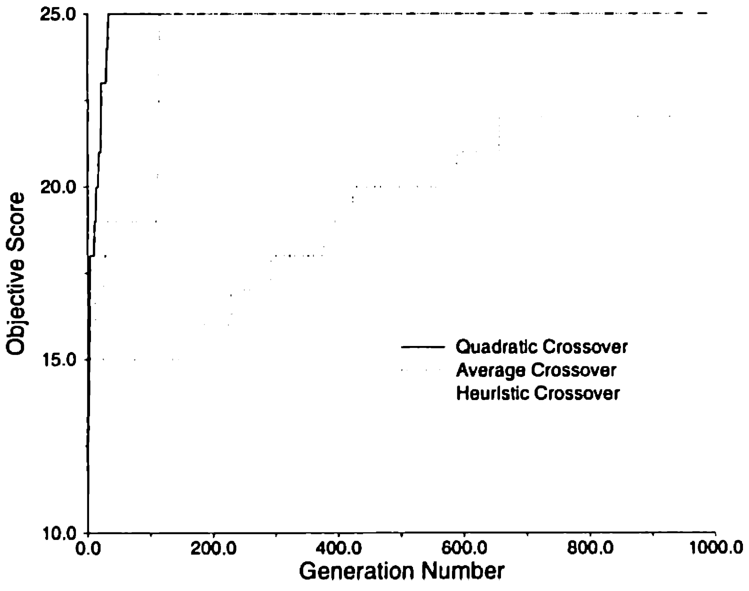


Figure 3.7b: Convergence plot for stair-steps (average performance)

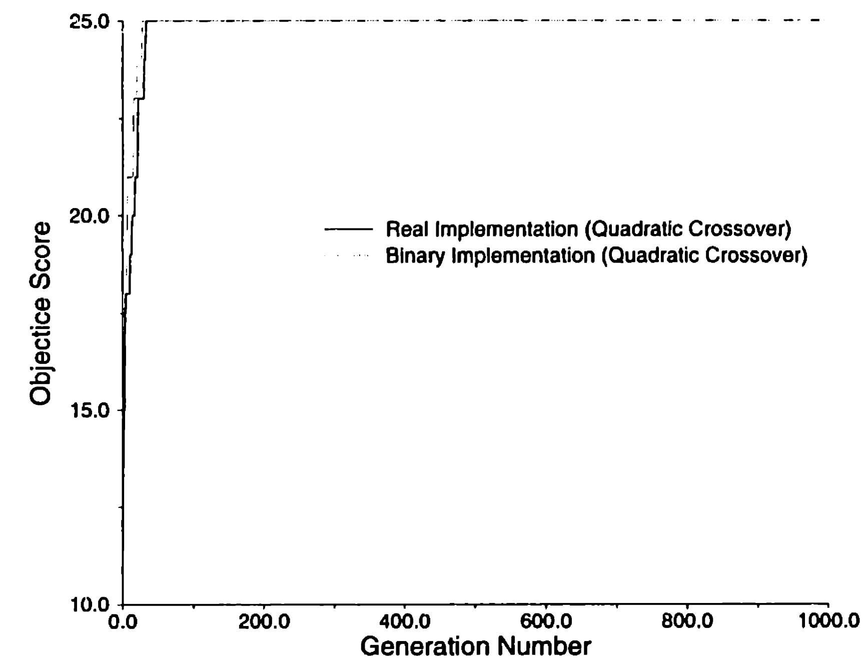


Figure 3.7c: Convergence plot for stair-steps (average performance)

3.2.4 Optimization of Inverted Shekel's Foxholes

The optimization problem is stated as follows:

$$\text{Maximize } f(x) = 500 - \cfrac{1}{0.002 + \sum_{j=1}^{25} \cfrac{1}{j + \sum_{i=1}^{2} \left(x_i - a_{ij}\right)^6}}$$

subject to

$$-65.536 \le x_i \le 65.536$$

Eshelman and Schaffer [1993] describe Shekel's foxholes as evenly spaced wells with sloped floors sunk in a plateau. A plot of the objective function is shown in Figure 3.8. There are 24 sub-optima. The global optimum is f* = 499.002.

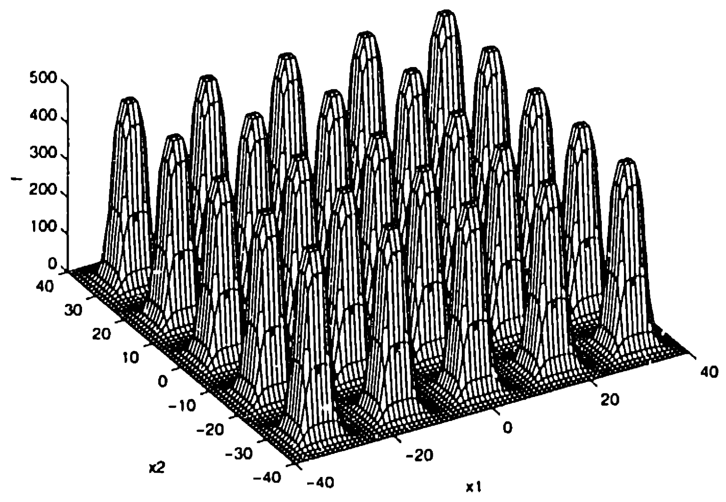


Figure 3.8: Plot of inverted Shekel's foxholes

The number of times that the optimum is found by the crossover operators and the binary implementation in five consecutive runs are listed in Table 3.4. The deviation of the optimum found by the crossover operators and the binary implementation from the global optimum is also shown in Table 3.4. The maximum number of generations for all runs is 1,000. Convergence plots for the different crossover operators and the binary implementation are shown in Figures 3.9a and 3.9b.

Table 3.4: Comparison of the performance of selected crossover operators for inverted Shekel's foxholes

Crossover	% time that optimum is found in 5 runs	average deviation from optimum (%)	max deviation from optimum (%)
Quadratic	20	0.2	0.4
Linear	0	1.4	1.4
Flat	0	1.2	1.4
BLX-0.5	20	0.4	0.4
Average	0	1.6	1.4
Heuristic	0	2.7	2.7
Uniform (Binary)	60	0.0	0.2

Quadratic crossover and BLX-0.5 found the exact optimum in 1 run out of 5 runs. The linear, flat, average, and heuristic crossovers did not find the optimum in any of the 5 runs. The binary implementation found the optimum in 3 out of 5 runs.

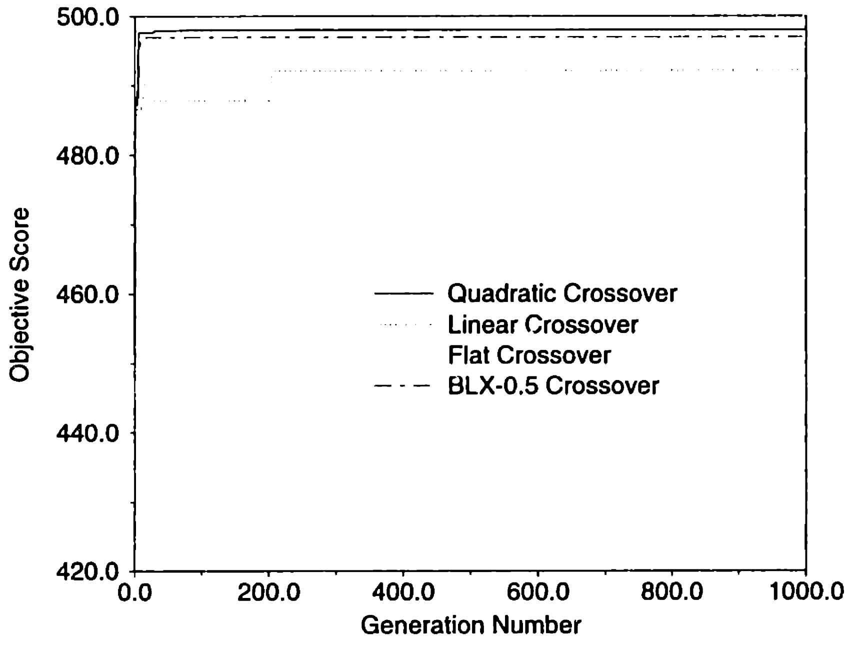


Figure 3.9a: Convergence plot for inverted Shekel's foxholes (average performance)

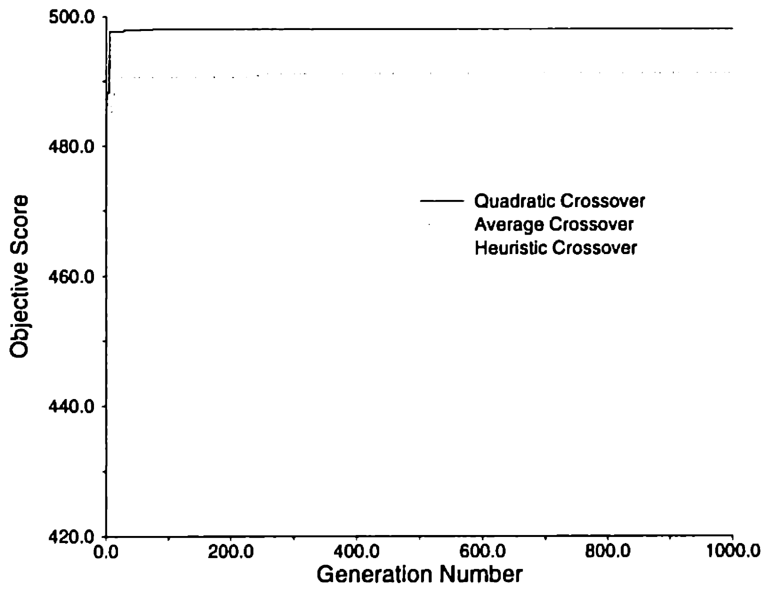


Figure 3.9b: Convergence plot for inverted Shekel's foxholes (average performance)

3.2.5 Optimization of Inverted Binary F6 Function

The optimization problem is stated as follows:

$$\text{Maximize} \quad f(x) = 0.5 - \frac{\left(\sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{\left(1.0 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$$

subject to

$$-10 \le x_1, x_2 \le 10$$

This function is from [Davis, 1991]. A plot of the objective function is shown in Figure 3.10. The global optimum is f* = 1 at x* = (0,0).

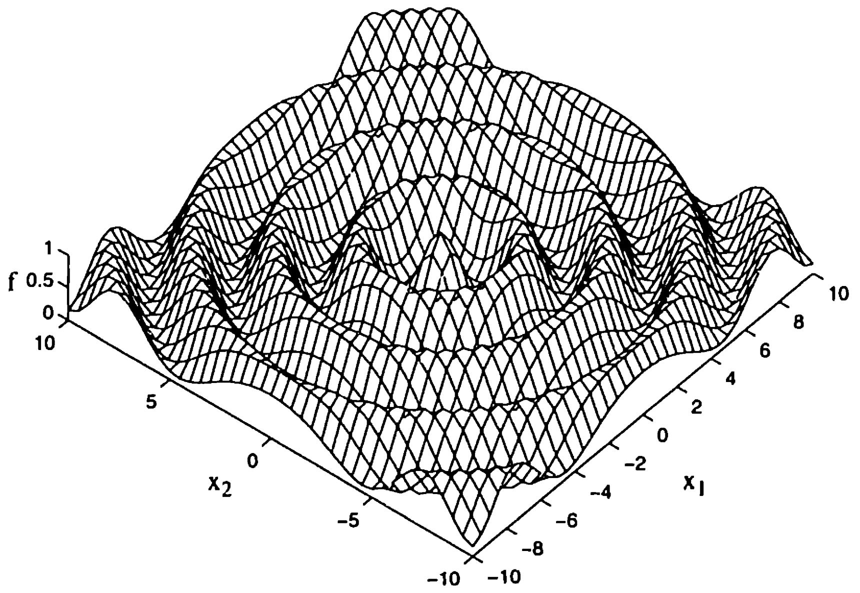


Figure 3.10: Plot of inverted binary F6 function

The number of times that the optimum is found by the crossover operators and the binary implementation in five consecutive runs are listed in Table 3.5. The maximum number of generations for all runs is 100.

Quadratic and BLX-0.5 crossovers found the exact optimum in 1 run out of 5 runs. The linear, flat, and heuristic crossovers did not find the optimum for any of the 5 runs. The performance of the average crossover is outstanding on this problem. One possible explanation for the outstanding performance of the average crossover on this problem is that the objective function is symmetrical about the origin and the optimum lies at the origin. To confirm this, the function is shifted off the origin so that the optimum is now at $(x_1,x_2) = (1,1)$. With this shift, the average crossover found the optimum in 1 out of 20 consecutive runs. However, shifting the function off the origin did not improve the performance of the other crossover methods on this problem. The poor performance of quadratic crossover has a lot to do with the nature of the objective function. The quadratic crossover may be more successful in finding the exact optimum for the original problem if two of the parents selected for crossover are nearly symmetric about the origin and the third parent is located near the origin, where the fitness of the third parent is higher than the fitness of the other two parents.

Table 3.5: Comparison of the performance of selected crossover operators for inverted binary F6 function

Crossover Operator	% time that optimum is obtained in 5 runs	Optimum obtained (on average)
Quadratic	20	0.990284
Linear	0	0.990284
Flat	0	0.990284
BLX-0.5	20	0.990284
Average	80	1
Heuristic	0	0.990284
Uniform (Binary)	0	0.990284

3.2.6 Optimization of Dynamic Control Problem

The optimization problem is stated as follows:

$$\text{Minimize} \quad f(x) = x_N + \sum_{k=0}^{N-1} \left(x_k^2 + u_k^2 \right)$$

subject to

$$x_{k+1} = x_k + u_k, \quad k=0,1,\ldots,N-1$$

$$-200 \le u_k \le 200, \quad x_0 = 100$$

This problem is from Michalewicz [1994]. The optimization problem has N=45 variables. The optimal value is given by

$$J^* = K_0 x_0^2$$

where K_k is the solution to the Riccati equation:

$$K_k = 1 + \frac{K_{k+1}}{(1 + K_{k+1})} \quad \text{and } K_N = 1$$

The optimal value is J* = 16180.4.

The number of times that the optimum is found by a selected number of crossover operators, including the quadratic crossover, in five consecutive runs are listed in Table 3.6. The maximum number of generations is 10,000. Convergence plots for the different crossover operators are shown in Figure 3.11.

The quadratic crossover found an optimum of 16180 in all 5 runs. The other crossover operators, including the binary implementation, did not find the optimum in any of the runs. Note that if the maximum number of generations is increased, some of the other crossover operators may eventually find the optimum. For the binary implementation, the control parameters also have to be changed.

Table 3.6: Comparison of the performance of selected crossover operators for the dynamic control problem

	% time that optimum is found in 5 runs	Average number of generations to obtain optimum	Average deviation from optimum	Maximum deviation from optimum
Quadratic	100	2420	0	0
Linear	0	▨	147.7	162
Flat	0	▨	37999	62829.7
BLX-0.5	0	▨	719.5	1035.2
Average	0	▨	8361.4	18665.4
Heuristic	0	▨	373.9	559.7
Uniform (Binary)	0	▨	227026	▨

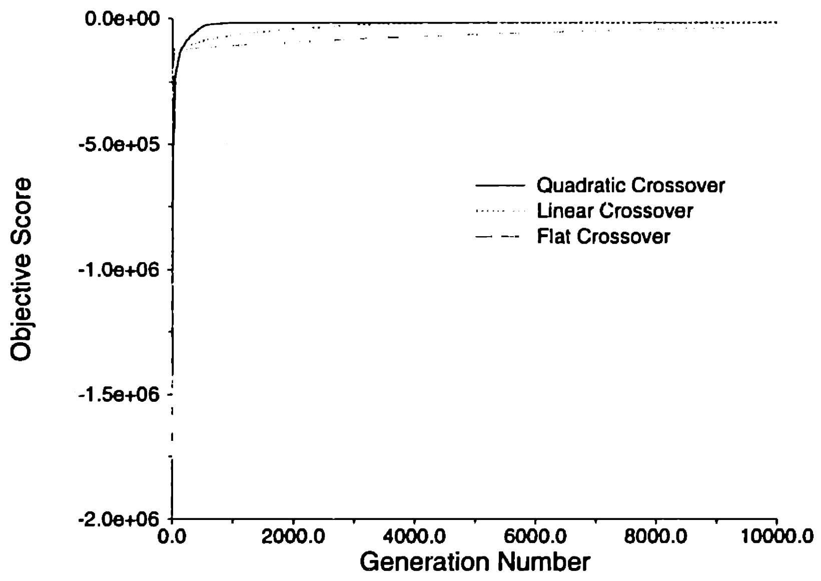


Figure 3.11a: Convergence plot for dynamic control problem (best performance)

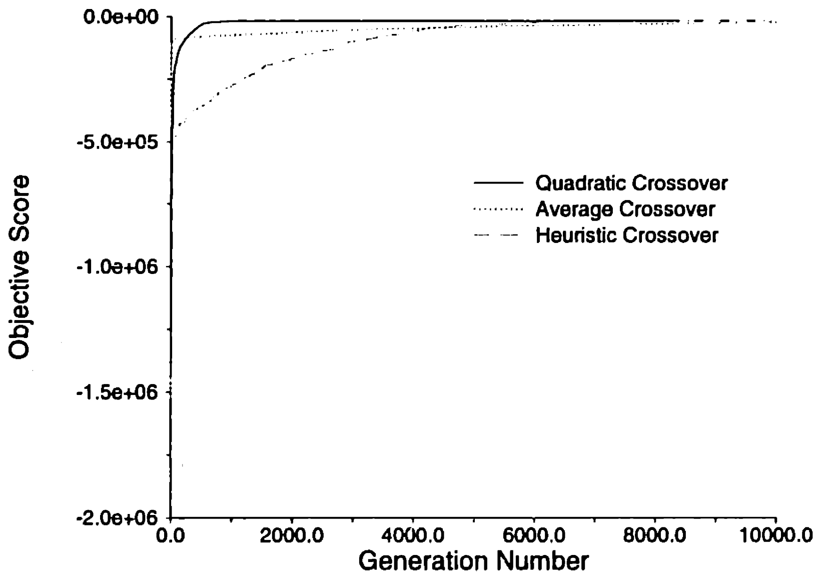


Figure 3.11b: Convergence plot for dynamic control problem (best performance)

3.2.7 Time Performance: Binary vs. Real Implementation

The time performance for the real implementation and binary implementation is compared in this section. The dynamic control problem is again considered here:

$$\text{Minimize} \quad f(x) = x_N + \sum_{k=0}^{N-1} \left(x_k^2 + u_k^2 \right)$$

subject to

$$x_{k+1} = x_k + u_k, \quad k=0,1,\ldots,N-1$$

$$-200 \le u_k \le 200, \quad x_0 = 100$$

where N is the number of variables. The binary implementation codes a variable with 30 bits, so a solution vector with N variables has $30 \cdot N$ bits. For both the real and binary implementation, the population size is 100, the probability of crossover is 1.0, the probability of mutation is 0.1, the replacement ratio is 0.5, and the maximum number of generations is 10,000. The real implementation uses quadratic crossover, and the binary implementation uses uniform crossover.

The results are shown in Table 3.7. The rate at which CPU time increases as a function of the number of variables is much higher for the binary implementation than it is for the real implementation (see Figure 3.12). The general conclusion is that the real implementation is more efficient than the binary implementation.

Table 3.7: CPU time (s) as a function of number of variables

N	5	10	15	20	25
Time (Float)	82	84	134	174	199
Optimum	16179	16180	16179.9	16179.9	16179.9
Time (Binary)	165	300	408	566	658
Optimum	16194.9	16218.4	17362.5	19764.7	50083.6

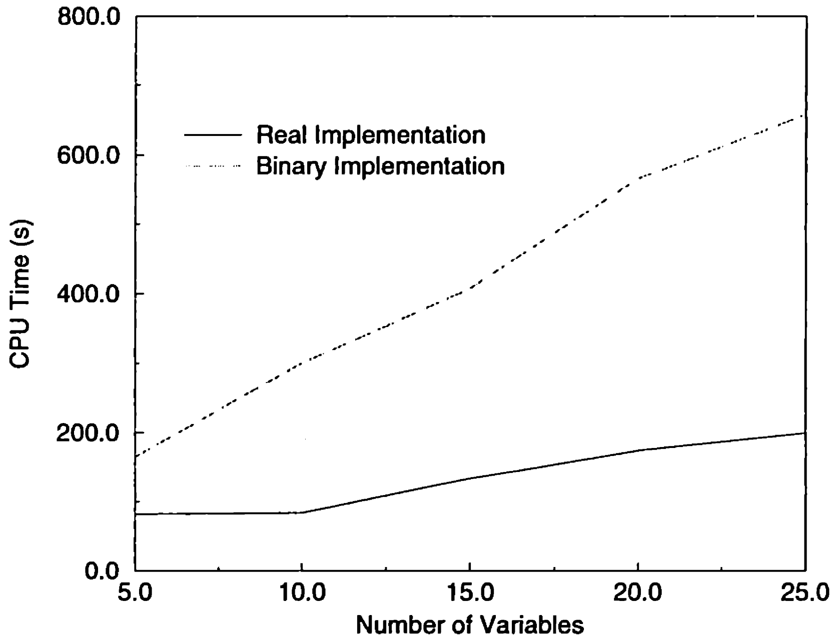


Figure 3.12: CPU time (s) as a function of number of variables for real and binary implementations

3.2.8 Summary of Results

The percentage of time that the optimum is found in five independent runs for all the test problems is summarized in Table 3.8. Quadratic crossover found the optimum in all five runs for four of the six problems. The linear and heuristic crossovers found the optimum in all five runs for three of the six probelms. The BLX-0.5 crossover found the optimum in all five runs for two of the test problems. The average and flat crossovers did not find the optimum for all five runs for any of the test problems. The performance of the quadratic crossover on the dynamic control problem is quite good, considering that none of the other crossover operators found the optimum in 10,000 generations. The general conclusion is that the quadratic crossover operator is very reliable and efficient in guiding the search to the global optimum.

Table 3.8: Summary of percentage of time that optimum is found in 5 runs for test problems

Crossover Operator	Parabolic Function	Rosenbrock Function	Stair-Step Function	Shekel's Function	Binary F6 Function	Dynamic Control
Quadratic	100	100	100	20	20	100
Linear	100	100	100	0	0	0
Flat	0	0	20	0	0	0
BLX-0.5	100	0	100	20	20	0
Average	40	20	0	0	80	0
Heuristic	100	100	100	0	0	0

Chapter IV

Genetic Algorithm for Numerical Optimization Problems: the GaNOP System

4.1 Overview

GaNOP is an acronym for *G*enetic *a*lgorithm for *N*umerical *O*ptimization *P*roblems. GaNOP optimizes unconstrained and constrained, linear and nonlinear optimization problems using genetic search algorithm. In mathematical terms, GaNOP minimizes (or maximizes) the objective function

$$f(\vec{x}) = f(x_1, x_2, \ldots, x_n)$$

subject to

$$h_i(\vec{x}) = h_i(x_1, x_2, \ldots, x_n) = 0; \; i = 1 \ldots p; \text{ and } p \le n$$

$$g_j(\vec{x}) = g_j(x_1, x_2, \ldots, x_n) \le 0; \; j = 1 \ldots m$$

in the domain $L_k \le x_k \le U_k$ where $L_k, U_k \in \Re$ and $k = 1 \ldots n$.

GaNOP is modeled after the genetic search algorithm GENITOR [Whitley, 1989]. The details of the GaNOP system are presented in this chapter.

4.2 Description of the GaNOP System

The GaNOP system is developed with a C++ genetic algorithm library, GAlib[1]. The following is a summary of the basic steps in a GaNOP evolutionary process:

1. GAlib is a library of C++ genetic algorithms objects that was developed by Matthew Wall at Massachusetts Institute of Technology. For more information on GAlib, refer to the URL address http://lancet.mit.edu/ga/ on the internet.

4.2.1 Preprocessing

This step involves the transformation of the optimization problem stated above to a GaNOP problem. The GaNOP problem is a constrained problem: the inequality constraints in the original problem are replaced with penalty functions and the equality constraints are eliminated, but the domain is bounded.

For minimization problems, the GaNOP problem is stated as follows:

$$\text{Maximize } F(x) = -f(x) - \sum_{i=1}^{k} \Phi_i$$

$$\text{subject to } L_k \le x_k \le U_k$$

where k is the number of equality constraints and Φ_i, a penalty function, is defined as

$$\Phi_i = \begin{cases} c_i g_i + d_i & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

where g_i is an inequality constraint and c_i and d_i are user-specified constants.

For maximization problems, the GaNOP problem is stated as follows:

$$\text{Maximize } F(x) = f(x) - \sum_{i=1}^{k} \Phi_i$$

$$\text{subject to } L_k \le x_k \le U_k$$

where all the definitions above still apply.

The equality constraints are not directly included in the GaNOP problem statement because it is almost impossible to exactly satisfy the equality constraints with the floating point representation. Perhaps if the equality constraint is relaxed a little, say $|h_i| < 1e-9$, then it will be possible to take care of the equality constraints with penalty functions. But in many of the problems, it will be much easier to identify a couple of independent variables and calculate the dependent variables using the expressions for the equality constraints. Eliminating equality constraints will generally result in additional inequality constraints that can be taken care of by penalty functions. Illustrative examples are included in the next two chapters.

There is no structured way to assign values to c_i and d_i in the penalty functions. Usually, small values are chosen for the constants, and if GaNOP violates constraints for more than the first 20% of the maximum number of generations, then the values of c_i and d_i are increased. The penalty awarded to the violation of constraints should be big enough to discourage GaNOP from moving to the infeasible region.

The genetic operators used in GaNOP ensure that the bound constraints are not violated.

4.2.2 Initialization

GaNOP starts with an initial population of single-chromosome individuals or potential solutions to the optimization problem. The initial population is created as follows: Given a chromosome $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$, each gene x_i is randomly assigned an allele that has a value between (and including) the upper and lower bounds prescribed for the ith allele, where i goes from 1 to n. This process is repeated for all the chromosomes in the initial population. Note that this initialization procedure may result in chromosomes that are infeasible, but the initialization procedure will not result in chromosomes that violate the bound constraints.

After initialization, the objective scores of the chromosomes are obtained by evaluating the objective function for each individual. The objective scores are mapped to the fitness scores using sigma truncation scaling:

$$f_i' = f_i - (f_{ave} - c \cdot \sigma)$$

where f_i' is the fitness score, f_i is the objective score of an individual, f_{ave} is the population's average objective score, c is a small integer, and σ is the population's standard deviation. The sigma truncation scaling sets possible negative f_i' to zero.

4.2.3 Evolution

Evolution starts with the initial population created during the initialization process. The transition from generation t to generation $t + 1$ is accomplished through three basic operations: reproduction, crossover, and mutation. The explanation of the three basic operations follows:

Reproduction involves the selection of three individuals (or parents) from the population at generation t using the roulette wheel selector. The roulette wheel selector picks an individual based on the magnitude of the fitness score of the individual relative to the fitness scores of the rest of the population. Indviduals with higher scores are more likely to be selected.

Crossover involves the production of a new offspring by combining the parents selected during reproduction. The combination process is performed by the quadratic crossover operator. The crossover operator is applied with a probability. If the crossover operator is not applied, the new offspring will be a copy of one of the parents.

The new offspring undergoes mutation. The mutation operator is Gaussian mutator. The Gaussian mutator used in GaNOP is slightly different from the one presented in Chapter II. The modified Gaussian mutator is defined as follows:

> If $V^t = \langle v_1, v_2, \ldots, v_k, \ldots, v_n \rangle$ is the new offspring, then each v_k has exactly equal chance of undergoing mutation. The result of a mutative process is
>
> $$V^{t+1} = \langle v_1, v_2, \ldots, v_k', \ldots, v_n \rangle$$
>
> where v_k' is randomly generated from a Gaussian curve with a mean of v_k. The standard deviation of the Gaussian curve is $0.5 \cdot (U_k - L_k)$ if the ratio of the generation number to the maximum number of generations is less than 0.75; the standard deviation is $0.1 \cdot (U_k - L_k)$ if the ratio of the number of generations to the maximum number of generations is greater than or equal to 0.75. Note that $L_k \le v_k \le U_k$. If $v_k' \notin [L_k, U_k]$, another v_k' will be generated until v_k' satisfies the bound constraints for v_k.

The process of selecting parents, combining parents to produce new offspring, and mutating the new offspring continues until the number of offspring produced is equal to the total number of individuals to be replaced in one generation. The total number of individuals to be replaced is given by the product of a user-specified replacement ratio and the population size.

The new offspring are then inserted into the population at generation t, and then the worst individuals are deleted from the population to make the new population for generation $t + 1$. The number of worst individuals deleted from the population is equal to the number of new offspring inserted into the population. Depending on the fitness values of the offspring in relation to the fitness values of the population at generation t, the offspring may or may not be a member of the new population. The new population is then evaluated; the objective scores are scaled using the sigma truncation scaling.

Evolution terminates when the maximum number of generations specified by the user has been completed. GaNOP keeps track of the best individual throughout the evolution process. A flowchart of GaNOP after preprocessing is shown in Figure 4.1.

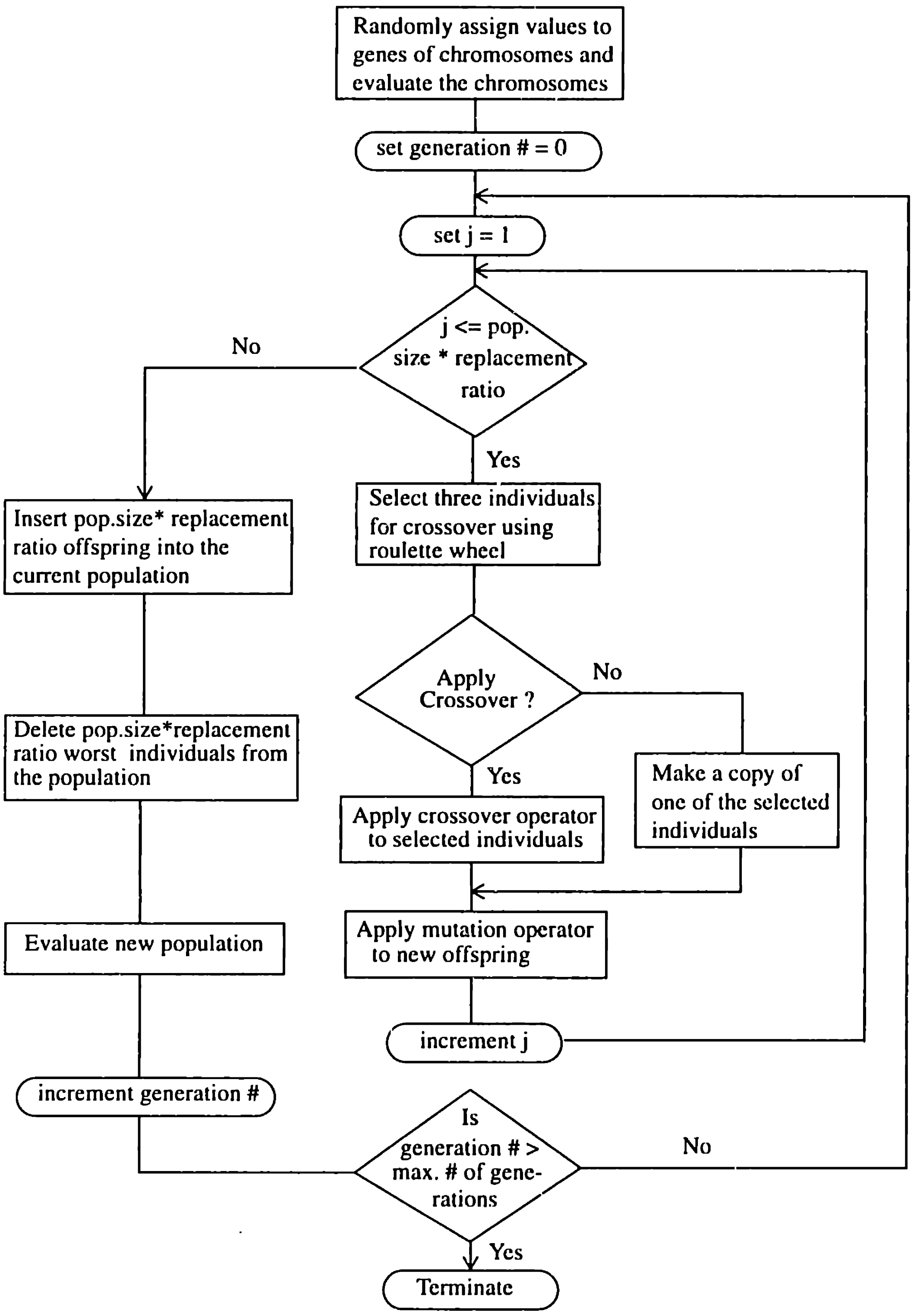

Randomly assign values to genes of chromosomes and evaluate the chromosomes
set generation # = 0
set j = 1
j <= pop. size * replacement ratio
No
Yes
Insert pop.size* replacement ratio offspring into the current population
Select three individuals for crossover using roulette wheel
Apply Crossover ?
No
Yes
Delete pop.size*replacement ratio worst individuals from the population
Make a copy of one of the selected individuals
Apply crossover operator to selected individuals
Apply mutation operator to new offspring
Evaluate new population
increment j
increment generation #
Is generation # > max. # of generations
No
Yes
Terminate

Figure 4.1: Flowchart of GaNOP

Chapter V

Optimization with GaNOP: *Theoretical Problems*

5.1 Overview

Unlike many optimization techniques, GaNOP does not impose any stringent conditions on the nature of the problem that it optimizes. In fact, the only two requirements are that the design domain should be bounded and the design variables should be continuous. The versatility of GaNOP is demonstrated by optimizing a wide variety of standard nonlinear programming theoretical test problems. The test problems are classified as theoretical problems because their global optima are known.

For every optimization problem considered, the optimum obtained with GaNOP will be compared to the global optimum of the problem. When possible, the performance of GaNOP will be compared to the performance of the optimization technique used in the source of a particular problem, where the parameters of interest are the accuracy of the result, the run-time, and the number of function evaluations required to obtain the optimum. Naturally, GaNOP should require more function evaluations than other optimization techniques that have some auxiliary knowledge of the nature of the objective function and design domain.

Optimization problems will be reformulated as GaNOP problems. Five independent runs are made for all the test problems, and the optimum found in each run is reported. For every test problem, the history of the optimization process for a selected run (usually the best) is presented. For the same selected run, the running percentage of the time that the different parts of the quadratic crossover[1] operator used in GaNOP are implemented will be reported. For any generation *i*, where *i* is greater than zero, the percentage of time that one part of the quadratic crossover, say quadratic interpolation, occurs is equal to the total number of quadratic interpolations (counted on a gene-by-gene basis) performed from generation 1 to generation *i* divided by the total number of crossovers from generation 1 to generation *i*. The total number of crossovers for any generation *i* is the product of the total number of individuals produced by quadratic crossover from generation 1 to generation *i* and the length of an individual.

The default values for the GaNOP control parameters are listed in Table 5.1. When necessary, the default values of the control parameters will be altered to guide GaNOP to the optimum solution. All runs are carried out on a SGI-Indy station.

1. The three parts in the quadratic crossover are quadratic interpolation, heuristic extrapolation, and random assignment. The quadratic crossover procedure is given in Chapter III.

Table 5.1: GaNOP control parameters

Control Parameters	Value
Population Size	100
Probability of Crossover	1.0
Probability of Mutation	0.1
Replacement Ratio	0.5
Number of Generations	Variable

5.2 Optimization of Quadratic Functions

Two quadratic functions are considered. The first function, the Rosen-Suzuki function, is taken from Hock and Schittkowski [1981]. The second function is a multimininma parabolic function from Corana et al. [1987]. The multiminima parabolic function in two dimensions has local minima of the order of 10^{10}.

The following is the description of the quadratic test functions and the optimization process of the functions with GaNOP.

5.2.1 Rosen-Suzuki Function

The minimization problem is stated as

$$\text{Minimize } f(\vec{x}) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$$

subject to

$$\begin{aligned} g_1 &= -8 + x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 \le 0 \\ g_2 &= -10 + x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 \le 0 \\ g_3 &= -5 + 2x_1^2 + x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 \le 0 \\ \vec{x} &\in [-50, 50] \end{aligned}$$

This problem features the minimization of a convex objective function subject to three quadratic inequality constraints. The design domain is unbounded in the original problem statement. The bounds on the variables have been introduced to make the optimization problem solvable by GaNOP. The bounds on the variables introduce four additional inequality constraints. The global optimum is at x*=(0,1,2,-1) with f(x*) = -44.

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(\vec{x}) = -f(\vec{x}) - \sum_{i=1}^{3} \Phi_i$$

where

$$\Phi_i = \begin{cases} 5 \cdot g_i + 5 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \text{for i = 1,2,3}$$

$$\vec{x} \in [-50, 50]$$

An initial population of four-gene individuals is randomly distributed over the entire design domain. The four genes correspond to the variables x_1, x_2, x_3, and x_4. The individuals are allowed to evolve for 1,000 generations with GaNOP keeping track of the best individual throughout the evolution process. The output for 5 different runs are shown in Table 5.2.

Table 5.2: Rosen-Suzuki output for 5 runs

Run #	Best Optimum Found (f)	No. of generations to obtain optimum	No. of function evaluations[a] required
1	-44	223	11,250
2	**-44**	**334**	**16,800**
3	-44	326	16,400
4	-44	379	19,050
5	-44	334	16,800

a. No. of function evaluations = Population Size + Replacement Ratio*Population Size*Number of Generations

GaNOP found the global optimum in all runs. A typical optimum found is

$\vec{x}^* = (-0.0005463221,\ 1.000618,\ 2.000213,\ -0.9996195)$

and $f(\vec{x}^*) = -44$

On average, GaNOP finds the global optimum in 334 generations with 16,800 function evaluations. The history of the optimization process for Run #2 is presented in

Table 5.3. A convergence plot of GaNOP for Run #2 is shown in Figure 5.1. The convergence plot shows that GaNOP finds the neighborhood of the optimum in less than 100 generations.

Table 5.3: History of optimization process for Rosen-Suzuki function (Run #2)

Generation #	x_1	x_2	x_3	x_4	F(x)
0	17.41715	-8.449738	7.078285	-0.3118286	-8726.227
100	0.00170557	1.011245	1.995385	-1.003309	43.99863
200	0.002754511	1.002352	2.001289	-0.9979787	43.99991
500	-0.0002485895	0.9997745	2.000242	-0.9997551	44
1000	-0.0002485895	0.9997745	2.000242	-0.9997551	44
CPU Run Time (s)					9
Number of Crossovers					50,000
Number of Mutations					20,025

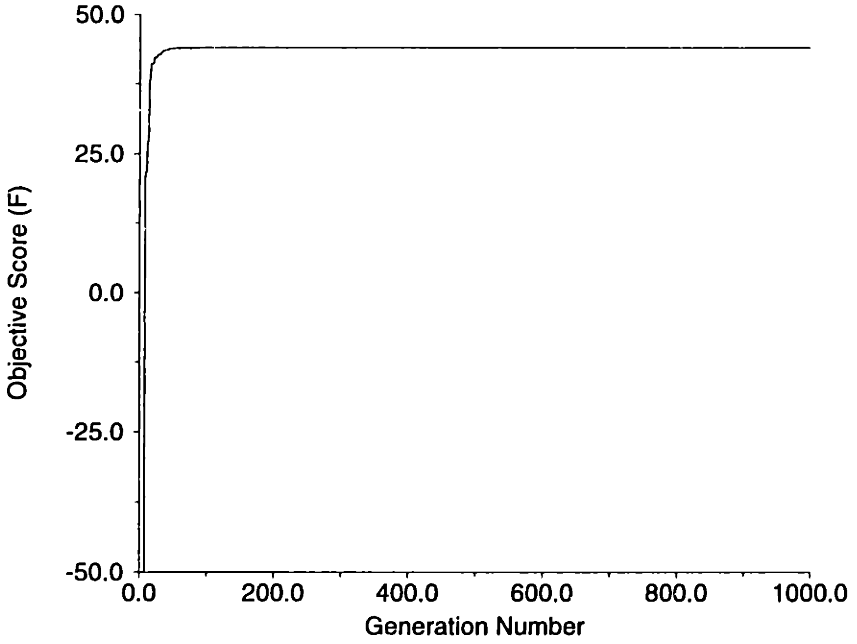


Figure 5.1: Convergence plot for Rosen-Suzuki function (Exploded View)

The running percentage of time that the different parts of the quadratic crossover are implemented for Run #2 is shown in Figure 5.2. The trend in Figure 5.2 is as follows: for the first 13 generations, the percentage of time that quadratic interpolation occurs increases from 56.5% to 57.5%. During this same period, the objective score increased from -8726.227 to 29.29354. For the next 57 generations, the percentage of time that quadratic interpolation occurs decreases from 57.5% to 50.2%. During the 57 generations, the best objective score increased from 29.29354 to 43.99001, where the global optimum is F* = 44 -- note that the objective score is determined by F(x) and not f(x). So in the first few generations when GaNOP is searching for a feasible region, the percentage of time that quadratic interpolation occurs increases. In this same period, the percentage of time that heuristic extrapolation occurs decreases. In the next phase when GaNOP starts to locate the optimum, the percentage of time that quadratic interpolation occurs starts to decrease, and the percentage of time that heuristic extrapolation occurs starts to increase. From generation 57 to generation 680, the percentage of time that quadratic interpolation and heuristic extrapolation occur remains fairly constant. Beyond generation 680, the percentage of time that quadratic interpolation occurs starts to decrease again, while the percentage of time that heuristic extrapolation occurs starts to increase. In summary, quadratic interpolation dominates when GaNOP is searching for the feasible region, and heuristic extrapolation dominates when GaNOP is fine tuning the optimum. Note that random assignments do not occur during quadratic crossover for the entire 1,000 generations.

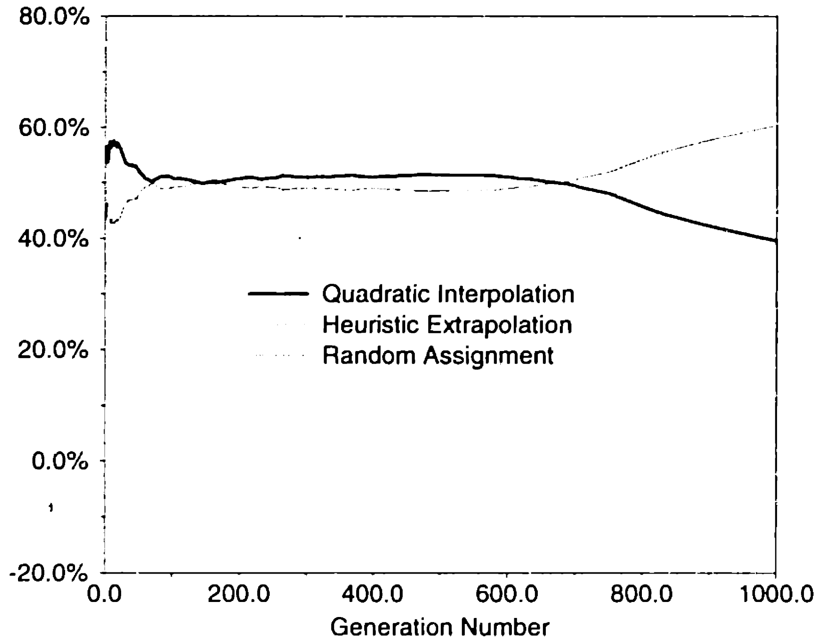


Figure 5.2: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for Rosen-Suzuki function

5.2.2 Parabolic Multiminima Function

The generic definition for the parabolic multiminima function, q_n, is stated as follows:

Let D_f be the domain of definition of the function q_n in n-space

$$D_f \equiv \left[x \in \Re^n : -a_1 \le x_1 \le a_1, \dots, -a_n \le x_n \le a_n ; \quad a \in \Re^n_+ \right]$$

D_f is a rectangular subdomain of $\Re^n$, centered around the origin whose width along each coordinate direction is determined by the corresponding component of the vector a.

Let D_m be the family of open, disjoint, rectangular subdomains of $\Re^n$ contained in D_f and defined as

$$d_{k_1, \dots, k_n} = \{ x \in D_f : k_i s_i - t_i < k_i s_i + t_i, \dots ,$$
$$k_n s_n - t_n < x_n < k_n s_n + t_n ; \ k_1, \dots, k_n \in I ;$$
$$t, s \in \Re^n_+ \quad ; t_i < \frac{s_i}{2}, i = 1, \dots, n \} ,$$

$$D_m \equiv \bigcup_{k_1, \dots, k_n \in I} d_{k_1, \dots, k_n} - d_{0, \dots, 0}$$

D_m is the open subset of D_f where the function q_n presents local minima. The vector **s** controls the grid steps along each axis, the grid points being the centers of these subdomains, while the vector **t** controls the size of these subdomains. The condition $t_i < s_i/2$ ensures that the subdomains are disjoint.

Let D_r be the closed subdomain of D_f complementary to D_r: $D_f - D_m$. The test function $q_n(x)$ of n real variables is defined as

$$q_n(x) : \quad D_f \rightarrow \Re_+$$

$$q_n(x) \equiv \sum_{i=1}^{n} d_i x_i^2 , \qquad x \in D_r , \qquad d \in \Re^n_+ \quad ,$$

$$q_n(x) \equiv c_r \sum_{i=1}^{n} d_i z_i^2 , \qquad x \in d_{k_1, \dots, k_n} , \qquad (k_1, \dots, k_n) \ne 0$$

where

$$z_i = \begin{cases} k_i s_i + t_i & if \quad k_i < 0 \\ 0 & if \quad k_i = 0 \\ k_i s_i + t_i & if \quad k_i > 0 \end{cases}$$

The components of the vector **d** determine the steepness of the paraboloid along the axes, while the coefficient c_r controls the depth of local minima relative to the function values along the boundaries of the regions $d_{k_1, \ldots, k_n}$. It is worth noting that the region $d_{0, \ldots, 0}$ belongs to the subdomain D_r, where q_n is not constant around the origin, which is the unique global minimum of q_n [Corana et al., 1987].

Corana et al. [1987] used the parameter settings listed in Table 5.4 to evaluate the performance of optimization algorithms:

Table 5.4: Control parameters for q_n

Parameters	Value
$a_{1,\ldots,n}$	10^4
$s_{1,\ldots,n}$	0.2 (0.1 for n = 10)
$t_{1,\ldots,n}$	0.05 (0.04 for n = 10)
c_r	0.15
$d_{1,\ldots,n}$	(1, 1000, 10, 100, 1, 10, 100, 1000, 1, 10)

The number of grid points associated with each dimension is $2a_i/s_i$. The total local minima of the test function $q_n(x)$ in its domain is 10^{5n}-1 [Corana et al., 1987].

A section of $q_n(x)$ along an axis *i* is shown in Figure 5.3a. A three-dimensional plot of q_2 for $d_1 = d_2 = 1$ is shown in Figure 5.3b. A contour plot of q_2 is also shown in Figure 5.3c. Figures 5.3a through 5.3c are good visual tools for observing the strongly discontinuous nature of the q functions and how easy it is for any unimodal optimization technique to get trapped in a local minimum.

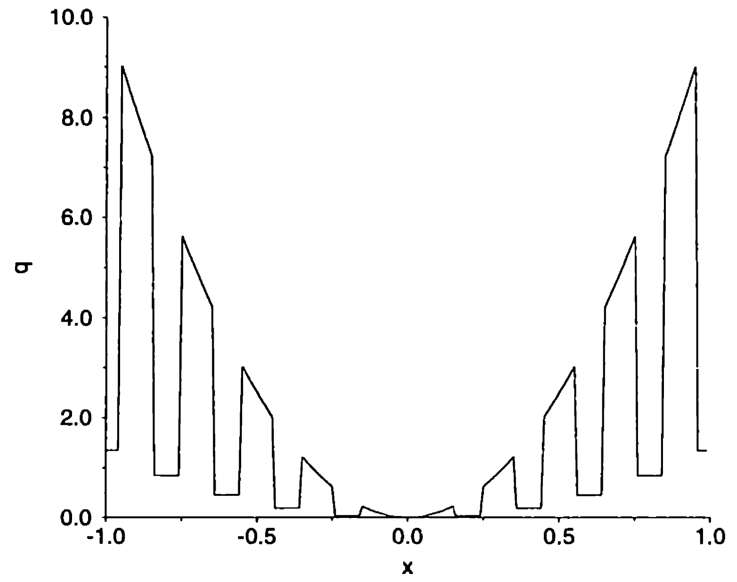


Figure 5.3a: A section of $q_n(x)$ along an axis i where $d_i = 10$

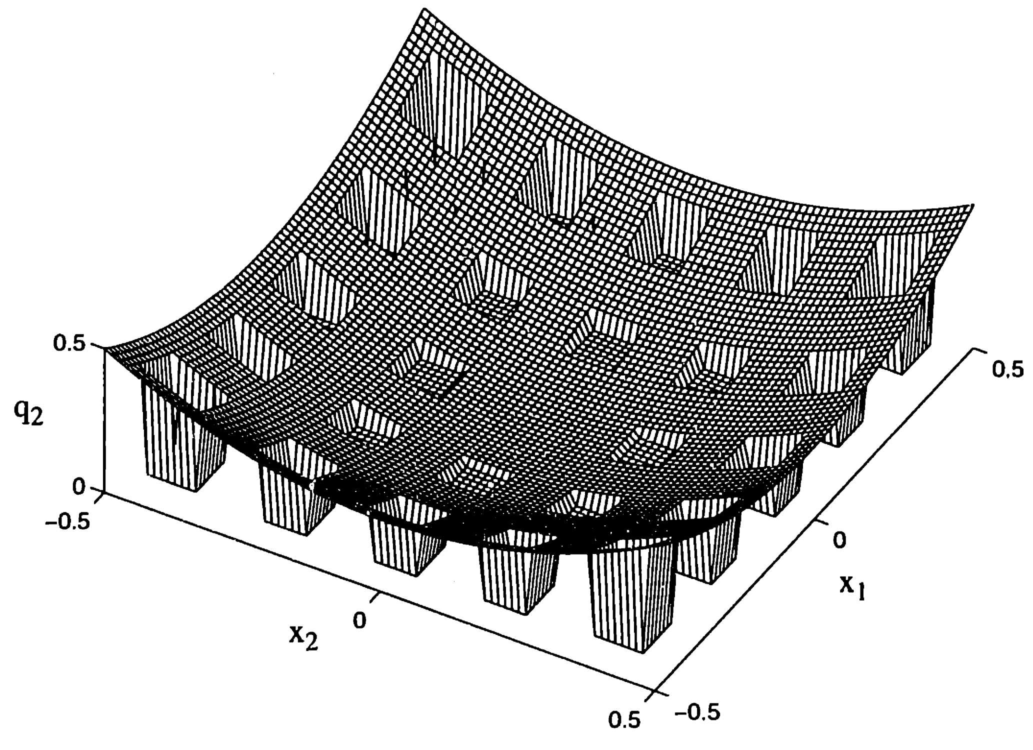


Figure 5.3b: Three dimensional plot of $q_2(x)$ for $d_1 = d_2 = 1$

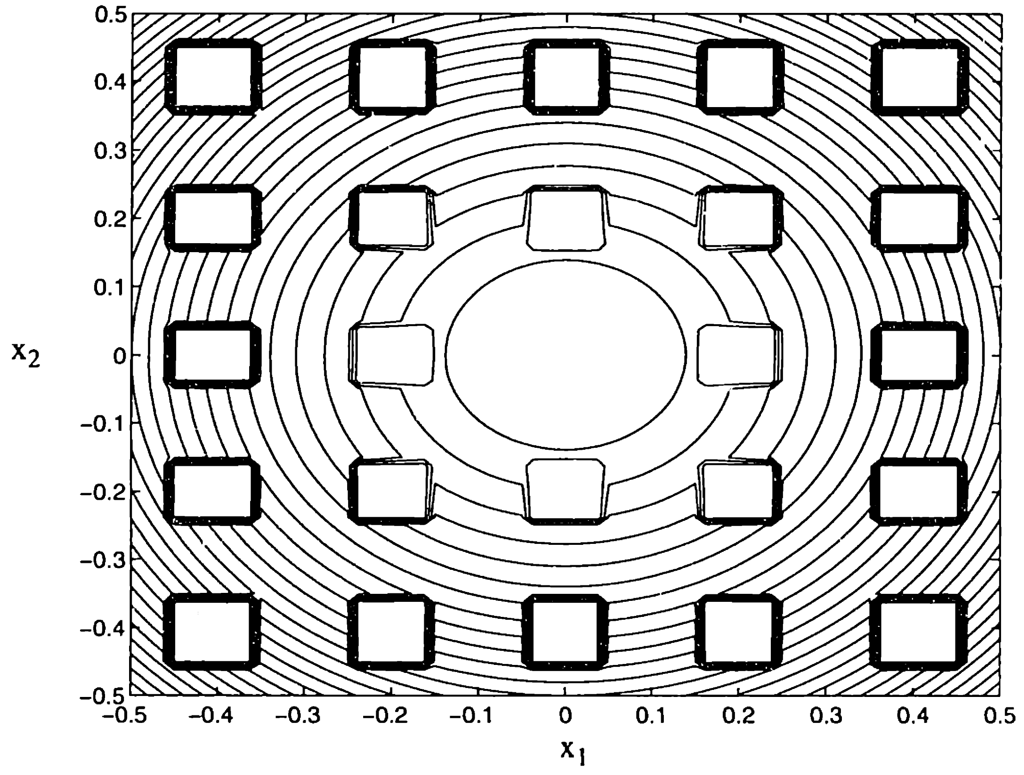


Figure 5.3c: Contour plot of $q_2(x)$ for $d_1 = d_2 = 1$

<u>Parabolic Multiminima Function in Two Dimensions</u>: The minimization problem is stated as follows:

$$\text{Minimize } f(x) = q_2(x)$$

$$\text{where } x \in D_f$$

The global optimum is $f_2(\grave{x}^*) = 0$ at $\grave{x}^* = (0,0)$.

GaNOP
The minimization problem is restated as follows:

$$\text{Maximize } F(\grave{x}) = -f(\grave{x})$$

$$\text{where } x \in D_f$$

An initial population of two-gene individuals is randomly distributed over the entire design domain. Each individual is a point in the problem space. The first gene in the individual corresponds to x_1 and the second gene corresponds to x_2. The individuals are allowed to evolve for 1,000 generations. The output for 5 different runs are shown in Table 5.5.

Table 5.5: 2-D parabolic multiminima output for 5 runs

Run #	Best Optimum Found (f)	Generations to Obtain Optimum	No. of Function Evaluations[a] Required
1	1.130439e-24	815	40,850
2	2.93360e-23	872	43,700
3	**3.127169e-25**	**346**	**17,400**
4	2.93360e-23	872	43,700
5	1.441338e-21	833	41,750

a. No. of function evaluations = Population Size + Replacement Ratio*Population Size*Number of Generations

GaNOP obtained the global optimum in all runs. A typical optimum found is

$$\vec{x}^* = (5.404767e\text{-}12,\ -1.115774e\text{-}14)$$
$$\text{and}\quad f(\vec{x}^*) = 2.9336e\text{-}23$$

The history of the optimization process for Run #3 is shown in Table 5.6. The convergence plot of GaNOP for Run #3 is shown in Figure 5.4. The convergence plot shows that GaNOP found the neighborhood of the optimum in less than 50 generations. In Run #3, GaNOP found the optimum in 346 generations with 17,400 function evaluations. This seems like a lot of function evaluations, but the number of function evaluations depends on the desired accuracy of the result. For instance, GaNOP obtained the optimum f = 1.9e-10 at generation 51 with 2,650 function evaluations.

Corana et al. [1987] reported a global minimum of 4.2e-10 using simulated annealing with 656,000 function evaluations. This is about 38 times the number of function evaluations required by GaNOP to obtain the optimum in Run #3. The Nelder-Mead simplex algorithm, a unimodal optimization technique, found the global optimum with 150,000 function evaluations after 240 restarts. Clearly, GaNOP performs better than simulated annealing and Nelder-Mead simplex algorithm on this problem.

Table 5.6: History of optimization process for parabolic multi-minima (Run #3)

Generation #	x_1	x_2	F
0	-1203.593	17.7793	-264539.2
100	-1.03491e-09	-5.272864e-11	-3.851348e-18
200	-7.975654e-11	-6.066459e-13	-6.729125e-21
500	1.645476e-13	-1.690092e-14	-3.127169e-25
1000	1.645476e-13	-1.690092e-14	-3.127169e-25
CPU Run Time (s)			8
No. of Crossovers			50,000
No. of Mutations			9,915

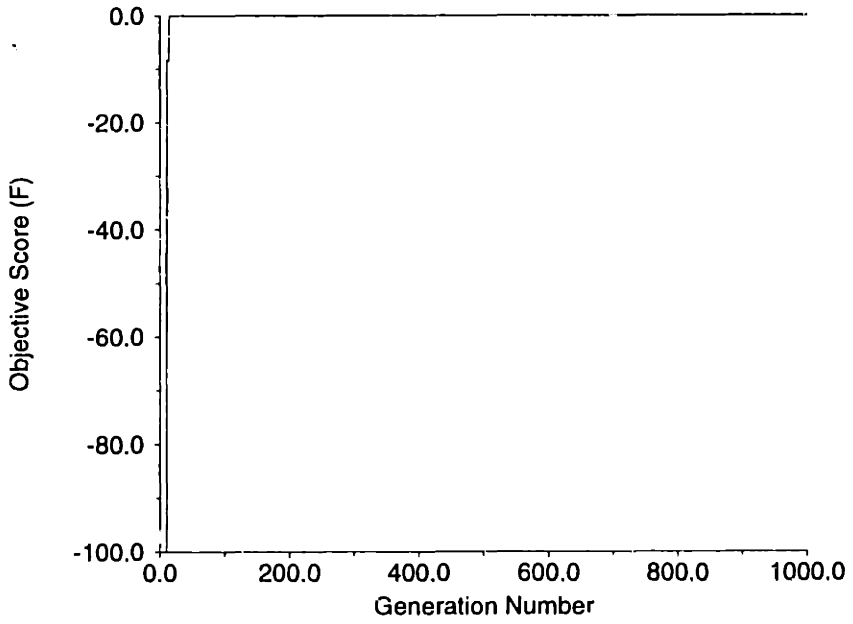


Figure 5.4: Convergence plot for 2-D parabolic multiminima function

The percentage of time that the different parts of the quadratic crossover are implemented for Run #3 is shown in Figure 5.5. For the first 15 generations, the percentage of time that quadratic interpolation occurs decreases from 64% to 61.4%. During this same period, the best objective score increased from -264539.2 to -0.018375. From generation 15 to generation 266, the percentage of time that quadratic interpolation occurs increases from 61.4% to 81.5%. During this same period, the best objective score (F) increased from -0.018375 to 9.440862e-22. Beyond generation 266, the percentage of time that quadratic interpolation occurs remains fairly constant, and so does the best objective score. For the 2-D parabolic multiminima function, quadratic interpolation dominates throughout the evolution process. This trend is different from the one observed for the Rosen-Suzuki function, where quadratic interpolation dominates during the search for the feasible region and heuristic extrapolation dominates when GaNOP is fine tuning the optimum. Note that heuristic extrapolation decreases when quadratic interpolation increases and increases when quadratic interpolation decreases. Again, there are no random assignments during quadratic crossover for the entire 1,000 generations.

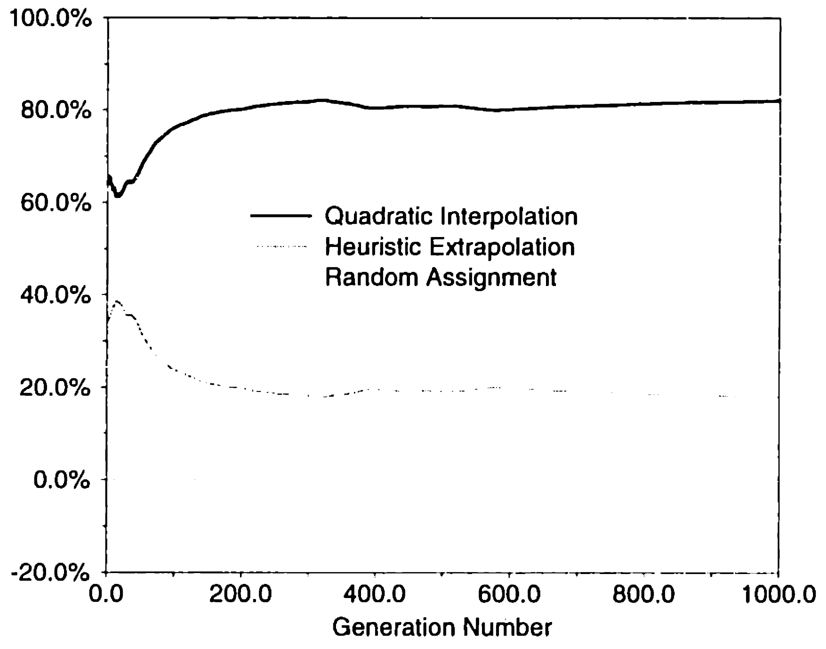


Figure 5.5: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for 2-D parabolic multiminima function

<u>Parabolic Multiminima Function in Four Dimensions</u>: The minimization problem is stated as follows:

Minimize $f(\vec{x}) = q_4(\vec{x})$

where $\vec{x} \in D_f$

The global optimum is $f_4(\vec{x}^*) = 0$ at $\vec{x}^* = (0,0,0,0)$.

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(\vec{x}) = -f(\vec{x})$$

$$\text{where } \vec{x} \in D_f$$

An initial population of four-gene individuals is randomly distributed over the entire design domain. The four genes correspond to the variables x_1, x_2, x_3, and x_4. The individuals are allowed to evolve for 2,000 generations. The output for 5 different runs are shown in Table 5.7.

Table 5.7: 4-D parabolic multiminima output for 5 runs[a]

Run #	Best Optimum Found (f)	No. of generations to obtain optimum	No. of function evaluations[b] required
1	6.229116e-13	1,940	48,600
2	**4.739943e-13**	**1,693**	**42,425**
3	4.739943e-13	1,693	42,425
4	5.498258e-13	1,937	48,525
5	4.739943e-13	1,693	42,425

a. Probability of Crossover = 0.95; Replacement Ratio = 0.25.
b. No. of function evaluations = Population Size + Replacement Ratio*Population Size*Number of Generations

GaNOP obtained the global optimum in all the five runs. A typical optimum found is

$$\vec{x}^* = (-2.240197e\text{-}08,\ 1.378791e\text{-}08,\ 1.434433e\text{-}07,\ 2.78615e\text{-}08)$$

$$\text{and } f(\vec{x}^*) = 4.739943e\text{-}13$$

The history of the optimization process for Run #2 is presented in Table 5.8. The convergence plot of GaNOP for Run #2 is shown in Figure 5.6. The typical optimum is obtained in 1,693 generations with 42,425 function evaluations. The best optimum obtained using simulated annealing with 1,464,000 function evaluations is 8.7e-8 [Corana et al., 1987]. The Nelder-Mead simplex algorithm failed on this problem.

Table 5.8: History of optimization process for 4-D parabolic multiminima (Run #2)

Generation #	x_1	x_2	x_3	x_4	F
0	3254.374	795.0283	-2808.437	5398.156	-5.453112e8
100	-0.228202	-9.832004e-3	-0.04551505	9.382609e-4	-0.003375
200	-0.228202	-9.832004e-3	-0.04551505	9.382609e-4	-0.003375
700	4.330514e-6	6.77095e-8	-3.003316e-7	-3.09445e-8	-2.433568e-11
1500	-1.415761e-6	-1.709767e-8	-5.360409e-8	1.956329e-7	-6.152664e-12
2000	-2.240197e-8	1.378791e-8	1.434433e-7	2.78615e-8	-4.739943e-13
CPU Run Time (s)					12
No. of Crossovers					47,533
No. of Mutations					20,068

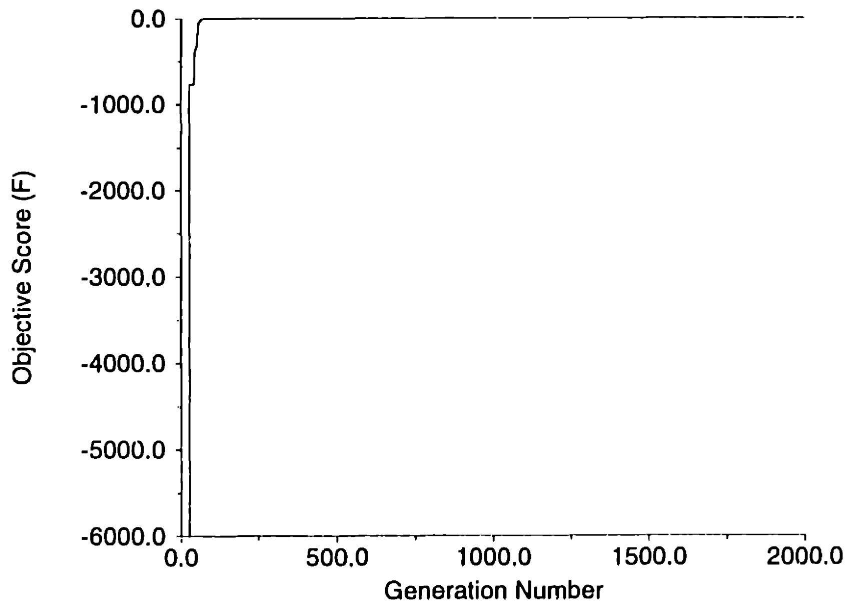


Figure 5.6: Convergence plot for 4-D parabolic multiminima function

The percentage of time that the different parts of the quadratic crossover are implemented for Run #2 is shown in Figure 5.7. The percentage of time that quadratic interpolation occurs steadily increases from 54% at generation 1 to 59.3% at generation 39. During this same period, the best objective score (F) increased from -5.453112e+08 to -767.2767. From generation 39 to generation 74, the percentage of time that quadratic interpolation occurs decreases from 59.3% to 56.8%. During this same period, the best objective score (F) increased from -767.2767 to -0.5246251. From generation 74 to generation 2,000, the percentage of time that quadratic interpolation occurs gradually rises from 56.8% to 71.6%. This trend is somewhat similar to the one observed for the 2-D version of the parabolic multiminima function. In the 2-D version, the initial increase in the percentage of the quadratic interpolation that occurs is not present. The heuristic extrapolation decreases when quadratic interpolation increases and increases when quadratic interpolation decreases. There are no random assignments during quadratic crossover for the entire 2,000 generations.

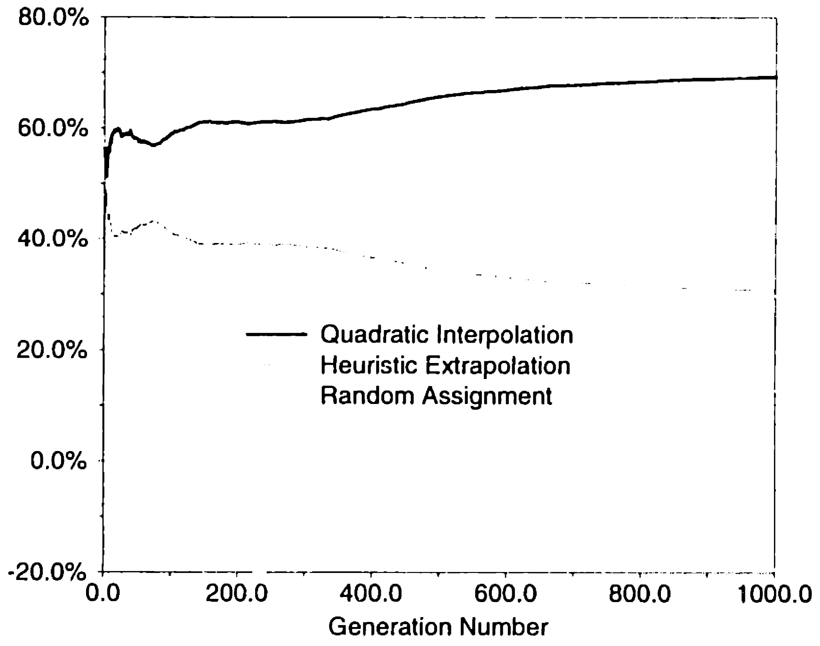


Figure 5.7: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for 4-D parabolic multiminima function.

Parabolic Multiminima Function in Ten Dimensions: The minimization problem is stated as follows:

$$\text{Minimize } f(\vec{x}) = q_{10}(\vec{x})$$

$$\text{where } \vec{x} \in D_f$$

The global optimum is $f_{10}(\vec{x}^*) = 0$ at $\vec{x}^* = (0,0,0,0,0,0,0,0,0,0)$.

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(\vec{x}) = -f(\vec{x})$$

$$\text{where } \vec{x} \in D_f$$

An initial population of ten-gene individuals is randomly distributed over the entire design domain. Each individual is a point in the problem space. The individuals are allowed to evolve for 2,500 generations.

GaNOP consistently remained trapped in the local minimum (i.e. one of the holes) adjacent to the global minimum in all the runs. A typical optimum found is

$$\vec{x} = (-0.022465421,\ -0.013882368,\ 0.018357892,\ -0.0097310301,\ -0.13446185,\ -0.00068439927,\ 0.018650901,\ 0.016743518,\ -0.0049647833,\ 0.024669537)$$

and $f(\vec{x}) = 0.00054$

Really, for the parabolic multiminima problem, GaNOP remains trapped in a local sub-optimum for dimensions greater than 4. Interestingly, the simulated annealing technique used in Corana et al. [1987] also consistently remained trapped in the local minimum ($f = 0.00054$) adjacent to the global minimum for the 10-D parabolic multiminima function.

The percentage of time that the different parts of the quadratic crossover are implemented for Run #5 is shown in Figure 5.8. The trend in Figure 5.8 is closer to the trend for the 4-D version of the parabolic multiminima function. Note that there are random assignments in this case.

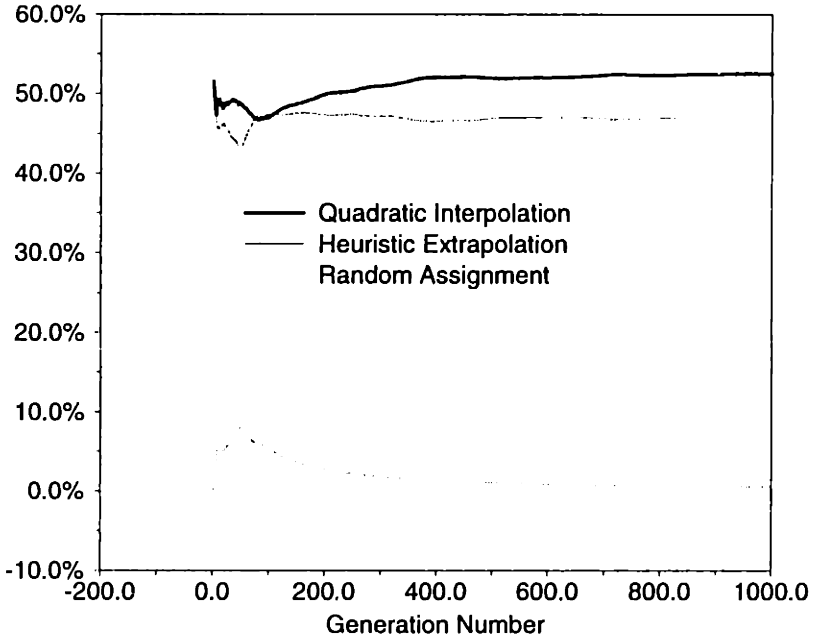


Figure 5.8: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for 10-D parabolic multiminima function

5.3 Optimization of Generalized Polynomial Functions

Three generalized polynomial functions are considered. The first objective function is a second-degree polynomial in two variables. The second objective function is a fourth-degree polynomial in four variables. The third objective function can be extended to any number of dimensions. The following is a description of the test functions and the optimization process with GaNOP.

5.3.1 Soland's Problem

This problem is taken from [Floudas and Pardalos, 1987]. It features a convex objective function with a nonconvex inequality constraint. The bounds on the variables introduce four additional linear inequality constraints.

Minimize $f(x, y) = -12x - 7y + y^2$

subject to

$$h = -2x^4 + 2 - y = 0$$
$$0 \le x \le 2 \text{ and } 0 \le y \le 3$$

A plot of the problem domain is shown in Figure 5.9. The best known solution is x* = 0.71751, y* = 1.470, and f* = -16.73889.

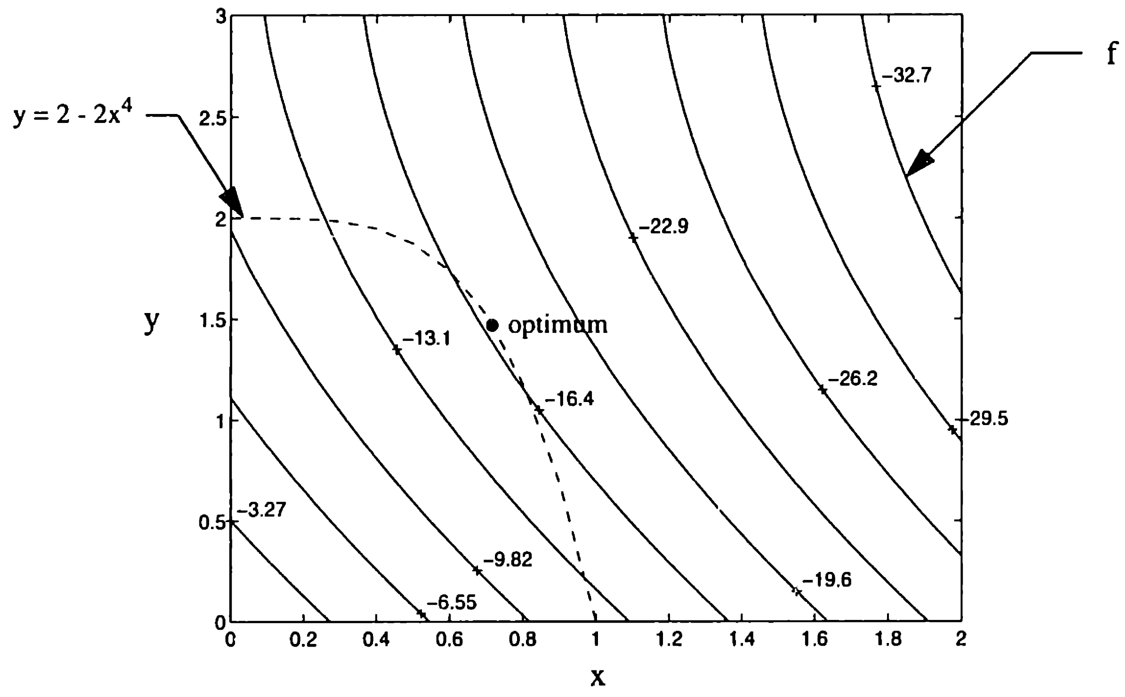


Figure 5.9: Graphical solution of Soland's problem

GaNOP

The first step in transforming the minimization problem to a GaNOP problem is to eliminate the equality constraint. From equality constraint h, $y = 2 - 2x^4$. The independent variable is x and the dependent variable is y. Two inequality constraints result from the bound constraints on y: $g_1 = 2x^4 - 2 \le 0$ and $g_2 = 2 - 2x^4 - 3 \le 0$. The minimization problem can now be rewritten in terms x only:

$$\text{Minimize } f'(x) = -12x - 7\left(2 - 2x^4\right) + \left(2 - 2x^4\right)^2$$

subject to

$$g_1 = 2x^4 - 2 \le 0$$

$$g_2 = 2 - 2x^4 - 3 \le 0$$

$$0 \le x \le 2$$

The minimization problem is restated as follows:

$$\text{Maximize } F(x) = -f'(x) - \sum_{i=1}^{2} \Phi_i$$

where

$$\Phi_i = \begin{cases} 5 \cdot g_i + 2 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and } 0 \le x \le 2$$

An initial population of one-gene individuals is randomly distributed over the entire design domain. The individuals are allowed to evolve for 100 generations with GaNOP keeping track of the best individual throughout the evolution process. GaNOP obtained the global optimum in all runs. A typical global optimum found is

$$x^* = 0.71753848, \quad y^* = 1.4698353, \quad \text{and } f^* = -16.738895$$

The global optimum was obtained in 4 generations with 300 function evaluations. The run-time for 100 generations is 1s. The percentage of time that the different parts of the quadratic crossover are implemented for Run #1 is shown in Figure 5.10. At the beginning of the evolution process, quadratic interpolation dominates. At generation 25^+, heuristic extrapolation starts to dominate. The general trend is that quadratic interpolation dominates when the individuals in the population are fairly dissimilar, and heuristic extrapolation dominates when the individuals in the population are fairly similar.

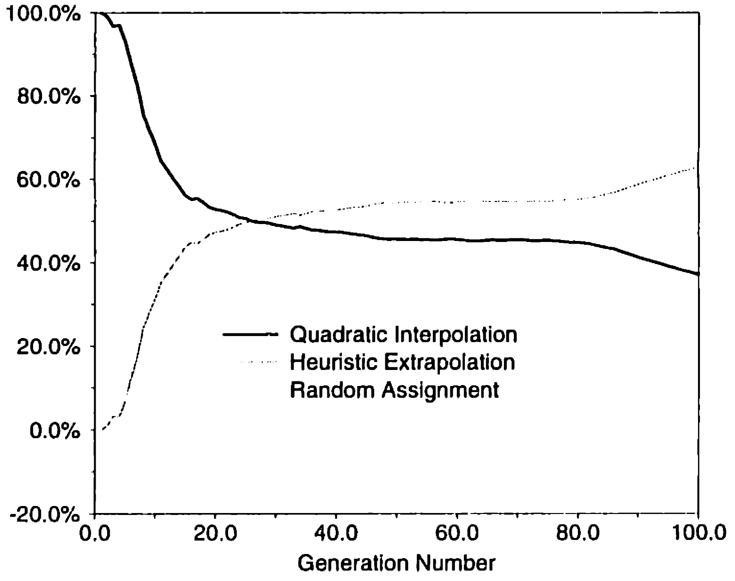


Figure 5.10: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for Soland's problem

5.3.2 Colville's Function

This problem is from [Michalewicz, 1994]:

$$\text{Minimize } f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 + 90\left(x_4 - x_3^2\right)^2 + (1 - x_3)^2$$
$$+ 10.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8\,(x_2 - 1)\,(x_4 - 1)$$

$$\text{where } -10.0 \le x_1, x_2, x_3, x_4 \le 10.0$$

The bounds on the variables introduce eight inequality constraints. The global solution is x* = (1,1,1,1) and f* = 0.

GaNOP

The minimization problem is restated as follows

$$\text{Maximize } F(x) = -f(x)$$

$$\text{where } -10.0 \le x_1, x_2, x_3, x_4 \le 10.0$$

An initial population of four-gene individuals is randomly distributed over the entire design domain. The individuals are allowed to evolve for 500 generations. The output for 5 different runs is presented in Table 5.9.

Table 5.9: Colville's output for 5 runs

Run #	Best Optimum Found (f)	Number of Generations to Obtain Optimum	Number of Function Evaluations[a] to Obtain Optimum
1	0	147	7450
2	**0**	**114**	**5800**
3	0	132	6700
4	0	114	5800
5	0	145	7350

a. No. of function evaluations = Population Size + Replacement Ratio*Population Size*Number of Generations

GaNOP obtained the exact global optimum in all runs. The history of the optimization process for Run #2 is presented in Table 5.10.

Table 5.10: History of optimization process for Colville's function (Run #2)

Generation #	x_1	x_2	x_3	x_4	F
0	1.5140095	1.5319357	1.4181671	4.4755135	-766.26251
50	0.92561758	0.86553949	1.0388691	1.0765578	-0.053369757
100	0.9999997	0.9999997	1	0.99999988	-1.1993606e-11
200	1	1	1	1	0
500	1	1	1	1	0
Run Time (s)					5
No. of Crossovers					25,000
No. of Mutations[a]					9

a. Probability of Mutation = 0.0001

In Runs #2 and #5, GaNOP obtained the global optimum in 114 generations with 5,800 function evaluations. This same problem was solved by Michalewicz's GENOCOP system (a genetic-algorithm based optimization system), and the best optimum obtained in 10,000 generations was x* = (1.000581, 1.001166, 0.999441, 0.998879) with f* = 0.0000012 [Michalewicz, 1994, p. 157]. GaNOP performs better than GENOCOP on this problem; although, it is not clear that a different control parameter setting, other than the setting used in the reported run with GENOCOP, may have led GENOCOP to obtain the optimum solution in fewer generations.

The percentage of time that the different parts of the quadratic crossover are implemented for Run #2 is shown in Figure 5.11. Again, quadratic interpolation dominates in the beginning of the evolution process and heuristic extrapolation dominates from generation 125^+. The observation that quadratic interpolation dominates when the individuals are dissimilar and that heuristic extrapolation dominates when the individuals are similar still holds for Colville's function.

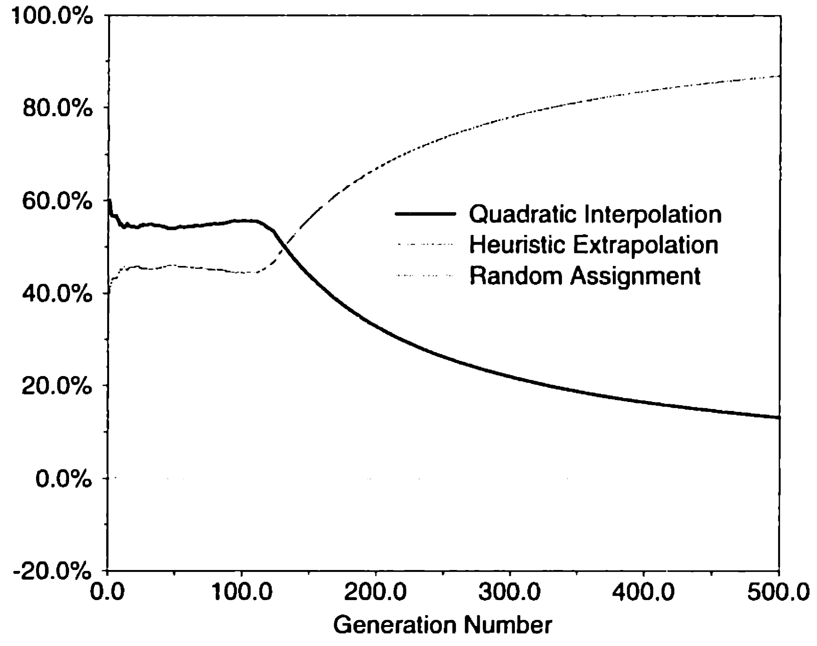


Figure 5.11: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for Colville's function

5.3.3 Rosenbrock Function

The Rosenbrock function [Rosenbrock, 1960] in *n* dimensions has the general form:

$$f_n(\vec{x}) = \sum_{k=1}^{n-1} \left[100\left(x_{k+1} - x_k^2\right)^2 + (1 - x_k)^2\right]$$

The Rosenbrock function is unimodal.

<u>Rosenbrock Function in Two Dimensions</u>: The minimization problem is stated as

$$\text{Minimize } f_2(\vec{x}) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

$$\text{where } \vec{x} \in [-2000, 2000]$$

The bounds on the variables x_1 and x_2 introduce four inequality constraints. A plot of $f_2(\vec{x})$ is shown in Figure 5.12a. A section of $f_2(\vec{x})$ along an axis $x_2 = 0$ is shown in Figure 5.12b. The global optimum is $f_2(\vec{x}^*) = 0$ at $\vec{x}^* = (1,1)$.

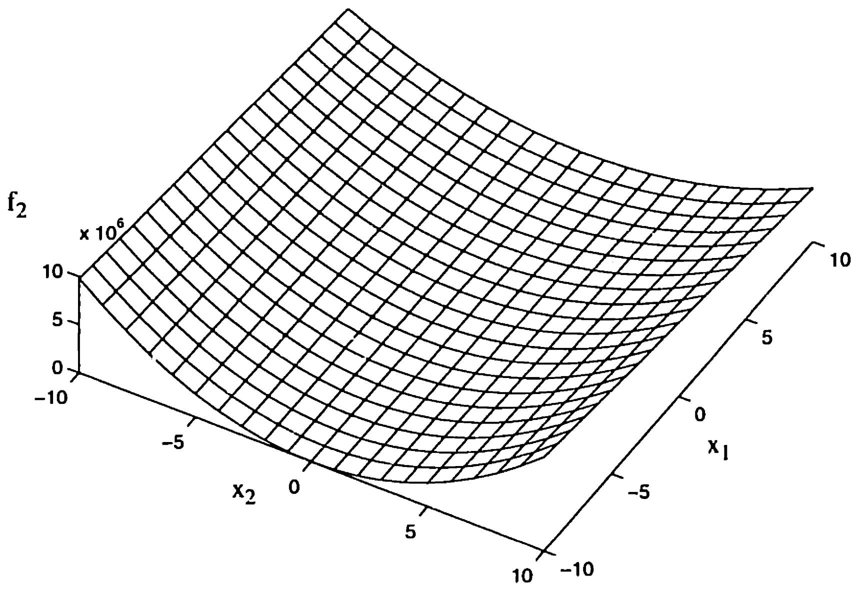


Figure 5.12a: Two-dimensional plot of the Rosenbrock function

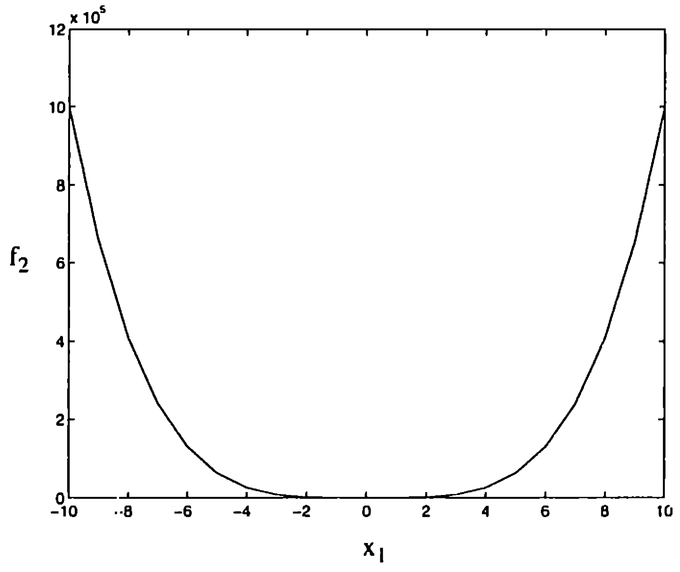


Figure 5.12b: A section of f_2 along axis $x_2 = 0$

The Rosenbrock function is a standard nonlinear programming test problem; thus it has been tackled by several optimization techniques. The techniques considered by Corana et al. [1987] are simulated annealing and Nelder-Mead simplex algorithms. Simulated annealing is a stochastic optimization technique and Nelder-Mead simplex algorithm is a deterministic optimization technique. The performance of GaNOP on the two-dimensional Rosenbrock function will be compared to the performance of simulated annealing and Nelder-Mead simplex algorithms on the same problem.

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(\vec{x}) = -f_2(\vec{x})$$

$$\text{where } \vec{x} \in [-2000, 2000]$$

An initial population of two-gene individuals is randomly distributed over the entire design domain. The first gene of the individual corresponds to x_1 and the second gene corresponds to x_2. The individuals are allowed to evolve for 500 generations. The output for 5 different runs are shown in Table 5.11.

Table 5.11: 2-D Rosenbrock output for 5 runs

Run #	f_2	Generations to Obtain Optimum	No. of Function Evaluations[a] Required
1	0	109	5550
2	**0**	**73**	**3750**
3	0	73	3750
4	0	62	3200
5	0	73	3750

a. No. of function evaluations = Population Size + Replacement Ratio*Population Size*Number of Generations

In all runs, the exact global optimum was obtained. The history of the optimization process for Run #2 is presented in Table 5.12. On average, the number of evaluations required to obtain the exact global optimum is about 3,750. Corana et al. [1987] reported a global minimum of 1.8e-10 using simulated annealing with 500,001 function evaluations. This is about 133 times the number of function evaluations required by GaNOP to obtain the exact global optimum. With Nelder-Mead simplex algorithm, a global optimum of 5.4e-11 was obtained. The Nelder-Mead simplex algorithm required 907 function evaluations [Corana et al., 1987]. Obviously, the Nelder-Mead simplex algorithm is more efficient that GaNOP on this problem. This result is not entirely surprising because the simplex technique utilizes some auxiliary information that is not available to GaNOP or simulated annealing, i.e. gradients of the objective.

The running percentage of time that the different parts of the quadratic crossover are implemented is shown in Figure 5.13. Again the trend observed in Figure 5.13 is similar to the trend observed for Colville's function.

Table 5.12: History of Optimization Process for 2-D Rosenbrock function (Run #2)

Generation #	x_1	x_2	F
0	-3.3985596	745.93115	-53931556
50	0.91201866	0.82748741	0.0095816767
100	1	1	0
500	1	1	0
CPU Run Time (s)	4		
No. of Crossovers	25,000		
No. of Mutations[a]	6		

a. Probability of Mutation = 0.0001

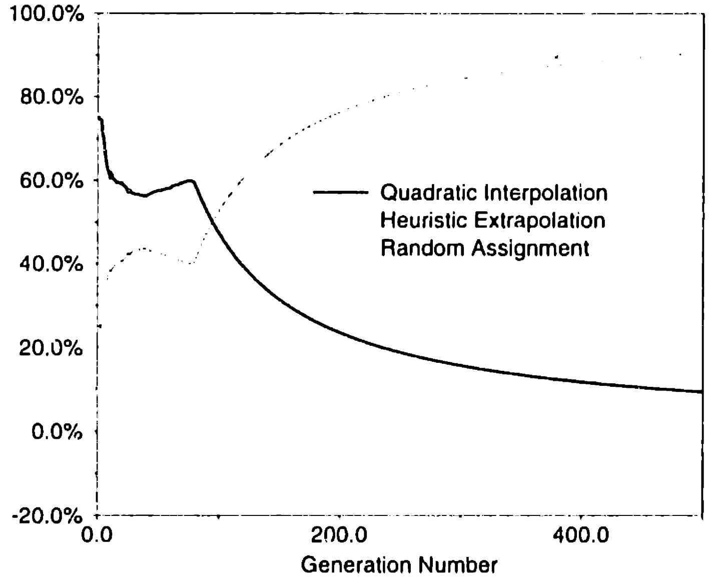


Figure 5.13: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for 2-D Rosenbrock function

<u>Rosenbrock Function in Four Dimensions</u>: The minimization problem is stated as

$$\text{Minimize } f_n(\vec{x}) = \sum_{k=1}^{3}\left[100\left(x_{k+1} - x_k^2\right)^2 + (1 - x_k)^2\right]$$

$$\text{where } \vec{x} \in [-2000, 2000]$$

The bounds on $\vec{x}$ introduce eight inequality constraints. The global minimum is $f_4(\vec{x}^*) = 0$ at $\vec{x}^* = (1,1,1,1)$.

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(\vec{x}) = -f_4(\vec{x})$$

$$\text{where } \vec{x} \in [-2000, 2000]$$

An initial population of four-gene individuals is randomly distributed over the entire design domain. The genes correspond to the variables x_1, x_2, x_3, and x_4. The individuals are allowed to evolve for 500 generations. The output for 5 different runs are presented in Table 5.13.

Table 5.13: 4-D Rosenbrock output for 5 runs

Run #	f	Generations to Obtain Optimum	No. of Function Evaluations[a] Required
1	**0**	**176**	**8,900**
2	0	255	12,850
3	0	176	8,900
4	0	184	9,300
5	0	215	10,850

a. No. of function evaluations = Population Size + Replacement Ratio*Population Size*Number of Generations

In all runs, the exact optimum was obtained. The history of the optimization process for Run #1 is presented in Table 5.14. On the average, the number of function evaluations required to obtain the exact global optimum is about 10,000. Corana et al. [1987] reported a global optimum of 1.8e-7 using simulated annealing with 1,328,001 function evaluations. This is about 133 times the number of function evaluations required by GaNOP to obtain the optimum. Using the Nelder-Mead simplex algorithm with 870 function evaluations a global optimum of 6e-18 was obtained [Corana et al., 1987]. Again, the performance of Nelder-Mead simplex algorithm is better than the performance of GaNOP on this problem.

Table 5.14: History of optimization process for 4-D Rosenbrock function (Run #1)

Generation #	x_1	x_2	x_3	x_4	F
0	302.80176	306.38721	283.63354	895.10278	-2.34384186e12
50	0.38929528	0.16872656	0.076927155	-0.013678871	-2.2187653
100	0.91013587	0.82839209	0.68642557	0.47364545	-0.13646547
200	1	1	1	1	0
500	1	1	1	1	0
CPU Run Time (s)					5
No. of Crossover					25000
No. of Mutations[a]					9

a. Probability of Mutation = 0.0001

The running percentage of time that the different parts of the quadratic crossover are implemented for Run #1 is shown in Figure 5.14. Again, the trend in Figure 5.14 is consistent with the other trends exhibited by Colville's function and the 2-D version of the Rosenbrock function.

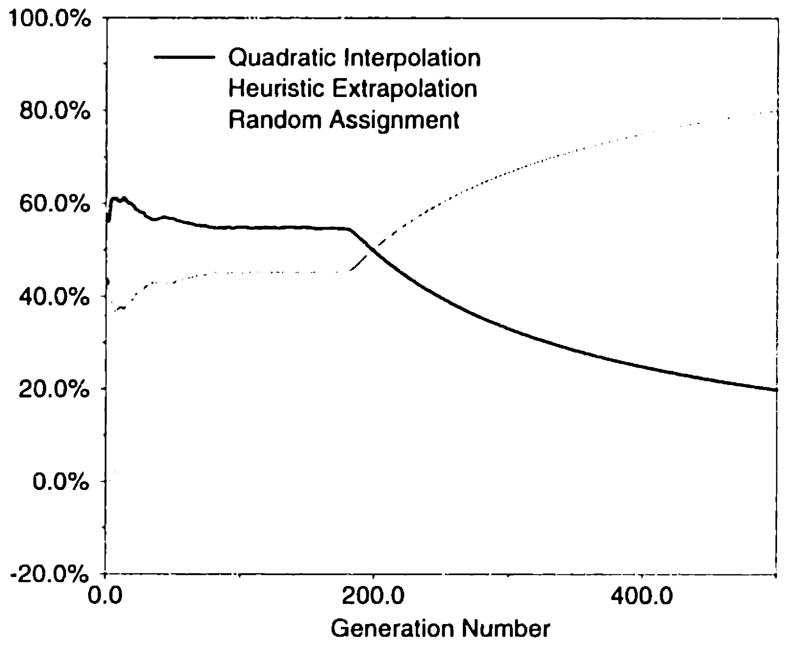


Figure 5.14: Running Percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for 4-D Rosenbrock function

<u>Rosenbrock Function in Ten Dimensions</u>: The minimization problem is stated as

$$\text{Minimize } f_{10}(\vec{x}) = \sum_{k=1}^{9}\left[100\left(x_{k+1} - x_k^2\right)^2 + (1 - x_k)^2\right]$$

$$\text{where } \vec{x} \in [-2000, 2000]$$

The bounds on the vector $\vec{x}$ introduce twenty inequality constraints. The global minimum is $f_{10}(\vec{x}^*) = 0$ with $\vec{x}^* = (1,1,1,1,1,1,1,1,1,1)$.

GaNOP

The minimization problem is restated as

$$\text{Maximize } F(\vec{x}) = -f(\vec{x})$$

$$\text{where } \vec{x} \in [-2000, 2000]$$

An initial population of ten-gene individuals is randomly distributed over the entire design domain. The individuals are allowed to evolve for a number of generations. The output for 5 different runs are shown in Table 5.14.

Table 5.15: 10-D Rosenbrock output for 5 Runs

Run #	Best Optimum Found (f)	Generations to Obtain Optimum	No. of Function Evaluations Required
1	**3.735785e-10**	**2780**	**139,100**
2	1.3627144e-10	3188	159,500
3	2.6421532e-09	2133	106,750
4	1.3597905e-09	2481	124,150
5	6.6015211e-09	3238	162,000

The best optimum obtained is

$f(\hat{x}^*) = 1.3627144e\text{-}10$

at

$\hat{x}^* = (1, 0.99999982, 0.9999997, 0.99999952, 0.99999923,$
$0.99999869, 0.99999768, 0.99999547, 0.99999088, 0.99998176)$

This optimum was obtained with 159,500 function evaluations. Corana et al. did not give results for 10-D Rosenbrock function. Hock and Schittkowski [1987] solved this problem using sequential quadratic programming (NLPQL), a deterministic optimization technique. The optimum obtained was 1.80818e-6 with 176 function evaluations.

The history of the optimization process for Run #1 is shown in Table 5.14. The convergence plot of GaNOP for the 10-D Rosenbrock function is shown in Figure 5.15. The running percentage of time that the different parts of the quadratic crossover are implemented for Run #1 is shown in Figure 5.16. The trend in Figure 5.16 is still similar to the trends observed for the 2-D and 4-D versions of the Rosenbrock function. Note that from about generation 1,000 to generation 2,000, the percentage of time quadratic interpolation and heuristic extrapolation are implemented is equal and fairly constant. During this period, the best objective score moved from -0.05364 to -1.321e-6. However, from generation 0 to 1,000, when quadratic interpolation dominates, the best objective score moved from -4.826e14 to -0.05364. Again the objective scores here are determined by F(x) and not f(x). This is still in line with the observation that quadratic interpolation dominates when the individuals are fairly dissimilar, or when GaNOP is still searching for the neighborhood of the optimum solution.

Table 5.16: History of optimization process for 10-D Rosenbrock function (Run #1.)

Generation #	f
0	-4.8258407e+14
100	101.18977
500	2.3539438
1000	0.053638354
2500	8.5464841e-08
5000	3.735785e-10
CPU Run Time (s)	67
No. of Crossovers	250,000
No. of Mutations[a]	384

a. Probability of Mutation = 0.00015

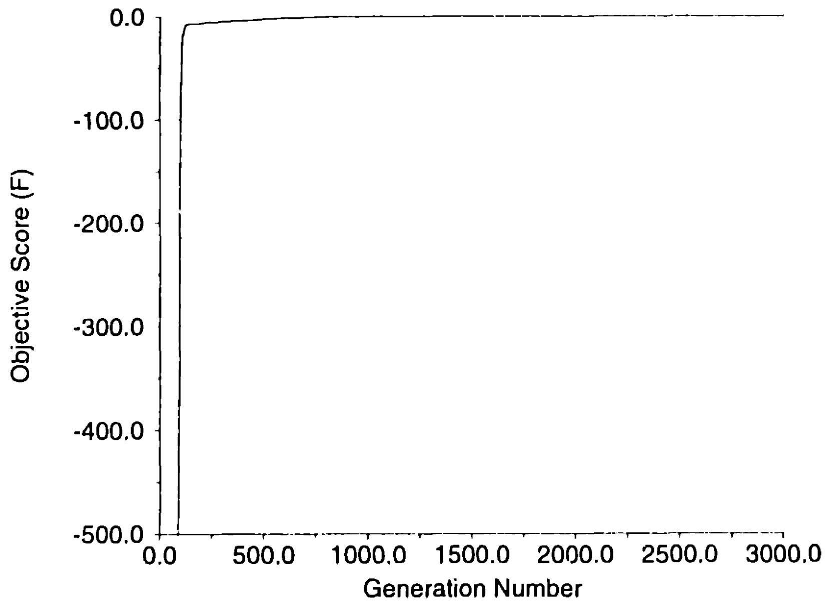


Figure 5.15: Convergence plot for 10-D Rosenbrock function

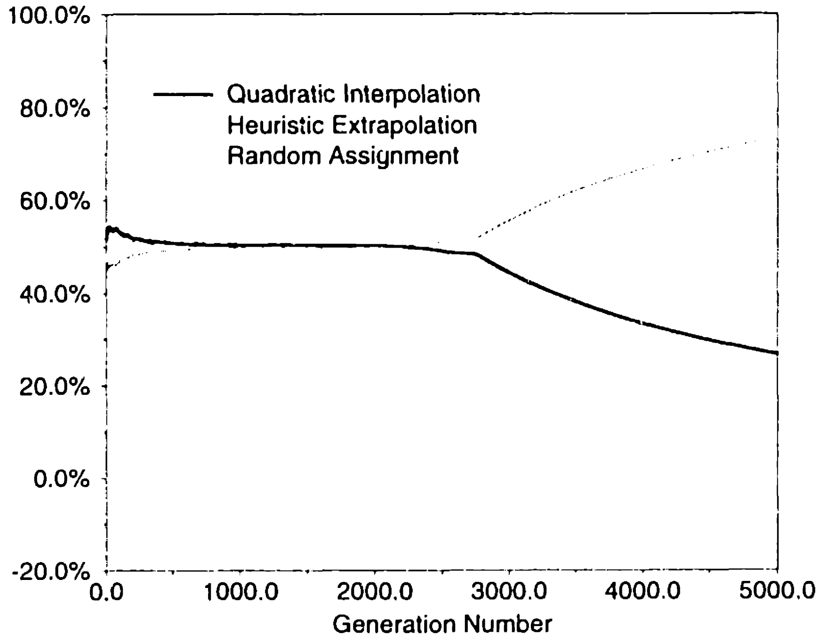


Figure 5.16: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for 10-D Rosenbrock function

5.4 Optimization of Linear Objective Function

This problem is taken from [Floudas and Pardalos, 1987].

Minimize $f(x, y) = -x-y$

subject to

$$g_1 = y - 2x^4 + 8x^3 - 8x^2 - 2 \le 0$$

$$g_2 = y - 4x^4 + 32x^3 - 88x^2 + 96x - 36 \le 0$$

$$0 \le x \le 3 \text{ and } 0 \le y \le 4$$

The problem has two nonlinear constraints; the objective function is linear with its known global optimum at (x*,y*) = (2.3295, 3.1783), and f* = -5.5079. The feasible region is almost disconnected [see Figure 5.17].

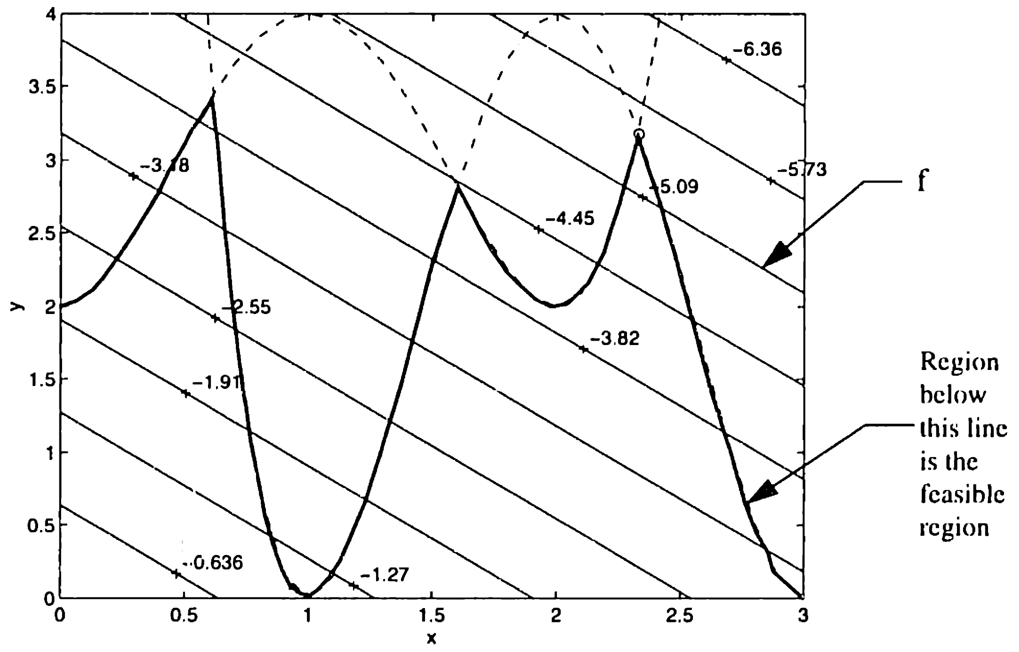


Figure 5.17: Graphical solution for the linear objective function

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(x, y) = x + y - \Phi_1 - \Phi_2$$

subject to

$$\Phi_i = \begin{cases} 5 \cdot g_i + 2 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \text{for i = 1,2}$$

$$0 \le x \le 3 \text{ and } 0 \le y \le 4$$

An initial population of two-gene individuals are randomly distributed over the entire design domain. The individuals are allowed to evolve for 100 generations. The global optimum was obtained in all runs. The typical optimum obtained is

(x*,y*) = (2.3295228, 3.1785152) with f* = -5.508038

The global optimum was obtained in 74 generations with 3,800 function evaluations. The history of the optimization process is presented in Table 5.16. The convergence plot of GaNOP for this problem in shown in Figure 5.18. The running percentage of time that the different parts of the quadratic crossover are implemented is shown in Figure 5.19. The trend in Figure 5.19 is similar to the trend observed for the Rosenbrock function in 10 dimensions.

Table 5.17: History of optimization process for the linear objective function

Generation #	x	y	F
0	2.3435702	2.6939869	5.0375571
10	2.3319812	3.1651652	5.4971466
50	2.329519	3.178484	5.5080032
100	2.3295228	3.1785152	5.508038
CPU Run Time (s)	2		
No. of Crossovers	5,000		
No. of Mutations	984		

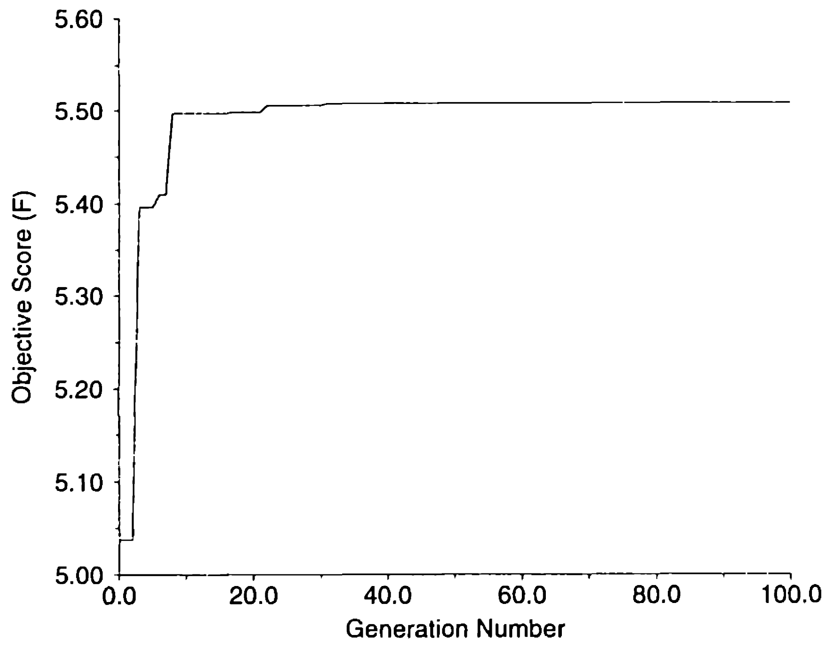


Figure 5.18: Convergence plot for linear objective function

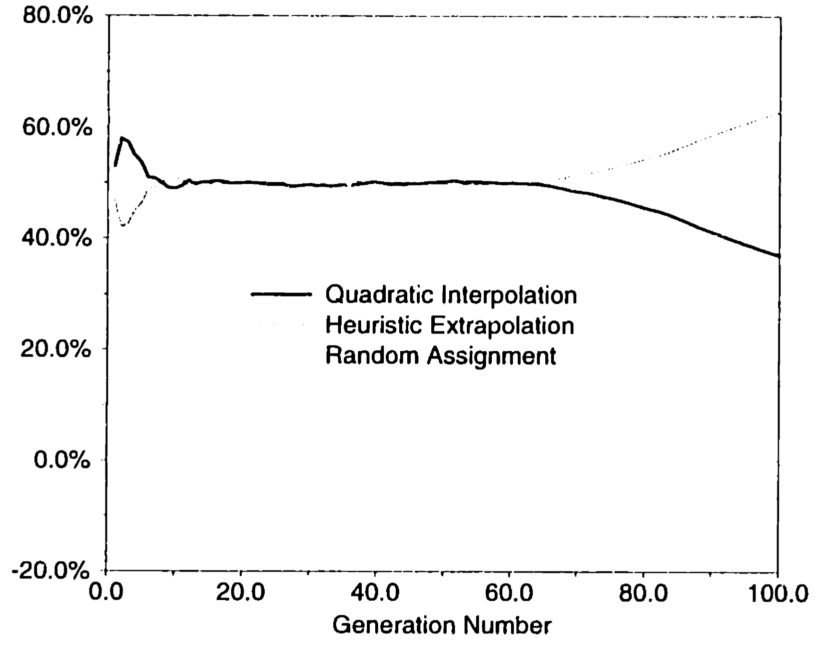


Figure 5.19: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for the linear objective function

5.5 Summary of Results

GaNOP obtained the global optimum in all the problems considered in this chapter, except for the 10-D parabolic multiminima function where it remained consistently trapped in one of the local minimum adjacent to the global optimum. The results obtained by GaNOP are accurate to at least six significant digits after the decimal point, except for the 10-D parabolic multiminima function. This shows that the quadratic crossover is effective in accurately finding the global optimum.

The observed trend in the running percentage of the time that the different parts of the quadratic crossover are implemented for the problems considered in this chapter is that quadratic interpolation dominates when the individuals are fairly dissimilar, or when GaNOP is search for the neighborhood of the optimum, and heuristic extrapolation dominates when the individuals are fairly similar, or when GaNOP has located the neighborhood of the optimum and is fine tuning the optimum. This observed trend does not apply to the parabolic multiminima function, where quadratic interpolation dominates throughout the evolution process. This is probably due to the strongly discontinuous nature of the function: quadratic interpolation gets the individuals out of the local minima, and heuristic extrapolation is probably more useful in the continuous bowl where the global optimum solution lies.

Chapter VI

Optimization with GaNOP: *Practical Problems*

6.1 Overview

In the previous chapter, the results of the optimization of selected theoretical problems with GaNOP were presented, and GaNOP was found to be effective in finding the global optimum to the desired accuracy for a wide class of nonlinear problems. In this chapter, practical problems are considered. Some of these problems have objective functions and constraints that are highly nonlinear with nonconvexities. The global optima for some of the problems are known, and only the best known global optima are available for the other problems. For all test problems, the optimum obtained by GaNOP will be compared to the global optimum or the best known global optimum.

Ten independent runs are made for each test problem, and the average performance of GaNOP for the ten runs is reported. Again, the running percentage of the time that the different parts of the quadratic crossover are implemented will be reported.

6.2 Coil Spring Design Problem

The standard definition of a coil spring design problem from [Arora, 1989, p. 450-454] is as follows:

$$\text{Minimize } f = (N+2)\cdot D\cdot d^2$$

subject to

$$g_1 = 1 - \frac{D^3\cdot N}{71785\cdot d^3} \le 0$$

$$g_2 = \frac{D\cdot(4D-d)}{12566\cdot d^3\cdot(D-d)} + \frac{2.46}{12566\cdot d^2} - 1 \le 0$$

$$g_3 = 1 - \frac{140.54\cdot d}{D^2\cdot N} \le 0$$

$$g_4 = \frac{D+d}{1.5} - 1 \leq 0$$

bounds:

$$0.05 \leq d \leq 0.2\,, \quad 0.25 \leq D \leq 0.5\,, \quad 2 \leq N \leq 15$$

where d is the wire diameter, D is the coil diameter, and N is the number of active coils. The design problem has three variables and four inequality constraints. The bound constraints introduce six additional inequality constraints. Arora [1989] solved the minimization problem using sequential quadratic programming. The optimum obtained is d* = 0.051699, D* = 0.35695, and N* = 11.289 with f* = 0.0126787.

GaNOP

The minimization problem is restated as follows:

Maximize $$F = -f - \sum_{i=1}^{4} \Phi_i$$

where $$\Phi_i = \begin{cases} g_i & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

The bounds on the variables remain the same.

A population of three-gene individuals is randomly distributed over the entire domain. The first gene corresponds to d, the second gene to D, and the third gene to N. The individuals are allowed to evolve for 500 generations with GaNOP keeping track of the best individual throughout the evolution process.

GaNOP obtained the optimum in all runs. A typical optimum obtained is

f* = 0.0126787
at d* = 0.0516931, D* = 0.356816, N* = 11.2973

On average, the optimum was obtained in 74 generations with 3,800 function evaluations. The history of the optimization process is given in Table 6.1. The convergence plot of GaNOP for this problem is shown in Figure 6.1.

The running percentage of time that the different parts of the quadratic crossover were implemented during the evolution process is shown in Figure 6.2. There is a sharp increase increase in the percentage of time that quadratic interpolation occurs for the first nine generations. During this period, the best objetive score (F) moved from -0.218985 to -0.0130907, where the optimum solution is -0.0126787. Overall, the percentage of time that heuristic extrapolation occurs is greater than the percentage of time that quadratic interpolation occurs.

Table 6.1: History of optimization process for coil spring design problem[a]

Generation #	d	D	N	F
0	0.0585839	0.409274	13.5899	-0.0218985
50	0.0517467	0.358102	11.2259	-0.0126823
100	0.0516977	0.356927	11.2908	-0.0126787
250	0.0516931	0.356816	11.2973	-0.0126787
500	0.0516931	0.356816	11.2973	-0.0126787
CPU Run Time (s)	4			
No. of Crossovers	25,000			
No. of Mutations	7,469			

a. Population Size = 100, Probability of Crossover = 1.0, Probability of Mutation = 0.1, Replacement Ratio = 0.5, Number of generations = 500

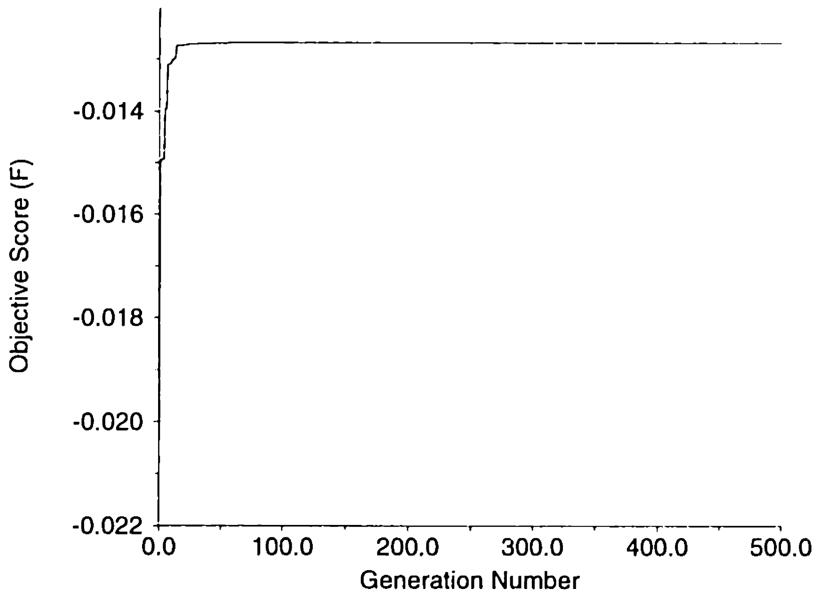


Figure 6.1: Convergence plot for coil spring design problem

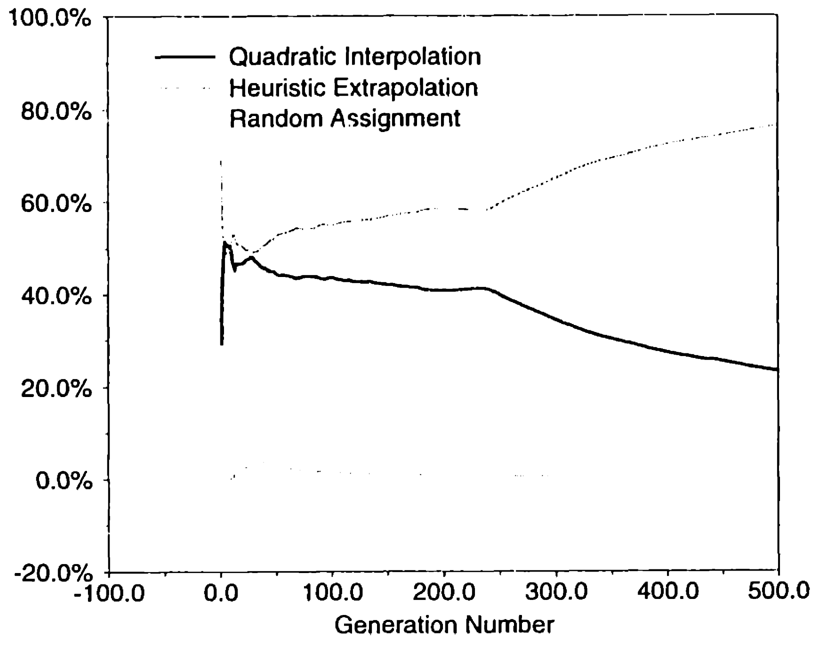


Figure 6.2: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for coil spring design problem

6.3 McGalliard's Problem

The following problem is taken from [Stephanopoulos and Westerberg, 1975]. The minimization problem is stated as follows:

$$\text{Minimize } f(x, u) = x_1^{0.6} + x_2^{0.6} + x_3^{0.4} + 2u_1 + 5u_2 - 4x_3 - u_3$$

subject to

$$h_1 = x_2 - 3x_1 - 3u_1 = 0$$

$$h_2 = x_3 - 2x_2 - 2u_2 = 0$$

$$h_3 = 4u_1 - u_3 = 0$$

$$g_1 = x_1 + 2u_1 - 4 \le 0$$

$$g_2 = x_2 + u_2 - 4 \le 0$$

$$g_3 = x_3 + u_3 - 6 \le 0$$

$$g_4 = x_1 - 3 \le 0$$

$$g_5 = u_2 - 2 \le 0$$
$$g_6 = x_3 - 4 \le 0$$

lower bounds:

$$x, u \ge 0$$

This problem features a nonconvex objective function with three linear equality constraints and six linear inequality constraints. The lower bounds on the variables introduce six additional inequality constraints. The optimum found by Stephanopoulos and Westerberg [1975] using the Hestenes' method of multipliers is f* = -11.96 at x* = (0.67,2,4) and u* = (0,0,0).

GaNOP

The first step in transforming the minimization problem to a GaNOP problem is to eliminate the equality constraints. This is accomplished as follows:

From the equality constraint h_1,

$$u_1 = \frac{x_2 - 3x_1}{3}$$

From the equality constraint h_2,

$$u_2 = \frac{x_3 - 2x_2}{2}$$

From the equality constraint h_3,

$$u_3 = 4u_1 = 4\left(\frac{x_2 - 3x_1}{3}\right)$$

The independent variables are now x_1, x_2, and x_3. The dependent variables are u_1, u_2, and u_3.

From the inequality constraint g_4 and lower bound constraint,

$$0 \le x_1 \le 3$$

From the inequality constraint g_6 and lower bound constraint,

$$0 \le x_3 \le 4$$

From the inequality constraint g_5 and upper bound constraint for x_3,

$$0 \le x_2 \le 2$$

The minimization problem can now be rewritten in terms of the independent variables only:

Minimize

$$f'(x) = x_1^{0.6} + x_2^{0.6} + x_3^{0.4} + 2\left(\frac{x_2 - 3x_1}{3}\right) + 5\left(\frac{x_3 - 2x_2}{2}\right) - 4x_3 - 4\left(\frac{x_2 - 3x_1}{3}\right)$$

subject to

$$g'_1 = x_1 + 2\left(\frac{x_2 - 3x_1}{3}\right) - 4 \le 0$$

$$g'_2 = x_2 + \left(\frac{x_3 - 2x_2}{2}\right) - 4 \le 0$$

$$g'_3 = x_3 + 4\left(\frac{x_2 - 3x_1}{3}\right) - 6 \le 0$$

new constraints (from **u** > 0):

$$g'_4 = -(x_2 - 3x_1) \le 0$$

$$g'_5 = -(x_3 - 2x_2) \le 0$$

$0 \le x_1 \le 3$, $0 \le x_2 \le 2$, and $0 \le x_3 \le 4$

The minimization problem is restated as follows:

$$\text{Maximize } F(x) = -f(x) - \sum_{i=1}^{5} \Phi_i$$

where

$$\Phi_i = \begin{cases} 10g'_i + 1 & \mathit{if}\ \ g'_i > 0 \\ 0 & \mathit{otherwise} \end{cases}, \quad \text{for i} = 1,\ldots,5$$

$0 \le x_1 \le 3$, $0 \le x_2 \le 2$, and $0 \le x_3 \le 4$

An initial population of three-gene individuals is randomly distributed over the entire design domain. The first gene corresponds to x_1, the second gene corresponds to x_2, and the third gene corresponds to x_3. The individuals are allowed to evolve for 150 generations with GaNOP keeping track of the best individual throughout the evolution process.

GaNOP obtained a better optimum than the reported optimum in all runs. The optimum found is

$f^* = -13.4019$
at $x^* = (0.166667, 2, 4)$ and $u^* = (0.5, 0, 2)$

On average, GaNOP finds the optimum in 35 generations with 1,850 function evaluations. The history of the optimization process is presented in Table 6.2. The convergence plot of GaNOP for this problem is shown in Figure 6.3. The running percentage of time that the different parts of the quadratic crossover are implemented during the evolution process is presented in Figure 6.4. Unlike the spring design problem, there is no initial increase in the percentage of time that quadratic interpolation occurs. Again, the percentage of time that heuristic extrapolation occurs is greater than the percentage of time that quadratic interpolation occurs. Note that there are random assignments during quadratic crossover.

Table 6.2: History of optimization process for McGalliard's problem[a]

Generation #	x_1	x_2	x_3	F
0	0.184763	1.43	3.69375	9.98548
10	0.165447	1.9962	3.99931	13.3851
50	0.166667	2	4	13.4019
100	0.166667	2	4	13.4019
150	0.166667	2	4	13.4019
CPU Run Time (s)	2			
No. of Crossovers	7500			
No. of Mutations	2146			

a. Population Size = 100, Probability of Crossover = 1.0, Probability of Mutation = 0.1, Replacement Ratio = 0.5, Number of Generations = 150

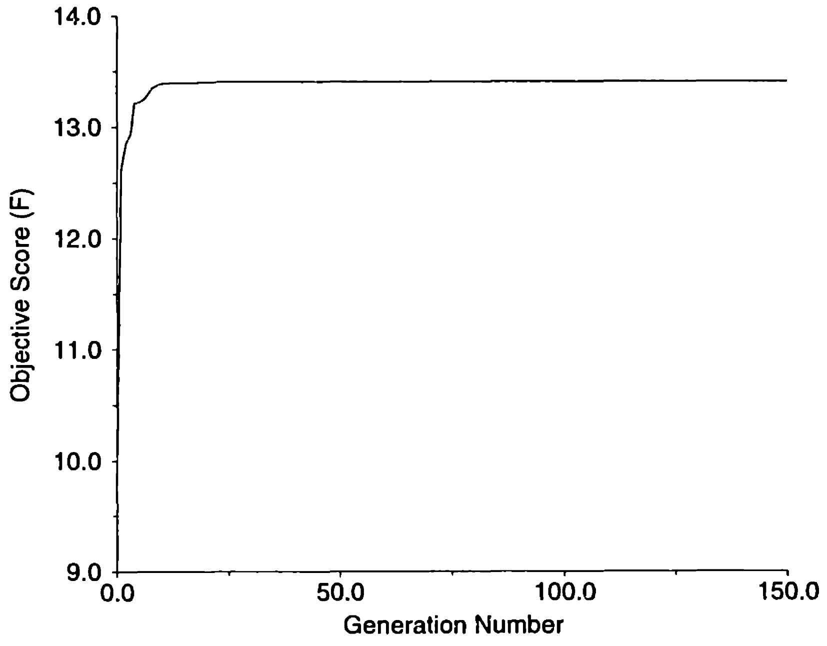


Figure 6.3: Convergence plot for McGalliard's problem

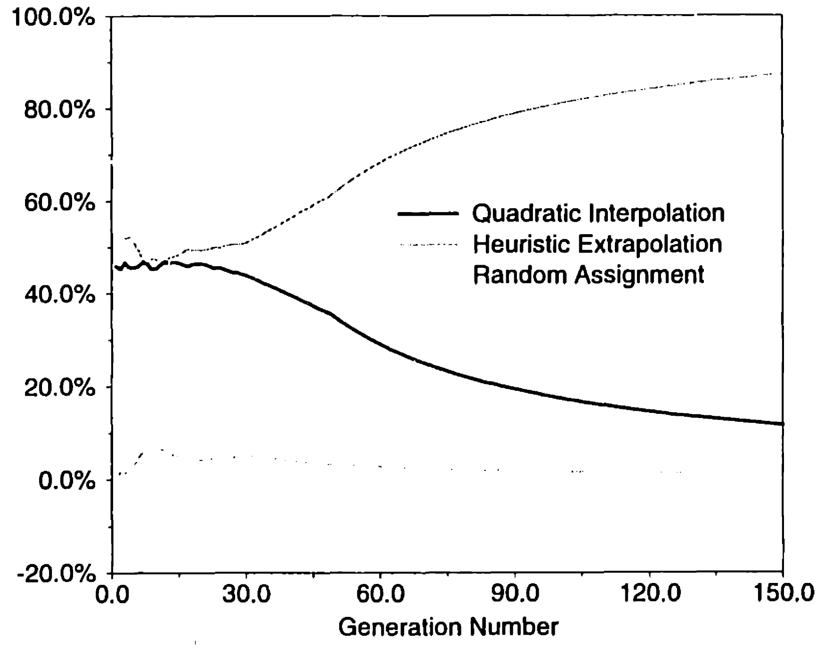


Figure 6.4: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for McGalliard's problem

6.4 Weight Minimization of a Speed Reducer

This problem is taken from [Chihsiung and Papalambros, 1996]. The minimization problem is stated as follows:

$$\text{Minimize } f(x) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.09334\right) - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.477\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right)$$

subject to

tooth bending stress: $g_1 = \dfrac{27}{x_1x_2^2x_3} - 1 \le 0$

tooth contact stress: $g_2 = \dfrac{397.5}{x_1x_2^2x_3^2} - 1 \le 0$

pinion deflection: $g_3 = \dfrac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0$

gear deflection: $g_4 = \dfrac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0$

pinion shaft stress: $g_5 = \dfrac{10}{x_6^3}\sqrt{\left(\dfrac{745x_4}{x_2x_3}\right)^2 + 16900000} \le 1100$

gear shaft stress: $g_6 = \dfrac{10}{x_7^3}\sqrt{\left(\dfrac{745x_5}{x_2x_3}\right)^2 + 147500000} \le 850$

geometry constraints: $g_7 = x_2x_3 - 40 \le 0$

$$g_8 = -\frac{x_1}{x_2} + 5 \le 0$$

$$g_9 = \frac{x_1}{x_2} - 12 \le 0$$

$$g_{10} = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11} = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

bounds: $2.6 \le x_1 \le 3.6$, $0.7 \le x_2 \le 0.8$, $17 \le x_3 \le 28$,

$$7.3 \le x_4, x_5 \le 8.3\,,\ 2.9 \le x_6 \le 3.9\,,\ \ 5.0 \le x_7 \le 5.5$$

The best solution found by Chihsiung and Papalambros [1996] using the convex cutting plane algorithm for generalized polynomial models is x* = (3.5, 0.7, 17.0, 7.3, 7.65, 3.34, 5.23) with f* = 2954.3. This solution violates constraint g_5.

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(x) = -f(x) - \sum_{i=1}^{11} \Phi_i$$

where

$$\Phi_i = \begin{cases} 100 \cdot g_i + 75 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i \ne 8, 9$$

$$\Phi_i = \begin{cases} 100 \cdot g_i + 145 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 8, 9$$

The bounds on the variables remain the same.

A population of seven-gene individuals is randomly distributed over the entire design domain. The individuals are allowed to evolve for 500 generations. The optimum found by GaNOP in all runs is

f* = 2956.906
at x* = (3.5, 0.69999999, 17, 7.3000002, 7.6521692, 3.3502147, 5.2292361)

This solution does not violate any constraint. GaNOP obtained the optimum solution in 193 generations with 9,750 function evaluations. The history of the optimization process is presented in Table 6.3. The running percentage of time that the different parts of the quadratic crossover operator are implemented is shown in Figure 6.6. The trend for the speed reducer problem is similar to the trend for McGalliard's problem.

Table 6.3: History of the optimization process for speed reducer problem[a]

Generation #	F
0	-3621.1467
50	-2960.4014
100	-2957.0173
250	-2956.906
500	-2956.906
CPU Run Time (s)	6
No. of Crossovers	25000
No. of Mutations	17356

a. Population Size = 100, Probability of Crossover = 1.0, Probability of Mutation = 0.1, Replacement Ratio = 0.5

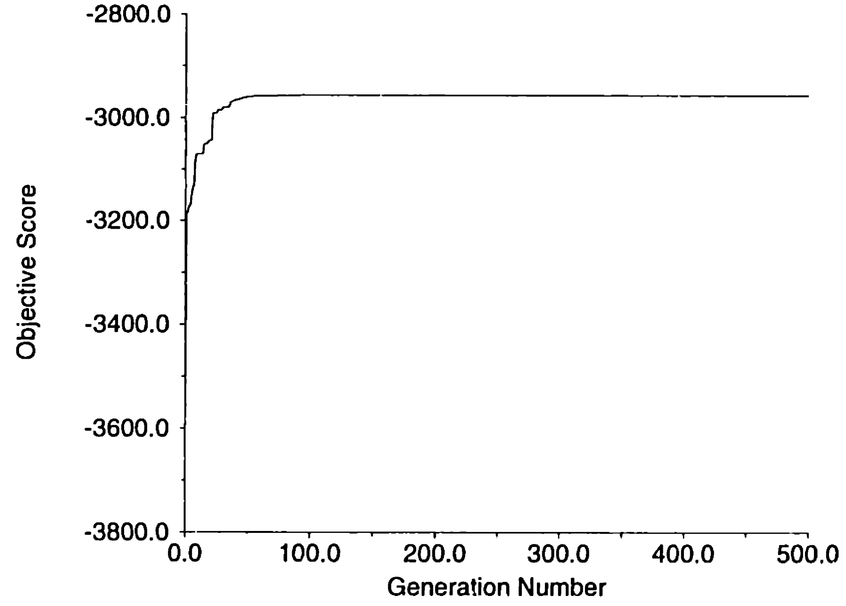


Figure 6.5: Convergence plot for the speed reducer problem

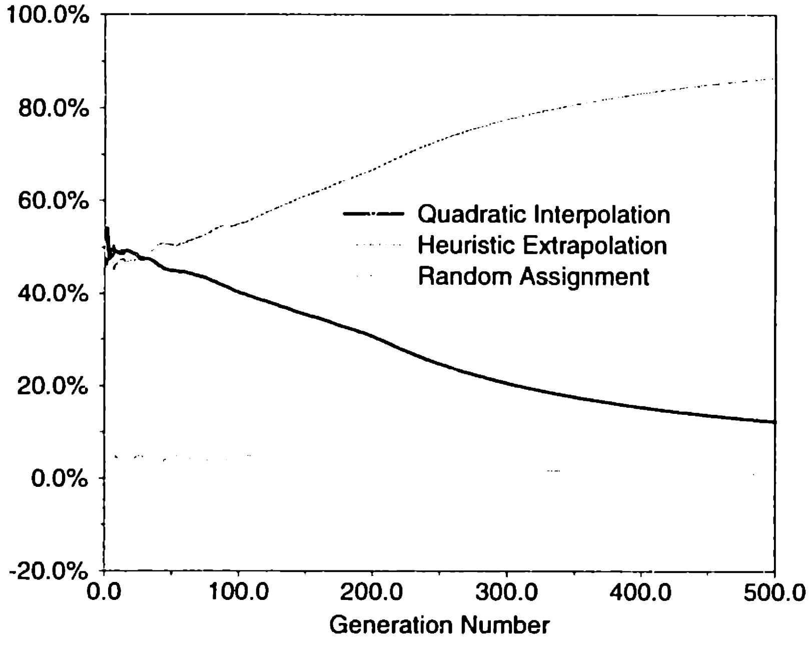


Figure 6.6: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossvoer for the speed reducer problem

6.5 Chip Layout Problem

The goal of this optimization process is to find the relative placement and shapes of chips such that the total chip area is minimized subject to linear and nonlinear constraints. The details of the problem can be found in [Dorneich and Sahinidis, 1995]. The optimization problem is stated as follows:

Minimize $f = x_a y_a + x_b y_b + x_c y_c + x_d y_d + x_e y_e + x_f y_f$

subject to

$$h_1 = -x_d + x_e = 0$$
$$h_2 = -x_a + x_b = 0$$
$$h_3 = -x_a + x_c - x_d + x_f = 0$$
$$h_4 = -y_c + y_f = 0$$
$$h_5 = -y_a - y_b + y_d + y_e = 0$$
$$g_1 = -x_b + x_c + 1 \le 0$$

$$g_2 = y_d - y_a + 1 \le 0$$
$$g_3 = -x_a y_a + 30 \le 0$$
$$g_4 = -x_b y_b + 20 \le 0$$
$$g_5 = -x_c y_c + 25 \le 0$$
$$g_6 = -x_d y_d + 25 \le 0$$
$$g_7 = -x_e y_e + 15 \le 0$$
$$g_8 = -x_f y_f + 20 \le 0$$

lower bounds:

$$x_a, x_b, x_f, y_a, y_c, y_e, y_f \ge 5$$
$$x_d, x_e, y_d \ge 4$$
$$x_c, y_b \ge 2$$

The global solution to the problem is x* = (5, 5, 4, 4, 4, 5), y* = (7.25, 4, 5, 6.25, 5, 5), and f* = 146.25.

GaNOP

The first step in using GaNOP is to transform the minimization problem to a GaNOP problem. This is accomplished as follows:

From the equality constraint h_1,

$$x_e = x_d$$

From the equality constraint h_2,

$$x_b = x_a$$

From the equality constraint h_3,

$$x_f = x_a + x_d - x_c$$

So x_a, x_c, and x_d are the independent variables; x_c, x_e, and x_f are the dependent variables.

From the equality constraint h_4,

$$y_f = y_c$$

From the equality constraint h_5,

$$y_e = y_a + y_b - y_d$$

So y_a, y_b, y_c, and y_d are the independent variables; y_e and y_f are the dependent variables.

The minimization problem can now be stated in terms of the independent variables only:

Minimize

$$f'(x) = x_a y_a + x_a y_b + x_c y_c + x_d y_d + x_d (y_a + y_b - y_d) + (x_a + x_d - x_c) y_c$$

subject to

$$g'_1 = -x_a + x_c + 1 \le 0$$
$$g'_2 = y_d - y_a + 1 \le 0$$
$$g'_3 = -x_a y_a + 30 \le 0$$
$$g'_4 = -x_a y_b + 20 \le 0$$
$$g'_5 = -x_c y_c + 25 \le 0$$
$$g'_6 = -x_d y_d + 25 \le 0$$
$$g'_7 = -x_d (y_a + y_b - y_d) + 15 \le 0$$
$$g'_8 = -(x_a + x_d - x_c) y_c + 20 \le 0$$
$$g'_9 = -(x_a + x_d - x_c) + 5 \le 0$$
$$g'_{10} = -(y_a + y_b - y_d) + 5 \le 0$$

lower bounds:

$x_a, y_a, y_c \ge 5$

$x_d, y_d \ge 4$

$x_c, y_b \ge 2$

This problem now has seven independent variables. The global optimum is $x_a = 5$, $x_c = 4$, $x_d = 4$, $y_a = 7.25$, $y_b = 4$, $y_c = 5$, $y_d = 6.25$ with f = 146.25.

The minimization problem can be restated as

$$\text{Maximize } F(x) = -f'(x) - \sum_{i=1}^{10} \Phi_i$$

where

$$\Phi_i = \begin{cases} g'_i + 20 & \text{if } g'_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

The lower bounds remain the same. Since GaNOP works with a bounded domain, the upper bound for all the variables is set to 30.

An initial population of 7-gene individuals is randomly distributed over the entire design domain. The individuals are allowed to evolve for 400 generations. The exact global optimum was obtained in all runs. The optimum found is

$$x_a = 5,\ x_c = 4,\ x_d = 4,\ y_a = 7.25,\ y_b = 4,\ y_c = 5,\ y_d = 6.25 \text{ with } f^* = 146.25$$

On average, the optimum was found in 319 generations with 16,050 function evaluations. The history of the optimization process is given in Table 6.4. The convergence plot of GaNOP for this problem is shown in Figure 6.7. The running percentage of the time that the different parts of the quadratic crossover are implemented is shown in Figure 6.8. The trend in Figure 6.8 is again similar to the trends in McGalliard's and the speed reducer problems. Note that there are random assignments during quadratic crossover for this problem. There is also an initial increase in the percentage of time that quadratic interpolation occurs during quadratic crossover.

Table 6.4: History of optimization process for chip layout problem[a]

Generation #	Penalty	F
0	130.059	-627.5
50	0	-156.442
100	0	-148.687
250	0	-146.252
400	0	-146.25
CPU Run Time (s)	6	
No. of Crossovers	20000	
No. of Mutations	14159	

a. Population Size = 100, Probability of Crossover = 1.0, Probability of Mutation = 0.1, Replacement Ratio = 0.5, Number of Generations = 400

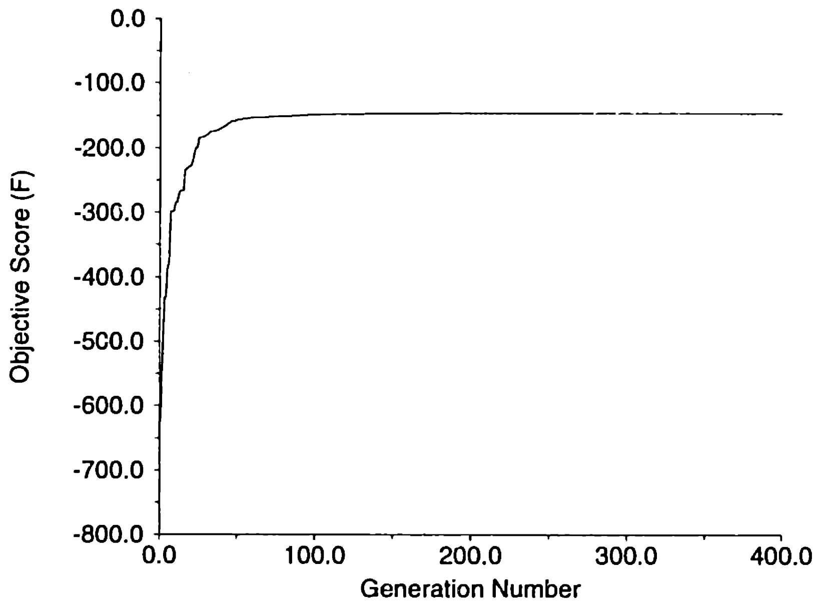


Figure 6.7: Convergence plot for chip layout problem

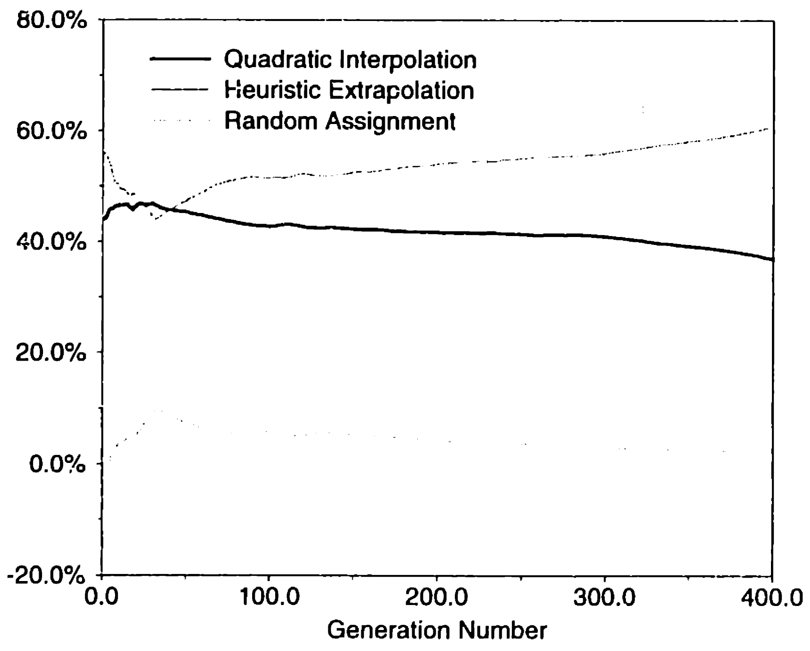


Figure 6.8: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for the chip layout problem

6.6 Three-Stage Membrane Separation Problem

This problem is taken from [Hock and Schittkowski, 1981]. The minimization problem is stated as follows:

Minimize $f(x) = x_{11} + x_{12} + x_{13}$

subject to

$$
\begin{aligned}
g_1 &= x_2 - x_3 \le 0 \\
g_2 &= x_1 - x_2 \le 0 \\
g_3 &= -1 + 0.002x_7 - 0.002x_8 \le 0 \\
g_4 &= 50 - f(x) \le 0 \\
g_5 &= f(x) - 250 \le 0 \\
g_6 &= -x_{13} + 1.262626x_{10} - 1.231059x_3x_{10} \le 0 \\
g_7 &= -x_5 + 0.03475x_2 + 0.975x_2x_5 - 0.00975x_2^2 \le 0 \\
g_8 &= -x_6 + 0.03475x_3 + 0.975x_3x_6 - 0.00975x_3^2 \le 0 \\
g_9 &= -x_5x_7 + x_1x_8 + x_4x_7 - x_4x_8 \le 0 \\
g_{10} &= -1 + 0.002\,(x_2x_9 + x_5x_8 - x_1x_8 - x_6x_9) + x_5 + x_6 \le 0 \\
g_{11} &= -x_2x_9 + x_3x_{10} + x_6x_9 + 500x_2 - 500x_6 - x_2x_{10} \le 0 \\
g_{12} &= -x_2 + 0.9 + 0.002\,(x_2x_{10} - x_3x_{10}) \le 0 \\
g_{13} &= -x_4 + 0.03475x_1 + 0.975x_1x_4 - 0.00975x_1^2 \\
g_{14} &= -x_{11} + 1.262626x_8 - 1.231059x_1x_8 \le 0 \\
g_{15} &= -x_{12} + 1.262626x_9 - 1.231059x_2x_9 \le 0
\end{aligned}
$$

bounds:

$$
\begin{aligned}
&0.1 \le x_1, x_2, x_3 \le 1,\ \ 0.0001 \le x_4 \le 0.1,\ \ 0.1 \le x_5, x_6 \le 0.9, \\
&0.1 \le x_7, x_8 \le 1000,\ \ 500 \le x_9 \le 1000,\ \ 0.1 \le x_{10} \le 500, \\
&1 \le x_{11} \le 150,\ \ 0.0001 \le x_{12}, x_{13} \le 150
\end{aligned}
$$

This problem features a linear objective function with 5 linear inequality constraints and 10 nonlinear inequality constraints. The optimum solution is

f* = 97.588409
at x* = (0.8037703, 0.899860, 0.9709724, 0.9999952, 0.1908154,
0.4605717, 574.0803, 74.08043, 500.0162, 0.1, 20.23413,
77.43755, 0.00673039)

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(x) = -f(x) - \sum_{i=1}^{15} \Phi_i$$

where

$$\Phi_i = \begin{cases} 50000 \cdot g_i + 20 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases}, \text{ for i = 1,...,15}$$

An initial population of 13-gene individuals is randomly distributed over the entire design domain. The individuals are allowed to evolve for 5,000 generations. A typical optimum found by GaNOP is

f* = 97.600624
x* = (0.80366808, 0.90004075, 0.97483724, 0.099943474, 0.19090439, 0.4969188, 574.2218, 74.221855, 500.01855, 0.1, 20.282104, 77.312241, 0.0062818807)

The deviation from the global optimum is 0.012215. This optimum is obtained in 4,992 generations with 249,700 function evaluations. The history of the optimization process is given in Table 6.5. The convergence plot for this problem is given in Figure 6.9. The running percentage of time that the different parts of the quadratic crossover are implemented is shown in Figure 6.10.

Table 6.5: History of optimization process for the membrane separation problem[a]

Generation #	F	penalty
0	-2307982.5	2307810.3
100	-805.84973	662.3385
CPU Run Time (s)	89	
No. of Crossovers	237,512	
No. of Mutations	324,958	

Table 6.5: History of optimization process for the membrane separation problem[a]

Generation #	F	penalty
500	-305.47379	211.21477
1000	-98.212151	0
2500	-97.679665	0
5000	-97.600624	0
CPU Run Time (s)	89	
No. of Crossovers	237,512	
No. of Mutations	324,958	

a. Population Size = 100, Probability of Crossover = 0.95, Probability of Mutation = 0.1, Replacement Ratio = 0.5, Number of Generations = 5000

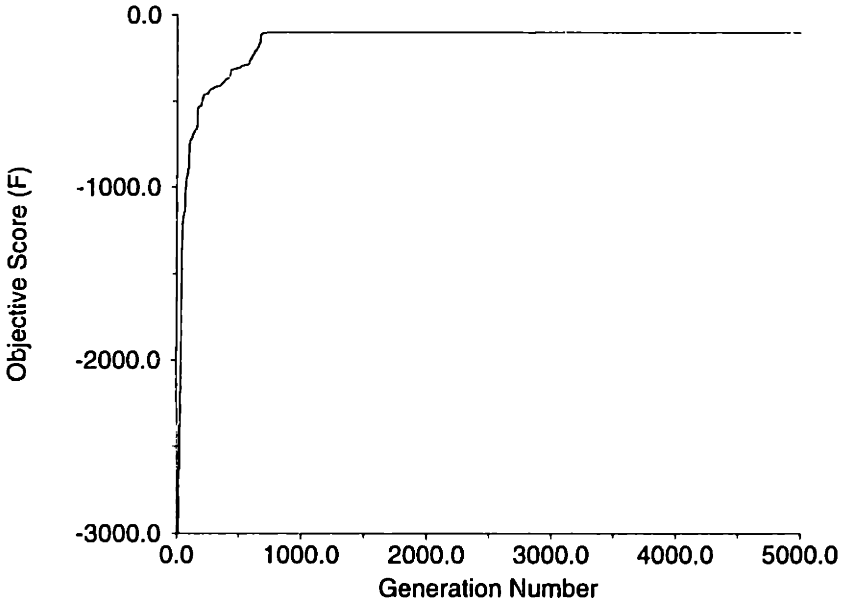


Figure 6.9: Convergence plot for the membrane separation problem

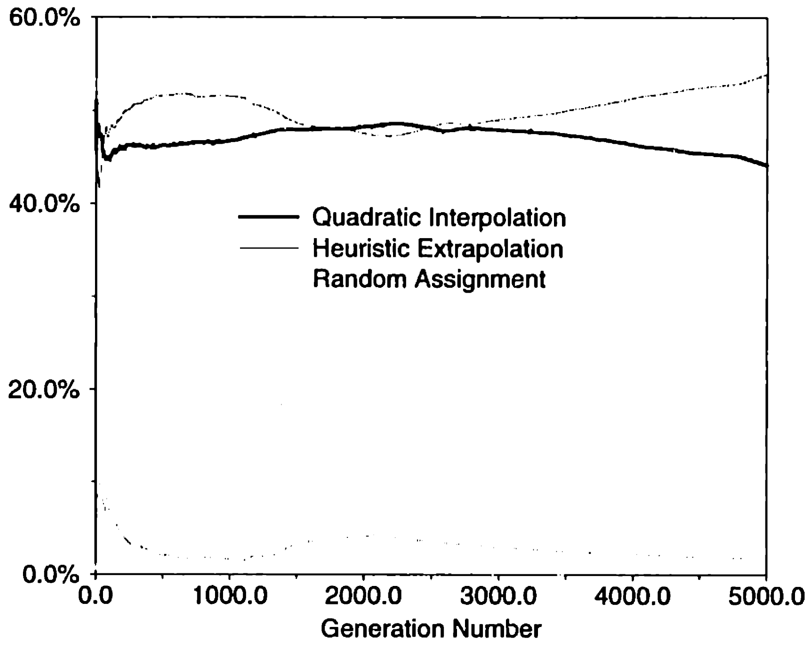


Figure 6.10: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for the membrane separation problem

6.7 Bartholomew-Biggs Problem

This problem is taken from [Hock and Schittkowski, 1981]. The minimization problem is stated as follows:

$$\text{Minimize } f(x) = \sum_{k=0}^{4} \Big(2.3x_{3k+1} + 0.0001x_{3k+1}^2 + 1.7x_{3k+2} +$$

$$0.0001x_{3k+2}^2 + 2.2x_{3k+3} + 0.00015x_{3k+3}^2 \Big)$$

subject to

$$g_j = -x_{3j+1} + x_{3j-2} - 7 \le 0$$

$$g_{j+4} = x_{3j+1} - x_{3j-2} - 6 \le 0$$

$$g_{j+8} = -x_{3j+2} + x_{3j-1} - 7 \le 0$$

$$g_{j+12} = x_{3j+2} - x_{3j-1} - 7 \le 0$$

$$g_{j+16} = -x_{3j+3} + x_{3j} - 7 \le 0$$
$$g_{j+20} = x_{3j+3} - x_{3j} - 6 \le 0$$
for j = 1,...4
$$g_{25} = -x_1 - x_2 - x_3 + 60 \le 0$$
$$g_{26} = -x_4 - x_5 - x_6 + 50 \le 0$$
$$g_{27} = -x_7 - x_8 - x_9 + 70 \le 0$$
$$g_{28} = -x_{10} - x_{11} - x_{12} + 85 \le 0$$
$$g_{29} = -x_{13} - x_{14} - x_{15} + 100 \le 0$$

bounds:

$$8 \le x_1 \le 21,\ 43 \le x_2 \le 57,\ 3 \le x_3 \le 16,\ 0 \le x_{3k+1} \le 90,$$
$$0 \le x_{3k+2} \le 120,\ 0 \le x_{3k+3} \le 60$$
for k = 1,...4

The problem features a quadratic objective function with 29 linear constraints. The optimum solution is x* = (8,49,3,1,56,0,1,63,6,3,70,12,5,77,18) with f* = 664.8204500

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize } F(x) = -f(x) - \sum_{i=1}^{29} \Phi_i$$

where

$$\Phi_i = \begin{cases} 10 \cdot g_i + 5 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{, for i = 1,...,29}$$

The bounds on the variables remain the same.

An initial population of 15-gene individuals is randomly distributed over the entire design domain. The individuals are allowed to evolve for 2,000 generations with GaNOP keeping track of the best individual.

The best optimum found by GaNOP is

f* = 665.734
at x* = (8, 48.1534, 3.8467, 1.00045, 52.0664, 6.2898e-16, 5.16778, 59.0663, 5.7659, 7.9083, 66.0549, 11.0368, 10.0282, 72.9355, 17.0364)

The deviation from the global optimum is 0.913. This optimum is obtained in 1,311 generations with 19,725 function evaluations. The history of the optimization process is shown in Table 6.6. The convergence plot for this problem is shown in Figure 6.11. The running percentage of time that the different parts of the quadratic crossover are implemented is shown in Figure 6.12.

Table 6.6: History of optimization process for Bartholomew-Biggs problem[a]

Generation #	F
0	-2429.33
100	-5.3334
500	-671.716
1000	-665.908
1500	-665.734
2000	-665.734
CPU Run Time (s)	12
No. of Crossovers	28751
No. of Mutations	40

a. Population Size = 60, Probability of Crossover = 0.96, Probability of Mutation = 0.0001, Replacement Ratio = 0.25, Number of Generations = 2000

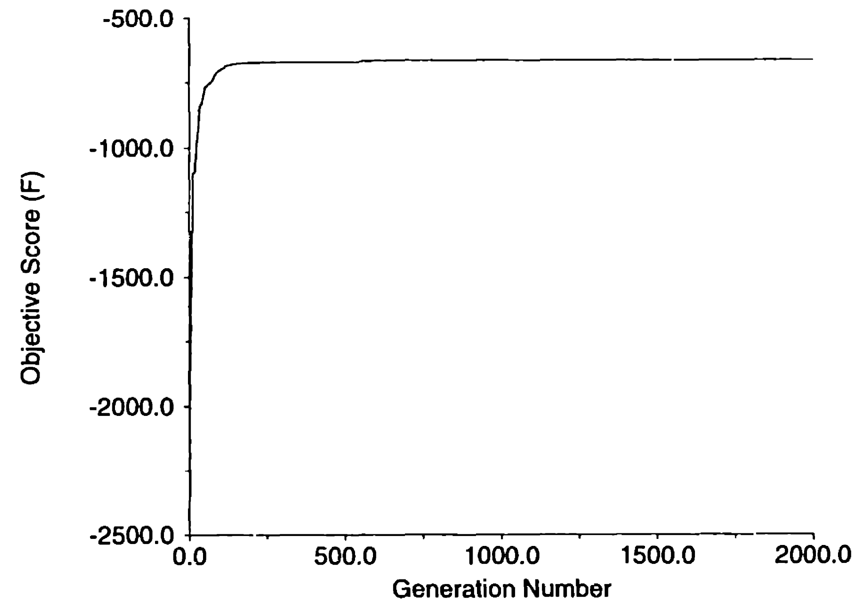


Figure 6.11: Convergence plot for Bartholomew-Biggs problem

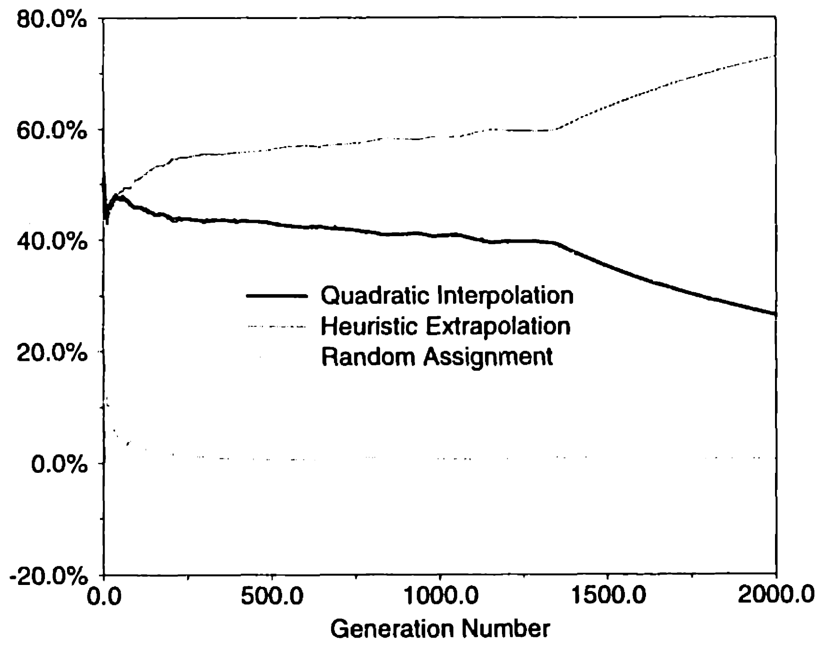


Figure 6.12: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for the Bartholomew-Biggs problem

6.8 Heat Exchanger Design

This problem is taken from [Hock and Schittkowski, 1981]:

$$\text{Minimize} \quad f(x_1, x_2, x_3) = x_1 + x_2 + x_3$$

subject to

$$\begin{aligned}
g_1 &= -1 + 0.0025\,(x_4 + x_6) \le 0 \\
g_2 &= -1 + 0.0025\,(-x_4 + x_5 + x_7) \le 0 \\
g_3 &= -1 + 0.01\,(-x_5 + x_8) \le 0 \\
g_4 &= 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0 \\
g_5 &= x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0 \\
g_6 &= x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0 \\
& 100 \le x_1 \le 10000, \quad 1000 \le x_2, x_3 \le 10000 \\
& \text{and } 10 \le x_4, x_5, x_6, x_7, x_8 \le 1000
\end{aligned}$$

This problem has eight variables with three linear and three nonlinear constraints. The best known global optimum is x* = (579.31, 1359.97, 5109.97, 182.02, 295.60, 217.98, 286.42, 395.60) and f* = 7049.25 [Hansen et al, 1989].

GaNOP

The minimization problem is restated as follows:

$$\text{Maximize} \quad F(x) = -f(x) - \sum_{i=1}^{6} \Phi_i$$

where

$$\Phi_i = \begin{cases} 10000 \cdot g_i + 1000 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \text{for i = 1,...,3}$$

$$\Phi_i = \begin{cases} 100 \cdot g_i + 5 & \text{if } g_i > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \text{for i = 4,...,6}$$

An initial population of 8-gene individuals is randomly distributed over the entire design domain. The individuals are allowed to evolve for 5,000 generations. The best optimum found by GaNOP is

x* = (605.629, 1344.67, 5099.37, 184.176, 296.025, 215.824, 288.151, 396.025)
f* = 7049.67

The typical optimum is
x* = (613.424, 1385.13, 5052.05, 184.8, 297.918, 215.2, 286.882, 397.918)
f* = 7050.61

The deviation of the best optimum from the global optimum is 0.42.

Table 6.7: History of optimization process for heat exchanger design problem (best optimum)[a]

Generation #	F	Penalty
0	-27978.9	16444.9
100	-7207.25	0
500	-7051.34	0
1000	-7049.67	0
2500	-7049.67	0
5000	-7049.67	0
CPU Run Time (s)		95
No. of Crossovers		375,000
No. of Mutations		333

a. Population Size = 100, Probability of Crossover = 1.0, Probability of Mutation = 0.0001, Replacement Ratio = 0.75, Number of Generations = 5000

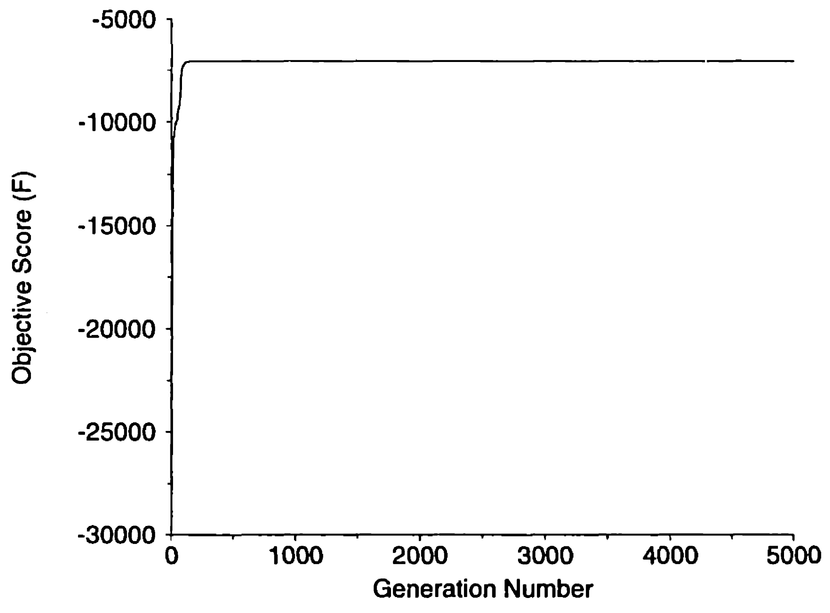


Figure 6.13: Convergence plot for heat exhanger design problem

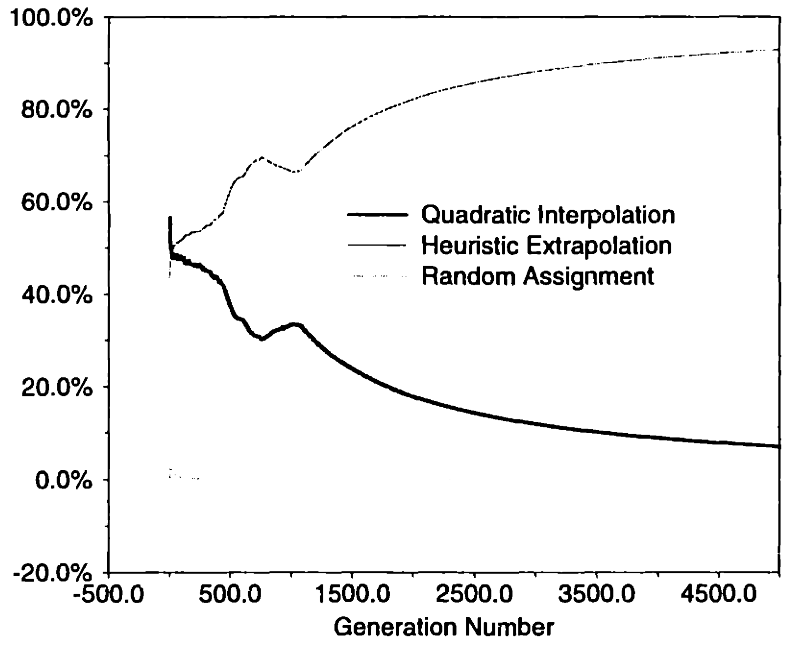


Figure 6.14: Running percentage of time that quadratic interpolation, heuristic extrapolation, and random assignment are performed during quadratic crossover for the heat exchanger design problem

6.9 Summary of Results

GaNOP obtained the reported global optimum for the coil spring design problem and the chip layout problem. GaNOP found better answers than the reported optimum for McGalliard's problem and the speed reducer problem. GaNOP did not find the exact optimum for the membrane separation problem; the percent deviation from the optimum (f*) is 0.0125%. GaNOP did not find the exact optimum for the Bartholomew-Biggs problem; the percent deviation from the optimum (f*) is 0.137%. GaNOP did not find the exact optimum for the heat exchanger problem; the percent deviation from the optimum (f*) is 0.006%.

The general trend observed for the percentage of time that the different parts of the quadratic crossover are implemented for the practical problems considered in this chapter is quite different from the general trend observed for the theoretical problems. For most of the practical problems, the heuristic extrapolation dominates throughout the evolution process. In the case of the theoretical problems, the quadratic interpolation usually dominates at the beginning of the evolution process and the heuristic extrapolation dominates at the end. However, one thing that is consistent in the trends for the theoretical and practical problems is that the quadratic interpolation usually shows an initial increase for the first few generations of the evolution process and then starts to drop with the progression of the evolution process.

Chapter VII

Conclusion

7.1 Summary

In the introductory chapter, genetic algorithms with real-valued chromosomes was proposed as an optimization technique for numerical optimization problems. The primary objection to using chromosomes with real-valued or floating-point genes in genetic search was discussed, and the conclusion was that the advantages of using real-coded genetic algorithm for numerical optimization problems far outweighed the objection to the use of real-valued genes in genetic search. This conclusion is the motivation for this thesis; the goal of the thesis was to develop new specialized genetic operators for the real-valued chromosomes.

A survey of the currently available genetic operators for real-valued chromosomes was presented in the second chapter. The observation was that the genetic operators (in particular, the crossover operators) had different strengths and weaknesses, sometimes making it necessary to combine two or more of these operators in a single genetic search.

In the third chapter, a new crossover operator, quadratic crossover, that uses polynomial interpolation and a pseudo-gradient local search technique to produce a new individual was proposed. The performance of the quadratic crossover was compared to the performances of the crossover operators presented in the second chapter. The quadratic crossover was found to be more reliable, and at times, more efficient than the other crossover operators for the selected standard genetic algorithm test problems considered.

A genetic-algorithm based numerical optimization system, GaNOP, was presented in the fourth chapter. This system uses the new quadratic crossover and a modified Gaussian mutator in genetic search.

In the fifth and sixth chapters, the effectiveness of GaNOP in finding the global optimum of a wide variety of theoretical and practical nonlinear test models to the desired accuracy was investigated. The quadratic operator, with the modified Gaussian mutator, guided GaNOP to the global optimum in 8 out of the 9 theoretical problems. The quadratic operator, with the modified Gaussian mutator, guided GaNOP to the reported global optimum in 2 out of 7 practical problems. GaNOP obtained better solutions than the reported optimum for another 2 of the practical problems. In the remaining three problems, GaNOP did not obtain the reported global optimum; although, GaNOP located the neighborhood of the optimum, i.e. the maximum percent deviation in the solution obtained is 0.13%. The primary conclusion is that the quadratic crossover is effective in accurately finding the global optimum for a wide variety of nonlinear test models.

7.2 Future Work

In GaNOP, the selection scheme during reproduction is the roulette wheel selection scheme. As shown in Chapter I, the roulette wheel selector can select the same individual more than once. The implication is that sometimes two or all of the parents selected for quadratic crossover can be identical, in which case, the quadratic interpolation part of the quadratic crossover will not be successfully implemented. If the three parents are identical, no useful information can be obtained from the heuristic extrapolation either. This means that the composition of the offspring will be determined by random assignments. To ensure that quadratic interpolation has a good chance of occurring during a single crossover, the parents selected for crossover should be distinctly different, especially during the first few generations when quadratic interpolation is very useful in locating the feasible region or the neighborhood of the optimum. A future direction would be to modify the roulette wheel selector such that a single individual gets selected only once during reproduction. It will also be interesting to see how other selection schemes affect the performance of GaNOP.

The performance of GaNOP on nontrivial problems is greatly affected by the control parameter settings. Unfortunately, the selection of the control parameter settings is still an art. However, this does not mean that control parameter settings that will guide GaNOP to the optimum based on the nature of the objective function and the constraints cannot be estimated. A future direction would be to find problems that are representative of the typical problems encountered in the real world. These problems will be classified according to the type of objective function and constraint(s). Empirical studies will be conducted on this problems, and a set of control parameter values that guide GaNOP to the global optimum will be identified. This information will be very useful for a designer that is not particularly interested in how GaNOP works but would like to find the global optimum to a design problem.

In almost all the test runs, GaNOP converged to an optimum before the specified maximum number of generations was completed. It would be nice to have GaNOP terminate just after it has converged to the optimum, rather than wait until the maximum number of generations is complete. A future direction would be to develop a better criterion for terminating GaNOP.

Bibliography

Antonisse, J. (1989). A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint. In Schaffer J. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 586-91). California: Morgan Kaufmann.

Arora, J. S. (1989). *Introduction to Optimum Design*. New York: McGraw-Hill Inc.

Chihsiung, Lo, & Papalambros, P. Y. (1996). A Convex Cutting Plane Algorithm for Global Solution of Generalized Polynomial Optimal Design Models. *Transactions of the ASME,* 118, 82-88.

Corana, A., et al. (1987). Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm. *ACM Transactions on Mathematical Software*, 13(3), 262-280.

Davis, L. (1991). Hybridization and Numerical Representation. In Davis L. (Ed.), *The Handbook of Genetic Algorithms* (pp. 61-71). New York: Van Nostrand Reinhold.

DeJong, Kenneth A. (1975). An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor.

DeJong, Kenneth A. (1993). Genetic Algorithms are not Function Optimizers. In Whitley, D. L. (Ed.), *Foundations of Genetic Algorithms* (pp. 5-17). California: Morgan Kaufmann.

Dorneich, M. C., & Sahindis, N. V. (1995). Global Optimization Algorithms for Chip Layout and Compaction. *Engineering Optimization*, 25(2), 131-154.

Eshelman, Larry J., & Schaffer, David J. (1993). Real-Coded Genetic Algorithms and Interval-Schemata. In Whitley, D. L. (Ed.), *Foundations of Genetic Algorithms 2* (pp. 187-202). California: Morgan Kaufmann.

Floudas, C. A., & Pardalos, P. M. (1987). *A Collection of Test Problems for Constrained Global Optimization Problems*. In Goos, G. & Hartmanis, J. (Eds.), (Vol. 455), Lecture Notes in Computer Science. New York: Springer Verlag.

Floudas, C. A., & Pardalos, P. M. (1992). Recent Advances in Global Optimization. In Princeton Series in Computer Science. New Jersey: Princeton University Press.

Fogel, D. B., & Atmar J. W. (1990). Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems. *Biological Cybernetics,* 63, 111-114.

Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. New York: Addison-Wesley.

Goldberg, D. E. (1990) Real-Coded Genetic Algorithms, Virtual Alphabets, and Blocking, IlliGAL Report 90001, Illinois Genetic Algorithms Laboratory Dept. of General Engineering University of Illinois at Urbana-Champaign, Urbana, IL.

Hock, W. & Schittkowski, K. (1981) Test Examples for Nonlinear Programming Codes. In Lecture Notes in Economics and Mathematical Systems. (Vol. 187). New York: Springer-Verlag.

Holland, John H. (1975) *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press.

Janikow, C.Z., & Michalewicz, J. (1991). An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms. In Belew, R. K., & Booker, L. B. (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 31-36). California: Morgan Kaufmann.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs* (2nd ed.). New York: Springer Verlag.

Peck, C. C., & Dhawan, A. P. (1995). Genetic Algorithms as Global Random Search Methods: An Alternative Perspective. *Evolutionary Computation*, 3(1), 39-80.

Radcliffe, N. J. (1991) Genetic Neural Networks on MIMD Computers, Ph.D. Dissertation, Dept. of Theoretical Physics, University of Edinburgh, Edinburgh, UK.

Schaffer, D. J., Carauna, R. A., Eshelman, L. J., & Rajarshi, D. (1989). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In Schaffer J. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 51-60). California: Morgan Kaufmann.

Schittkowski, K. (1987). More Test Examples for Nonlinear Programming Codes. In Lecture Notes in Economics and Mathematical Systems. (Vol. 282). Springer-Verlag.

Stephanopoulos, G., & Westerberg, A. W. (1975). The Use of Hestenes' Method of Multipliers to Resolve Dual Gaps in Engineering System Optimization. *Journal of Optimization Theory and Applications*, 15(3), 285-309.

Whitley, D. (1989). The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In Schaffer J. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 116-121). California: Morgan Kaufmann.

Wright, A., (1991) Genetic Algorithms for Real Parameter Optimization, *Foundations of Genetic Algorithms.* In Rawlins, G. J. E. (Ed.), *Foundations of Genetic Algorithms* (pp. 205-218). California: Morgan Kaufmann.

Zhigljavsky, A. A. (1991) *Theory of Global Random Search.* Boston: Kluwer Academic Publishers.