Genetic Invariance: A New Type of Genetic Algorithm

Michael J. Lewchuk

April 1992

Abstract

Genetic algorithms are adaptive search algorithms which generate and test a population of individuals, where each individual corresponds to a solution. They then adapt to the information obtained from testing, seeking superior solutions by selecting and combining solutions of above average value. As the number of superior individuals in the population increases, the number of inferior individuals decreases. This thesis introduces Genetic Invariance, a similar family of generate and test problem solvers which uses a different selection and replacement strategy. In the best case, it achieves superior solutions without eliminating inferior characteristics. Although characteristics may initially be associated with inferior solutions, they may prove to be superior when combined with other particular characteristics. Mathematical analysis of lower bounds of Genetic Invariance on a simple function is given, and several properties of Genetic Invariance are explained using this analysis. A comparison and contrast is done to show how the two selection strategies achieve optimization in different ways. An analysis of the assumption and strategies of each system explains likely beneficial and detrimental effects of each system, while empirical analysis is given which demonstrates these effects. Together, they show each system's features and drawbacks.

Contents

List of Figures

List of Tables

1 Genetic Algorithms

1.1 Introduction

A Genetic Algorithm is an adaptive search strategy which employs selection of fitter individuals, similar to Darwinian evolutionary theory. In this approach, a population of individuals is chosen from the set of possible solutions. A number of (not necessarily distinct) individuals from the population are chosen based on their relative performance. These individuals are mated, and the children produced form the next generation. This selection and replacement strategy creates several interesting phenomena. Section 2 defines various terms used in the analysis of Genetic Algorithms. Section 3 explains the structure of the Genetic Algorithm, while section 4 explains the Schema Theorem, the fundamental theorem of Genetic Algorithms, and shows that it states that in an ideal case, a characteristic grows at an exponential rate based on its quality. While research has shown that this promotes superior characteristics, Genetic Algorithms also have their flaws. Section 5 explains various problems that have been noticed in applying the Schema Theorem. Section 6 summarizes various solutions that have been proposed and their effects on the Genetic Algorithm.

The remaining part of this thesis is organized as follows: Chapter 3 introduces Genetic Invariance. This generate-and-test problem solver is similar to Genetic Algorithms except that the selection and replacement strategy is different. Instead of selecting superior individuals to achieve superior characteristics in the population, Genetic Invariance simply mates pairs of closest value. A local separation is achieved, which produces a global separation, and thus global optimization. Analysis of Genetic Invariance on a restricted problem is given, which shows several things about the nature of Genetic Invariance.

Chapter 4 compares and contrasts Genetic Invariance and Genetic Algorithms. It is not the intent of this thesis to show that one is better than the other. Instead, their optimization methods will be compared, and their performance on several functions will be given. The strengths and weaknesses of each system will be explored, and an overview of the implementation details will be given.

1.2 Definitions

A Genetic Algorithm requires a *population* to operate. A population is a set of *individuals*, taken from the set of possible solutions. The population is denoted as $\mathcal{P}$ and $\mathcal{P}_i$ is an individual in $\mathcal{P}$. An individual can be thought of as a series of binary digits. More generally, $\mathcal{P}$ can be a series of characters, called *alleles*. Let the number of alleles in $\mathcal{P}_i$ be l, and the jth allele in individual i be $\mathcal{P}_{ij}, 1 \leq i \leq n, 1 \leq j \leq l$. *Parents* are individuals which are *mated* in some way to produce *children*. A new population, the next *generation*, is chosen from the original population plus the children. The original population, randomly chosen from the domain, is $\mathcal{P}^{(0)}$, and the population after t generations is $\mathcal{P}^{(t)}$. Each generation, the Genetic Algorithm generates $\mathcal{P}^{(t+1)}$ from $\mathcal{P}^{(t)}$.

With these terms defined, it is now possible to define a *schema* (plural: schemata). A schema is a subset of the domain. It can be represented as a vector of l symbols from the set $\{0, 1, \#\}$, much like an individual. The hyperplane is defined as the set of individuals which match the defined positions (0 or 1) in the schema. The # character matches both 0 and 1. Thus, the schema 01##11#0 matches both 01101100 and 01001110, but not 00010110. The notion of a schema is used in analysing the propogation of superior groups of values from generation to generation. Schemata have two characteristics. The *order* of a schema is the number of fixed (non #) symbols in the schema and the *defining length* is the length from the first to last fixed position. Thus, the schema ##01#10#010# has order 7 and defining length 9.

The Hamming distance between two individuals, $H_d(\mathcal{P}_i, \mathcal{P}_j)$ is the number of positions in which the two individuals differ in value. Thus, 010010 and 101001 have a Hamming distance of 5. We define the Hamming closure of 2 individuals $\mathcal{P}_i$ and $\mathcal{P}_j$, $H_c(\mathcal{P}_i, \mathcal{P}_j)$, to be the schema h with the smallest order that includes both $\mathcal{P}_i$ and $\mathcal{P}_j$:

$$h_k = \begin{cases} 0 & \text{if} \quad \mathcal{P}_{ik} = \mathcal{P}_{jk} = 0 \\ 1 & \text{if} \quad \mathcal{P}_{ik} = \mathcal{P}_{jk} = 1 \\ \# & \text{if} \quad \mathcal{P}_{ik} \neq \mathcal{P}_{jk} \end{cases}$$

We also define the Hamming size of this space, $H_s(\mathcal{P}_i, \mathcal{P}_j)$, to be the size of the set defined by h. Thus, 010010 and 101001 have a Hamming closure of ###0## and a Hamming size of 32.

Now that alleles, Hamming space, and schemata have been defined, it is possible to define *crossover*. Crossover is a method modeled after sexual genetic reproduction, which takes 2 individuals as parents and produces 2 children. Thus, crossover is a function $X : (\mathcal{P}_i, \mathcal{P}_j) \rightarrow (\mathcal{P}'_i, \mathcal{P}'_j)$. There are 2 types of crossover used in evolutionary systems: point crossover and uniform crossover. Point crossover requires a parameter, which is the number of points at which crossover occurs. This number is usually constant, and is usually kept at 1 or 2. An a point crossover divides the 2 individuals at a random points, and exchanges alternating groups. An example of 2 point crossover is given in table 1. The dashes indicate where crossover occurs.

$\mathcal{P}_i$	111 – 11000 – 1100
$\mathcal{P}_j$	101 – 01011 – 0010
$\mathcal{P}_i^{'}$	111 – 01011 – 1100
$\mathcal{P}_j^{'}$	101 – 11000 – 0010

Table 1: Point Crossover

Uniform crossover swaps each column with some probability, checking each column independently. Uniform crossover can be done by generating a bit string of length l, with a 0 indicating no swap and a 1 indicating a swap. Thus, 001101011100 would indicate that bits 3,4,6,8,9, and 10 are to be swapped between the parents to obtain the children. An example of uniform crossover with this string is given in table 2. The swap vector is denoted as x.

$\mathcal{P}_i$	110010111000
$\mathcal{P}_j$	101011101011
x	001101011100
$\mathcal{P}_i^{'}$	111011101000
$\mathcal{P}_j^{'}$	100010111011

Table 2: Uniform Crossover

Uniform crossover is used in this thesis because it is more general, and thus more powerful than point crossover in diverse problem solving. Uniform crossover can produce all children produceable by all possible point crossovers, but any particular a-point crossover cannot produce all children produceable by uniform crossover. This is easily understood: since each position in uniform crossover has some probability of being swapped or kept, it is possible to swap or keep any possible subset of the individuals. An a-point crossover must keep the first few elements, swap the next few, and so on.

This produces 2 restrictions: the number of groups that can be swapped or kept, and the location of each group that can be swapped or kept. Each group must be a contiguous set of bits. The generality of uniform crossover adds to the exploration power of the genetic algorithm. In tests conducted by Syswerda [Sys99], Uniform crossover performed better over a wide range of functions than 1 or 2 point crossover.

1.3 The Structure of the Genetic Algorithm

Genetic Algorithms have several characteristics.

1. Selecting parents is done by allocating each individual a probability of being selected equal to its value divided by the cumulative value of the population. This is known as *roulette wheel selection* [Hol73]. An example of roulette wheel selection is given in figure 2.

2. *Mutation*, randomly complementing values in the population, is used to introduce random alleles at a slow rate. Mutation can be implemented in various ways. The method used in this paper is: For each bit $\mathcal{P}_{ij}$ in $\mathcal{P}$, generate a random number between 0 and 1. Complement $\mathcal{P}_{ij}$ if the random number is less than the chosen mutation rate, u.

3. There is no widely accepted termination condition, although Genetic Algorithms usually terminate after a certain number of generations or after the population is made up of individuals which are similar enough that little useful work can be done. The characteristic of moving from a varying population to a uniform one is called *convergence*, and if an individual dominates the population after it has converged, it is said that the population has converged on this individual.

A diagram of the Genetic Algorithm is given in figure 1. An informal discussion of the characteristics of Genetic Algorithms can be found in [Gol89b].

1.4 The Schema Theorem

In 1973, Holland published a paper on the optimal allocation of trials to subsets of a space, based on the perceived relative values of those subsets. One theorem in [Hol73], expanded upon in [Hol75], became known as the *Schema Theorem*, the fundamental theorem of Genetic Algorithms. It stated

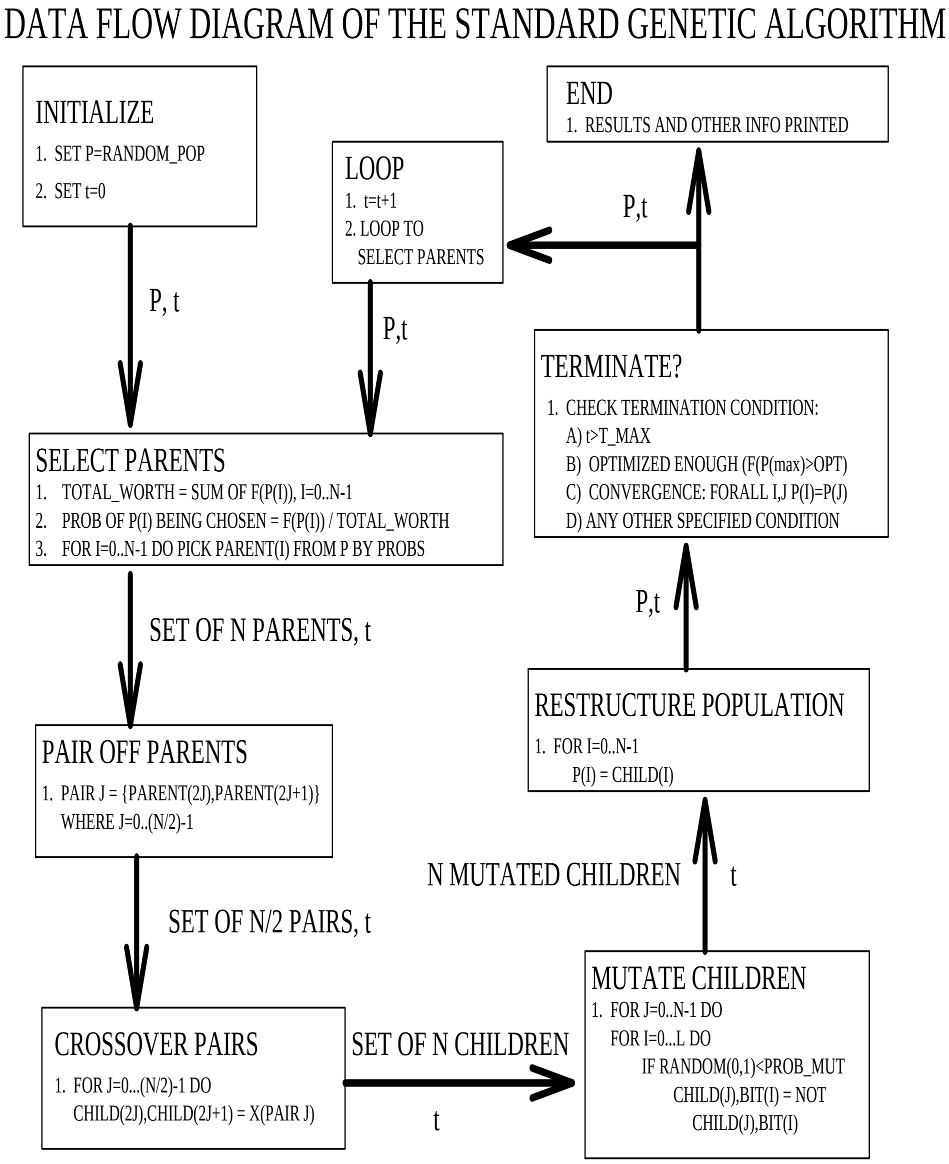
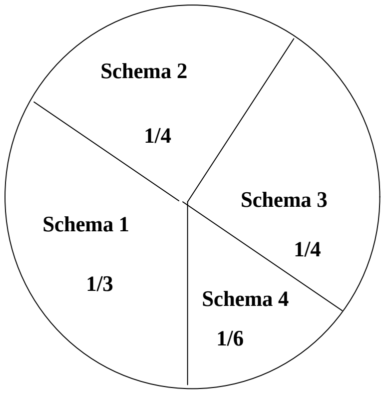


Figure 1: Structure of a Genetic Algorithm



Schema	Value
1	20
2	15
3	15
4	10
Total	60

Figure 2: Roulette Wheel Selection

that roulette wheel selection caused an exponential increase in the quantity of superior schemata.

Consider a schema s. The number of individuals in $\mathcal{P}^{(t)}$ which are members of s is defined as $m_s(t)$. Let the population be of near infinite size, and maintain a relatively constant average value, $\bar{\mu}$. Also, assume that the schema s is likely to be unaffected by crossover. Thus, $m_s(t+1) = \sum_{\mathcal{P}_i^{(t)} \in s} \left(\frac{f(\mathcal{P}_i^{(t)})}{\bar{\mu}} \right)$. In a very large population, the average values of the individuals in s will approach the average value of schema s. Let the average value of s be $\mu(s)$. Thus, $m_s(t+1) = \sum_{\mathcal{P}_i^{(t)} \in s} \frac{\mu(s)}{\bar{\mu}}$. Since the number of individuals of schema s in $\mathcal{P}^{(t)}$ is $m_s(t)$, $m_s(t+1) = m_s(t)\frac{\mu(s)}{\bar{\mu}}$. Thus, an exponential increase in superior schemata results from assigning each individual a probability of mating equal to its value.

This theorem makes several unrealistic assumptions. The population is of finite size and will not maintain a constant average value. In fact, the increase of the number of superior schemata raises the overall value of the population. Also, schemata *perceived* to be superior will increase. The schemata may not necessarily be superior; all that is required is that the samples show them to be superior. A schema of high order may not get an accurate sampling. If the population is not large enough, schemata of high order may not even be represented in the population. In [Hol75], Holland states that if schemata have a high variance compared to the difference in their average values, sampling errors can cause the wrong schema to be preferred. Assuming that a schema does increase exponentially, it must survive from generation to generation. Certain crossover schemes, such as uniform crossover, tend to cause many alleles to swap. Thus, even if a schema is of high value, it may disappear during crossover. Conversely, schemata can appear from the crossover of similar parents. Grefenstette and Baker [GB89] discussed these and other difficulties with the Schema Theorem.

1.5 Problems with Proportional Allocation

Although allocating probabilities of mating based on the value of each individual relative to the total worth of the population does result in the increase of superior schemata, there are several problems that can occur due to this selection system.

If the superior individuals in $\mathcal{P}$ do not represent characteristics present in the optimal value, the Genetic Algorithm will be led away from the optimal value. This property is called *deception*. Bethke [Bet80] and Goldberg [Gol88, Gol89a], used Walsh functions to analyse the deceptiveness of a function. In this analysis, average values of low order schemata in a function are examined. The accuracy with which low order schemata predict the value of higher order schemata determines the deceptiveness of the function. In particular, if alleles are highly *epistatic*, that is, the value of the function depends greatly on patterns among many alleles instead of the values of single alleles, the function may be very difficult for a Genetic Algorithm to solve. Note that some epistatic functions, such as $f(x) = x^2$, are quite predictable and thus are still easily solved.

Another problem Genetic Algorithms have is premature convergence. When the population converges, little further exploration can be done, since the genetic algorithm will waste much time generating individuals which have already been evaluated. Mutation corrects this by introducing some randomness into the population. This randomness will be kept in future generations if it is of high worth, while it will be discarded if it causes the value of the individual to deteriorate.

Interference from mutation can undermine the effectiveness of the selection system. If the difference between average and superior individuals is small, mutation may change optimized alleles into random alleles at a faster rate than roulette-wheel selection changes random alleles into optimal alleles. DeJong [DeJ75] pointed out that a common practice in Genetic Algorithms research was to increase the mutation rate when the Genetic Algorithm did not converge to a very high valued individual. This led to the Genetic Algorithm being stifled by a high mutation rate.

If the ratio between superior and average schemata is low, an effect called *genetic drift* will occur. DeJong [DeJ75] stated that even in a population where no particular schema is preferred, the population will increasingly deviate from the norm, until the population converges. Goldberg and Segrest [GS87] calculated the time to convergence of a simple 1-bit population (each individual can be 0 or 1) of size n. The estimated time to convergence (in number of generations) is $\frac{(2r-1)^2}{4r(1-r)}n$. Although this case is simplistic, it does indicate what happens in a large population, if there is no significant difference between average and superior schemata for many generations. The population starts to converge to an arbitrary individual. This causes a loss of schemata from the original population, and thus schemata which have a higher value later on may not be creatable from the new population. A further complication is that an intuitive solution, increasing the mutation rate in an attempt to re-introduce lost alleles, does not work. DeJong [DeJ75] states that this usually does not help the Genetic Algorithm much, and adds the problem of interference from mutation to genetic drift.

1.6 Modifications to the Genetic Algorithm

As the problems with Genetic Algorithms were explored, modifications designed to correct these faults emerged. This section explores the ways in which Sharing, Crowding, Elitism, Steady-state Genetic Algorithms, and Parallel Genetic Algorithms were designed to correct problems with the genetic algorithm.

Sharing was created to allow Genetic Algorithms to explore many possible peaks, rather than a single peak. Goldberg and Richardson [GR87] designed sharing to de-emphasize having copies of one superior schema. Instead, copies would be kept of many varying relatively superior schema. The sharing function must have the following characteristics:

- $sh(\mathcal{P}_i, \mathcal{P}_i) = 1$
- $sh(\mathcal{P}_i, \mathcal{P}_j) \rightarrow 0$ as $H_d(\mathcal{P}_i, \mathcal{P}_j) \rightarrow \infty$
- $\forall \mathcal{P}_i, \mathcal{P}_j$, $0 \leq sh(\mathcal{P}_i, \mathcal{P}_j) \leq 1$

Goldberg and Richardson suggested an exponential sharing function

$$sh(\mathcal{P}_i, \mathcal{P}_j) = \begin{cases} 1 - \left(\frac{H_d(\mathcal{P}_i, \mathcal{P}_j}{\text{share}_{\text{max}}}\right)^{\alpha} & \text{if} \quad H_d(\mathcal{P}_i, \mathcal{P}_j) < \text{share}_{\text{max}} \\ 0 & otherwise \end{cases}$$

where $share_{max}$ was the Hamming distance at which the sharing value, $sh(\mathcal{P}_i, \mathcal{P}_j)$, dropped to 0, and α was a positive constant. The value of an individual, $f_{share}(\mathcal{P}_i)$ is $\frac{f(\mathcal{P}_i)}{\sum_{j=1}^{n} sh(\mathcal{P}_i, \mathcal{P}_j)}$. This way, an individual's value is reduced by any individuals with a Hamming distance of α. Thus, the number of copies of a schemata present in the population will be proportional to its relative value.

DeJong [DeJ75] suggested *crowding* as a way to slow down the convergence rate. In this model, the population was not the set of children produced from the previous population. Instead, each child replaces one individual in the population, with a higher probability of replacing an individual containing similar alleles. DeJong found that this was most effective when combined with a *generation gap*, G, $0 \leq G \leq 1$. The generation gap G indicates what fraction of the population is replaced each generation. Thus, each generation Gn children are generated, and replace individuals in $\mathcal{P}^{(t)}$ to form $\mathcal{P}^{(t+1)}$. DeJong stated that a small generation gap and small amounts of crowding caused the Genetic Algorithm to perform better.

Note that crowding is similar to sharing, but works on the restructuring process rather than the selection process. Both accomplish the same goal: the reduction of duplicate genetic material and thus the diversification of the population, but operate at different times and in different ways.

Elitism, keeping the best individual seen, is a simple and practical heuristic to improve the performance of Genetic Algorithms. In his thesis [DeJ75], DeJong commented that elitism is usually beneficial to a Genetic Algorithm. Elitism maintains the most superior individual in the population, and thus maintains the most superior schemata. Thus, it is natural that this method improves the performance of a Genetic Algorithm. In this thesis, Elitism is implemented by replacing the minimal element in the population with the maximal element seen. While allowing the best element to be kept, this method also increases the rate at which the Genetic Algorithm converges.

A Steady State Genetic Algorithm is a genetic algorithm which only replaces a constant number of individuals in the population during each generation. As defined in [Sys99], a Steady State Genetic Algorithm has replaced kt individuals after t generations, while the Genetic Algorithm has replaced nt. The Steady State Genetic Algorithm is similar to the simple genetic algorithm, including the use of roulette wheel selection, but the population is restructured by deleting population members. One member must be deleted for each child produced, and thus the size of the population remains constant. The individuals to be deleted are probabilistically selected based on worth. The lower an individual's worth, the more likely it is to be deleted. Syswerda tested Steady State Genetic Algorithms [Sys91] and stated that "at least for some problems, steady state genetic algorithms do find as good or better solutions in much less time" than simple genetic algorithms. In the steady state approach, a schema of excellent fitness is immediately available for use, while a simple genetic algorithm must wait until the next generation to take advantage of superior schemata.

A Parallel Genetic Algorithm is an algorithm which performs standard Genetic Algorithm selection and mating on various *subpopulations* to achieve global optimization in the entire population. In [Müh91], Mühlenbein described a Parallel Genetic Algorithm. In this model, a genetic algorithm is run on each subpopulation, which produces some superior schemata. Individuals have a small probability of migrating between subpopulations. When this occurs, the schemata of one subpopulation are introduced into the other subpopulation. When superior schemata from a subpopulation are introduced into another subpopulation, they dominate the population if they are superior to the schemata in that subpopulation. In this way, superior genetic material propogates from one subpopulation to all subpopulations. Eventually, all of the subpopulations contain copies of all superior schemata, which are combined into one superior individual. Thus, local optimization of low order superior schemata plus a propagation of individuals results in a global optimization of high order superior schemata.

1.7 Conclusion

A Genetic Algorithm is an adaptive search algorithm based on the Schema Theorem. It uses roulette wheel selection to simulate Darwinian evolution in the population. This is a result of the Schema Theorem, the fundamental theorem of Genetic Algorithms. Function optimization has been used to test the performance of Genetic Algorithms. Although Genetic Algorithms work well on many of the functions that they were tested on, there have been a number of problems with optimizing some functions. Deception, premature convergence, interference from mutation, and genetic drift are all possible when optimizing with a Genetic Algorithm. Deception and premature convergence cause lower optimal values to be found, while interference from mutation and genetic drift actually cause the Genetic Algorithm to perform little useful work whatsoever. Several solutions have been proposed, including Sharing, Crowding, Elitism, Steady-State Genetic Algorithms, and Parallel Genetic Algorithms. Sharing and Crowding reduce the amount of duplication of schemata in the population, elitism keeps the most superior individual (and thus, the most superior schemata) in the population, Steady-State Genetic Algorithms allow for improvements in schemata to be taken advantage of immediately, while Parallel Genetic Algorithms use the convergence of subpopulations to superior schemata of low order and the propogation of individuals between subpopulations to achieve global optimization of long order superior schemata.

2 Genetic Invariance

2.1 Introduction

This chapter introduces a new adaptive problem solver, Genetic Invariance, which has at least two changes from Genetic Algorithms. First, invariance is enforced by replacing parents by their children. In this manner, the alleles in any column of the population never vary; they may be exchanged between individuals but may not be added or deleted. Second, the selection system involves choosing the closest valued pair for mating. This causes a local change of value, which in turn causes a global change of value. This method is of particular interest to us because it does not specifically select for optimal value. Instead, side effects of the mating process lead to overall optimization. Section 2 shows the structure of Genetic Invariance in more detail. Section 3 shows how Genetic Invariance operates by proving lower bounds on a restricted case of a simple problem. Section 4 elaborates on the results in section 3, and show how they can be extended to general function optimization using Genetic Invariance.

2.2 The Structure of Genetic Invariance

Like Genetic Algorithms, Genetic Invariance uses a population $\mathcal{P}$ of n individuals, each having l alleles. Unlike Genetic Algorithms, the population is ranked, with $\mathcal{P}_1^{(t)}$ being the individual in $\mathcal{P}^{(t)}$ with the lowest f value and $\mathcal{P}_n^{(t)}$ being the individual with the highest f value. Individuals with the same value are arbitrarily ranked. $\mathcal{P}$ can be thought of as an n by l matrix of bits, $\mathcal{P}_{ik}$.

The main steps of the algorithm are:

1. Randomly select an initial population $\mathcal{P}^{(0)}$. Set $t = 0$.

2. Select two individuals, $\mathcal{P}_i, \mathcal{P}_j, i < j$, such that the difference in their function values, $f^-(\mathcal{P}_i, \mathcal{P}_j)$, is minimal over all pairs. If two have equal differences, choose the pair in which j is maximized. If two have equal differences and identical maximal individuals, choose the pair in which i is maximized.

3. Mate these two with a crossover, and insert them back into the population. Although any crossover can be used, it has been stated that we will use uniform crossover throughout this paper.

4. Increment t. If the termination condition is not met, goto step 2.

Genetic Invariance is diagrammed in figure 3.

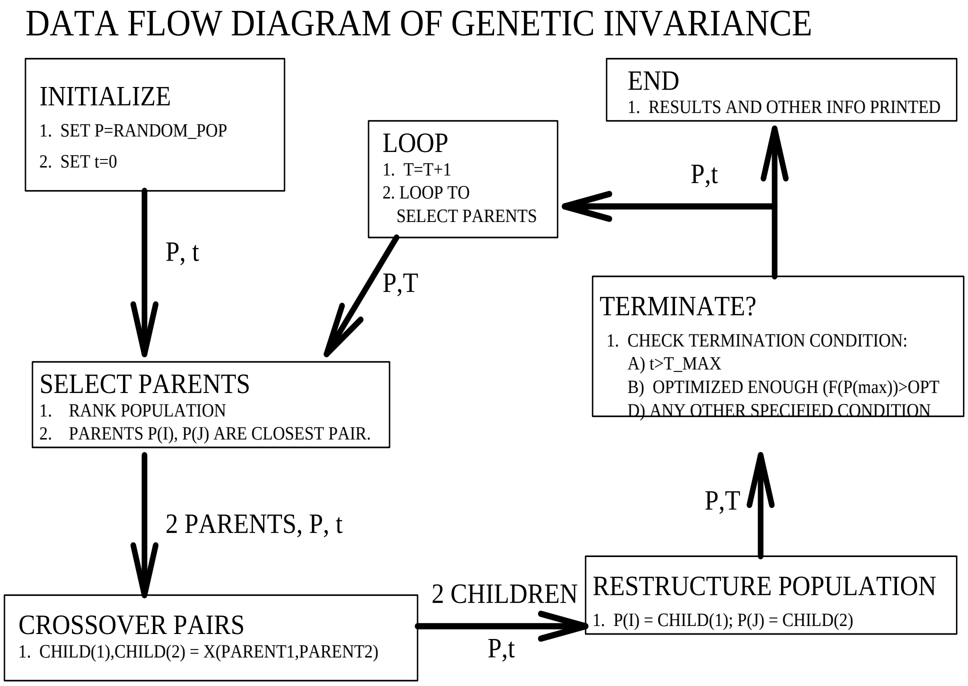


Figure 3: Structure of Genetic Invariance

Notice that Genetic invariance does not use mutation, and always replaces the parents with their children. Because of this, the alleles in any column do not change; only their positions within each column change. This is known as invariance. A formal definition of invariance is:

$$\forall k, t_1, t_2, \sum_{i=1}^{n} \mathcal{P}_{ik}^{(t_1)} = \sum_{i=1}^{n} \mathcal{P}_{ik}^{(t_2)}.$$

We defined a *mating cycle* to be parents repeatedly producing offspring which in turn become the parents in the next generation. Thus, $\mathcal{P}_i^{(t)}$ and $\mathcal{P}_j^{(t)}$ mate to produce $\mathcal{P}_i^{(t+1)}$ and $\mathcal{P}_j^{(t+1)}$, which in turn mate to produce $\mathcal{P}_i^{(t+2)}, \mathcal{P}_j^{(t+2)}$, and so on for some period of time. Each cycle has a definite starting point, namely the first time $\mathcal{P}_i^{(t)}$ and $\mathcal{P}_j^{(t)}$ mate to produce $\mathcal{P}_i^{(t+1)}$

and $\mathcal{P}_j^{(t+1)}$, but it may not have an endpoint. If the current mating cycle is of infinite length, we say that Genetic Invariance has *stagnated*. Ideally Genetic Invariance should run until stagnation. However, since this is not generally detectable, Genetic Invariance, like the Genetic Algorithm, is run for some pre-determined period of time.

2.3 Mathematical Analysis

2.3.1 Introduction

This section gives a worst case performance analysis of Genetic Invariance on a simple function. Because of the modifications to the selection and restructuring systems, the Schema Theorem of Genetic Algorithms does not apply. The analysis proves lower bounds on a restricted case of the function $f_0(x)$, the number of 1s in the binary string x. The bounds are the least possible maximum when stagnation occurs, which we call the LMS point. The restrictions on the initial population will be lessened, and heuristics will be added which increase the LMS point. This will lead to a discussion of the nature of Genetic Invariance in the next section.

2.3.2 A Special Case

Consider a very special population $\mathcal{P}$ and evaluation f, where $f(p)$ is function f_0, the number of 1s in p. The initial population is chosen randomly under the constraint that the number of 1s in any column of $\mathcal{P}_{i,j}$ is exactly one. We assume that $n > 2$ and $l > 0$ since two individuals stagnate by definition (producing a random search over their Hamming Closure) and an individual size of zero is meaningless.

We define $T(x)$ to be the xth triangular number, $\frac{x(x+1)}{2}$; $\triangle(y)$ to be the smallest integer x such that $T(x) \geq y$; and the f sum of two individuals $f^+(\mathcal{P}_x, \mathcal{P}_y)$ to be $f(\mathcal{P}_x) + f(\mathcal{P}_y)$.

What is the least maximum under stagnation, $\mathrm{LMS}(n, l)$, under these conditions?

It is obvious from the definitions of mating, selection, and ranks that:

Lemma 2.1 *If $\mathcal{P}_i$ and $\mathcal{P}_j$ mate then $j = i + 1$.*

Note that the lemma 2.1 implies that stagnation can only occur on adjacent individuals, since the stagnation pair is a mating pair. Later, we will introduce constraints to the selection strategy which will change this.

Lemma 2.2 *Let $f^+(\mathcal{P}_i, \mathcal{P}_{i+1}) beaconstant \delta$. If stagnation on $\mathcal{P}_i, \mathcal{P}_{i+1}$ has occurred, then there exists with probability 1 some time t such that $f(\mathcal{P}_{i+1}^{(t)}) = \delta$ and $f(\mathcal{P}_i^{(t)}) = 0$.*

Proof: Consider 2 strings a and b such that a contains all of the 1s in $H_c(\mathcal{P}_i^{(t)}, \mathcal{P}_{i+1}^{(t)})$. Since a is within the Hamming Closure of $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$, the probability is 1 that $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ will produce a as one of their descendants. The complementary child must, by definition, be the string of zeroes. ∎

Lemma 2.3 *If Genetic Invariance stagnates on $\mathcal{P}_i, \mathcal{P}_{i+1}$, then either $i = 1$ or $f(\mathcal{P}_i) = f(\mathcal{P}_{i+1}) = 0$.*

Proof: Assume $i > 1$. $0 = f(\mathcal{P}_i) < f(\mathcal{P}_{i+1})$ by lemma 2.2.
$f(\mathcal{P}_0) \leq f(\mathcal{P}_i) = 0$ because we are ranking our individuals.
$f(\mathcal{P}_0) \geq 0$ by definition of f, hence $f(\mathcal{P}_0) = 0$.
$f^-(\mathcal{P}_i, \mathcal{P}_0) = 0$, but since we are mating the closest pair of highest rank, $f^-(\mathcal{P}_{i+1}, \mathcal{P} i) = 0$, which contradicts $0 = f(\mathcal{P}_i) < f(\mathcal{P}_{i+1})$.
Therefore, $i \leq 1$ or $f(\mathcal{P}_i) \geq f(\mathcal{P}_{i+1})$, which reduces to $i = 1$ or $f(\mathcal{P}_i) = f(\mathcal{P}_{i+1}) = 0$ by definition of f and i. ∎

It is interesting to note that in the case $f(\mathcal{P}_i) = f(\mathcal{P}_{i+1}) = 0$, the lower zeroes, $\mathcal{P}_k, k < i$ will have a 0 probability of *ever* being chosen as mates. Thus, Genetic Invariance is said to *implicitly eliminate* zero-value individuals both in this case and in general when f has 0 epistasis.

Consider lemma 2.3. In terms of the evaluation function, we may write

Lemma 2.4 *If Genetic Invariance stagnates on $\mathcal{P}_i, \mathcal{P}_{i+1}$ then*

$$\forall j \neq i, f^+(\mathcal{P}_i, \mathcal{P}_{i+1}) < f^-(\mathcal{P}_{j+1}, \mathcal{P}_j). \textit{ or } f(\mathcal{P}_i) = f(\mathcal{P}_{i+1}) = 0$$

Proof: Consider $i = 1$. The algorithm mates the highest ranked pair if two or more pairs have the same f differences, thus $\forall j > i, f^-(\mathcal{P}_j, \mathcal{P}_{j+1}) > f^-(\mathcal{P}_i, \mathcal{P}_{i+1})$. But consider lemma 2.2. $\mathcal{P}_2$ will eventually be $f^+(\mathcal{P}_i, \mathcal{P}_{i+1})$, and $\mathcal{P}_1$ will be 0. Thus, $f^+(\mathcal{P}_i, \mathcal{P}_{i+1}) < f^-(\mathcal{P}_{j+1}, \mathcal{P}_j)$. By lemma 2.3, we know that for the stagnant pair, either $i = 1$ or $f(\mathcal{P}_i) = f(\mathcal{P}_{i+1}) = 0$, proving the lemma. ∎

Lemma 2.5 *$LMS(n, l) = \triangle(l)$ is either $n \geq \triangle(l) + 2$ if n is a triangular number, or $n \geq \triangle(l) + 1$ if it is not.*

Proof: l is nonzero, therefore some individuals must have nonzero values. Let the stagnation individuals have values 0 and $\epsilon - 1$, where $\epsilon - 1 = f(\mathcal{P}_i) + f(\mathcal{P}_{i+1})$ (by lemma 2.2). The individuals must have values equal to or greater than $0, 0, ..., 0, \epsilon - 1, 2\epsilon - 1, 3\epsilon - 1, ..., k\epsilon - 1$. The minimum values possible are achieved when $\epsilon = 1$, giving the sequence $0, 0, ..., 0, 0, 1, 2, 3, 4, ..., \triangle(l)$. This sequence, which has at least two leading 0s, sums to l if l is a triangular number. Thus, if l is triangular, and $n \geq k + 2 = \triangle(l) + 1$, this triangle can be constructed. If it is not, eliminate the integer $T(\triangle(l)) - l$ from the sequence to achieve the correct sum. This requires $n \geq k + 2 - 1 = k - 1 = \triangle(l) + 1$, since the triangle is missing one individual (the one that was deleted). ■

Lemma 2.6 *If* $3 < n < \triangle(l) + 2$ *then* $LMS(n, l) = (n - 2) + \lceil \frac{l - T(n-2)}{n-2} \rceil$.

Proof: Consider an arithmetic triangle on the highest $n - 2$ positions. This accounts for $T(n - 2)$ bits, leaving us with (l-$T(n$-$2)$) bits to position. The condition for stagnation is $\forall j, f^-(P_{j+1}, \mathcal{P}_j) > f^+(\mathcal{P}_{i+1}, \mathcal{P}_i)$, but although this implies an arithmetic triangle is the least possible distribution, each individual may have any value, provided that any pair differs by at least 1. Consider adding a value δ to some $\mathcal{P}_j$. This means that δ must be added to all $\mathcal{P}_k, k \geq j$. But how do we distribute the other alleles among the population for a lowest possible bound? Considering the above statements, we can only do this by evenly distributing the value among all of the $n - 2$ individuals above $\mathcal{P}_2$.

Note that we cannot add to $\mathcal{P}_i$ nor to $\mathcal{P}_{i+1}$, because to do so would necessitate that the value of ϵ be increased by 1 throughout the population. Since $\frac{T(n-1)}{n-2} \leq (n - 1)$ when $n > 3$, the worst case occurs when each of the nonzero individuals are increased equally, with any partial increase being added from the top down, resulting in a worst case LMS point of $(n - 2) + \lceil \frac{l - T(n-2)}{n-2} \rceil$. ■

Lemma 2.7 *If* $n = 3$, *then* $LMS(n, l) = l - \lfloor \frac{l-1}{3} \rfloor$.

Proof: In this case $\frac{T(n-1)}{n-2} > (n - 1)$, so an arithmetic triangle distribution over the population is worse than distributing the remaining value over the $n - 2 = 1$ remaining individuals. The lowest stable state is $f(\mathcal{P}_3) - f(\mathcal{P}_2) > f(\mathcal{P}_2) - f(\mathcal{P}_1) = f(\mathcal{P}_2)$ and $f(\mathcal{P}_1) + f(\mathcal{P}_2) + f(\mathcal{P}_3) = f(\mathcal{P}_2) + f(\mathcal{P}_3) = l$, which reduces to $f(\mathcal{P}_3) > \frac{2l}{3}$. Note that this is a strict inequality, so when l is divisible by 3, we must add 1 to the MAX value. Thus, LMS$(n,l) = l - \lfloor\frac{l-1}{3}\rfloor$. ■

THEOREM **2.1** *The least maximum under stagnation, $LMS(n,l)$, is*

$(n-2) + \frac{(l-T(n-2))}{(n-2)}$	*if* $3 < n < \triangle(l) + m$
$\triangle(l)$	*if* $n \geq \triangle(l) + m$ *and* $n > 3$
$l - \lfloor\frac{l-1}{3}\rfloor$	*if* $n = 3$

where $m = 2$ if l is a triangular number, 1 if it is not.

Proof: Theorem 2.1 is proven by lemmas 2.6, and 2.7. ■

Thus, an arithmetic triangle is formed, which results in an LMS value proportional to $\triangle(l)$.

2.3.3 Improving the Algorithm

Consider two individuals $\mathcal{P}_x$ and $\mathcal{P}_y$. If their Hamming Distance is less than 2, then crossover produces individuals $\mathcal{P}_x$ and $\mathcal{P}_y$, since these are the only individuals in $H_c(\mathcal{P}_x, \mathcal{P}_y)$. Since mating two individuals with a Hamming distance of 0 or 1 will cause stagnation, and so is unprofitable in the Gene Invariance approach, we add the heuristic of never mating individuals with Hamming Distances of less than 2. How does the new LMS point, LMS2(n,l), differ from the previous one?

Lemma 2.8 *The heuristic has no effect on the choice of parents if both selected parents have positive values under f.*

Proof: Let $\mathcal{P}_x$ and $\mathcal{P}_y$ be two individuals with positive values under f. By the definition of f, each must have at least one 1 in it. But these 1s cannot be in the same position (since there is only one 1 per column), thus at least two bit positions in $\mathcal{P}_x$ and $\mathcal{P}_y$ are not identical. ■

Lemma 2.9 *If $n = 3$, $LMS2(n,l)$ is*

$LMS(3,l)$	*if* $l > 6$
l	*if* $l \leq 6$

Proof: All of the claims of lemma 2.7 hold, except that in 2.7 the bottom two individuals may have a Hamming distance of less than 2. When does this occur? When $\lfloor \frac{l-1}{3} \rfloor \leq 1$, i.e. $l \leq 6$. Thus, $l > 6 \Longrightarrow \mathrm{LMS2}(3,l) = \mathrm{LMS}(3,l)$.

Now consider $l \leq 3$. By enumeration of the possible states it is provable that $\mathrm{LMS2}(3,l) = l$.

Consider $4 \leq l \leq 6$. By lemma 2.7, it is known that we must satisfy the same stagnation conditions as before, with the exception that if the bottom two individuals have Hamming Distance 1, they are not mated. The stagnation distribution is $(0, 1, l-1)$. But the lower two will not be able to mate now that they have Hamming distance 1, so the upper two will mate, causing the GA to stagnate. Since we assume the uppermost value to be maximized at stagnation, Genetic Invariance is said to stagnate at the value l. ■

Lemma 2.10 *If* $n > 3$*,* $LMS2(n,l)$ *is*

$(3n-4) + \lceil \frac{l-(3T(n-1)-n)}{(n-2)} \rceil$	*if* $3 < n < \triangle(\frac{l-n}{3}) + 1$
$3\triangle(\frac{l}{3}) - 1$	*if* $n > 3, n \geq \triangle(\frac{l-n}{3}) + 1$

Proof: Genetic Invariance cannot stagnate at $(f(\mathcal{P}_i), f(\mathcal{P}_{i+1})) = (0,0)$ or $(0,1)$ since $H_d(P_i, P_{i+1}) < 2$. Thus, consider again lemma 2.5, with ϵ having a minimum value of 2. This will produce an arithmetic sequence $(0, 2, 5, 8, 11, ..., 3m-1)$, which is the minimum for stagnation. The sum is $3T(m-1) - m$; the highest individual is $3(m-1) - 1 = 3m-4$. $m = \lceil \frac{\triangle(l+m)}{3} \rceil$, and so

$$m = \begin{cases} \triangle(\frac{l}{3}) + 1 & \text{if } 3T(\triangle(\frac{l}{3})) - l > \triangle(\lceil \frac{l}{3} \rceil) \\ \triangle(\lceil \frac{l}{3} \rceil) & \text{otherwise} \end{cases}$$

which is lower bounded by $\triangle(\lceil \frac{l}{3} \rceil)$. As in 2.6, the LMS2 bound is calculated by distributing any remaining values evenly over the triangle. ■

THEOREM **2.2** *Thus,* $LMS2(n,l)$ *is*

$(3n-4) + \frac{l-(3T(n-1)-n)}{(n-2)}$	*if* $3 < n < \triangle(\frac{l-n}{3}) + 1$
$3\triangle(\frac{l}{3}) - 1$	*if* $n > 3, n \geq \triangle(\frac{l-n}{3}) + 1$
$l - \lfloor \frac{l-1}{3} \rfloor$	*if* $n = 3$ *and* $l > 6$
l	*if* $n = 3$ *and* $l \leq 6$

Proof: By lemmas 2.8, 2.9, and 2.10 we can conclude theorem 2.2. ∎

Thus, by not mating any pair with Hamming distance less than 2, we have increased the slope of the arithmetic triangle from 1 to 3, thus increasing the LMS value from $\triangle(l)$ to $3\triangle(\frac{l}{3})$.

2.3.4 Extending the Special Case

In the previous chapters, we assumed an initial population where there was only one 1 per column. Consider an initial population $\mathcal{P}$ with a total of b ones that are distributed almost evenly between the l columns of $\mathcal{P}$ (ie. either $\lfloor \frac{b}{l} \rfloor$ or $\lceil \frac{b}{l} \rceil$. Let d be the number of columns that have at least 2 bits set to 1. Also, assume that we are *not* implementing the feature which mates individuals only if they have a Hamming Distance of 2 or more. What is the arbitrary-b LMS point LMS(b, n, l), in comparison to LMS(n, l)?

Lemma 2.11 *When $b \leq l$, $LMS(b, n, l) = LMS(n, b)$.*

Proof: Consider $b < l$. Some columns will be all zeroes, and others will only have one 1 per column. This is directly mappable to the case where $l = b$, since the extra zeroes do not affect the method in any way. Thus, LMS(b, n, l) =LMS(n, b).

LMS(n, l) is defined to be LMS(b, n, l), since in the original case there are l 1s in the population, one per column. ∎

Lemma 2.12 *Consider $n = 3, b > l$. Then, for any mating pair, the maximal individual will have a value of at least d, d defined above.*

Proof: Consider the columns which contain the replicated 1s. Given $n = 3$, there are 3 individuals, at least 2 of which contain a 1. Therefore for any mating pair, at least one of them will have a 1 in that column, therefore the pair can produce an individual which has a 1 in each of the d columns. Therefore at stagnation the maximal split insures $\mathcal{P}_n^{(t)} \geq d$. ∎

Lemma 2.13 *If $l < b$ and $n = 3$, then $LMS(b, n, l)$ is*

$l - \lfloor \frac{b}{3} \rfloor$	*if* $l < b < \frac{3l}{2}$
$b - l$	*if* $\frac{3l}{2} \leq b < 2l$
l	*if* $b \geq 2l$

Proof: First of all, consider the case $b \geq 2l$. By lemma 2.12, any mating pair can mate so that the higher individual will have a value of l. Thus, by our definition of stagnation, when they stagnate the maximum value reached will be l.

Next, consider $\frac{3l}{2} \leq b < 2l$. By lemma 2.12, any mating pair can produce a value of $b - l$. Genetic Invariance can stagnate at exactly this value, on the individuals with values: $(2l - b, b - l, b - l)$. The upper individuals can be identical, and since we give priority to upper individuals upon mating, Genetic Invariance will stagnate on them.

Now consider $l < b < \frac{3l}{2}$. First we show that stagnation will not occur when mating the top pair. Assume stagnation occurs on the top pair. Thus, the population will be:

Element	individual 1s	replicate 1s
$\mathcal{P}_3$	i	$b-l$
$\mathcal{P}_2$	0	x
$\mathcal{P}_1$	$2l-b-i$	$b-l-x$

where "individual 1s" refers to 1s in columns with one 1 in them, and "replicate 1s" refers to 1s in columns with more than one 1 in them. Note that the position of these columns in the individual is irrelevant; the analysis only uses the number of 1s in these columns. This matrix of individuals is called a *stagnation matrix*. At stagnation, it is assumed that the top individuals has the maximal value (maximum split), so all of the duplicate 1s in the top pair will move to the top individual, and all of the non-duplicate 1s in the top pair will likewise be in the top individual. i and x are variables in this example, and can take on any (feasible) value. Claim: this situation is contradictory. Proof: By definition of stagnation: $i + b - l - x \leq x - (2l - b - i + b - l - x)$ which reduces to $b \leq 3x$, i.e. $-x \leq -\frac{b}{3}$. Since the number of 1s can never be less than zero in any part of this, $0 \leq b - l - x \leq b - l - \frac{b}{3} = \frac{2b}{3} - l$. Since $\frac{2b}{3} < l$ by the initial statement $b < \frac{3l}{2}$, $\frac{2b}{3} - l < l - l = 0$, thus $0 < 0$.

Consider the case $l < b < \frac{3l}{2}$ where stagnation occurs on the bottom. The lower-pair stagnation matrix becomes:

Element	individual 1s	replicate 1s
$\mathcal{P}_3$	i	x
$\mathcal{P}_2$	$2l-b-i$	$b-l$
$\mathcal{P}_1$	0	$b-l-x$

Thus $i + x - (2l - b - i + b - l) \geq 2l - b - i + b - l - (b - l - x)$, and so $b + 3i \geq 3l$, i.e. $i \geq l - \frac{b}{3}$. Consider $i = \lfloor l - \frac{b}{3} \rfloor$; $x = 0$. Substituting into the above array gives the least stagnation point, $l - \lfloor \frac{b}{3} \rfloor$. ■

Lemma 2.14 *If $b > l$ and $n > 3$, $LMS(b, n, l)$ is*

$\lceil \frac{b-1}{n} \rceil$	*if* $b - l \geq \lceil \frac{b-1}{n} \rceil$
$b - l + LMS(n, b - n(b - l))$	*if* $b - l < \lceil \frac{b-1}{n} \rceil$

Proof: Let $\delta = b - l$ and $\tau = \lceil \frac{b-1}{n} \rceil$. Consider $\delta \geq \tau$. It is possible to distribute the bits in the following manner: $(P_1, \mathcal{P}_2, ..., \mathcal{P}_n) = (\lfloor \frac{b}{n} \rfloor, ..., \lfloor \frac{b}{n} \rfloor, \lceil \frac{b}{n} \rceil, ..., \lceil \frac{b}{n} \rceil)$. It is possible for Genetic Invariance to stagnate if the first two identical-valued individuals (individuals whose f difference is 0) are identical. In order for this to occur, the total number of duplicate bits (δ) must be greater than or equal to the value of either individual (their values are the same). Consider the breakpoint where values of $\lfloor \frac{b-1}{n} \rfloor$ change to $\lceil \frac{b-1}{n} \rceil$. If ($b$ modulo n) = 1, there is only one top individual, otherwise there is more than 1. Thus when (b modulo n) = 1, stagnation can occur at this point if $\delta \geq \lfloor \frac{b}{n} \rfloor$, otherwise δ must be at least $\lceil \frac{b}{n} \rceil$. If $\delta \geq \tau$, this case holds, and the maximum under stagnation in both cases is τ.

Now consider $\delta < \tau$. Consider the stagnation pair $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ to have all of the duplicates. This is clearly the worst case, since at stagnation the concern is with the separation they can achieve. All individuals can now have a value of $b - l$ without interfering with $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ mating, except that all $P_j, j > i$ must differ by at least 1. Thus, the worst case is constructed by letting all of the individuals have the value $b - l$, followed by building an arithmetic triangle from $\mathcal{P}_{i+2}$ to $\mathcal{P}_n$, inclusive. If the triangle grows to fill the entire population, except the two bottom individuals, distribute the remaining value evenly over the top $n - 2$ individuals. In this case, the number of duplicates does not affect the maximum value. ■

THEOREM **2.3** *The arbitrary-b LMS point , $LMS(b,n,l)$, is*

$LMS(n,b)$	*if* $b \leq l$
$l - \lfloor \frac{b}{3} \rfloor$	*if* $b > l$ *and* $n = 3$ *and* $l < b < \frac{3l}{2}$
$b - l$	*if* $b > l$ *and* $n = 3$ *and* $\frac{3l}{2} \leq b < 2l$
l	*if* $b > l$ *and* $n = 3$ *and* $b \geq 2l$
$\lceil \frac{b-1}{n} \rceil$	*if* $b > l$ *and* $n > 3$ *and* $b - l \geq \lceil \frac{b-1}{n} \rceil$
$b - l + LMS(n, b - n(b-l))$	*if* $b > l$ *and* $n > 3$ *and* $b - l < \lceil \frac{b-1}{n} \rceil$

Proof: This is proven by lemmas 2.11, 2.12, 2.13 and 2.14. ∎

Thus, the limit has decreased from a triangular distribution to an even distribution, even though we have increased the total value of the population.

2.3.5 Putting the Heuristics Together

Consider the following heuristic mating system:

1. Any pair with Hamming distance less than 2 is not allowed to mate.
2. Pick the matable pair(s) of smallest f difference.
3. If more than one pair has equal f differences, pick the pair(s) with largest Hamming distance.
4. If more than one pair has equal Hamming distances, pick the pair of highest rank.

What happens to the LMS point under this heuristic mating strategy? Let this be the Heuristic-LMS point, HALMS(b,n,l). We still assume that the bits are distributed as evenly as possible among the columns.

What does this introduce to the LMS strategy? It re-introduces the heuristic which prevents mating of pairs with Hamming distance of 0 or 1, thereby not mating pairs which provably stagnate. Also, a check for individuals with the largest Hamming distance is introduced. This check is useful, since individuals with larger Hamming distances will tend to separate easier than individuals with low Hamming distances; mating individuals with very low Hamming distances usually leads to stagnation. How does this affect the ALMS point?

Lemma 2.15 *If $b \leq l$, then $HALMS(b,n,l) = LMS2(n,b)$.*

Proof: The only improvement we introduce beyond the LMS2 heuristics is that of mating individuals with larger Hamming distances if we have pairs with identical f differences. But consider the stagnation points: they all stagnate on the bottom with f differences less than that of the other individuals, thus the extra Hamming-distance heuristic does not increase the limit point. Neither does it decrease the limit point, since all of the arguments in lemmas 2.8, 2.9, and 2.10 still hold. Note that it is impossible to have data in which the bottom two individuals have equal f differences to the other individuals but larger Hamming distances, because the Hamming distance between any two individuals is their f sum in this case (since every column contains at most one 1), and the f sum of $\mathcal{P}_1$ with 0 is less than or equal to the f sum of $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ where $i > 1$. Note that the case $b < l$ is directly mappable to the case of a smaller l, $l = b$. ■

Lemma 2.16 *If $b \geq 2l$, then HALMS$(b, n, l) = l$.*

Proof: First of all, note that when $b > 2l$, the HALMS point must be no less than l for the same reasons given in lemma 2.12. The value of l is achievable if any two individuals mate. If none mate, then recall that each column must have at least one 1 in it. Since their Hamming distance is at most 1, if any individual has a 1 in a column, all of the other individuals must have a 1 in that column, except for 1 column. So each member of the population must have a value of at least $l - 1$. Then there must be at least 1 individual with its bit set to 1 in one of the remaining columns, which leads to a value of l. ■

Lemma 2.17 *Consider $n = 3, 2l > b > l$. It is possible to split the HALMS value into the following disjoint groups, with the following conditions and values. $HALMS(b, n, l)$ is*

MIN for stagnation on $\mathcal{P}_2$ and $\mathcal{P}_3$	*if $\frac{3l}{2} \leq b < 2l$*
MIN for stagnation on $\mathcal{P}_1$ and $\mathcal{P}_2$	*if $l < b < \frac{3l}{2}$*

Proof: Consider splitting up the cases of stagnation into stagnation on the top two individuals, stagnation on the middle two individuals, and stagnation on the outer two individuals. If we find the restrictions those stagnation cases imply, we can partition the space into several ranges, which are hopefully disjoint. This will be done by exclusion: stagnation cases will be shown to imply that certain b values are impossible, thus those b values cannot lead to that type of stagnation, therefore they must be calculated from the remaining type(s).

Consider stagnation on the outer two individuals. Although $\mathcal{P}_1 \leq \mathcal{P}_2 \leq \mathcal{P}_3$, $\mathcal{P}_2$ is within Hamming distance of 1 of both $\mathcal{P}_1$ and $\mathcal{P}_3$. The Hamming distance of the outer two individuals must be at least 2, since they mate, and can be no greater than 2, since $H_d(\mathcal{P}_1, \mathcal{P}_3) \leq H_d(\mathcal{P}_1, \mathcal{P}_2) + H_d(\mathcal{P}_2, \mathcal{P}_3)$. A 1 must be present in at least one column of one individual, and if it is present in one column of one individual, it must be present in that column of the other two individuals, except for two columns. This is because of the Hamming distance between the individuals. Thus, $f(\mathcal{P}_1) \geq l-2$. As for the remaining two columns, this situation can only happen if $f^{-}(\mathcal{P}_1, \mathcal{P}_3) = 2$, thus $f(\mathcal{P}_3) \geq 2 + l - 2 = l$. Since the limit point this gives is clearly not the lowest (since it is the highest possible value), it can be ignored for the purpose of establishing a lower bound, if *any* other limit can be found.

Next, consider stagnation on the top pair. The stagnation matrix is:

Element	individual 1s	replicate 1s
$\mathcal{P}_3$	i	$b-l$
$\mathcal{P}_2$	0	x
$\mathcal{P}_1$	$2l-b-i$	$b-l-x$

since we assume that the top individual mated has maximum value.

At stagnation, $i + b - l - x \leq x - (2l - b\ i + b - l - x)$, by mating rule (2), and $b - l - x \geq 0$. These inequalities reduce to $b \leq 3x$ and $x + l \leq b$, which leads to $b \leq 3x \leq 3b - 3l$ thus $b \geq \frac{3l}{2}$. Thus if $b < \lceil \frac{3l}{2} \rceil$, the lower bound cannot be achieved by upper pair stagnation.

Lastly, the lower pair stagnation matrix

Element	individual 1s	replicate 1s
$\mathcal{P}_3$	i	x
$\mathcal{P}2$	$2l-b-i$	$b-l$
$\mathcal{P}1$	0	$b-l-x$

leads to the inequalities: $2l - b - i + x \leq 2i + x - l$ from mating rule (2), and $2l - b - i \geq 0$, since all matrix entries must be at least 0. These equations lead to $3i > 3l - b$ and $2l - b \geq i$ which imply $3l > 2b$, thus $b < \frac{3l}{2}$. Thus if $b \geq \frac{3l}{2}$, the HALMS point cannot be calculated by a lower stagnation matrix.

Thus, we have proven the lemma, namely that the following limits on b imply the following stagnation matrices be used to compute the HALMS value:

limit	upper pair	lower pair	outer pair
$b \geq \frac{3l}{2}$	yes	no	no
$b < \frac{3l}{2}$	no	yes	no

■

Lemma 2.18 *Consider $n = 3$ and Genetic Invariance having stagnated on $\mathcal{P}_2$ and $\mathcal{P}_3$. The Heuristic ALMS point, $HALMS(b, n, l)$, is*

$ALMS(b, 3, l)$	*if* $\lfloor \frac{3l}{2} \rfloor + 4 \leq b < 2l$
$\lfloor \frac{l}{2} \rfloor + 3$	*if* $(\lfloor \frac{3l}{2} \rfloor + 1 \leq b \leq \lfloor \frac{3l}{2} \rfloor + 3)$ *and* $(b < 2l)$
l	*if* $b = \frac{3l}{2}$

Proof: By the arguments in lemma 2.13, stagnation can never happen at a value of less than $b-l$. The case $(2+2l-b, b-l-2, b-l)$ is valid when $\lfloor \frac{3l}{2} \rfloor \leq b$. The uppermost $b - l - 2$ bits are duplicates, with the other 2 duplicates in $\mathcal{P}_1$. This ensures stagnation, resulting in a limit of ALMS$(b, 3, l) = b - l$.

Next, consider the upper pair stagnation matrix from lemma 2.17. The mating technique described in the introduction allows us to conclude the following stagnation equations:

1. $i + (b - l - x) \geq 2$
2. $i + (b - l - x) > x - (2l - b - i + b - l - x)$
3. $i + (b - l - x) = x - (2l - b - i + b - l - x)$ and
   $i + (b - l) - x \geq 2l - b - i + (b - l)$

where equation (1) and either equation (2) or (3) is true. This reduces to (1) and either $(x > \frac{b}{3})$ or $(x = \frac{b}{3}, i \geq l - \frac{b}{3})$. If $x = \frac{b}{3}$ then $f(\mathcal{P}_n) = i + b - l \geq x + 2 > l$, which is a contradiction. So consider $x > \frac{b}{3}$. $b - l \geq x > \frac{b}{3}$, therefore $b > \frac{3l}{2}$. Interestingly enough, $x < \lfloor \frac{l}{2} \rfloor + 3$ implies $i = \lfloor \frac{l}{2} \rfloor + 3 - x$, due to heuristic (1). The only exception is the case $b = \frac{3l}{2}$ where we must use $x = \frac{b}{3}$ to get a minimum bound of l. If $b - l$ is small because l is small, heuristic (1) may hamper this solution. When is this the case? By substitution of the minimum value into the array, it is evident that the top 2 individuals will be far enough apart when $l \geq 5$ for $b > \frac{3l}{2}$ and $l \geq 4$ for $b = \frac{3l}{2}$. ■

Lemma 2.19 *Consider the case $n = 3$, with stagnation on the bottom. The stagnation point, $HALMS(b, n, l)$, is*

$\lceil \frac{b-1}{3} \rceil + 3$	*if* $\lfloor \frac{3l}{2} \rfloor - 2 \leq b \leq \lceil \frac{3l}{2} \rceil - 1$
$l - \lfloor \frac{b}{3} \rfloor$	*if* $l < b \leq \lfloor \frac{3l}{2} \rfloor - 3$

Proof: Consider the stagnation point $(b-l, b-l+\lfloor \frac{b-l}{3}, \lfloor l - \lfloor \frac{b}{3})\rfloor$ much like in lemma 2.13. This holds when the Hamming distance of the bottom two individuals is less than 2, which is when $b > \lfloor \frac{3l}{2} \rfloor - 3$. When this case does not hold, consider the lower pair stagnation matrix from lemma 2.13. From this and our heuristics, we get the following:

1. $2l + x \geq b + i + 2$
2. $3l < b + 3i$

The limit point is obtained by combining $f(\mathcal{P}_3) = i + x$ with (2) and (1), which produce limits on i and x respectively $f(\mathcal{P}_3) = i + x \geq \lceil \frac{b}{3} \rceil + 2$. ∎

For l extremely small, some of the stagnation conditions required above cannot be met. This is when $l \leq 4$. In this case, the formulas produce lower bound values greater than l, which is clearly not correct. These cases are accounted for in the analysis of HALMS(b, n, l), $l \leq 4$.

Lemma 2.20 *HALMS$(b, n, l) = l$ for $n = 3, l \leq 4, l \leq b$.*

Proof: Proof is by enumeration of cases. ∎

Lemma 2.21 *Consider Genetic Invariance stagnating on $\mathcal{P}_i$ and $\mathcal{P}_j$, with b arbitrary and $n > 3$. Define function C where $C(\mathcal{P}_x)$ is the set of columns in $\mathcal{P}_x$ which have the value 1. The population can be divided into an (ordered) list of ϕ groups of individuals $g_1, ..., g_\phi$ such that the following properties hold:*

1. *Each group is contiguous.*
2. *Let $\mathcal{P}_q$ be a member of group g_r. Then, for any member $\mathcal{P}_k$ with $k > q$, we have $C(\mathcal{P}_k) \subseteq C(\mathcal{P}_q)$.*
3. *The groups are maximal with respect to properties (1) and (2).*

4. *The f difference between any 2 (adjacent) groups is at least 3. Since the list is ranked, a difference of at least 3 between adjacent groups implies an f difference of at least $3k$ between group i and group $i+k$, for any groups i and $i+k$.*

Proof: Start with each individual being in its own group. This will validate properties (1) and (2) (since no two individuals are in the same group yet). To validate properties (3) and (4), the groups must be merged into larger groups, such that at the end, each group's individuals are separated from another group's individuals by at least 3. Note that property (3) will usually eliminate the trivial solution of having each individuals in its own group, while property (2) will most likely ensure that they do not all merge into the same group. The partition is unique, since the process to construct it from the trivial partition $\phi = n$, $g_k = \{\mathcal{P}_k\}$ does not allow any choices to be made.

We must now state the partition merge rule. This rule will be proven to be consistent with properties (1) and (2) (applying them to a consistent set will give you a consistent set) and will be shown to attain a unique solution which validates property (3). Finally, the unique solution will be proven to validate property (4). Note that since the merge is a set merge, the order in which it is applied is irrelevant.

The first thing we have to find out is when not to merge two groups. This is rather obvious. In order to merge groups, they must be consecutive groups in the list of groups (which will result in a consecutive list of individuals within the group) and they must maintain property (2). Property (2) is maintained by noticing the properties of the highest individual of the lower group ($\mathcal{P}_x$) and the lowest individual of the higher group ($\mathcal{P}_{x+1}$). Property (2) will be maintained iff $C(\mathcal{P}_x) \subseteq C(\mathcal{P}_{x+1})$. This is due to the transitive nature of subset. If the subset property holds on these two individuals, it will hold on any within the combined set. If it doesn't there is at least one pair in any contiguous set starting at $\mathcal{P}_k, k \leq x$ and continuing to $\mathcal{P}_q, x \leq q$ which does not follow this property, namely $\mathcal{P}_x, \mathcal{P}_{x+1}$, thus *any* attempt at building other partitions will result in smaller groups, clearly violating point (3).

Property (4) can be proven by case analysis on the possibilities of both the stagnation pair $\mathcal{P}_i, \mathcal{P}_j$ and the group boundary individuals $\mathcal{P}_x$, P_{x+1} such that $\mathcal{P}_x$ is the highest individual in group g_y and $\mathcal{P}_{x+1}$ is the highest individual in group g_{y+1}. The following table represents this case analysis. Note that at maximum split of $\mathcal{P}_i$ and $\mathcal{P}_j$, their f difference and Hamming distance must be equal.

$f^-(\mathcal{P}_i, \mathcal{P}_j)$	$f^-(\mathcal{P}_x, \mathcal{P}_{x+1})$	$H_d(\mathcal{P}_x, \mathcal{P}_{x+1})$	Possible?	Reason
< 2			No	Rule 1
2	< 2		No	Rule 2
2	2	≥ 2	No	Rule 3
2	2	< 2	No	$f^- \leq H_d$
2	≥ 3		Yes	
≥ 3	≥ 3		Yes	

The first 3 cases are eliminated by the first 3 mating rules. The fourth case is eliminated because $f^-(\mathcal{P}_x, \mathcal{P}_{x+1}) \leq H_d(\mathcal{P}_x, \mathcal{P}_{x+1})$. The last 2 cases support the lemma in any case, therefore they need not be analysed. Thus, by case analysis, any two group boundary individuals must have a separation of at least 3. Note that because of the definition of groups and stagnation, the stagnation pair must be in the same group. ■

Lemma 2.22 *Let M_z be the value of the maximal invidual ME_z in the group g_z. Then, $\sum_{z=1}^{\phi} M_z \geq l$.*

Proof: To prove this, it is only necessary to show that each column contributes at least one 1 to the overall sum. Thus the total sum must be greater than or equal to l. Consider any column x. $b > l$ therefore $\exists y | P_{y,x} = 1$. To put it another way, $x \in C(\mathcal{P}_y)$. But we know that the maximum individual in the group is a C-function superset of this individual, thus $\exists z, \mathcal{P}_y | ((x \in C(\mathcal{P}_y)) \wedge (C(\mathcal{P}_y) \subset C(ME_z)))$, thus $x \in (C(ME_z))$, and since $M_z = f(ME_z)$, column x contributes its 1 through at least one group, namely z. Thus, $\sum_{z=1}^{\phi} M_z \geq l$. ■

Lemma 2.23 *HALMS(b, n, l) =LMS2(b, n, l) if $n > 3$.*

Proof: Consider lemmas 2.21 and 2.22, and 2.10. Consider each group as if it was an individual. These groups form a sequence of integers summing to l, differing by at least 3, Therefore by lemma 2.10, the top group must have a value at least equal to LMS2(b, ϕ, l). Note that in lemma 2.10 we used the difference of the bottom individuals to prove the property of the sequence, while in this case the property of the sequence was proven, therefore the values of the bottom two individuals do not matter. The triangle must be at least $(0, 2, 5, 8, ...)$, with some value added to the entire series. In the worst case, an arithmetic triangle will be formed with the remaining value being distributed evenly over the groups. Since the value of ϕ is generally not known, the value n may be used as a limit on ϕ, which results in the limit being LMS2(b, n, l). ■

One thing of further interest is that if each group has an internal f-difference of δ, the triangle will be constructed with an inter-individual difference of at least $3+\delta$, producing a lower bound of $(3+\delta)\mathrm{MIN}(\triangle(\frac{l}{3+\delta}), N) + \frac{\text{any remainder}}{n}$.

THEOREM **2.4** *Thus, $HALMS(b, n, l)$ is*

$LMS2(n, b)$	*if* $(b \leq l)$ *or* $(3 < n)$
l	*if* $(n = 3)$, $((b = \frac{3l}{2})$ *or* $((l \leq 4), (l < b))$ *or* $(b \geq 2l))$
$ALMS(b, 3, l)$	*if* $(n = 3)$, $(b > \lfloor\frac{3l}{2}\rfloor + 4)$, $(5 \leq l < b < 2l)$
$\lfloor\frac{l}{2}\rfloor + 3$	*if* $(n = 3)$, $(\lfloor\frac{3l}{2}\rfloor + 1 \leq b \leq \lfloor\frac{3l}{2}\rfloor + 3)$, $(5 \leq l)$
$\lceil\frac{b-1}{3}\rceil + 3$	*if* $(n = 3)$, $(\lfloor\frac{3l}{2}\rfloor - 2 \leq b \leq \lceil\frac{3l}{2}\rceil - 1)$, $(5 \leq l)$
$l - \lfloor\frac{b}{3}\rfloor$	*if* $(n = 3)$, $(5 \leq l < b \leq \lfloor\frac{3l}{2}\rfloor - 3)$

Proof: This theorem is proven by lemmas 2.15, 2.16, 2.18, 2.19, 2.20, 2.23. ■

Thus, the beneficial effect of the Hamming distance heuristic also applies to the case where the number of 1s is arbitrary. The population can be grouped where each group is differs in value from the previous group by at least 3, achieving a lower bound of $\triangle(\frac{l}{3})$.

2.4 Analysis of the Nature of Genetic Invariance

The previous section described lower bounds achievable on a simple function. Even though this analysis has been done on a very simple function, it shows several things about the way Genetic Invariance operates.

Genetic Invariance makes several assumptions about the nature of the function and the nature of the population. Genetic Invariance assumes that the population contians a fairly broad range of alleles in each column. This can usually be guaranteed by insuring that each allele is given an equal proportion of each column in $\mathcal{P}$. So, if there are a alleles in a particular column, the number of each allele will be $\lceil\frac{n}{a}\rceil$ or $\lfloor\frac{n}{a}\rfloor$. Genetic Invariance also assumes that the function can be ordered in some manner, having a distinct minimum (even though the number of minimands might be high) and a distinct maximum. This assumption is true if we are using functions from finite ranges to finite domains.

To determine the effectiveness of Genetic Invariance, we need to know both the assumptions about the population, and how Genetic Invariance operates. Genetic Invariance operates through value propagation. Mating cycles occur, which cause superior schemata to propagate to the upper individual in the mating pair, and inferior schemata to propagate to the lower individual. This is seen most clearly in lemma 2.2, where all of the value propagates to the upper individual in the mating pair. It was stated in the previous section that stagnation occurs when the mating individuals have a Hamming distance of less than 2. However, notice that the Hamming distance between the mating pair determines the number of possible individuals that can be generated. Thus, a mating pair should be chosen so that it has a large enough Hamming distance to separate. However, note that a mating cycle is a random walk of the Hamming closure of the parents, so unless a relatively small increase in value is desired, it is likely that the cycle will not terminate quickly enough to produce adequate results within a small period of time. Thus, Genetic Invariance mates the closest pair, hoping that the Hamming difference is large enough to produce a noticeable separation. Genetic Invariance assumes that the required improvement ($f^-(\mathcal{P}_i, \mathcal{P}_j) - f^-(\mathcal{P}_k, \mathcal{P}q)$, where individuals k and q have the second smallest f difference) is small enough so that relatively little random search is required to attain it.

Thus, the overall effect is that $f^-(\mathcal{P}_i, \mathcal{P}_{i+1})$ will increase, and thus $f^-(\mathcal{P}_1, \mathcal{P}_n)$ will increase as superior schemata move from lower individuals to higher individuals and inferior schemata move from higher individuals to lower individuals. This is seen in lemma 2.5 where an arithmetic triangle is constructed in the worst case because of the minimal difference between individuals. The minimal difference between the mating pair when stagnation occurs thus controls the minimal difference in value between population members. In the simple function, the stagnant pair was always found at the bottom, leading to a minimal difference of 0 or 2, depending on whether individuals with Hamming distance less than 2 were required to mate. The other individuals provably had f differences greater than this value, leading to a difference in value of 1 or 3. On more general functions, this observation about Genetic Invariance should still hold: the lower bound on optimization is governed by the minimum guaranteeable separation between individuals.

Although this thesis does not consider the performance of Genetic Invariance over general functions, there are some characteristics of Genetic Invariance which would seem to apply to more general functions. The C function in lemma 2.21 indicates that general lower bounds on local separation (which directly affect global performance) can be done by exploring the relation between the Hamming distances of individuals and the possible separation of the individuals. The question then becomes "how much optimization must occur before the Hamming distances of nearby individuals converge". Answering this question will lead to a fundamental theorem of Genetic Invariance.

2.5 Conclusion

A new adaptive algorithm, Genetic Invariance, is presented. Genetic Invariance is a system which uses a ranked population, and mates adjacent pairs of minimal function difference. Mathematical analysis is given which proves lower bounds on a specific function, $f_0(x) =$ the number of ones in x. This analysis provides a basis for the discussion on the properties of Genetic Invaraiance. While the mathematical analysis was on a restricted case of a simple function, it did indicate several properties of Genetic Invariance which would apply to more general functions. The separation of superior and inferior schemata causes a separation of value in individuals. Eventually, a minimum difference between individuals is reached, resulting in a minimum difference between each pair of adjacent individuals in the population. Since a function with a finite domain has a defined lower bound, it is possible to put a lower bound on the value of the maximal individual in the population at stagnation. Analysis indicates that allele differences between low valued and high valued schemata may provide clues about the usefulness of Genetic Invariance on a problem.

3 A Comparison of Genetic Invariance and Genetic Algorithms

3.1 Introduction

This chapter will compare and contrast genetic algorithms and Genetic Invariance. The emphasis will be placed on how the two methods work on similar principles, even though they have different optimization strategies. It is not the intent of this chapter to claim that either method is better than the other. This chapter will concentrate on the similarities and differences of these systems and how these similarities and differences affect function optimization.

Section 2 will examine the similarities and differences in the structures of the two systems, and how each structure achieves optimization. The characteristics of these methods will be compared and contrasted.

Genetic Algorithms are often tested by function optimization. Section 3 will show the results obtained by using the two systems to optimize the DeJong functions, a set of functions designed to test the performance of Genetic Algorithms over a variety of function types. Genetic Algorithms and Genetic Invariance will be compared to each other and to a simple elitist random search.

Section 4 will discuss an implementation of the two systems, including data structures for efficient selection of parents. The optimal runtime for each method will be given in terms of the number of generations elapsed.

3.2 A Comparison of the Two Systems

Genetic Algorithms and Genetic Invariance are both evolutionary systems. But what are their similarities and differences? A comparison of the two systems will help understand in what way each system solves problems and what problems each will work better on.

Recall from chapter 1 that Genetic Algorithms are based on the Schema Theorem. This theorem assumes that a sampling of the population by proportional selection of fitter individuals is feasible. As the Genetic Algorithm searches for the optimal solution, the population converges to a particular individual (or, in the case of mutation, to a set localized around a particular individual), and so does less and less exploration. At some point, the algorithm terminates, and produces a solution. Problems such as premature convergence, interference from mutation and genetic drift occur due to the selection system, when inadequate sampling is done or the sampling does not reveal significantly superior schemata.

Chapter 2 shows that Genetic Invariance assumes that schemata are separable. Local separation produces global optimization. If schemata are not separable, stagnation occurs. But both work well on zero epistatic problems (although Genetic Algorithms sometimes needs a large population compared to that of Genetic Invariance).

Although both are evolutionary algorithms, the Genetic Algorithm can be thought of as a more active problem solving system, while Genetic Invariance is more passive. Genetic Invariance actively promotes superior individuals. Genetic Invariance simply lets schemata propagate up or down in the population depending on their relative worth. Thus, the Genetic Algorithm only produces maximands (and, thus, maxima), at the end, while Genetic Invariance produces both maximands and minimands. Thus, a researcher using Genetic Invariance can simply look at the entire population, can analyze the similarities and differences in the top, bottom, and middle of the population to see what did work, what did not, and what was neither good nor bad.

But Genetic Invariance employs some of the features designed to improve Genetic Algorithms. Only two individuals are selected and replaced at a time, similar to Steady-State Genetic Algorithms. Recall from chapter 1 that this causes any superior or inferior schemata produced to be immediately available to reproduce in the next generation. Sharing and crowding are not needed, since genetic diversity is naturally maintained in Genetic Invariance. Although all alleles (thus, all schemata) may not be producable from any particular pair, they will be present in the population, and so available for use. Genetic Invariance is also similar to Parallel Genetic Algorithms in that local optimization is propagated between subpopulations to achieve global optimization. In the case of Genetic Invariance, each adjacent pair can be thought of as a subpopulation. Although these features do not necessarily make Genetic Invariance better than Genetic Algorithms, they do appear to add to its potential.

A summary of the characteristics of Genetic Algorithms and Genetic Invariance is given in table 3. Each method has its successes and failures. The next section will explore how these successes and failures are shown by empirical analysis.

	Genetic Algorithm	Genetic Invariance
Optimization Method	Convergence	Stagnation
f Evals per Generation	n	2
Diversity	Artificial	Natural
Endpoint	Convergence	Stagnation
End Values	Maxima	Maxima,Minima
"Simple" Functions	0 Epistatic (?)	0 Epistatic

Table 3: Characteristics of Each Method

3.3 Empirical Analysis

3.3.1 Introduction

This section shows the results of testing Genetic Algorithms, Genetic Invariance, and random search on the DeJong test suite. These results are analysed, showing several properties of each system.

3.3.2 The DeJong Test Suite

Function optimization is a common test of Genetic Algorithms. Among the functions Genetic Algorithms are tested on, the most common is the DeJong Test Suite. DeJong [DeJ75] designed 5 functions with varying properties. Although the properties of the functions, namely modality, convexity, and continuity, indicate that the functions are of differing natures, this is only true in Cartesian space (and thus, only true for Cartesian function optimizers). The notion of modality, continuity, and convexity is much different in Hamming space.

We added the function f_0 to the 5 DeJong test functions, f_1 to f_5, to see how each method would perform on this allegedly simple function. The functions are listed in table 4.

Functions f_1 through f_5 are the 5 DeJong test functions, copied from [Gol89b]. For f_0, the variable x is a binary number of arbitrary length. No further encoding was required. For f_1 to f_5, each x_i was encoded as a sign-magnitude integer, and divided by an appropriate amount to obtain the range. For example, f_2 was encoded as a 24-bit binary number, composed of 2 12-bit sign-magnitude integers, having 1 sign bit and 11 data bits each. Function f_0 was maximized, and functions f_1 to f_5 were minimized by maximizing $f_{max} - f$. The "noise" in function 4 was created by the following function: $-6 + \sum_{i=1}^{12}$ RANDOM(0,1), where RANDOM(0,1) is a random

f	Mathematical Expression	Limits	f_{max}
f_0	The number of 1 bits in x	$0 \leq x < 2^l$	
f_1	$\sum_{i=1}^{3} x_i^2$	$-5.12 < x_i < 5.12$	81
f_2	$100(x_1^2 - x_2^2)^2 + (1 - x_1)^2$	$-2.048 < x_i < 2.048$	3700
f_3	$\sum_{i=1}^{5} \lfloor x_i \rfloor$	$-5.12 < x_i < 5.12$	25
f_4	$\sum_{i=1}^{30} ix_i^4 + NOISE(-6,6)$	$-1.28 < x_i < 1.28$	1030
f_5	$\frac{1}{0.002+\sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}}$	$-65.536 < x_i < 65.536$	500

Table 4: Table of Test Functions

value from the uniform distribution on (0,1). Function f_4 was also designed to test performance on a large number of dimensions. f_5's constants, a_{ij}, are found in table 5.

First dimension

-32	-16	0	16	32
-32	-16	0	16	32
-32	-16	0	16	32
-32	-16	0	16	32
-32	-16	0	16	32

Second Dimension

-32	-32	-32	-32	-32
-16	-16	-16	-16	-16
16	16	16	16	16
32	32	32	32	32
0	0	0	0	0

Table 5: Constants for DeJong function f_5

3.3.3 Empirical Results of Genetic Algorithms

Graphs 4 to 26 show how the value of the maximal individual in the Genetic Algorithm's population varies with the number of function evaluations performed. One function evaluation is performed for each individual tested. The Genetic Algorithm is used without any modifications such as Crowding or Sharing, or the use of Parallel Genetic Algorithms or Steady State Genetic Algorithms. However, a small mutation rate (0.01), was present and the Genetic Algorithm was tested both with and without elitism. The graphs show the averages obtained over 20 test runs.

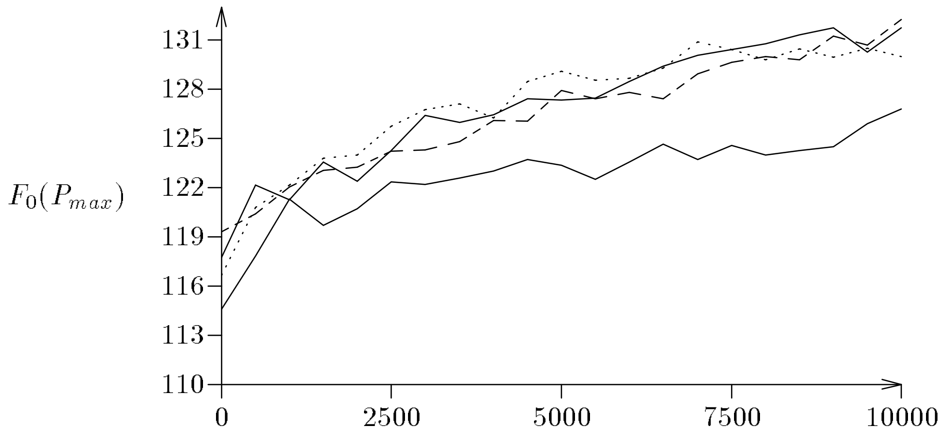
The genetic algorithm performs well on most of the functions. Figures 4 to 9 show how the value of the maximal individual in the population varies over time for all 6 functions. Figure 10 confirms DeJong's note that small populations initially optimize faster, but larger populations outperform them in the long run [DeJ75].

Figure 11 shows that Genetic Algorithms perform better with Elitism. Elitism with mutation keeps improving its maximum value until it has achieved the maximal value. Even though the population can converge to n copies of a single individual, mutations are retained if they are beneficial. Thus an extremely slow growth rate will be noticed even after the population has converged. This is because mutation, with a very low probability, will cause some individuals to rise slightly in value. Thus, even though growth is still possible, it may require hundreds or thousands of generations.

Genetic Algorithms perform poorly without mutation. In the problems sampled, the maximum value of the population in the simple Genetic Algorithm with no mutation rises in value and then falls in value as the population converges. This is the point where the superiority of superior schemata is balanced, and outweighed, by the quantity of inferior schemata. Genetic drift starts to occur, slowly decreasing the overall worth of the population as drift occurs towards the more numerous lower valued schemata. This is shown in figures 12 and 13.

Elitism, naturally, does not allow for the previous sort of drop in maximum value. This method has the effect of distributing the maximal individual's schemata over the inferior individuals. Thus, the current maximal schemata will replace inferior individuals, resulting in a higher growth rate. Figure 11 shows the effects of adding elitism to the Genetic Algorithm.

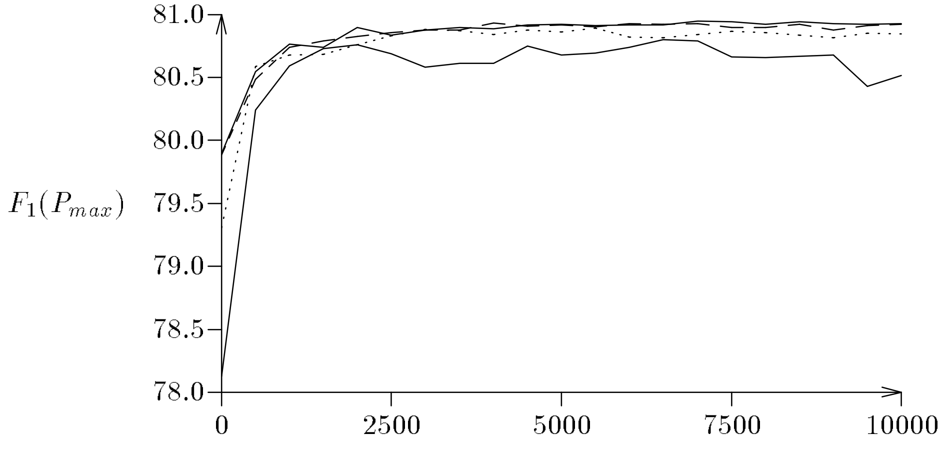
Mutation is a complex issue. While it does allow for greater genetic diversity, it also has its drawbacks. It increases the growth rate by mutating inferior schemata, while it decreases the growth rate by mutating superior schemata. Thus, at the beginning, the mutation rate will allow for greater diversity at very little expense, since the growth rate will usually be high and will not be greatly affected by a slight mutation rate. At the end, where the difference between superior and average schemata is much less, interference from mutation occurs, resulting in oscillation in a wave-like pattern. This is shown by the solid and dashed lines in figure 10.



Function Evaluations

$n = 40, u = 0.00$ (lower line)
$n = 100, u = 0.00$
$n = 150, u = 0.00$
$n = 200, u = 0.00$ (upper line)

Figure 4: Performance of the Genetic Algorithm on f_0



Function Evaluations

$n = 50, u = 0.01$ (lower line)
$n = 100, u = 0.01$
$n = 150, u = 0.01$
$n = 200, u = 0.01$ (upper line)

Figure 5: Performance of the Genetic Algorithm on f_1

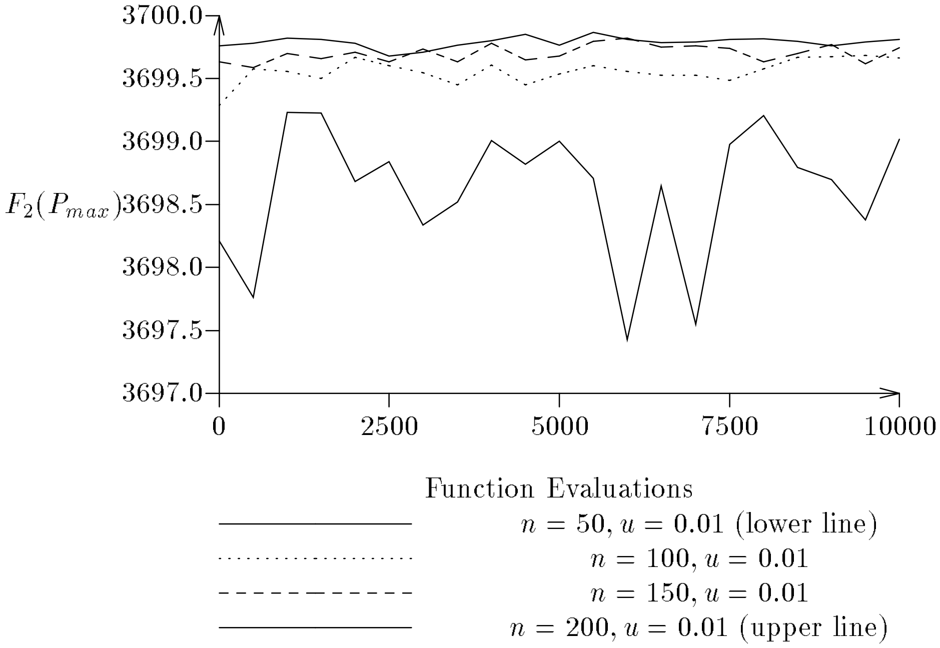


Figure 6: Performance of the Genetic Algorithm on f_2

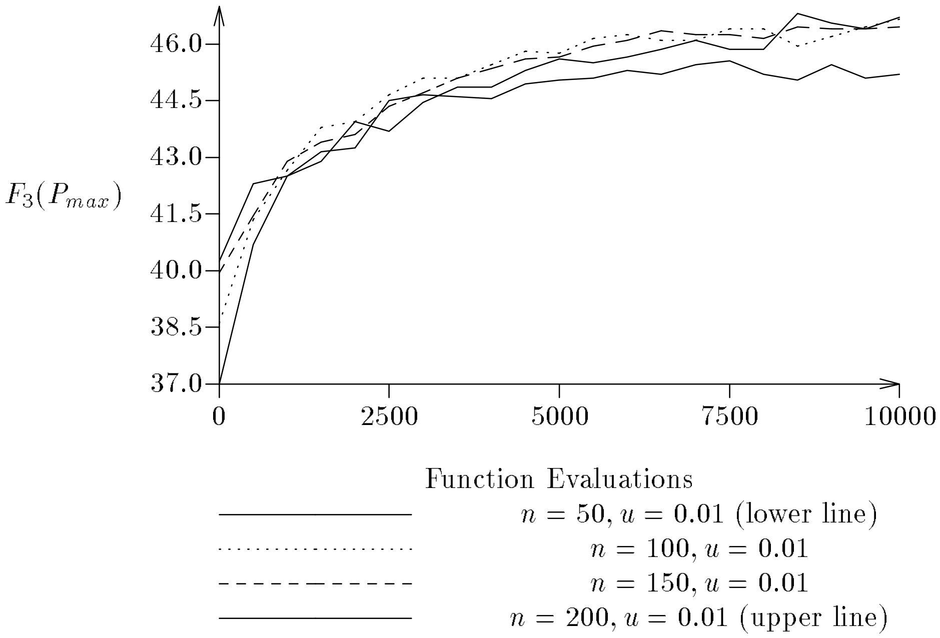


Figure 7: Performance of the Genetic Algorithm on f_3

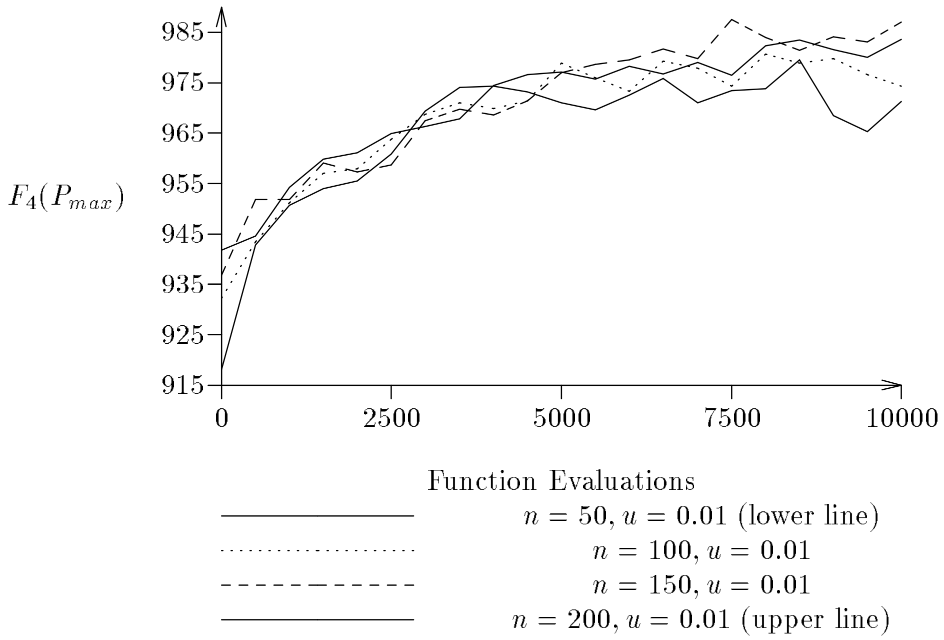


Figure 8: Performance of the Genetic Algorithm on f_4

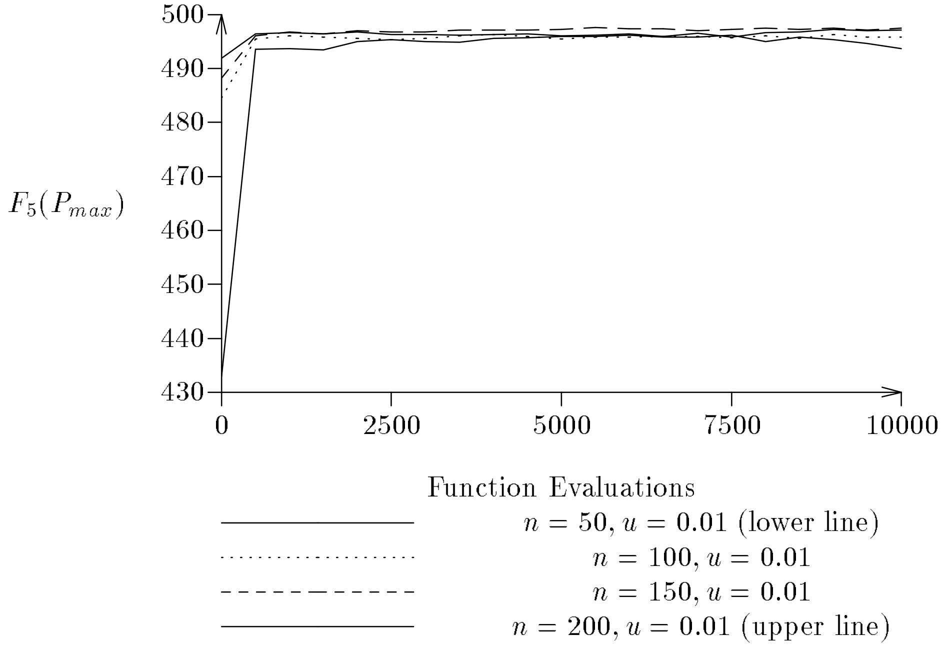


Figure 9: Performance of the Genetic Algorithm on f_5

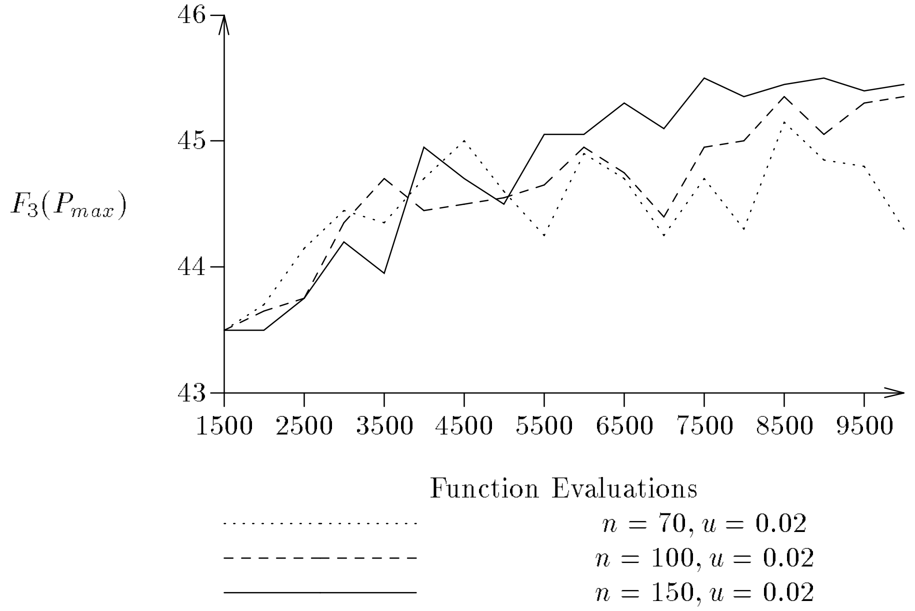


Figure 10: Small and Large Populations, with Mutation

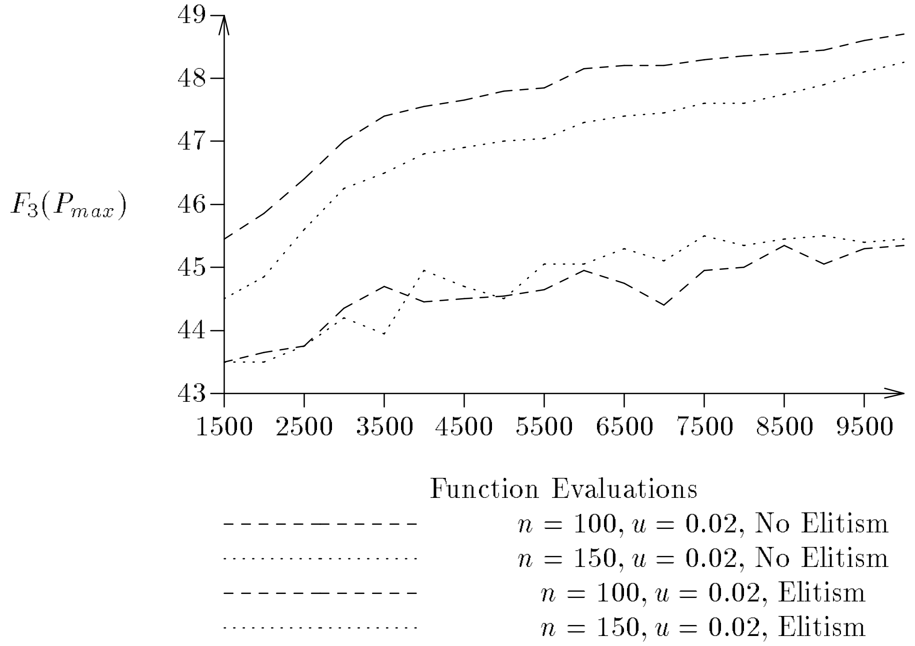


Figure 11: The Simple GA and GA with Elitism, Both with Mutation

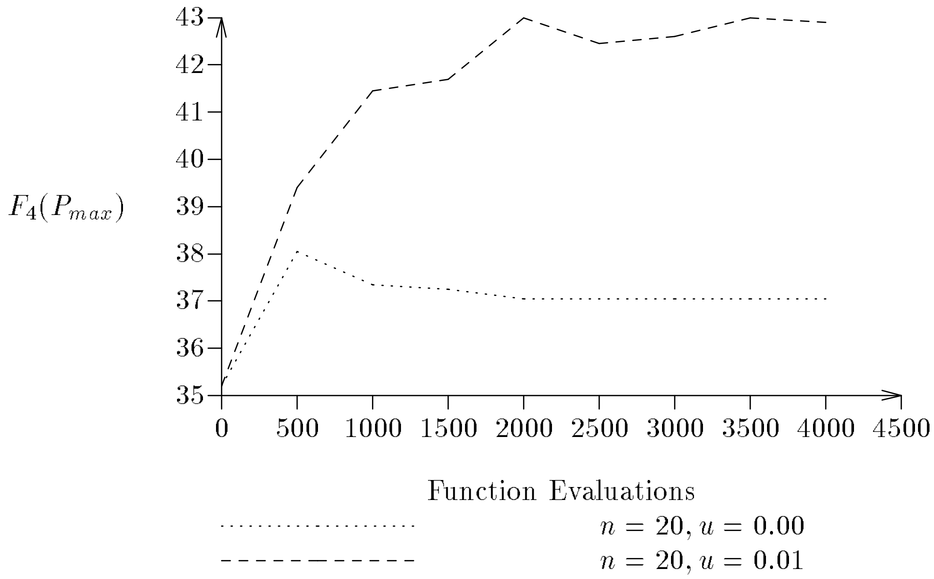


Figure 12: Genetic Algorithm at Zero and Nonzero Mutation

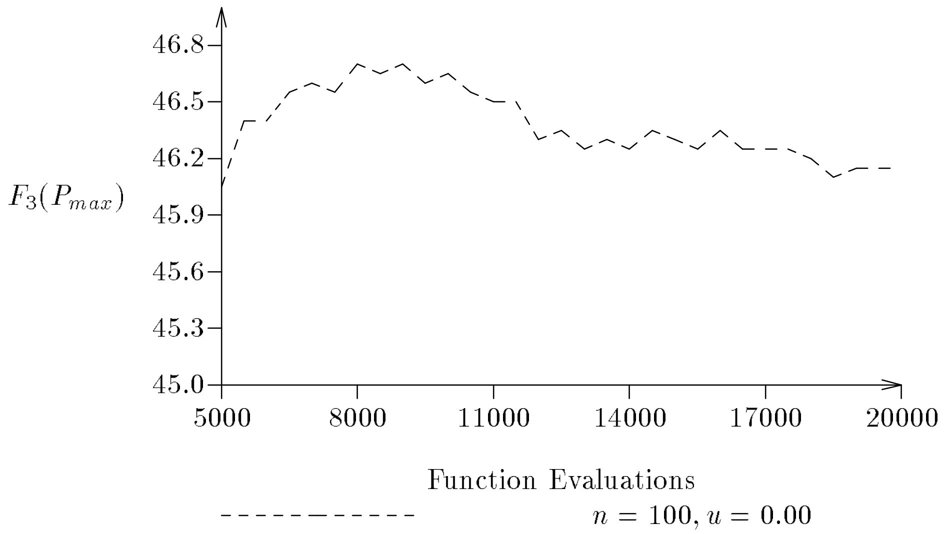


Figure 13: Genetic Algorithm's Maximum Falls as it Converges

3.3.4 Empirical Results of Genetic Invariance

Genetic Invariance was run as described in chapter 2. As with Genetic Algorithms, the graphs show how the maximum value in the population varies with the number of function evaluations performed. The graphs show averages obtained over 20 runs.

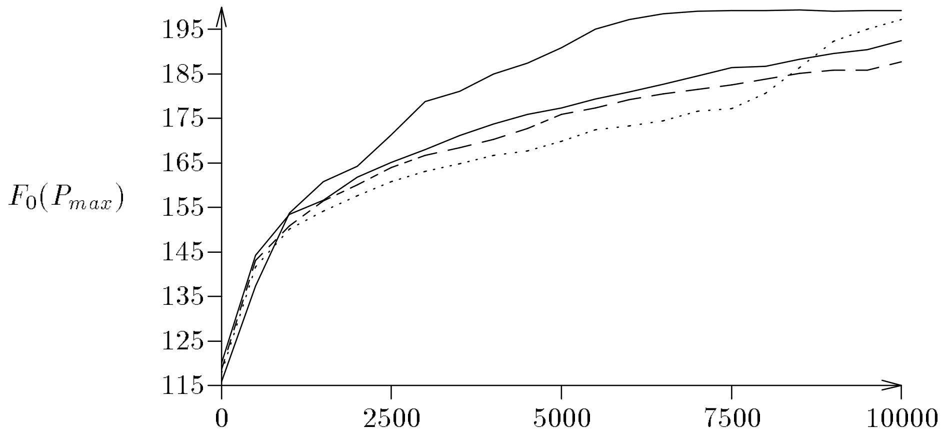
The first function of note is f_0. As can be expected for a method that assumes zero epistasis, Genetic Invariance performed extremely well on this problem. This is, in fact, one of the easiest non-trivial functions for Genetic Invariance to solve. All of the population sizes, from 20 to 200, performed well on all of the string sizes, from 20 to 200. Figure 14 shows that Genetic Invariance performs well on this problem, even with low population sizes compared to the number of bits.

Figures 15, 17, and 18 show that Genetic invariance performed well on functions f_1, f_3, and f_4. However, figure 16 shows that Genetic Invariance performed poorly on function f_2 and did not do much on function f_5. There are several possible causes for Genetic Invariance's poor performance on f_5. First of all, the multimodality of the two functions can cause disturbance in Genetic Invariance, since two conflicting maxima (maxima from different peaks) will tend to combine and may cause a notable reduction in both values. Second, the large range that was being optimized by Genetic Invariance may have adversely affected the precision of the optimization. Both problems not only required that the tested algorithm search for a point near the optimal, they also required a close search of the optimal space. Genetic Invariance concentrates on exploration, and so is not designed to concentrate search on an area. These problems should be investigated in future papers.

Overall, genetic invariance performed well on functions with a small range compared to the size of the domain. It performed particularly well on functions with near-zero epistasis, or functions that can be ranked to near-zero epistasis. From the way the functions were designed, it can be said that Genetic Invariance is not highly affected by multidimensionality (actually, epistasis, not dimensionality, is the primary concern of Genetic Invariance). Genetic Invariance performed poorly when asked to optimize a function with a large range and small domain.

3.3.5 Empirical Analysis of Random Search

Random search is simple to understand. It involves randomly generating an individual, testing it, and keeping the best individual seen.



Function Evaluations

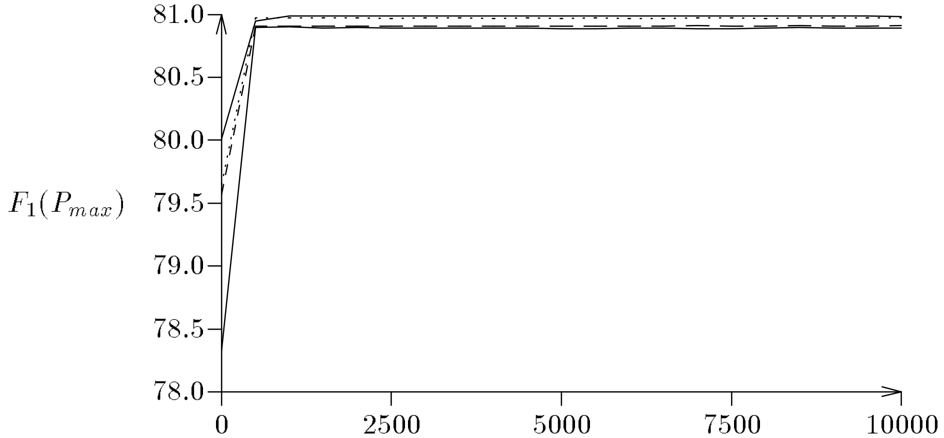
$n = 40, u = 0.00$ (lower line)

$n = 100, u = 0.00$

$n = 150, u = 0.00$

$n = 200, u = 0.00$ (upper line)

Figure 14: Genetic Invariance on Function f_0



Function Evaluations

$n = 50, u = 0.00$ (lower line)

$n = 100, u = 0.00$

$n = 150, u = 0.00$

$n = 200, u = 0.00$ (upper line)

Figure 15: Genetic Invariance on Function f_1

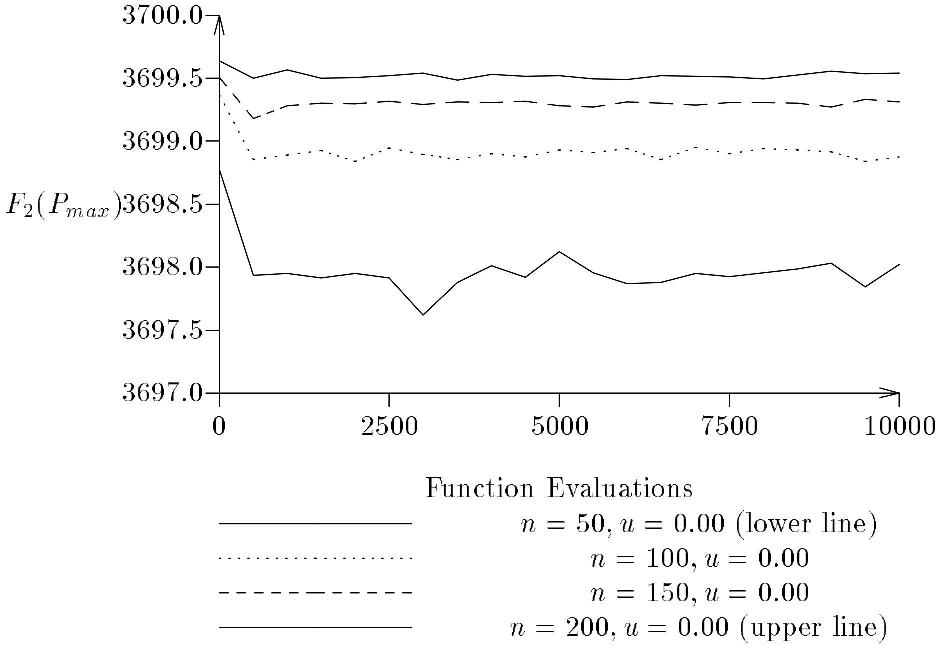


Figure 16: Genetic Invariance on Function f_2

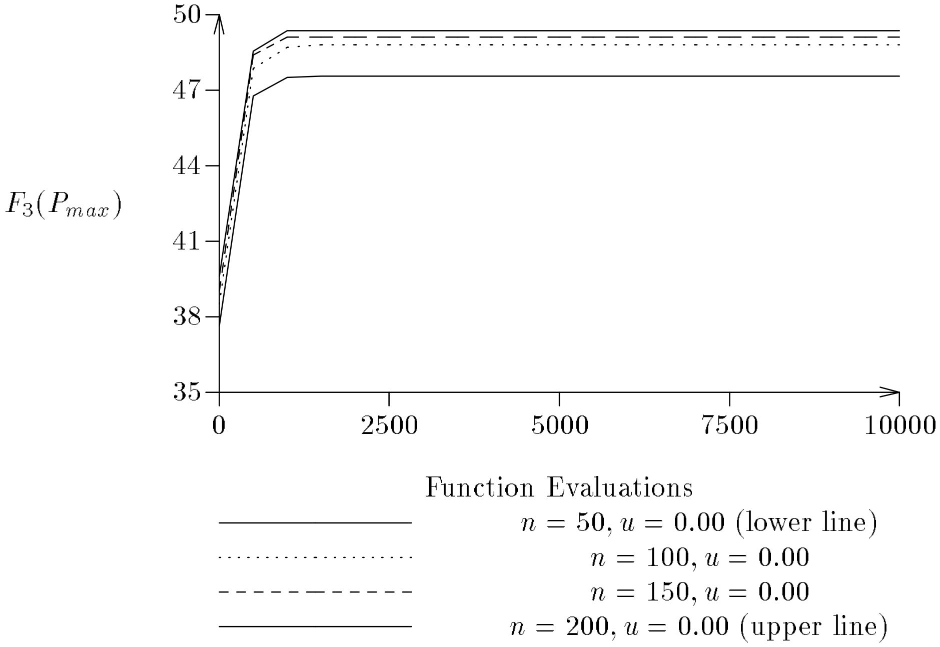


Figure 17: Genetic Invariance on Function f_3

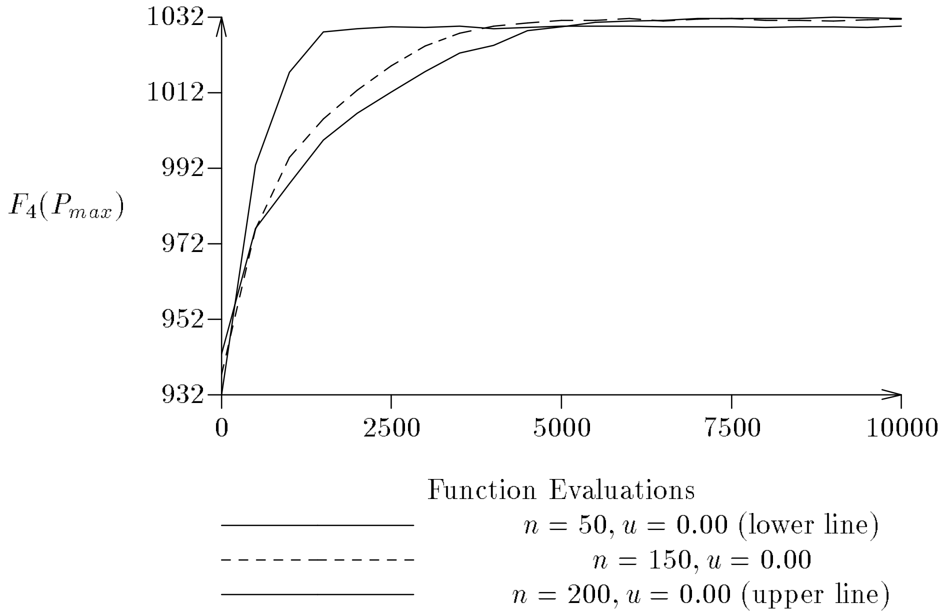


Figure 18: Genetic Invariance on Function f_4

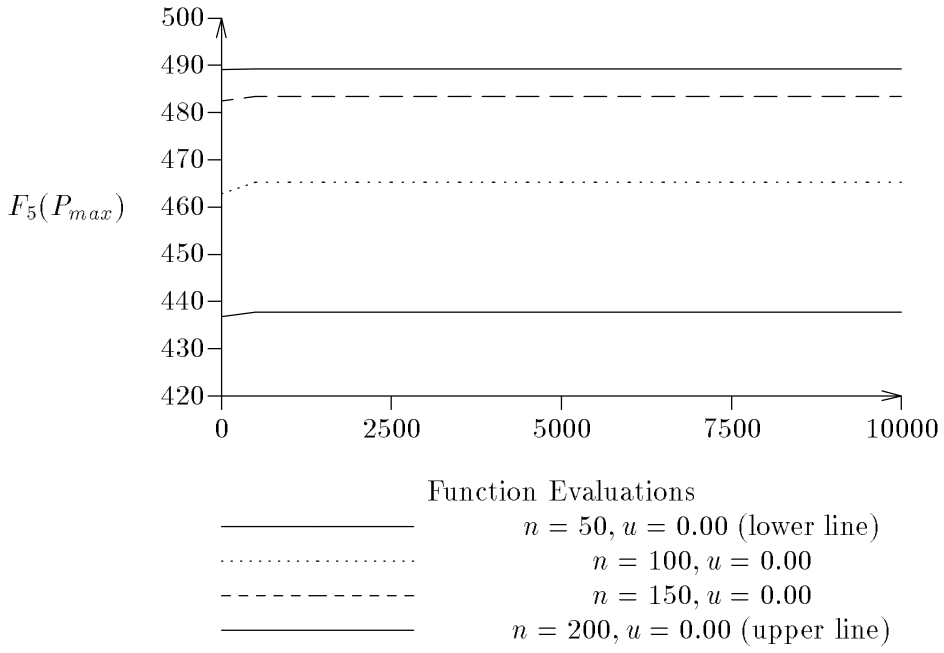


Figure 19: Genetic Invariance on Function f_5

Random search is not expected to outperform any other problem solver on any specific problem. However, it is useful to know in which cases a problem solver utterly fails at its task. Figures 20 through 25 show the performance of random search on functions f_0 through f_5.

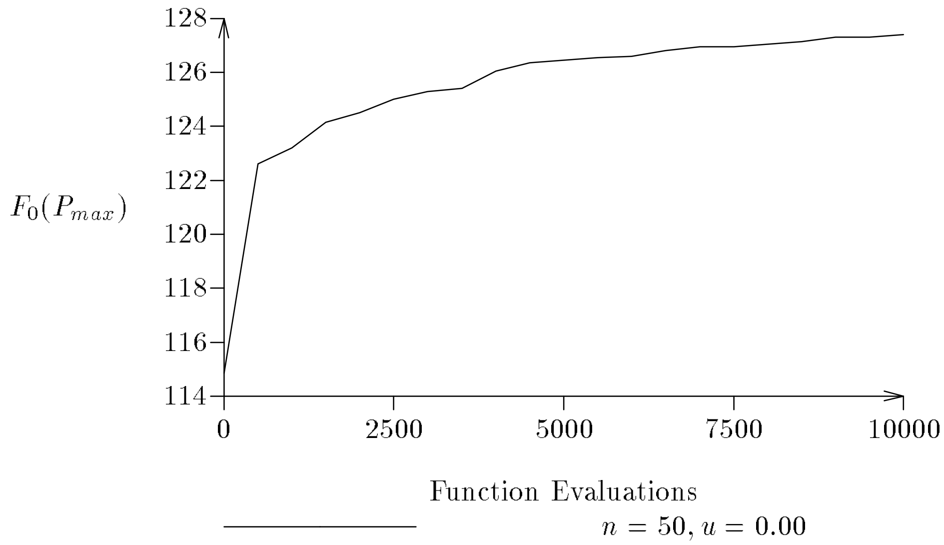


Figure 20: A Random Search on Function f_0

Random search used an elitist approach: each "generation" 50 individuals were tested and the current maximum individual was replaced if the population contained a higher individual. This provides yet another search technique that can be compared with the two evolutionary approaches.

3.3.6 A Comparative Analysis

For function f_0, Genetic Invariance clearly outperformed the simple genetic algorithm, as can be seen from figures 4 and 14. Figure 26 shows what happens when a small population is used on this function. The result is genetic drift, causing only small amounts of real optimization and large amounts of convergence to average valued schemata.

For function f_1, Genetic Invariance outperformed both random search and the simple genetic algorithm. The random search performed poorest, even doing less than the genetic algorithm with a relatively small population size (20). Figures 5, 15, and 21 show that Genetic Invariance performed both more efficiently and more quickly. However a random search, if done long enough, would outperform either method. The simple genetic algorithm

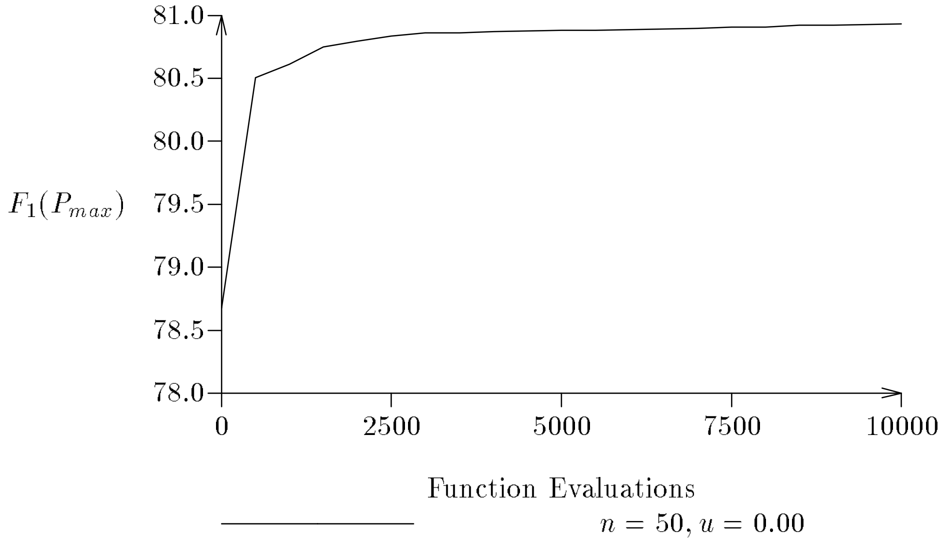


Figure 21: A Random Search on Function f_1

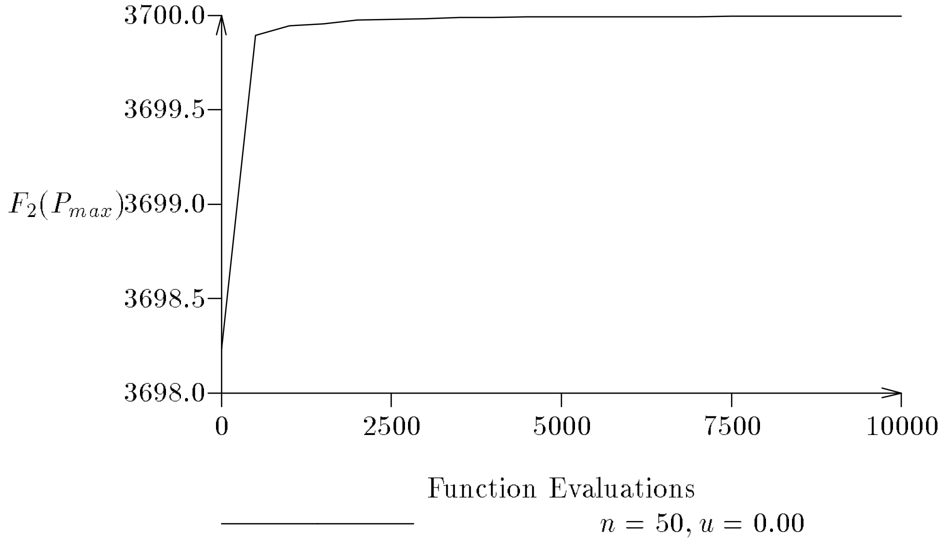


Figure 22: A Random Search on Function f_2

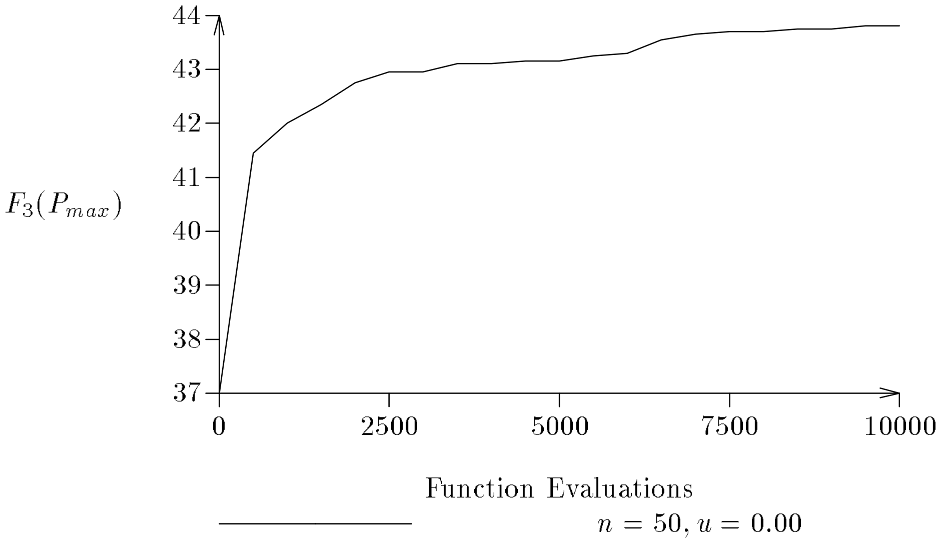


Figure 23: A Random Search on Function f_3

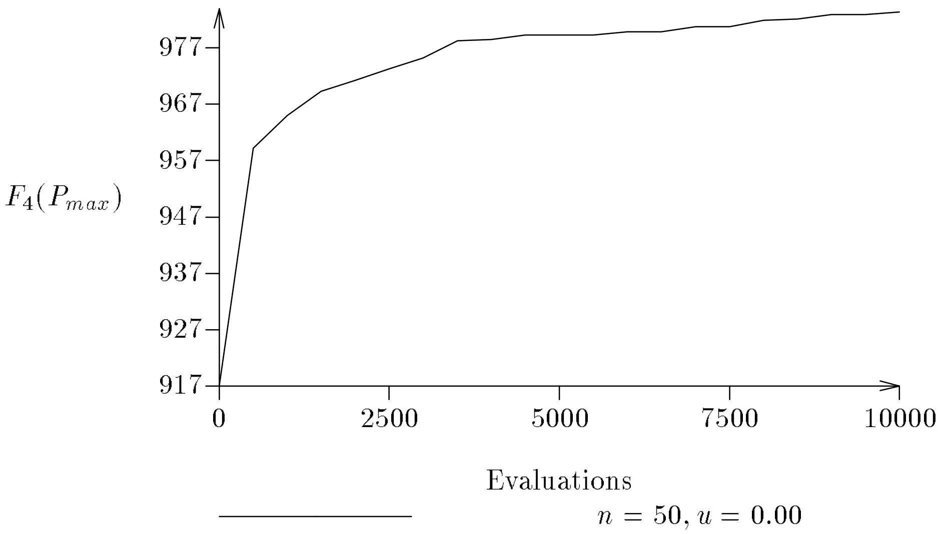


Figure 24: A Random Search on Function f_4

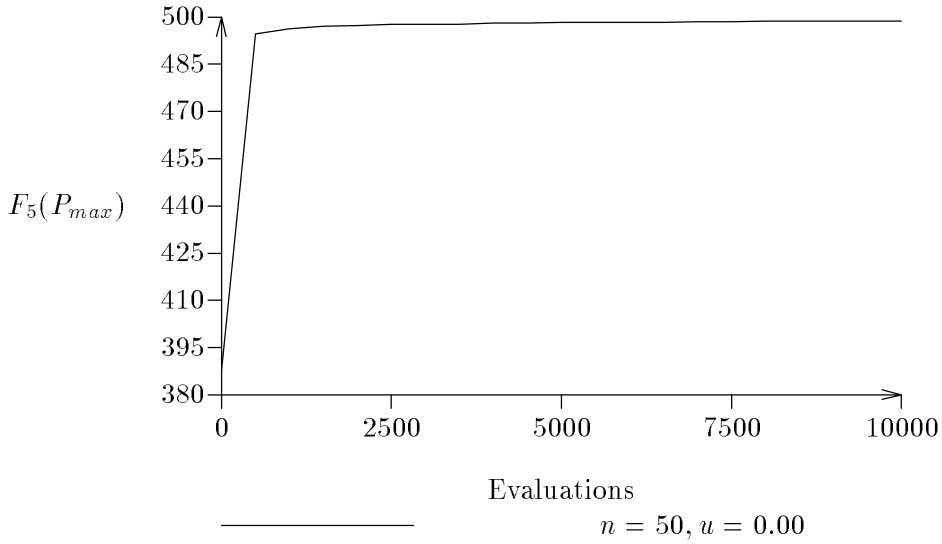


Figure 25: A Random Search on Function f_5

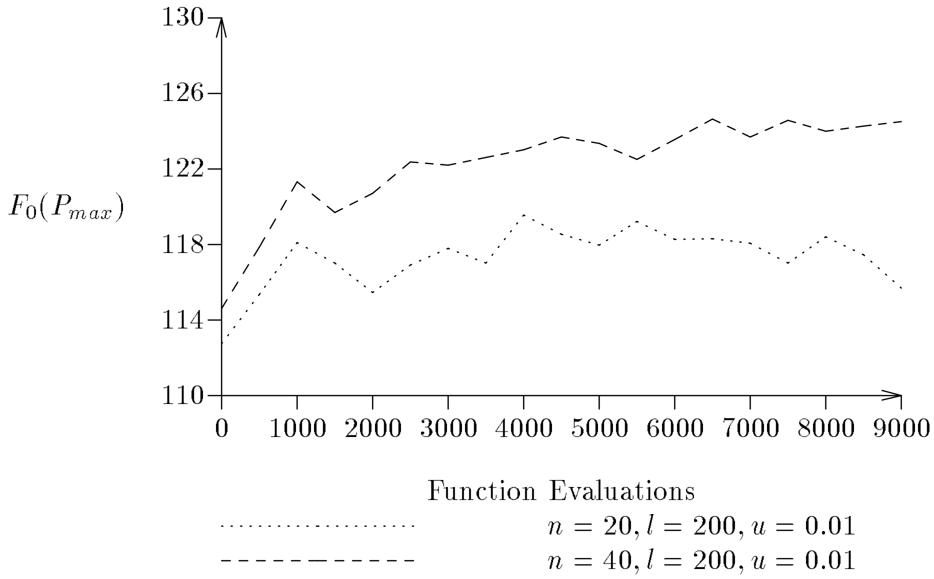


Figure 26: Genetic Drift in Small Population on Function f_0

fluctuated at high values where the speed of convergence was outweighed by mutation. At this point, it would have been profitable to decrease the mutation rate, probably to 0, and let the genetic algorithm converge.

Function f_2 appears to be an anomaly, since a random search appears to outperform both the simple genetic algorithm and Genetic Invariance. However, the "spreading" nature of Genetic Invariance was not designed to optimize a function with such a high range. This distributive nature in fact is bad for this function since the maximal value actually drops. The Genetic Algorithm converges on this function, but not quickly enough to optimize it. The maximal value can only grow another 1 point in 3700, less than 0.03 percent of the total value. Genetic drift becomes a prominent factor since the amount of growth is almost meaningless compared to the overall value of the population. Convergence will be random, and schemata of only moderate worth can be dominant in the converged population. Elitist Genetic Algorithms would perform better, since superior schemata would be preserved. Elitist random search does not have a problem with convergence or separation size. It slowly and steadily increases in value, making it perform better than either of the two evolutionary systems.

The step property of function f_3 makes it an interesting function. The values of an elitist random search will tend to form a step-like graph similar to the graph of the function as the random search take longer to reach higher steps. The graphs show that the maximum value of the $10,000$ random selections does not outperform the simple genetic algorithm. The simple genetic algorithm reaches fair values in moderate time. Genetic Invariance reaches excellent values in small amounts of time, clearly outperforming both other techniques.

For function f_4, Genetic Invariance is also the clear winner. The simple genetic algorithm performed better than a random search, but was still growing slowly at $10,000$ evaluations. However, Genetic Invariance performed quite well, achieving optimal values in little time. Note that although the maximum value of the non-random component was 1030, the fluctuation of the random factor produced results above this value. Thus, a maximum value of 1036 was achievable, but only through both a superior individual and a superior value from the random factor.

Function f_5 was optimized well by both the random search and the simple genetic algorithm. The genetic algorithm achieved higher results in less time than the random search. Unfortunately, Genetic Invariance performed poorly, hardly achieving any improvement over the initial population. This is most likely the result of stagnation early on, and suggests that methods to reduce the probability of early stagnation would be beneficial to Genetic Invariance.

3.3.7 Conclusion

The three strategies, Genetic Invariance, Genetic Algorithms, and Random Search, were tested on f_0 and the DeJong Test Suite. Results showed many of the characteristics of both systems, including genetic drift and interference from mutation in Genetic Algorithms, and stagnation in Genetic Invariance. Genetic Invariance worked well on low-epistatic functions, where schemata were easily separable, while Genetic Algorithms worked well on problems where convergence aided optimization. Random search performed better on one function where the amount of optimization possible was extremely small compared to the total overall function range.

3.4 Data Structures

How can Genetic Invariance and Genetic Algorithms be implemented efficiently? Recall from figures 1 and 3 that the two methods select individuals from the population, mate them with crossover, the Genetic Algorithm performs mutation, and then the population is restructured. Each function evaluation is assumed to take at least $O(l)$ time, since there are l bits in each individual. Thus, the rest of the algorithm should optimally take at least $O(l)$ time per individual evaluated.

In the case of Genetic Algorithms, crossover, mutation, and population restructuring take $O(nl)$ time. Selection can be implemented efficiently by assigning each individual a third characteristic: the sum of the values of previous individuals. In this manner, each parent can be chosen in $O(log(n))$ time by performing a binary search on the sum of values. Each generation, n function evaluations are performed, resulting in $O(nl)$ time per generation. Thus, Genetic Algorithms require $O(nl+nlog(n))$ time per generation. Since $log(n)$ should be smaller than l (otherwise, the population size would exceed the function's domain size, and thus an exhaustive search would be possible), the runtime is $O(nl)$ per generation.

In Genetic Invariance, only two individuals are selected each generation, resulting in $O(l)$ time for crossover. Recall that no mutation is performed in Genetic Invariance. Two function evaluations take $O(l)$ time. Thus, selection and restructuring should optimally take $O(l)$ time. $O(log(n))$ selection can be done using a heap of function differences [AVAU83]. Consider a mating cycle. After it is over, the two individuals will split, separating two other pairs somewhere in the population. Thus, five function differences need to be deleted and re-inserted each generation: the differences between the two mating individuals, the difference between each of them and its non-mating neighbor, and the difference between the two pairs which will be split. Five heap insertions take $O(log(n))$ time, thus $O(log(n))$ time per individual is required. Restructuring the population can be accomplished by making it a binary tree [AVAU83]. Deletions and insertions can be accomplished in $O(log(n))$ time, and the ability to quickly select individuals is maintained by having pointers from the individuals in the heap to the 2 adjacent individuals in the tree. Thus, in Genetic Invariance, a constant number of individuals are updated in $O(log(n))$ time. Since $log(n) < l$, Genetic Invariance can be said to update a constant number of individuals in $O(l)$ time.

3.5 Conclusion

Similarities and differences between the two methods were discussed. Genetic Algorithms were shown to be an active function optimizer in that it concentrated its effort on finding optimal values. Genetic Algorithms use roulette wheel selection to implement Darwinian selection, which does perform adequate optimization, but can lead to premature convergence, interference from mutation, and genetic drift.

Genetic Invariance causes local separation, which in turn causes global separation. Superior schemata are propagated upwards in the population, to combine with more superior schemata, while inferior schemata propagate down, to combine with more inferior schemata. The result is optimization without convergence. While Genetic Invariance does not have problems with interference from mutation, genetic drift, or premature convergence, it has the problem of stagnation. If two mating individuals cannot separate, no more work can be done. Several features used to improve the Genetic Algorithm are natural to Genetic Invariance. Genetic diversity and the use of localized mating and mating few individuals at once are all inherent to Genetic Invariance.

Empirical results of the Genetic Algorithm and Genetic Invariance were discussed. Random search was also tested, to determine the effectiveness of each method. Each was tested on function f_0 and the five DeJong functions. The maximal value seen at any given time was graphed against time, represented by the number of function evaluations done. Genetic Algorithms showed the features of premature convergence, genetic drift, and interference from mutation. Even so, they performed well on several of the functions. Elitism caused an improvement in the overall performance of the Genetic Algorithm. Genetic Invariance worked well on three DeJong functions, but performed poorly on the others. Genetic Algorithms outperformed both other problem solvers on function f_5, while Genetic Invariance worked better on functions f_0, f_1, f_3, and f_4. Random search performed best on function f_2, where the amount of optimization possible was insignificant compared to the overall value of the function.

Finally, Data structures suitable for each were discussed. A simple binary search may be used to make the Genetic Algorithm selection process efficient, while Genetic Invariance is more complex, requiring a tree and a heap to run efficiently. Both methods resulted in the minimum runtime of $O(l)$ per function evaluation, under the assumption that each function evaluation takes at least $O(l)$ time.

4 Conclusion

An adaptive search algorithm is a system that modifies is search strategy based on the data it collects. A Genetic Algorithm is an adaptive search algorithm that uses generate-and-test data collection and employs a population of elements. The initial population is randomly generated, and subsequent populations are created by using processes modelled after genetic sexual and asexual reproduction. Genetic Algorithms use roulette wheel selection based on the Schema Theorem. The Schema Theorem states that individuals should be assigned a probability of mating proportional to their relative values, which causes schemata to grow in quantity at a rate exponentially proportional to their quality. Function optimization was noted to be the standard method for testing Genetic Algorithms. Several problems occur when using Genetic Algorithms, including deception, premature convergence, interference from mutation, and genetic drift. Sharing, Crowding, Elitism, Steady-State Genetic Algorithms, and Parallel Genetic Algorithms have been proposed by other researchers to solve these problems.

We introduced Genetic Invariance, an evolutionary system based on the separation of value. This method was of particular interest to us because no where does it specifically select for optimal values. Rather, optimization is accomplished by side effects of its overall behavior. We have given a mathematical analysis on a restricted case of a simple function, and various properties of Genetic Invariance were noted. Implications of the properties and mathematical analysis were given, showing that in general, Genetic Invariance separates superior and inferior schemata, causing a separation between the minimal and maximal individuals in the population. This, combined with the fact that the functions we are using have defined lower limits, indicate that it is possible to put a lower bound on the maximal value achieved by Genetic Invariance.

Similarities and differences between the two methods were discussed. Genetic Algorithms use roulette wheel selection, which causes the population to converge. Genetic Invariance uses local separation, to indirectly cause global optimization. Both were tested on the DeJong functions. Genetic Invariance worked better on four of the functions, but performed poorly on the other two. Genetic Algorithms worked well on five functions, outperforming Genetic Invariance on one of the functions. They were tested against Random search, which performed better on one function where the amount of optimization possible was considerably smaller than the overall function value. Data structures were given for both, which indicated that an $O(l)$ runtime per function evaluation was possible for each system.

The nature of this thesis was not one of determining which system is better. Each method is better for a different set of problems. Instead, this thesis explores the similarities and differences of the two methods. Genetic Invariance requires separability and produces separation. Genetic Algorithms require separation and produce optimization. An exploration of a combination of the two methods would be useful, since each can succeed in cases where the other will fail. Allowing each method to run for a number of generations is a possibility, as Genetic Algorithms will trim unprofitable schemata from the population while Genetic Invariance will separate the population, which will reduce the probability of genetic drift.

Maintaining invariance does not require that adjacent pairs be mated. Any pair can be chosen as parents as long as they are replaced by their children. Other methods are more flexible, and are likely to produce better results. Useful optimization can involve mating any pair, assigning probabilities of mating based on Hamming distance and similarity of value. The pair would then be mated, and replace the parents if some condition was met. Two conditions suggest themselves as being feasible. If the children produced have a higher difference in value, they can replace their parents. Alternatively, the children could replace their parents if the children's cumulative or maximal value exceeds the parents' value.

For parallel computers, Parallel Genetic Invariance is possible. Selecting a number of closest pairs, then mating them can be done, although steps would have to be taken to insure that no pair interferes with any other pair.

Genetic Diffusion, adding mutation to Genetic Invariance is possible. This is not recommended, since the mutation rate should be extremely small and directed at modifying the values of the lowest ranked individuals. This may produce a high maximum, but will also cause the minimum to increase. This is not recommended if an overall view of the function is desired. Another method is replacing the lowest individual by a random individual every few generations. This will have an effect similar to mutation: a higher maximum is possible, but at the expense of an overall view of the function.

Genetic Algorithms have the Schema Theorem. A similar fundamental theorem of Genetic Invariance would allow for more effective research in Genetic Invariance. This theorem would show what function properties lead to local separation and which do not. Also, the amount of separation that can be achieved before stagnation can be mentioned.

This thesis is only a start in exploring the fundamentals of Genetic Invariance and evolutionary systems. Although some basic explanations about the nature of Genetic Invariance were given, further research is necessary before Genetic Invariance is understood even as well as Genetic Algorithms. From this, the nature of evolutionary systems can be studied, and the beneficial and unnecessary features of these systems can be explored. This will lead to more efficient evolutionary systems in particular, and more efficient problem solvers in general.

References

[AVAU83] John E. Hopcroft Alfred V. Aho and Jeffrey D. Ullman. *Data Structures and Algorithms.* Addison Wesley Publishing Company, 1983.

[Bet80] Albert D. Bethke. *Genetic Algorithms for Function Optimization.* PhD thesis, University of Michigan, Department of Computer and Communication Sciences, 1980.

[DeJ75] K.A. DeJong. *Analysis of Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975.

[GB89] John J. Grefenstette and James E. Baker. How genetic algorithms work: A critical look at implicit parallelism. In John J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Third International Conference on Genetic Algorithms.* Erlbaum, 1989.

[Gol88] David E. Goldberg. *Genetic Algorithms and Walsh Functions, Part 1: A gentle Introduction.* TCGA report 88006, 1988.

[Gol89a] David E. Goldberg. *Genetic Algorithms and Walsh Functions, Part 2: Deception and its Analysis.* TCGA report 89001, 1989.

[Gol89b] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[GR87] Goldberg and Richardson. Genetic algorithms with sharing for multimodal function optimization. In John J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Erlbaum, 1987.

[GS87] David E. Goldberg and Philip Segrest. Finite markov analysis of genetic algorithms. In John J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8. Erlbaum, 1987.

[Hol73] John H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput.*, 2(2):88–105, 1973.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[Müh91] H. Mühlenbein. Evolution in time and space – the parallel genetic algorithm. In Gregory J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337. Morgan Kaufmann, 1991.

[Sys91] G. Syswerda. A study of reproduction in generational and steady-state genetic algorithms. In Gregory J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

[Sys99] G. Syswerda. Uniform crossover in genetic algorithms. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1999.

A Glossary

b	The number of 1 bits in the population.
$C(x)$	$\{w \mid x_w = 1\}$.
d	The number of duplicate columns in the population.
f	A function.
f_0	The number of 1 bits in x, $0 < x < 2^l$
f_1	$\sum_{i=1}^{3} x_i^2$, $-5.12 < x_i < 5.12$
f_2	$100(x_1^2 - x_2^2)^2 + (1 - x_1)^2$, $-2.048 < x_i < 2.048$
f_3	$\sum_{i=1}^{5} \lfloor x_i \rfloor$, $-5.12 < x_i < 5.12$
f_4	$\sum_{i=1}^{30} ix_i^4 + NOISE(-6, 6)$, $-1.28 < x_i < 1.28$
f_5	$\frac{1}{0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}}$, $-65.536 < x_i < 65.536$
$f^+(\mathcal{P}_x, \mathcal{P}y)$	The f sum: $f(\mathcal{P}_x) + f(\mathcal{P}_y)$.
$f^-(\mathcal{P}_x, \mathcal{P}y)$	The f difference: $\lvert f(\mathcal{P}_x) - f(\mathcal{P}_y) \rvert$.
$H_d(x, y)$	the Hamming distance between x and y.
l	Length of an individual.
n	Population size.
$\mathcal{P}$	A Population of n individuals.
$\mathcal{P}_i$	The ith individual of $\mathcal{P}$.
$\mathcal{P}_{ik}$	The kth bit of $\mathcal{P}_i$.
$\mathcal{P}^{(t)}$	The population after t restructurings.
t	The current time step.
$T(x)$	The xth triangular number.
$\triangle(x)$	The inverse of $T(x)$.
u	The mutation rate .
$X(\mathcal{P}_i, \mathcal{P}_j)$	The crossover of $\mathcal{P}_i$ and $\mathcal{P}_j$.
ϕ	The number of C-superset groups in $\mathcal{P}$.

Table 6: Definitions of Commonly Used Variables

Allele The value of a column in an individual.

ALMS point The minimal value achievable by Genetic Invariance when it stagnates on f_0, with the b 1s evenly distributed over the columns.

Convergence The restriction of search to a smaller domain.

Crossover A method of combining 2 parents, which produces 2 children.

Defining Length The defining length of a schema s is the distance between the first and last fixed alleles of s.

Elitism Ensuring that the fitness of the population does not drop. If $f_{max}(\mathcal{P}^{(t)}) < f_{max}(\mathcal{P}^{(t+1)})$, then the fittest individual in $\mathcal{P}^{(t)}$ replaces the least fit in $\mathcal{P}^{(t+1)}$.

Epistasis A function is epistatic if there is a dependence among bits such that the contribution of one bit to the value of the individual is dependent on the other bits in the individual. A function without bit dependence has 0 epistasis.

Genetic Algorithm An adaptive search strategy using roulette wheel selection to achieve Darwinistic evolution of superior individuals.

Genetic Drift A problem with Genetic Algorithms which occurs if superior individuals are of only slightly higher value than average individuals.

Genetic Invariance An adaptive search strategy employing local separation to achieve global separation, and thus, optimization.

HALMS point The minimal value achievable by Genetic Invariance when it stagnates on f_0, with the b 1s evenly distributed over the columns, with several heuristics added to Genetic Invariance.

Higher $\mathcal{P}_y$ is higher than $\mathcal{P}_x$ in Genetic Invariance if $x > y$.

Individual An l bit binary string.

Invariance $\forall t_1, t_2, j, \sum_{i=1}^{n} \mathcal{P}_{ij}^{(t_1)} = \sum_{i=1}^{n} \mathcal{P}_{ij}^{(t_2)}$.

Initialization A method of creating $\mathcal{P}^{(0)}$.

LMS Point The least maximum under stagnation in the 1 bit per column case of f_0.

Lower $\mathcal{P}_x$ is lower than $\mathcal{P}_y$ in Genetic Invariance if $x > y$.

Mating Scheme A method of generating $\mathcal{P}^{(t+1)}$ from $\mathcal{P}^{(t)}$.

Mutation The probability that $\mathcal{P}'_{ik}$ is replaced by $\neg\mathcal{P}'_{ik}$ when children are generated.

Order The order of a schema s is the number of fixed alleles in s.

Selection The process of choosing parents from $\mathcal{P}^{(t)}$.
Roulette Wheel Selection: The probability that $\mathcal{P}_i$ will be selected as a parent is $\frac{f(\mathcal{P}_i)}{\sum_{j=1}^{n} f(\mathcal{P}_j)}$.

Stagnation The time at which Genetic Invariance is no longer productive. Thus, the point at which the value of $\mathcal{P}_n$ cannot increase.

Value Divergence Increase of $f^-(\mathcal{P}_1, \mathcal{P}_n)$ in Genetic Invariance.

Table 7: Definitions of Commonly Used Terms