Comparison of convergence behavior in the Simple genetic algorithm and the Infinite population model

Michael D. O'Conner

The University of Montana

Follow this and additional works at: https://scholarworks.umt.edu/etd

Let us know how access to this document benefits you.

Recommended Citation


This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.
The University of MONTANA

Permission is granted by the author to reproduce this material in its entirety, provided that this material is used for scholarly purposes and is properly cited in published works and reports.

** Please check "Yes" or "No" and provide signature **

Yes, I grant permission
No, I do not grant permission

Author's Signature

Date 1 August, 1997

Any copying for commercial purposes or financial gain may be undertaken only with the author's explicit consent.
COMPARISON OF CONVERGENCE BEHAVIOR
IN
THE SIMPLE GENETIC ALGORITHM
AND
THE INFINITE POPULATION MODEL

by
Michael D. O'Conner
B.A., The University of Montana, 1971
B.S., The University of Montana, 1992
presented in partial fulfillment of the requirements
for the degree of
Master of Science
The University of Montana
1997

Approved by:
[Signature]
Chairperson
[Signature]
Dean, Graduate School
8-5-97
Date
Comparison of Convergence Behavior in the Simple Genetic Algorithm and The Infinite Population Model (123 pp)

Director: Alden H. Wright, Ph.D.

Two paradigms of genetic algorithms are the Simple Genetic Algorithm (SGA) and Vose's Infinite Population Model (IPM). We initialize these two algorithms with identical initial populations, and use proportional selection, one-point crossover, and no mutation.

The convergence behavior of the SGA and IPM is compared for various bitstring lengths, population sizes, and epistasis levels in order to determine whether the deterministic IPM can be a good predictor of the behavior of the stochastic SGA. Bitstrings are limited to length 4, 6, and 8, due to the computation time required for the matrix multiplications used in the IPM algorithm. Population sizes are chosen on the basis of three criteria: constant size, size proportional to the square root of the search space, and size proportional to bitstring length. Epistasis is developed from the perspective of Walsh space, providing a clean and simple method of calculation. This presentation of epistasis is shown to be compatible with those of Davidor and Aizawa.

The fitness vector developed for the genetic algorithms is the combination of a purely linear fitness vector and a purely epistatic fitness vector in proportions determined by a specified constant, $\alpha$, which is a function of the length of the linear and epistatic vectors and the desired epistasis for the resulting fitness vector. This enables the creation of fitness vectors having a predetermined epistasis.

For this class of fitness functions, the IPM becomes a better predictor of SGA behavior as the SGA population size increases. However, the specific behavior upon which the comparison is based should be stated if epistasis varies, because increasing epistasis may cause a comparison based on one behavior (e.g., SGA convergence to a stable fixed point) to improve, while a comparison based on another behavior (e.g., SGA and IPM convergence to the same population) may degrade.
Contents

1 Introduction 1

2 Background 4

1. The Simple Genetic Algorithm (SGA) ......................................................... 4
   1.1. Terms and Representations ................................................................ 4
   1.2. The Algorithm .................................................................................... 4
   1.3. Distinguishing Characteristics of the SGA .................................. 5

2. The Fitness Function and Epistasis ............................................................ 10

2.1. Fitness and Fitness Functions ......................................................... 10

2.2. The Walsh Transform ....................................................................... 13

2.3. Epistasis ................................................................................................. 22

3. The Vose Infinite Population Model (IPM) .............................................. 31

3.1. Notation and Definitions ................................................................... 31

3.2. Population Vector ................................................................................ 32

4. Trajectory, Fixed Points, Stability, Convergence ................................. 37

4.1. Trajectory .............................................................................................. 37

4.2. Fixed Points .......................................................................................... 37

4.3. Stability ................................................................................................. 37

4.4. Convergence ........................................................................................... 38
3 Objectives

4 Methodology

1. Design ........................................... 43
   1.1. The General Algorithm ................. 43

2. Implementation Details ....................... 44
   2.1. Random Number Generator ............ 44
   2.2. Parameters Fixed For All Experiments 45
   2.3. Parameters Which Vary Between Experiments 45
   2.4. Fast Walsh Transform .................. 46
   2.5. Creation of Initial Populations ........ 46
   2.6. Creation of a Fitness Function Vector 47
   2.7. The IPM .................................. 48
   2.8. The SGA .................................. 48
   2.9. Stability Calculation ................... 48

3. Verification of Correctness .................. 48

4. Experiments ................................... 49

5 Results ........................................ 51

1. Charts ........................................... 51

2. Observations ................................... 107
   2.1. IPM as Predictor ......................... 107
   2.2. Transition to Fully Epistatic .......... 109
   2.3. Random Fitness Vectors ............... 112

6 Conclusion ..................................... 113

7 Appendix — Notation Dictionary ........... 116

1. Thesis Notation ................................. 116

iv
2. **Davidor Notation** ................................................................. 118
3. **Aizawa Notation** ................................................................. 118
4. **Vose Notation** ................................................................. 119

8 **References** 121
List of Figures

2.1 Constant, Linear, and Epistatic Components of the Fitness Function 23

5.1 SGA converges to Stable Fixed Point, \( \ell = 4 \) .......................... 52
5.2 SGA converges to Stable Fixed Point, \( \ell = 6 \) .......................... 53
5.3 SGA converges to Stable Fixed Point, \( \ell = 8 \) .......................... 54
5.4 SGA converges to Stable Fixed Point, \( r_e = 128 \) ....................... 55
5.5 SGA converges to Stable Fixed Point, \( r_e = 8\sqrt{2^\ell} \) ................ 56
5.6 SGA converges to Stable Fixed Point, \( r_e = 4\ell \) ........................ 57
5.7 SGA converges to Almost-Stable Fixed Point, \( \ell = 4 \) ................. 58
5.8 SGA converges to Almost-Stable Fixed Point, \( \ell = 6 \) ................. 59
5.9 SGA converges to Almost-Stable Fixed Point, \( \ell = 8 \) ................. 60
5.10 SGA converges to Almost-Stable Fixed Point, \( r_e = 128 \) .............. 61
5.11 SGA converges to Almost-Stable Fixed Point, \( r_e = 8\sqrt{2^\ell} \) ........ 62
5.12 SGA converges to Almost-Stable Fixed Point, \( r_e = 4\ell \) .............. 63
5.13 IPM and SGA converge to populations having the same fitness, \( \ell = 4 \) 65
5.14 IPM and SGA converge to populations having the same fitness, \( \ell = 6 \) 66
5.15 IPM and SGA converge to populations having the same fitness, \( \ell = 8 \) 67
5.16 IPM and SGA converge to populations having the same fitness, \( r_e = 128 \) 68
5.17 IPM and SGA converge to populations having the same fitness, \( r_e = 8\sqrt{2^\ell} \) 69
5.18 IPM and SGA converge to populations having the same fitness, \( r_e = 4\ell \) 70
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.19</td>
<td>Convergence to Optimum vs Population Size, $\ell = 4$</td>
<td>72</td>
</tr>
<tr>
<td>5.20</td>
<td>Convergence to Optimum vs Population Size, $\ell = 6$</td>
<td>73</td>
</tr>
<tr>
<td>5.21</td>
<td>Convergence to Optimum vs Population Size, $\ell = 8$</td>
<td>74</td>
</tr>
<tr>
<td>5.22</td>
<td>IPM and SGA converge to identical populations, $\ell = 4$</td>
<td>75</td>
</tr>
<tr>
<td>5.23</td>
<td>IPM and SGA converge to identical populations, $\ell = 6$</td>
<td>76</td>
</tr>
<tr>
<td>5.24</td>
<td>IPM and SGA converge to identical populations, $\ell = 8$</td>
<td>77</td>
</tr>
<tr>
<td>5.25</td>
<td>IPM and SGA converge to identical populations, $r_t = 128$</td>
<td>78</td>
</tr>
<tr>
<td>5.26</td>
<td>IPM and SGA converge to identical populations, $r_t = 8\sqrt{2^\ell}$</td>
<td>79</td>
</tr>
<tr>
<td>5.27</td>
<td>IPM and SGA converge to identical populations, $r_t = 4\ell$</td>
<td>80</td>
</tr>
<tr>
<td>5.28</td>
<td>Hamming Distance to Optimum String: IPM (250 experiments), $\ell = 4$</td>
<td>81</td>
</tr>
<tr>
<td>5.29</td>
<td>Hamming Distance to Optimum String: IPM (300 experiments), $\ell = 6$</td>
<td>82</td>
</tr>
<tr>
<td>5.30</td>
<td>Hamming Distance to Optimum String: IPM (250 experiments), $\ell = 8$</td>
<td>83</td>
</tr>
<tr>
<td>5.31</td>
<td>Hamming Distance to Optimum String: IPM, $\ell = 4, 6, 8$</td>
<td>84</td>
</tr>
<tr>
<td>5.32</td>
<td>SGA Zero Hamming Distance to Optimum (by Population Size), $\ell = 4$</td>
<td>85</td>
</tr>
<tr>
<td>5.33</td>
<td>SGA Zero Hamming Distance to Optimum (by Population Size), $\ell = 6$</td>
<td>86</td>
</tr>
<tr>
<td>5.34</td>
<td>SGA Zero Hamming Distance to Optimum (by Population Size), $\ell = 8$</td>
<td>87</td>
</tr>
<tr>
<td>5.35</td>
<td>SGA Zero Hamming Distance to Optimum (by String Length), $r_t = 128$</td>
<td>88</td>
</tr>
<tr>
<td>5.36</td>
<td>SGA Zero Hamming Distance to Optimum (by String Length), $r_t = 8\sqrt{2^\ell}$</td>
<td>89</td>
</tr>
<tr>
<td>5.37</td>
<td>SGA Zero Hamming Distance to Optimum (by String Length), $r_t = 4\ell$</td>
<td>90</td>
</tr>
<tr>
<td>5.38</td>
<td>SGA Generations to Convergence vs Epistasis, $\ell = 4$</td>
<td>91</td>
</tr>
<tr>
<td>5.39</td>
<td>SGA Generations to Convergence vs Epistasis, $\ell = 6$</td>
<td>92</td>
</tr>
<tr>
<td>5.40</td>
<td>SGA Generations to Convergence vs Epistasis, $\ell = 8$</td>
<td>93</td>
</tr>
<tr>
<td>5.41</td>
<td>SGA Generations to Convergence vs Epistasis, $r_t = 128$</td>
<td>94</td>
</tr>
<tr>
<td>5.42</td>
<td>SGA Generations to Convergence vs Epistasis, $r_t = 8\sqrt{2^\ell}$</td>
<td>95</td>
</tr>
<tr>
<td>5.43</td>
<td>SGA Generations to Convergence vs Epistasis, $r_t = 4\ell$</td>
<td>96</td>
</tr>
<tr>
<td>5.44</td>
<td>IPM Generations to Convergence vs Epistasis, $\ell = 4$</td>
<td>97</td>
</tr>
<tr>
<td>5.45</td>
<td>IPM Generations to Convergence vs Epistasis, $\ell = 6$</td>
<td>98</td>
</tr>
</tbody>
</table>
5.46 IPM Generations to Convergence vs Epistasis, \( \ell = 8 \) ........................................ 99
5.47 IPM Generations to Convergence vs Epistasis, \( r_\ell = 128 \) ................................. 100
5.48 IPM Generations to Convergence vs Epistasis, \( r_\ell = 8\sqrt{2^\ell} \) ........................ 101
5.49 IPM Generations to Convergence vs Epistasis, \( r_\ell = 4\ell \) ................................. 102
5.50 Sample Scatterplot .......................................................... 104
5.51 Fitness vs Hamming Distance to Optimum (Epistasis \( \frac{9}{5} - \frac{4}{9} \)) ........................ 105
5.52 Fitness vs Hamming Distance to Optimum (Epistasis \( \frac{5}{5} - \frac{2}{9} \)) ........................ 106
5.53 Alpha vs Epistasis for \( \ell = 3, 4, 6, 8 \) ..................................................... 110
5.54 Empirical \( \alpha \) vs \( \varepsilon \) for \( \ell = 3, 4, 6, 8 \) ..................................................... 111
List of Tables

2.1 Proportional Fitness Example ............................................. 6

5.1 Random Fitness Vector Statistics ................................. 112
5.2 Random Fitness Vector Epistasis Bounds .................. 112
Chapter 1

Introduction

The set of all possible solutions to a problem is called its “solution space”. One approach to solving an optimization problem is to evaluate all the members of its solution space, retaining always the optimum of those evaluated. If the number of solutions is so large that an exhaustive search of the solution space is impractical, then it is possible that an algorithm which guides the search toward the more likely solutions will be feasible. For example, a hill-climbing (“greedy”) algorithm examines solutions “near” the current one, and replaces the current solution with a better one. The process is repeated, and stops when there is no nearby solution which is better than the current solution. However, what is obtained may be only a local optimum, rather than the global one. Another algorithm, simulated annealing, attempts to overcome the “local optimum” trap of greedy algorithms. It does so by applying the “greedy” choice to cases where a (randomly selected) nearby solution is better. However, an inferior solution has a non-zero probability of becoming the current solution. This probability is based on the ratio of the difference between the current and proposed solutions to a threshold value. As proposed solutions are evaluated, the threshold value is gradually decreased, thereby reducing the likelihood of accepting a solution worse than the current one. It can be shown that the global optimum will be
found by the simulated annealing algorithm if the threshold decreases at a sufficiently slow rate. In contrast to the search for the optimal solution in simulated annealing, which is based on the physical process of annealing, solutions in a genetic algorithm "evolve" in a manner which models the biological principles of sexual reproduction and survival of the fittest.

A genetic algorithm begins with an initial population, a collection of solutions which are (typically) selected at random, with replacement, from the solution space. Each solution in the population is assigned a fitness value determined by the function being optimized. The population is then modified in a three-step process to produce a new population, which represents the next generation. The three steps are: selection — in which the principle of survival of the fittest is applied to the current population of solutions; crossover — in which new solutions are produced via recombination of pairs of solutions in analogy to sexual reproduction; mutation — in which the solutions are diversified via random modifications, in analogy to genetic mutation.

The fitness-based selection step is central to the genetic algorithm. It provides the only guiding influence of the optimization objective function upon the search of the solution space. By implementing an "evolutionary bias" toward the selection of higher fitnesses solutions, lower fitness solutions tend to "die out" of the population.

The crossover step broadens the subset of the solution space under consideration by separating a pair of solutions into components, then combining these components to form new solutions — which likely have more in common with their "parents" than randomly-generated solutions would.

The mutation step has the potential to enlarge the current subset of the solution space under consideration by modifying a solution in such a way that it contains a component not present in any "parents".

Since crossover and mutation are random processes not guided by fitness, it is likely that many of the solutions produced by these processes will be inferior to the
"parent" solutions used to produce them. The next selection step should eliminate the inferior solutions.
Chapter 2

Background

1. The Simple Genetic Algorithm (SGA)

1.1. Terms and Representations

When the Simple Genetic Algorithm is used as a search heuristic:

1. The solution space is the set of all possible solutions.

2. The term "individual" refers to one possible solution, one member of the solution space. Each solution is encoded as a bitstring of length $\ell$.

3. A population is a bag (a set with duplications) containing individuals, and usually refers to the individuals comprising a particular generation. The population size (the number of individuals in the bag) is represented by $r$.

1.2. The Algorithm

The term "Genetic Algorithm" encompasses a diverse collection of solution-space-search algorithms. The Simple Genetic Algorithm is an iterative, stochastic genetic algorithm which can be described as follows:
1. Create the initial population by selecting $r$ individuals at random (with replacement) from the solution space.

2. Determine the fitness of each individual in the current population.

3. Until a new generation of $r$ individuals is created from the current population,
   (a) Use the selection procedure to obtain two "parent" solutions from the current population,
   (b) Apply crossover to the parents to obtain a "child" solution,
   (c) Apply mutation to the child,
   (d) Put the child in the new generation.

4. The newly-created generation becomes the current population.

5. If the stopping criteria is not satisfied, then continue from step 2: otherwise, stop.

1.3. Distinguishing Characteristics of the SGA

The simple genetic algorithm can be defined by how it handles key features of genetic algorithms, which include:

1. The Selection Method

   "Survival of the fittest" refers to the procedure to select those individuals which will contribute genetically to the next generation. The SGA bases selection on proportional fitness: the probability that an individual will be selected is the ratio of its fitness to the aggregate fitness of all individuals in its generation.

   For example, suppose we have individuals $x_1$, $x_2$, $x_3$, $x_4$, and $x_5$, with fitnesses 3, 1, 9, 5, and 7 respectively. The sum of fitnesses is thus 25. The relative
fitness of $x_1$ is therefore $3/25$ or 0.12, which also represents the probability that $x_1$ will be selected. Similarly, the highest probability of selection would be 0.36 for individual $x_3$, as seen in Table 2.1.

<table>
<thead>
<tr>
<th>fitness of individual</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>probability of selection</td>
<td>0.12</td>
<td>0.04</td>
<td>0.36</td>
<td>0.20</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Table 2.1: Proportional Fitness Example

Proportional fitness has a drawback when a genetic algorithm is used for optimization: whenever the population is reduced to several types of individuals of approximately the same fitness, the bias toward selecting the best-fit individuals becomes small. Consequently, the search loses much of the effectiveness of a fitness-based approach.

2. Crossover Methods

Crossover refers to the merging of genetic material taken from two individuals ("parents") from one generation to form a new individual ("child") for the next generation. The SGA also allows cloning as a means of passing genetic information from one generation to the next. The crossover rate is the probability that crossover will be performed to produce a child, rather than cloning one of the parents.

When dealing with binary strings, three crossover methods are prevalent: one-point, two-point, and uniform.

(a) One-Point Crossover
The simplest way to exchange portions of fixed-length bitstrings is to “cut” both strings at the same bit position (the “crossover point”), then create the new strings by attaching the first part of each string to the last part of the other. For a string with $\ell$ bits, the randomly selected crossover point falls in the range 1 to $\ell - 1$ (A crossover point of 0 or $\ell$ would indicate a cloning).

For example, with the crossover point at position 6, the strings

\[
\begin{align*}
011001 & | 0110001001101 & \text{parent 1} \\
010010 & | 1100100111010 & \text{parent 2}
\end{align*}
\]

produce children

\[
\begin{align*}
011001 & | 1100100011010 & \text{child 1} \\
010010 & | 0110001001101 & \text{child 2}
\end{align*}
\]

The SGA randomly selects one of the children to add to the next generation.

(b) Two-Point Crossover

A bitstring in which the first and last bits are assumed to be adjacent forms a closed loop. In that case, two crossover points must be specified in each string in order to obtain the components to be exchanged. (One-point crossover can be viewed as a special case of two-point crossover in which one of the crossover points has been decided already). The crossover points are randomly selected in the range 0 to $\ell - 1$ for an $\ell$-bit string.

For example, consider crossover points at positions 6 and 15:

\[
\begin{align*}
011001 & | 0110001000101 & \text{parent 1} \\
010010 & | 1100100010110 & \text{parent 2}
\end{align*}
\]

produce

\[
011001 | 11001000101101 & \text{child 1}
\]
Again, the SGA randomly selects one of the children to add to the next generation.

(c) Uniform Crossover

Suppose a child is created by selecting each of its bits randomly from one parent or the other. In a sense, the number of crossover points and their positions are selected at random. This is most easily described in terms of “masking”. A mask is a binary string of the same length as the parents: A 0 in a position indicates the first parent provides that bit value for the child, and a 1 indicates the source is the other parent. (If it were necessary to produce both children, the mask and its bitwise complement would be used.)

For example,

\[
\begin{array}{ccccccc}
011 & 01 & 01 & 01 & 10 \\
101 & 00 & 11 & 01 & 10 \\
111 & 00 & 10 & 01 & 10 \\
101 & 01 & 01 & 01 & 10 \\
\end{array}
\]

(To simplify verification, the strings above are split where the mask changes between 1 and 0 — which corresponds to switching between parent strings — and the chosen bits are underlined.)

This example could be described as 7-point crossover for a linear string or 8-point crossover for a “looped” string. Conversely, one-point and two-point crossover can be considered to be examples of masking, where the mask has either all 0’s or all 1’s contiguous, or both. In that representation, the mask for the one-point crossover example above would be either 00000011111111111 or its complement 111110000000000000, and the
mask for the two-point crossover would be either 0000001111111110000 or its complement 11111000000001111, depending on which child is to be created.

It is possible for both parents to have identical bit-patterns in some positions of their representations. In that case, the child will be limited to those bits in those positions, and cannot represent a significant portion of the possible strings. (For an n-bit overlap of the two strings, the child can represent only \((1/2)^n\) of the possible strings.)

For example, the parents

\[101001\] and \[01100\]

can not produce a child with a bit-pattern like

**0*** \(\text{or} \) ****1* \(\text{or} \) **0*1*

because neither parent has a 0 in the 3rd position or a 1 in the 5th position. (An * means the bit can be either a 1 or a 0.)

The effect of crossover is to add variation to the search, but only within the limits imposed by the parents' bit patterns.

3. The Mutation Method

Mutation is a random modification in the representation of an individual, and occurs without interaction with any other individuals. The mutation rate is the probability that a bit will be changed. It is applied to each bit in an individual.

For example, a mutation rate of 0.001 means each bit has a 1 in 1000 chance of changing. It can also be interpreted to mean approximately every thousandth bit (on average) used in the creation of new individuals will be changed to its complement.
Mutation has the potential to add diversity to the search by modifying a child to obtain a bit pattern not achievable solely by crossover (because of the limitations imposed by parents' bit patterns, as described above).

2. The Fitness Function and Epistasis

2.1. Fitness and Fitness Functions

The fitness of an individual corresponds to the degree of optimality of the solution represented by that individual. The assignment of fitnesses to individuals is done by a fitness function. We assume that all fitnesses are non-negative. An example of an arbitrary assignment of fitnesses to all individuals represented by 3-bit strings is:

<table>
<thead>
<tr>
<th>string</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>101</td>
<td>4</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>3</td>
</tr>
</tbody>
</table>

Generally, a fitness function is expressed as a rule for calculating the fitness of a given bitstring, based on its bit pattern.

An important example of such a rule is the affine fitness function \( f \): Each of the \( \ell \) bits of string \( B \) is a variable \( b_i \), and the fitness of the string \( B = b_1 \ b_2 \ \ldots \ b_\ell \) is

\[
f(B) = f_B = c_0 + c_1 \ b_1 + c_2 \ b_2 + \ldots + c_\ell \ b_\ell
\]
Consider an affine fitness function for 3-bit strings, with $c_0 = 3$, $c_1 = -2$, $c_2 = 5$, and $c_3 = 0.47$. The fitness of string $b_1b_2b_3 = 101$, for example, is

$$f_{101} = c_0 + c_1(1) + c_2(0) + c_3(1) = 1.47$$

and the complete fitness function is

<table>
<thead>
<tr>
<th>string</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>3</td>
</tr>
<tr>
<td>001</td>
<td>3.47</td>
</tr>
<tr>
<td>010</td>
<td>8</td>
</tr>
<tr>
<td>011</td>
<td>8.47</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1.47</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>6.47</td>
</tr>
</tbody>
</table>

The fitness function maps three binary digits to a real number by summing $c_0$ and the $c_i$ coefficients which are associated with the "1" bits in the string.

The fitness function can also be represented as a (column) vector. To do so, associate each (unique) bitstring of length $\ell$ with a dimension in $\mathbb{R}^N$, where $N = 2^\ell$ is the number of possible bitstrings. (The bitstrings associated with the components of the column vector will appear in ascending order of their associated decimal values.) Let each bitstring's fitness value be recorded as the component of the fitness vector in that dimension. Thus, the fitness function is represented by a vector of dimension $N = 2^\ell$ with real coefficients. For the 3-bit affine fitness function above, the 8-dimensional fitness vector is

$$(3 \ 3.47 \ 8 \ 8.47 \ 5 \ 5.47 \ 10 \ 10.47)^T.$$
The simplest affine fitness function is the "counting ones" fitness function, in which each coefficient $c_i = 1$ and the additive constant $c_0 = 0$. The "ones" fitness of a string equals the number of 1's in its binary representation.

Affine fitness functions have some useful properties.

1. Complementing bit position $i$ always changes the fitness by $c_i$, regardless of the bits at the other positions.

2. Under crossover, the sum of the fitnesses of the parents equals the sum of the fitnesses of the children. (Each bit position's contribution to fitness is independent of the other bit positions in the string. Crossover exchanges bits between strings, but does not change their positions in the strings, so their contributions are not changed. Therefore, the same values are summed in the children as in the parents.)

Affine fitness functions are not adequate to express all fitness schemes. If a fitness function is not affine, it is said to be "epistatic". For example, a fitness function in which each individual is assigned a randomly selected fitness value will almost always be epistatic.

The term "epistasis" has its origin in genetics, where it refers to an interaction between loci in a genetic string. To illustrate how this interaction affects a fitness function, consider an epistatic fitness function in which only two of the bit positions interact. Then four fitness values will be associated with the four strings which differ only in those two bit positions. Let $f_{00}$ be the fitness value assigned to the string having the 00 pattern in those two positions, $f_{01}$ assigned to the string having the 01 pattern, $f_{10}$ to the string with 10, and $f_{11}$ to the string with 11. An affine fitness function would also produce four values for those same four strings. But in the epistatic case, the fitness change associated with changing the first bit depends on what the second bit is: $f_{00} - f_{10}$ need not equal $f_{01} - f_{11}$. For affine functions,
however, the fitness change would always be the same in both cases. (This is the first of the useful properties of affine fitness functions, mentioned above. The second property is also lost with epistatic fitness functions.)

Consider the space of fitness functions over strings of length \( \ell \). Let \( N = 2^\ell \), the number of dimensions in the space. For each fitness function

- the domain is the \( N \) strings of length \( \ell \) (each string representing one of the \( N \) dimensions of the the fitness function vector),

- the range is the set of non-negative real numbers \( \mathbb{R} \) (the fitnesses of the strings, or equivalently, the components of the fitness function vector),

- every fitness function is a point in \( \mathbb{R}^N \),

- the space of fitness functions is the non-negative orthant of \( \mathbb{R}^N \).

The set of affine functions is a linear subspace of \( \mathbb{R}^N \). (A scalar multiple of a affine function is affine, and the sum of affine functions is affine.) Thus the space of affine fitness functions is the intersection of the non-negative orthant of \( \mathbb{R}^N \) with the space of affine functions.

### 2.2. The Walsh Transform

Consider the Walsh Transformation matrix of dimension \( 2^{\ell+1} \) by \( 2^{\ell+1} \)

\[
W^{\ell+1} = 2^{-\frac{1}{2}} \begin{pmatrix}
  W^\ell & W^\ell \\
  W^\ell & -W^\ell
\end{pmatrix}
\]

where

\[
W^0 = (1)
\]

and \( \ell \) is the bitstring length. (The superscript indicates the dimension.)
Lemma 2.1 The Walsh matrix is symmetric (i.e., $W = W^T$).

Proof: (by induction)

Base Case: $W^0$ is trivially symmetric, as is $W^1$.

Induction Hypothesis: $W^\ell = (W^\ell)^T$

Induction Step: Clearly, $-W^\ell$ is symmetric. Then, $W^{\ell+1} = (W^{\ell+1})^T$ is also symmetric, because

$$2^{-\frac{1}{2}} \begin{pmatrix} W^\ell & W^\ell \\ W^\ell & -W^\ell \end{pmatrix} = 2^{-\frac{1}{2}} \begin{pmatrix} (W^\ell)^T & (W^\ell)^T \\ (W^\ell)^T & -(W^\ell)^T \end{pmatrix} = 2^{-\frac{1}{2}} \begin{pmatrix} W^\ell & W^\ell \\ W^\ell & -W^\ell \end{pmatrix}^T.$$

Lemma 2.2 The Walsh matrix is orthogonal (i.e., $W^T W = I$).

Proof: (by induction)

Base Case: ($\ell = 0$ is trivial: $(1)(1) = (1)$, which is $I$ in one dimension.) Consider $\ell = 1$:

$$(W^1)^T W^1 = 2^{-\frac{1}{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} 2^{-\frac{1}{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = 2^{-1} \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = I.$$ 

Induction Hypothesis: $(W^\ell)^T W^\ell = I$.

Induction Step:

$$(W^{\ell+1})^T W^{\ell+1} = (2^{-\frac{1}{2}})^2 \begin{pmatrix} W^\ell & W^\ell \\ W^\ell & -W^\ell \end{pmatrix} \begin{pmatrix} W^\ell & W^\ell \\ W^\ell & -W^\ell \end{pmatrix}$$

$$= 2^{-1} \begin{pmatrix} 2W^\ell W^\ell & 0 \\ 0 & 2W^\ell W^\ell \end{pmatrix}$$

$$= 2^{-1} \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} = I.$$
by the induction hypothesis, since $W^\ell W^\ell = (W^\ell)^T W^\ell$ by the symmetry of $W$.

From $W^T W = I$, and the definition of inverse ($W^{-1} W = WW^{-1} = I$), it is also clear $W^T = W^{-1}$. Thus, we have the useful relationship $W = W^T = W^{-1}$.

**Notation**

Let $\vec{1}$ represent the vector of all ones, $(1 1 \ldots 1)^T$, and $\vec{0}$ represent the vector of all zeros, $(0 0 \ldots 0)^T$.

Define vector $\vec{d}$ by $\vec{d} = (2^0 2^1 \ldots 2^{\ell-1})^T$.

Associate vector $\vec{x} = (x_1 x_2 \ldots x_\ell)^T$, where $x_i \in \{0, 1\}$, with the number $\vec{x}^T \vec{d}$ and with string $S_x = x_\ell \ldots x_2 x_1$. For example, $(1 1 0)^T \equiv 3 \equiv 011$.

For some matrix $A$ having $r$ rows and $c$ columns, let $A_k$ denote the (column) vector whose $r$ elements are the $k^{th}$ column of $A$, where $k = 0, 1, \ldots, c-1$.

**Lemma 2.3** $W_0 = (2^{-\frac{1}{2}}) \vec{I}$.

**Proof:** (by induction)

**Base Case:** $W^0 = (1)$, so $W_0 = (1) = \vec{I}$.

**Induction Hypothesis:** $W_\ell^\ell = (2^{-\frac{1}{2}}) \vec{I}$.

**Induction Step:**

$$W_{\ell+1}^\ell = \left[ 2^{-\frac{1}{2}} \begin{pmatrix} W^\ell & W^\ell \\ W^\ell & -W^\ell \end{pmatrix} \right] = 2^{-\frac{1}{2}} \begin{pmatrix} W^\ell \\ W^\ell \end{pmatrix} \cdot 2^{-\frac{1}{2}} \begin{pmatrix} 2^{-\frac{1}{2}} \\ \vec{I} \end{pmatrix} = (2^{-\frac{\ell+1}{2}}) \vec{I}$$

by the induction hypothesis.
For string length \( \ell \), define the \( 2^\ell \times \ell \) matrix \( X^* \) by

\[
X^*_j = (2^{\frac{j}{2}})W_{2j}
\]

(2.1)

where \( j = 0, 1, \ldots, \ell - 1 \). For example, with \( \ell = 3 \),

\[
X^* = \begin{pmatrix}
+1 & +1 & +1 \\
-1 & +1 & +1 \\
+1 & -1 & +1 \\
-1 & -1 & +1 \\
+1 & +1 & -1 \\
-1 & +1 & -1 \\
+1 & -1 & -1 \\
-1 & -1 & -1
\end{pmatrix}
\]

Let \( X = \frac{1}{2}(\tilde{1}\tilde{1}^T - X^*) \). Then, for \( \ell = 3 \),

\[
X = \begin{pmatrix}
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 1 \\
0 & 1 & 1 \\
1 & 1 & 1
\end{pmatrix}
\]

Observe that the rows of \( X \) are the vector representations of the consecutive integers from 0 to \( 2^\ell - 1 \). Note also that \( X^* = (\tilde{1}\tilde{1}^T - 2X) \). \( X^* \) is the bipolar (\( \{+1, -1\} \)) form of the unipolar (\( \{0, 1\} \)) \( X \): Where \( X^* \) has an entry of +1, \( X \) has an entry of 0, and where \( X^* \) has an entry of -1, \( X \) has an entry of 1.
Define the fitness vector \( f^t \) by 

\[
\begin{pmatrix}
 f^t_0 \\
 f^t_1 \\
 f^t_2 \\
 \vdots \\
 f^t_{2^t-1}
\end{pmatrix}
\]

where \( f^t_x = f^t(\vec{x}) \), and where the (optional) superscript indicates the bitstring length. For example,

\[
f^2 = \begin{pmatrix}
 f^2_0 \\
 f^2_1 \\
 f^2_2 \\
 f^2_3
\end{pmatrix}
\]

We view the Walsh transform as a change-of-basis transformation. In other words, if \( f \in \mathbb{R}^N \), then \( f_i \) is the \( i^{th} \) coordinate of \( f \) in the "standard" basis \( (e_0, e_1, \ldots, e_{2^t-1}) \), whereas \( \hat{f}_i \) is the \( i^{th} \) coordinate of the same vector in the Walsh basis \( (W_0, W_1, \ldots, W_{2^t-1}) \). \( (W e_i = W_i \text{ since } W = W^T. ) \) Note our numbering of the \( 2^t \) basis elements' subscripts begins at 0 rather than 1.

Note \( \hat{f} \) represents \( Wf \), the fitness vector \( f \) in the Walsh basis. \( (f \leftrightarrow W \rightarrow \hat{f}) \)

**Theorem 2.1** \( f(\vec{x}) \) is an affine vector of \( \vec{x} = (x_1, x_2, \ldots, x_t)^T \), \( x_i \in \{0, 1\} \leftrightarrow \hat{f}_k = 0 \) for \( k \neq 0, 1, 2, 4, \ldots, 2^t - 1 \)

**Proof of \( \Rightarrow \):** Let \( \vec{c} = (c_1, c_2, \ldots, c_t)^T \).

Then for affine function \( f \)

\[
f_x = c_0 + \vec{x}^T \vec{c}
\]

and

\[
f = \begin{pmatrix}
 f_0 \\
 f_1 \\
 \vdots \\
 f_{2^t-1}
\end{pmatrix}
= c_0 I + \vec{X} \vec{c} = c_0 I + \frac{1}{2}(I^T - X^*) \vec{c} = I(c_0 + \frac{1}{2}I^T \vec{c}) - \frac{1}{2} X^* \vec{c}.
\]

Thus,

\[
\hat{f} = W f = W I(c_0 + \frac{1}{2}I^T \vec{c}) - \frac{1}{2} W X^* \vec{c}.
\]

(2.2)
To evaluate the first term, begin with the orthogonality of $W$, that inner products of columns (or rows) equal zero unless the operands are equal, in which case the result is 1.

$$W_i^T W_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

In particular,

$$W_0^T W_0 = (2^{-\frac{1}{2}}) \bar{I}^T (2^{-\frac{1}{2}}) \bar{I} = (2^{-\frac{1}{2}}) (2^{-\frac{1}{2}}) 2^\ell = 1.$$

Using this and the fact that $\bar{I} = (2^{\frac{1}{2}}) W_0$ (see Lemma 2.3),

$$W_i^T \bar{I} = W_i^T (2^{\frac{1}{2}}) W_0 = \begin{cases} (2^{\frac{1}{2}}) & \text{if } i = 0 \\ 0 & \text{otherwise.} \end{cases}$$

So $W \bar{I}$ can be rewritten as

$$W \bar{I} = (2^{\frac{1}{2}}) \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The expression $(c_0 + \frac{1}{2} \bar{I}^T \bar{c})$ equals $\frac{1}{2}(2c_0 + c_1 + \cdots + c_\ell)$, a constant, so the first term in Equation 2.2 becomes

$$W \bar{I}(c_0 + \frac{1}{2} \bar{I}^T \bar{c}) = \frac{1}{2} (2^{\frac{1}{2}}) \begin{pmatrix} 2c_0 + c_1 + \cdots + c_\ell \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

For the second term in Equation 2.2, use $X_j^* = (2^{\frac{1}{2}}) W_2^\ell$, (Equation 2.1) and the orthogonality of $W$ to obtain

$$\frac{1}{2} W X^* = \frac{1}{2} (2^{\frac{1}{2}}) V^\ell.$$
where $Y^\ell$ is a $2^\ell \times \ell$ matrix with entries

$$Y_{ij} = \begin{cases} 1 & \text{if } i = 2^j \\ 0 & \text{otherwise.} \end{cases}$$

Then $Y^\ell \bar{c} = (0 \ c_1 \ c_2 \ 0 \ 0 \ 0 \ c_4 \ 0 \ \ldots)^T$ since only rows $1, 2, 4, 8, \ldots, 2^j$ of $Y^\ell$ have a non-zero entry.

And finally, combining terms, $\hat{f} = Wf$ can be written as

$$\hat{f} = \frac{1}{2} \begin{pmatrix} 2c_0 + c_1 + \cdots + c_\ell \\ -c_1 \\ -c_2 \\ 0 \\ -c_3 \\ 0 \\ 0 \\ -c_4 \\ 0 \\ \vdots \end{pmatrix}$$

(2.3)

Because of the structure of $Y^\ell$, it is clear that $\hat{f}_k = 0$ for $k \neq 0, 1, 2, 4, \ldots, 2^{\ell-1}$, as was to be shown.

Proof of $\Leftarrow$ : Given $f^\ell = (\hat{f}_0 \ \hat{f}_1 \ \hat{f}_2 \ 0 \ \hat{f}_4 \ 0 \ 0 \ 0 \ \hat{f}_8 \ 0 \ \ldots)^T$, define the constants $c'_1, c'_2, \ldots, c'_\ell$ in terms of the $\hat{f}_0, \ \hat{f}_1, \ \hat{f}_2, \ \hat{f}_4, \ \ldots, \ \hat{f}_{2^{\ell-1}}$ as follows:

$$c'_i = -2 \left(2^{-\frac{i}{2}}\right) \hat{f}_{2^{i-1}}$$

for $i = 1, 2, \ldots, \ell$. 
Then, by defining $c_0$ to be

$$c_0 = \left(2^{-\frac{1}{2}}\right) (\hat{f}_0 + \hat{f}_1 + \hat{f}_2 + \cdots + \hat{f}_{2^t-1}) = \left(2^{-\frac{1}{2}}\right) \vec{I}^T \hat{f}$$

it is possible to write $\hat{f}$ in terms of the $c_i$'s to obtain

$$\hat{f} = \frac{1}{2} \left(2^\frac{t}{2}\right) \begin{pmatrix} 2c_0 + c_1 + \cdots + c_t \\ -c_1 \\ -c_2 \\ 0 \\ -c_3 \\ 0 \\ 0 \\ -c_4 \\ 0 \\ \vdots \end{pmatrix}.$$

It is clear from equation 2.3 that this represents the Walsh transformation of the linear function $f' = c_0 + \vec{x}^T \vec{c}$ where $\vec{c} = (c_1, c_2, \ldots, c_t)$. In other words, $\hat{f} = Wf'$, where $f'$ is a linear function, as was to be shown.

This theorem indicates that the space of affine functions is the space spanned by the following subset of the Walsh basis vectors (corresponding to the non-zero components of $\hat{f}$): $\{W_0, W_1, W_2, W_4, \ldots, W_{2^t}\}$.

**Arbitrary Fitness Vector**

Consider an arbitrary fitness vector $f = (f_0, f_1, \ldots, f_{2^t-1})^T$, whose components $(f_i)$ are non-negative numbers.
Then \( \hat{f} \) has a natural orthogonal decomposition

\[
\begin{pmatrix}
\hat{f}_0 \\
\hat{f}_1 \\
\hat{f}_2 \\
\hat{f}_3 \\
\vdots \\
\hat{f}_{2^t-1}
\end{pmatrix}
= \begin{pmatrix}
\hat{f}_0 \\
\hat{f}_1 \\
\hat{f}_2 \\
\hat{f}_3 \\
\vdots \\
\hat{f}_{2^t-1}
\end{pmatrix} = \begin{pmatrix}
0 \\
\hat{f}_1 \\
\hat{f}_2 \\
0 \\
\vdots \\
\hat{f}_{2^t-1}
\end{pmatrix} + \begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
\vdots \\
\hat{f}_{2^t-1}
\end{pmatrix} = C_f + L_f + E_f.
\]

The \( \hat{f} \) is decomposed into

- a "constant" part: \( C_f = \hat{f}_0 W_0 \),
- a "linear" part: \( L_f = \sum_{i=0}^{t-1} \hat{f}_{2^i} W_{2^i} \), and
- an "epistatic" (non-linear) part: \( E_f = \hat{f} - C_f - L_f \),

each of which represents the Walsh transformation of a corresponding decomposition on \( f \):

\[
\hat{f} = Wf = WC_f + WL_f + WE_f.
\]

By applying \( W^{-1} \) to \( \hat{f} \), the decomposition of \( f \) into constant, linear, and epistatic components \( (C_f, L_f, \text{ and } E_f) \) can be obtained. Thus, \( f = C_f + L_f + E_f \) can be
written as

\[
\begin{pmatrix}
  c_0 + r_0 \\
  c_0 + c_1 + r_1 \\
  c_0 + c_2 + r_2 \\
  c_0 + c_1 + c_2 + r_3 \\
  c_0 + c_3 + r_4 \\
  \vdots \\
  c_0 + c_1 + \cdots + c_2 + r_{2^t - 1}
\end{pmatrix} = \begin{pmatrix}
  c_0 \\
  c_0 \\
  c_0 \\
  c_0 \\
  c_0 \\
  \vdots \\
  c_0
\end{pmatrix} + \begin{pmatrix}
  0 \\
  c_1 \\
  c_2 \\
  c_1 + c_2 \\
  c_3 \\
  \vdots \\
  c_1 + \cdots + c_t
\end{pmatrix} + \begin{pmatrix}
  r_0 \\
  r_1 \\
  r_2 \\
  r_3 \\
  r_4 \\
  \vdots \\
  r_{2^t - 1}
\end{pmatrix}
\]

where

\[
c_0 = \left(2^{\frac{t}{2}}\right) (T^T \hat{J}) \quad \text{for } C_f,
\]

\[
c_i = -2 \left(2^{\frac{i}{2}}\right) \hat{J}_{2^i} \quad \text{for } L_f,
\]

and for $E_f$,

\[
r_0 = f_0 - c_0, \quad \text{and}
\]

\[
r_i = f_i - (c_0 + e_i^T L_f),
\]

where $e_i$ is the unit vector in the $i$ direction.

### 2.3. Epistasis

**A Definition for Epistasis**

A natural measure for the epistasis of a fitness vector is the proportion of fitness attributable to epistasis. We will define the epistasis $\epsilon$ of fitness vector $f$ to be

\[
\epsilon_f = \frac{E_f^2}{L_f^2 + E_f^2}
\]

where $E_f^2$ represents $E_f^T E_f$ and $L_f^2$ represents $L_f^T L_f$. 
Figure 2.1: Constant, Linear, and Epistatic Components of the Fitness Function

This definition of epistasis is a logical development from earlier work by Davidor [2] and Aizawa [3]. We will present their work using their notations, and indicate the correspondence between the terms used in their notations and ours.

Recall that our definition of the Walsh matrix associates a \( 2^{-\frac{j}{2}} \) factor with both \( W^{-1} \) and \( W \), a symmetry which simplifies theoretical development, whereas the "standard" (Paley) definition associates \( 2^{-\frac{j}{2}} \) only with \( W^{-1} \). Due to that difference between Walsh definitions, a factor of \( 2^{-\frac{j}{2}} \) will appear at times when the formulas from Davidor and Aizawa are rewritten in our notation.

In his paper on GA-Hardness, Davidor [2] approaches epistasis as a "non-linearity" effect. For a linear fitness function, the fitness of any string \( S \) can be found in two ways:

1. Obtain it directly from the fitness function: \( v(S) \).

2. Obtain it by adding to the "average fitness of all strings" \( \bar{V} = \frac{1}{N} \sum_{i=0}^{N-1} v(S_i) \), the
effect \( E_j(b) \) which each bit \( b \) at each bit position \( j \) has on \( \bar{V} \). (The "effect" is the (average) fitness deviation (from the average fitness of all strings) attributable to having a particular bit at a particular position in a string.) An \( \ell \times 2 \) table of such effects is pre-calculated by finding the average fitness of all strings having bit \( b \in \{0, 1\} \) at bit position \( j \), then subtracting \( \bar{V} \). The calculated genic fitness \( A(S) \) of string \( S \) is then the average fitness \( \bar{V} \) plus the sum of the "effects" due to each bit at each position in the string: \( A(S) = \sum E_j(b) + \bar{V} \).

For a non-linear (epistatic) function, however, the fitness values resulting from the two methods are not the same, and Davidor defines the quantity *epistatic variance* \( \sigma_e^2 \) based on the difference:

\[
\sigma_e^2 = \frac{1}{N} \sum_{S \in \{0, 1\}^\ell} (v(S) - A(S))^2 = \frac{1}{N} \sum_{S \in \{0, 1\}^\ell} \left( v(S) - \sum_{j=1}^{\ell} E_j(b) + \bar{V} \right)^2
\]

where \( N \) is the number of strings used in the calculation.

This method produces an epistasis value for the fitness function when applied to the solution space (for which \( N = 2^\ell \)), and an estimate of the fitness function's epistasis when applied to a sample ("population") of \( N \) strings.

Aizawa [3] presents a formalization of genetic algorithms which makes it possible to combine the characterization of both the search space and the genetic operators within the same framework. One of the basic quantities defined by Aizawa is the total variance \( \sigma_A^2 \) of the solution space

\[
\sigma_A^2 = 2^{-L} \sum_{x \in \mathcal{A}} (F(x) - \mu_A)^2 = 2^{-L} \sum_{x \in \mathcal{A}} F(x)^2 - \mu_A^2
\]

where \( \mathcal{A} \) is the solution space, \( F(x) \) is the fitness of string \( x \), and \( \mu_A = 2^{-L} \sum F(x) \) is the mean fitness.

Aizawa partitions the search space \( \mathcal{A} \) into subspaces. The partitions corresponding to the standard (Cartesian) basis are denoted \( u_1, u_2, \ldots \). The function \( f \) is the
effect on average fitness due to a certain bit pattern — a generalization of Davidor's approach — to allow consideration of multiple bit positions. However, for the standard (Cartesian) partitioning \( (u_i) \), \( f_{(u_i)}(x^{(u_i)}) \) reduces to precisely Davidor's \( E_i(b) \).

In converting from Davidor's notation to Aizawa's: \( N \) becomes \( 2^L \), \( V \) becomes \( \mu_A \), and \( E_i(b) \) becomes \( f_{(u_i)}(x^{(u_i)}) \). Thus, Davidor's definition of epistasis variance becomes

\[
\sigma_e^2 = 2^{-L} \sum_{x \in A} \left( F(x) - \sum_{i=1}^{\ell} f_{(u_i)}(x^{(u_i)}) - \mu_A \right)^2
\]

in Aizawa's notation, which Aizawa shows to be

\[
\sigma_e^2 = \sigma_A^2 - \sum_{i=1}^{\ell} (\omega_{2i-1})^2
\]

where \( \omega_{2i-1} \) is the \( 2^{i-1} \)-th Walsh function (Paley).

To convert Aizawa's notation to ours,

1. \( \omega_i \) becomes \( \left( 2^{-\frac{i}{2}} \right) \hat{f}_i \), and note

\[
\sum_{i=1}^{\ell} (\omega_{2i-1})^2 = 2^{-\ell} \left( \hat{f}_1^2 + \hat{f}_2^2 + \hat{f}_4^2 + \ldots + \hat{f}_{2^{i-1}}^2 \right) = 2^{-\ell} \mathcal{L}_j^2.
\]

2. \( F(x) \) becomes \( f_x \), so that \( \sum F(x)^2 = \sum \hat{f}_i^2 = \hat{f}^2 = \hat{f}^2 \),

since \( (Wf)^T(Wf) = (f^TW^T)(Wf) \).

3. \( \mu_A = 2^{-\ell} \sum_{x \in A} F(x) \) becomes

\[
\mu_A = 2^{-\ell} \sum_i f_i = \left( 2^{-\frac{\ell}{2}} \right) Wf = \left( 2^{-\frac{\ell}{2}} \right) \hat{f}_0 = \left( 2^{-\frac{\ell}{2}} \right) \mathcal{C}_f
\]

since \( W_0f = \left( 2^{-\frac{\ell}{2}} \right) \sum f_i \).

4. \( \sigma_A^2 = 2^{-\ell} f^T f - \frac{1}{2} \hat{f}_0^2 \).

Considering these developments by Davidor and Aizawa, it would seem logical to define the epistasis of a fitness function to be the ratio of epistasis variance \( \sigma_e^2 \)
of Davidor [2] to total variance $\sigma_A^2$ of Aizawa [3]. This is, in fact, equivalent to our definition (see Equation 2.8), as shown below:

**Proposition 2.1** $\varepsilon_f$ is the ratio of epistasis variance $\sigma_e^2$ to total variance $\sigma_A^2$

$$\varepsilon_f = \frac{\sigma_e^2}{\sigma_A^2}.$$  

**Proof:** Combining the definitions of $\sigma_e^2$ and $\sigma_A^2$ to isolate $\sigma_A^2$,

$$\sigma_A^2 = \sigma_e^2 + \sum (\omega_{2i-1})^2 = 2^{-L} \sum F(x)^2 - \mu_A^2.$$  

Thus,

$$\sigma_e^2 + 2^{-L} L_f^2 = 2^{-L} \tilde{f}^2 + 2^{-L} \sigma_f^2.$$  

Comparing with $\tilde{f}^2 = C_f^2 + L_f^2 + \varepsilon_f^2$, we find

$$\sigma_e^2 = 2^{-L} \varepsilon_f^2.$$  

From $\sigma_A^2 = \sigma_e^2 + \sum (\omega_{2i-1})^2$ we have

$$\sigma_A^2 = 2^{-L} \varepsilon_f^2 + 2^{-L} L_f^2.$$  

So

$$\frac{\sigma_e^2}{\sigma_A^2} = \frac{\varepsilon_f^2}{\varepsilon_f^2 + L_f^2}$$

and comparison with Equation 2.8 confirms this is $\varepsilon_f$, as desired. 

As an example of the calculation of $\varepsilon_f$, consider Goldberg’s minimally deceptive function over 3 bits (taken from Davidor [2]):

The fitness vector is $\tilde{f} = (7, 5, 5, 0, 3, 0, 0, 8)^T$ and

$$W \tilde{f} = \tilde{f} = \frac{1}{\sqrt{8}} (28, 2, 2, 8, 6, 12, 12, -14)^T.$$  

From $\tilde{f}^2$, we obtain the lengths of the components:
\[ f_0^2 = 98.0 \quad \text{(constant)} \]
\[ f_1^2 = 0.5 \quad \text{(linear)} \]
\[ f_2^2 = 0.5 \quad \text{(linear)} \]
\[ f_3^2 = 8.0 \quad \text{(epistatic)} \]
\[ f_4^2 = 4.5 \quad \text{(linear)} \]
\[ f_5^2 = 18.0 \quad \text{(epistatic)} \]
\[ f_6^2 = 18.0 \quad \text{(epistatic)} \]
\[ f_7^2 = 24.5 \quad \text{(epistatic)} \]

The linear component is 5.5, the epistatic component 68.5.

Reeves and Wright [4] examined epistasis from the perspective of experimental design methodology, obtaining precisely the results above, which differ from Davidor’s results by a factor of \(2^{-L}\) (the source of which, in our case, is due to the definition of the Walsh transformation matrix, as described above).

**Controlling Epistasis Levels**

It is possible to design a class of fitness vectors which exhibit a spectrum of degrees of epistasis.

The most direct way to obtain such a class is to make the fitness contribution at each bit position in a string be contingent upon the bit values at \(k\) other positions, as well as at that position: the larger the value of \(k\), the higher the epistasis level. Under this model, the epistasis level increases by discrete amounts as the number of interdependent bit positions increases.

Kaufmann’s NK Fitness Landscape [7] illustrates this method. Here \(N\) represents the number of bits comprising an individual, and \(K\) is the number of (adjacent) bits contributing to the fitness at each bit position in the individual. The fitness value of an individual is the sum of the fitness values of each bit position. When \(K = 0\) the
fitness function produced is affine, since the fitness contribution at each bit position is selected according to the bit value at that position. Each increase in $K$ adds another bit position to the list of interdependent bit positions, producing a more epistatic fitness function.

An alternative method to regulate the level of epistasis in a fitness vector is to obtain the fitness value of each component by combining the fitnesses obtained for that component from a linear fitness vector and an epistatic fitness vector. The $\alpha$-linear scheme accomplishes this by combining the contributing vectors in a proportion controlled by the constant $\alpha$.

**The $\alpha$-Linear Fitness Vector:** Given a purely linear fitness vector $L$ (where $WL = C_L + L$), a purely epistatic fitness vector $E$ (where $WE = C_E + E$), and constant $\alpha \in [0, 1]$, the $\alpha$-linear fitness vector $V$ is defined by

$$V = (1 - \alpha)L + (\alpha)E.$$ 

Thus,

$$WV = (1 - \alpha)(C_L + L) + (\alpha)(C_E + E)$$

$$= (1 - \alpha)C_L + (\alpha)C_E + (1 - \alpha)L + (\alpha)E$$

$$= C + (1 - \alpha)L + (\alpha)E.$$ 

So $V$ in terms of $L$ and $E$ is

$$V = WC + (1 - \alpha)WL + (\alpha)WE.$$  

(2.9)

The purely epistatic fitness vector considered here is created in Walsh space. This and equation 2.9 taken together suggest a way to convert an arbitrary fitness vector $f$ into an $\alpha$-linear fitness vector $V_f$: 

1. Transform the arbitrary fitness vector \( f = (f_0, f_1, \ldots, f_{2^\ell-1})^T \) to Walsh space and decompose it to produce \( \hat{f} = C_f + L_f + E_f \).

2. Create the \( \alpha \)-linear fitness vector:

\[
\mathcal{V}_f = C_f + (1 - \alpha)L_f + (\alpha)E_f.
\]

3. Transform \( \mathcal{V}_f \) to "standard" space: \( \mathcal{V}_f = W \mathcal{V}_f \). Because \( W \mathcal{V}_f \) has the potential to produce negative fitness values when epistasis is high, adjust the constant term \( C_f \) accordingly: When \( \min\{(W \mathcal{V}_f)_i\} < 0 \), let \( V' = W \mathcal{V}_f \), and let \( C_f = W \left( C_f - \min\{(V')_i : i = 0, 1, \ldots, 2^\ell - 1\} \right) \).

**An example:** For \( \ell = 2 \), consider fitness vector \( f \) with arbitrary components selected from the interval \([0,1]\): \( f = (0.54735, 0.76116, 0.60295, 0.83653)^T \). Then

\[
\hat{f} = \begin{pmatrix}
1.37400 \\
-0.22369 \\
0.00988
\end{pmatrix} = C_f + L_f + E_f = \begin{pmatrix}
1.37400 \\
0 \\
0
\end{pmatrix} + \begin{pmatrix}
0 \\
-0.22369 \\
0
\end{pmatrix} + \begin{pmatrix}
0 \\
0 \\
0.00988
\end{pmatrix}.
\]

Create the \( \alpha \)-linear fitness vector: \( \mathcal{V}_f = C_f + (1 - \alpha)L_f + (\alpha)E_f \).

Transform back to "standard" space, \( V_f = W \mathcal{V}_f = WC_f + (1 - \alpha)WL_f + (\alpha)WE_f \), and the \( \alpha \)-linear fitness vector is

\[
V_f = \begin{pmatrix}
0.68700 \\
0.68700 \\
0.68700 \\
0.68700
\end{pmatrix} + (1 - \alpha) \begin{pmatrix}
-0.14459 \\
0.07910 \\
-0.07910 \\
0.14459
\end{pmatrix} + (\alpha) \begin{pmatrix}
0.00494 \\
-0.00494 \\
-0.00494 \\
0.00494
\end{pmatrix}.
\]

The degree of epistasis of the \( \alpha \)-linear fitness vector \( V \) can be increased by arbitrarily small amounts by increasing the real number \( \alpha \in [0,1] \).
The Relationship Between $\alpha$ and Epistasis

The constant of proportionality ($\alpha$) can be calculated from the desired level of epistasis ($\varepsilon$) by using the linear and epistatic Walsh-space vectors ($L_f$ and $E_f$).

Consider an arbitrary fitness vector $f$ in Walsh space. Then the $\alpha$-linear fitness vector $V_f$ can be written (with subscripts omitted) as

$$V = C + (1 - \alpha)L + (\alpha)E,$$

with epistasis

$$\varepsilon = \frac{(\alpha)^2E^2}{(1 - \alpha)^2L^2 + (\alpha)^2E^2}.$$

Dividing numerator and denominator by $(\alpha)^2E^2$ gives

$$\varepsilon = \frac{1}{(1 - \frac{\alpha}{\varepsilon})^2 \frac{L^2}{E^2} + 1}.$$

Inverting yields

$$\frac{1}{\varepsilon} = (1 - \frac{\alpha}{\varepsilon})^2 \frac{L^2}{E^2} + 1$$

and thus

$$\left(\frac{1 - \varepsilon}{\varepsilon}\right) = \left(1 - \frac{\alpha}{\varepsilon}\right)^2 \frac{L^2}{E^2}.$$

If

$$K = \left(\frac{1 - \varepsilon}{\varepsilon}\right) \frac{E^2}{L^2},$$

then

$$\sqrt{K} = \left(\frac{1 - \alpha}{\alpha}\right)$$

and

$$\alpha = \frac{1}{\sqrt{K}} - \frac{\alpha}{\sqrt{K}}.$$

Combining the $\alpha$ terms and simplifying yields

$$\alpha = \frac{1}{\sqrt{K} + 1}.$$
So $\alpha$, in terms of the known quantities $\varepsilon$ (the desired epistasis level), $||L||$ (the length of the linear component of $\hat{f}$), and $||E||$ (the length of the epistatic component of $\hat{f}$), is expressed as

$$\alpha = \frac{1}{\sqrt{\frac{1-\varepsilon}{\varepsilon} ||E|| + 1}}$$

or

$$\alpha = \frac{||L||}{\sqrt{\frac{1-\varepsilon}{\varepsilon} ||E|| + ||L||}}.$$  \hfill (2.10)

Thus it is possible to convert an arbitrary fitness vector into an $\alpha$-linear fitness vector of specified epistasis.

3. The Vose Infinite Population Model (IPM)

A genetic algorithm performs the same series of operations on each generation in order to obtain the next generation. The Vose Infinite Population model [1] gives an exact probability distribution for the individuals chosen for the next generation. If populations are represented appropriately, then the expectation of this probability distribution can be shown to be the expected next population of the genetic algorithm. The Vose model gives an exact expression for this expected next generation population. This expected next generation population is also the next generation population under the simplifying assumption that the population is infinite.

The general principle is to model the initial population as a vector, then apply to that vector the operators which perform selection, crossover, and mutation, thereby obtaining a vector representation for the next generation.

3.1. Notation and Definitions

Several key elements of notation are defined in this section.
Recall the association between the vector $\vec{b} = (b_1 b_2 \ldots b_\ell)^T$, where $b_i \in \{0, 1\}$, the integer $\vec{b}^T \vec{d}$, where $\vec{d} = (2^0 2^1 \ldots 2^{\ell-1})^T$, and the string $S_\ell = b_\ell \ldots b_2 b_1$. This means, for example, the vector $(0 1)^T$, the integer 2, and the string 10 are associated with each other.

The domain $\Omega$ of the SGA is the set of binary strings of length $\ell$, and these are associated with the integers 0, 1, ..., $N - 1$, where $N = 2^\ell$. Associate the vector $\vec{1}$ with the integer $n - 1$, which when written as a string is all 1's. Associate the vector $\vec{0}$ with the integer 0, which when written as a string is all 0's.

The $k^{th}$ column of the $n \times n$ identity matrix is denoted $e_k$. Thus the $e_k$ are the standard basis vectors for $\mathbb{R}^n$.

For $x, y \in \Omega$, $x \oplus y$ denotes the bitwise exclusive-OR of $x$ and $y$, $x \otimes y$ denotes the bitwise AND of $x$ and $y$, and $\overline{x}$ denotes the bitwise complement of $x$. Note $\overline{x} = \vec{1} \oplus x$.

For predicate expression $expr$, $[expr]$ has value 1 if $expr$ is TRUE, and 0 otherwise.

Let $\delta_{i,j}$ represent $[i = j]$, and define $\sigma_k$ to be the $n \times n$ permutation matrix whose $ij^{th}$ entry is $\delta_{i \oplus k, j}$. Note $(\sigma_k x)_i = x_{i \oplus k}$. Thus, for example, the $4 \times 4$ permutation matrix for $k = 2$ is

$$
\sigma_2 = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}.
$$

### 3.2. Population Vector

Rather than maintain a population of strings, a population can be represented as a vector $\vec{x}$, where the $i^{th}$ component $x_i$ is the proportion of the total population represented by the string associated with the integer $i$.

By representing a population as a vector having $2^\ell$ non-negative real components whose sum is 1, a population vector is also a point in the positive orthant of $\mathbb{R}^n$.

For example, in population $\{00, 01, 11, 01, 01, 00, 01, 01\}$, string 00 occurs 2 times,
The population size is 8, and the population vector is 

\[ \vec{x} = \left( \begin{array}{cccc} 2 & 5 & 0 & 1 \\ 8 & 8 & 8 & 8 \end{array} \right)^T = (.250 \ 0.625 \ 0.000 \ 0.125)^T. \]

Note that information regarding the size of the population is lost in this representation, making it appropriate for finite or infinite populations.

The domain (set of all populations) of the IPM is the unit simplex

\[ \Lambda = \{ (x_0 \ \ldots \ x_{2^t-1})^T : \vec{x}^T \vec{1} = 1, x_j \geq 0 \}. \] (2.12)

Thus, the vectors \( e_k \) are the vertices of \( \Lambda \) and correspond to populations composed entirely of the one particular string associated with the integer \( k \).

Note that \( x_i \) represents the fraction of string \( i \) in the population, which is also the probability of occurrence of the string in that population. Thus, a population vector can also be regarded as a real-valued probability vector.

**Proportional Selection (\( \mathcal{F} \))**

The proportional selection process bases the probability of selection of any particular string on the fraction of the aggregate fitness of the population which that string's fitness represents.

We can perform (proportional) selection on a population represented in vector format. Let \( c_i \) represent the number of strings associated with integer \( i \), and \( r \) represent the population size. From the definition of \( \vec{x} \), \( x_i = \frac{c_i}{r} \) or \( c_i = rx_i \). Let \( f_i \) represent the fitness of string \( i \). Then the fitness due to all strings of type \( i \) is \( rf_i x_i \), total fitness is \( \sum rx_i f_i = r \vec{f}^T \vec{x} \), and the proportional fitness of string \( i \), which is also the probability of selecting string \( i \), is given by

\[ \frac{rx_i f_i}{r \vec{f}^T \vec{x}} = \frac{f_i x_i}{\vec{f}^T \vec{x}} = \frac{(F \vec{x})_i}{\vec{f}^T \vec{x}}, \]

where \( F \) is the \( n \times n \) diagonal matrix (obtained from \( \vec{f} \)) with entries \( F_{ij} = \delta_{ij} f_i \).
Record the proportional fitness of each string $i$ in vector $\vec{f}$, as was done for the population vector: $\vec{f} = (f_0 \ f_1 \ \ldots \ f_{n-1})^T$. Then proportional selection applied to population $\vec{x}$ produces the population

$$F(\vec{x}) = \frac{(x_0 f_0 \ x_1 f_1 \ \ldots \ x_{n-1} f_{n-1})^T}{\vec{f}^T \vec{x}} = \frac{F \vec{x}}{\vec{f}^T \vec{x}} = \frac{F \vec{x}}{\vec{f}^T F \vec{x}}.$$

The $F(\vec{x})$ is a probability vector.

As an example, consider the population $\vec{x} = (\frac{2}{5} \ \frac{3}{8} \ \frac{5}{8} \ \frac{1}{3})^T$, and the "ones" fitness vector $\vec{f} = (0 \ 1 \ 1 \ 2)^T$. The probability of selecting string $i$ from population $\vec{x}$ under proportional selection is the $i^{th}$ component of the population vector

$$F(\vec{x}) = \frac{(f_0 x_0 \ f_1 x_1 \ f_2 x_2 \ f_3 x_3)^T}{(f_0 x_0 + f_1 x_1 + f_2 x_2 + f_3 x_3)\ \ (0 \ 0 \ .625 \ .250)^T}$$

$$= \frac{(0 \ 0 \ .625 \ .250)^T}{(0 + 0 + .625 + .250)^T}$$

$$= \left( \begin{array}{cccc} 0 & 0 & \frac{5}{7} & \frac{2}{7} \end{array} \right).$$

Crossover

Consider the effect of crossover on the population. The proportion of string $k$ in the new population is based on an accumulation of all possible ways string $k$ is produced by a crossover between string $i$ and string $j$, taking into consideration the probability of selecting string $i$ and string $j$, as well as the probability (which is affected by the crossover method) that they produce string $k$.

Typically, crossover does not occur in every instance, but at some specified rate $C$. In the absence of crossover, a child is produced by selecting and cloning one of the parents.

Let $k \in \Omega$ represent a crossover mask which when applied to parents $i$ and $j$ produces children $(i \oplus k) \oplus (j \oplus k)$ and $(j \oplus k) \oplus (i \oplus k)$, one of which is selected (with equal probability) and retained. Let $\chi_k$ denote the probability that $k$ is selected to
be the mask.

We refer to vector $\chi \in \Omega$ as a *crossover (probability) distribution*. Such a crossover distribution is determined by the method of crossover: One-point crossover occurring at rate $C$ corresponds to the crossover distribution

$$
\chi_i = \begin{cases} 
1 - C & \text{if } i = 0, \\
C/(\ell - 1) & \text{if } \exists p \in (0, \ell) \text{ such that } i = 2^p - 1, \\
0 & \text{otherwise}.
\end{cases}
$$

Uniform crossover at rate $C$ is given by

$$
\chi_i = \begin{cases} 
1 - C + C2^{-\ell} & \text{if } i = 0, \\
C2^{-\ell} & \text{if } i > 0.
\end{cases}
$$

**Mutation**

Mutation can be handled in a manner similar to crossover: The probability that the child string mutates to become a specific string is the product of the probabilities that each bit of the child string is mutated or not, as is appropriate to obtain the desired string.

For example, let $R$ represent the (bitwise) mutation rate — the probability that a bit flips (changes to its complement). Then the probability that string 1010 mutates to string 0011, expressed in terms of the probability that a bit will flip ($R$) or will not flip ($1 - R$), is $(R)(1 - R)(1 - R)$ or $(1 - R)^2$. The vector $\mu \in \Omega$ is called the *mutation (probability) distribution*. Such a mutation distribution is determined by the method of mutation: Bitwise mutation at rate $R$ corresponds to the mutation distribution

$$
\mu_i = R^{\ell_i} (1 - R)^{\ell - \ell_i}
$$

where $\ell_i$ yields the number of 1's in string $i$ of length $\ell$, and $\ell - \ell_i$ therefore yields the number of 0's.
Mixing ($\mathcal{M}$)

Crossover and mutation operate simultaneously on a population in the IPM, by means of a process given the name mixing and represented by the function $\mathcal{M}$.

$\mathcal{M}$ can be calculated in terms of the $n \times n$ mixing matrix $M$, defined as follows: Element $M_{ij}$ of $M$ represents the probability that crossover and mutation applied to individuals $i$ and $j$ will produce 0, the "zeros" vector $(0 \ldots 0)^T$. Thus, $\bar{x}^T M \bar{x}$ is the probability that crossover and mutation applied to population $\bar{x}$ produces 0. And, equivalently, it is the fraction of the population (which resulted from crossover and mutation applied to population $\bar{x}$) which is represented by the individual comprised of all zeros.

Represent the probability that parent $i$ mutates to string $i \oplus u$ as $\mu_u$. Then Vose [1] showed

$$M_{i,j} = \sum_{u,v,k \in \Omega} \mu_u \mu_v \frac{\chi_k + \chi^-_k}{2} \left[ ((i \oplus u) \otimes k) \oplus ((j \oplus v) \otimes \overline{k}) = 0 \right].$$

The mixing function $\mathcal{M} = (M_0 \ M_1 \ldots \ M_{2^t-1})$ applied to population $\bar{x}$, has components

$$\mathcal{M}_i(\bar{x}) = \bar{x}^T \sigma_i \sigma_i \bar{x} = e_i^T \mathcal{M}(\bar{x}) = \sum_{u,v \in \Omega} \bar{x}_u \bar{x}_v M_{u \oplus i, v \oplus i}$$

Vose [1] showed that $\mathcal{M}_k(\bar{x})$ represents the probability of occurrence of string $k$ in the population obtained from population $\bar{x}$ by means of crossover and mutation.

The IPM Iteration Function $\mathcal{G}$

The IPM is represented by the function

$$\mathcal{G}(\bar{x}) = \mathcal{M}(\mathcal{F}(\bar{x}))$$

where $\mathcal{G}(\bar{x})$ is the new population resulting from one iteration of the genetic algorithm on population $\bar{x}$. 
4. Trajectory, Fixed Points, Stability, Convergence

4.1. Trajectory

As mentioned above, an infinite population can be represented as a point in the unit simplex $\Lambda$ (see Equation 2.12), which is in the positive orthant of $n$-space, $(\mathbb{R}^+)^n$.

Let $\bar{x}_t = G^t(\bar{x}_0)$, where $t = 1, 2, \ldots$, and $\bar{x}_0$ is the initial population. Then a sequence of consecutively-generated $\bar{x}_t$'s defines a trajectory in $(\mathbb{R}^+)^n$.

4.2. Fixed Points

A fixed point of $G$ is a population $\bar{y}$ such that $\bar{y} = G(\bar{y})$.

4.3. Stability

A point $\bar{x}$ in the unit simplex $\Lambda$ (see Equation 2.12) is a stable fixed point of $G$ if for every neighborhood $U$ about $\bar{x}$ there exists a neighborhood $V$ such that for each $\bar{q} \in V$ the trajectory $\bar{q}, G^1(\bar{q}), G^2(\bar{q}), \ldots$ lies in $U$.

A stable fixed point $\bar{x}$ is asymptotically stable if all trajectories beginning in some neighborhood of $\bar{x}$ converge to $\bar{x}$.

Differential of $G$

From Vose [1]:

- The differential of $M$ at $\bar{x}$ is the unique linear transformation $dM_{\bar{x}}$ satisfying

$$\lim_{\bar{y} \to \bar{0}} \frac{M(\bar{x} + \bar{y}) - (M(\bar{x}) + dM_{\bar{x}} \bar{y})}{||\bar{y}||} = 0.$$ 

- The twist of an $n \times n$ matrix $A$, denoted $A^*$, is the matrix with entries $A_{i,j}^* = A_{i \otimes j,i}$.

- The differential of $M$ at $\bar{x} \in \Lambda$ is $dM_{\bar{x}} = 2 \sum_{\sigma_u} M^* \sigma_u \bar{x}$. 
- The differential of $G$ at $\bar{x} \in \Lambda$

(calculated from the chain rule: $dG_{\bar{x}} = dM_{F_{\bar{x}}} \circ dF_{\bar{x}}$) is

$$dG_{\bar{x}} = \frac{1}{1^T F_{\bar{x}}} dMSF P,$$

where $P = I - \bar{x} \frac{1^T F}{1^T F_{\bar{x}}}$ and $S = \frac{F_{\bar{x}}}{1^T F_{\bar{x}}}$.

**Spectral Radius of $dG_{\bar{x}}$**

The spectrum of a square matrix $A$ is the set of eigenvalues ($\lambda$'s) satisfying $Av = \lambda v$, where $v$ represents an eigenvector.

From Vose and Wright [5]:

- The spectrum of $dG$ at vertex $e_k$ is

$$\text{spec}(dG_{e_k}) = \left\{ \sum_u (\chi_u + \chi_{\bar{u}}) [u \otimes i = 0] : i = 1, 2, \ldots, n-1 \right\} \cup \{0\}.$$  

- The spectral radius (the maximum modulus of the eigenvalues) of $dG$ is

$$\rho(dG_{e_k}) = \max\{|\lambda| : \lambda \in \text{spec}(dG_{e_k})\}.$$  

- A standard result of dynamical system theory (Belitskii and Lyubich [6]) is that for a fixed point $\bar{x}$ of $G$: If $\rho(dG_{\bar{x}}) < 1$, then $\bar{x}$ is asymptotically stable; if $\rho(dG_{\bar{x}}) > 1$, then $\bar{x}$ is unstable.

Thus, a point $\bar{x} \in \Lambda$ is a stable fixed point of $G$ when

$$\rho(dG_{\bar{x}}) < 1.$$  

(2.13)

**4.4. Convergence**

The term *convergence* has been used imprecisely to describe genetic algorithm behavior, since a genetic algorithm allowing mutation does not converge in the sense intended by the term.
A finite-population genetic algorithm with a non-zero mutation rate will go from one population to another in a stochastic manner. When the general composition of the populations remains mostly similar over many generations, some people might accept this situation as convergence.

On the other hand, a genetic algorithm with crossover but no mutation can approach a state where nothing changes from generation to generation. As the similarity of population members increases, crossover will tend to preserve the similarities. If a population is composed entirely of one type of individual, there is no way crossover and selection acting alone will be able to alter it. (For the IPM, such a population is a vertex of the simplex.)

For this thesis (where the mutation rate is zero), the IPM is said to converge if its trajectory from population $\bar{x}$ goes to some fixed point $\bar{c}_x$ in the simplex. $\bar{c}_x$ is called the convergent population.

$$\lim_{t \to \infty} G^t(\bar{x}) = \bar{c}_x.$$  

When applied to the (finite population) SGA having mutation rate zero, convergence means that all members of population $\bar{x}$ are identical, and $\bar{x}$ is called the convergent population.
Chapter 3

Objectives

As genetic algorithm optimization algorithms, the finite population model (SGA) and the infinite population model (IPM) operate differently, the SGA being stochastic, the IPM deterministic.

Vose [1] presents the following theorem regarding the behavior of the SGA and IPM:

Given $k > 0$, $\epsilon > 0$, and $0 < \gamma < 1$, there exists an integer $N$ such that with probability at least $\gamma$ and for all $0 \leq t < k$

$$r > N \implies ||r^t(x) - G^t(x)|| < \epsilon$$

In other words, GIVEN a bound $k$ on the number of generations, a bound $\epsilon$ on the separation of the SGA ($r$) and IPM ($G$) at generation $t$ (where $t \leq k$), and a lower bound $\gamma$ on probability, THERE EXISTS a lower bound on population size ($N$) such that when population $r$ exceeds bound $N$, then with a probability of at least $\gamma$ the divergence of the SGA and IPM populations will be less than the specified amount $\epsilon$ for $k$ generations.

Therefore, one objective of this thesis is to examine the question: At what population size does the fixed-point structure of the IPM become an effective predictor of
the behavior of the SGA?

Examining all aspects of behavior for the two algorithms is not feasible. Therefore, this thesis is limited to examining aspects of convergence behavior, and to that end will not use mutation. This "zero mutation" choice has the added benefit of providing a simple specification for the stopping criteria for the SGA: When all individuals in the population are identical, stop. In all comparisons, the initial population and the fitness vector are the same for the SGA and the IPM.

Areas of comparison for convergence behavior include:

1. How often does the SGA converge to a stable fixed point of the IPM? The following quotation from Vose [1] indicates the behavior of the IPM with regard to fixed points: "The expected behavior of all nearby populations is to converge towards a stable fixed point. Fixed points with this property are called attractors. A natural hypothesis is that attractors indicate locations within Λ where the SGA is predisposed to be near." (Vose [1])

Thus, we take one point of comparison to be whether the SGA follows the IPM behavior with regard to stable fixed points.

Convergence to an almost-stable fixed point may be of interest also, because it allows for the possibility that the fitnesses generated for two points are very nearly equal, and provides a meaningful measure for taking such cases into account when fitnesses are near-optimal.

2. How close to each other are the fitnesses of the convergent IPM and SGA populations? How close are they to optimal?

3. Do the IPM and SGA converge to the same population? This is perhaps the purest measure of the IPM as a predictor of SGA behavior. When not the same
population, the magnitude of the separation between them (and of each from
the optimum) may be of interest.

The epistasis of the fitness vector is not directly addressed in these questions, but
we expect it will have an influence on the behavior of the two genetic algorithms.
Thus, another objective is to answer the following questions concerning the effects of
epistasis:

1. Will an increase in epistasis make it more difficult for the genetic algorithms
to find the optimum? Two measures of “difficulty” to examine are: frequency
of convergence to the optimum, and number of generations required in order to
obtain convergence.

2. Will one of the genetic algorithms be affected more than the other by increasing
epistasis? Two points of comparison for this question are: convergence to the
same fitness, and convergence to the same population.

3. Is there a level of epistasis beyond which further increases in epistasis produce
little changes in convergence behavior? Are there ranges of epistasis in which
behavior remains essentially constant?

4. How does the distribution of fitnesses among strings at various hamming dis-
tances from the optimum string change in the $\alpha$-linear fitness vector as epistasis
increases? For a linear fitness vector ($\alpha = 0$), individuals of high fitness gen-
erally tend to be near the optimum in hamming distance. Increasing epistasis
(and hence $\alpha$) can redistribute the relative fitnesses of individuals, perhaps
raising the fitnesses of distant individuals above the new levels for the nearer
individuals. (This is of interest because it introduces the potential to skew con-
vergence behavior away from the optimum, as well as to change the optimum
to a different string.)
Chapter 4

Methodology

1. Design

Our basic method for data acquisition is to perform an experiment: Run the IPM and SGA to convergence (no mutation), where both start from the same initial population and use the same fitness vector. The parameters to be varied from experiment to experiment are string length, population size, and epistasis.

Rather than use existing genetic algorithm programs, which introduces the logistics problem of coordinating the input data (initial population, fitness vector, experiment parameters) for two programs and the two sets of raw data files produced by those programs, we decided to write a program which accepts input parameters and runs the experiments, saving the raw data from each experiment as a unit.

1.1. The General Algorithm

One execution of this algorithm constitutes one experiment.

(1) Set the parameters for the current experiment.
(2) Create a random SGA population, then an IPM population from it.
(3) Create a fitness function vector with a given epistasis.
(4) Run the IPM to convergence, or for the maximum number of generations.
(5) If the IPM does not converge
(6) then return to (3),
(7) otherwise
(8) run the SGA to convergence to a uniform population $P$, 
(9) calculate the stability of $P$, 
(10) save the results of this experiment, and 
(11) if more parameter sets exist
(12) then return to (1),
(13) otherwise STOP.

2. Implementation Details

The general algorithm is repeated for each set of input parameters, with multiple runs for the same parameter set being controlled by one of the parameters. Some aspects of the algorithm are elaborated below.

2.1. Random Number Generator

Of fundamental importance in our program is the creation of random numbers. We used the random number generator from *Communications of the ACM* [8] (giving it the name ACM_rand) to create a 32-bit random number by concatenating the 16 most significant bits of two random numbers generated by ACM_rand. This operation was called UL_Rand.

In order to create an individual, the first $\ell$ bits from the UL_rand number were selected.

For fitness values, a UL_Rand number was converted to a decimal value in the range $[0,1)$. This operation was called D_Rand.
2.2. Parameters Fixed For All Experiments

In this thesis, some GA parameter values are fixed for all experiments:

- The *crossover rate* is set at 0.6, which falls within the generally accepted range for effective performance. This means that for 40% of the pairs selected at random for crossover, one member (chosen at random) is instead inserted directly into the next generation.

- Mutation rate is set to zero (and the mutation portion of the program is not implemented) so that the SGA can run to full convergence. (Mutation could disrupt the population in which all individuals in the population are identical.)

- IPM convergence tolerance is 0.00001. This implies that the IPM population vector $\vec{x}$ will be considered fully converged when $||\vec{x} - e_k|| < (0.00001)^{1/2}$ for some vertex $e_k$ (vertex: see below Equation 2.12).

- One-point crossover is used exclusively in this thesis.

- The maximum number of generations allowed for the IPM to converge is set at 2000 generations.

- The maximum number of SGA generations is set high enough (2000 generations) that it will never be reached for parameter settings within the ranges used in this thesis.

- The number of experiments to run for each particular $(\ell, r, \epsilon)$ parameter setting is 50.

2.3. Parameters Which Vary Between Experiments

The parameters to be varied between experiments in order to produce data sets for analysis are binary string length $(\ell)$, population size $(r)$, and degree of epistasis.
Also, by varying the starting seed for the random number generator, multiple data sets for an experiment can be obtained.

### 2.4. Fast Walsh Transform

Vose and Wright [9] point out the complexity of mixing is $O(n^3)$ in standard space and $O(n^2)$ in Walsh space. Also, selection is $O(n)$ in standard space and $O(n^2)$ in Walsh space. For reasons of efficiency, then, we use the Walsh space implementation of the IPM, and use the Fast Walsh Transform to convert the initial population vector and fitness vector to the Walsh basis. (The cumulative effect of round-off error attributable to the Fast Walsh Transform makes it unreasonable to repeatedly convert the population vector between standard space for selection and Walsh space for mixing.)

The Fast Walsh Transform is also used to convert the Walsh basis population to standard basis in order to check for convergence.

(It was also used when converting the population and fitness vectors to standard space as part of the verification of implementation correctness.)

### 2.5. Creation of Initial Populations

A comparison of SGA and IPM behaviors clearly must begin with the use of a common starting point: the initial population. Since the SGA and IPM use different representations for populations, we create an initial population in SGA format, then use it to create another population in the IPM format.

A matter of concern, especially for smaller populations, is that a particular bit position may have the same value for all members of the population. If that occurs, the search space accessible from the initial population will contain the optimally fit
string only half the time, on average. If two bits of the convergence population are predetermined by the initial population, the optimum string is only accessible a fourth of the time, and so on.

To avoid this sort of influence on the convergence behavior by the initial population, an initial population is not accepted unless every bit is represented at every bit position somewhere in the population.

2.6. Creation of a Fitness Function Vector

To make an $\alpha$-linear fitness vector,

1. Generate random fitness vector $f$ using UL_Rand: $f_i \in [0, 1)$ for $i = 0, \ldots, 2^t - 1$.
2. Transform $f$ to Walsh space and separate it into linear and epistatic vectors.
3. Calculate the proportionality constant $\alpha$.
4. Create the $\alpha$-linear fitness vector $\hat{f}$, keeping the same $C_f$, and make a duplicate fitness vector from the result.
5. Transform the duplicate to standard space. If any negative values appear in the vector, increase $C_f$ in $\hat{f}$ to offset the most negative one. (It should be noted that negative fitnesses seem to occur in vectors created in this fashion only if $\alpha = 1$.)
6. To avoid tiny negative terms like -1.09e-16 (which are essentially zero, and are likely due to truncation errors occurring during the Fast Walsh Transform), add 1.0e-12 to $C_f$. (This small quantity is essentially zero, and will not affect the epistasis anyway.)
7. Transform $\hat{f}$ to standard space.
2.7. The IPM

The IPM is considerably slower than the SGA (due to matrix multiplications), and may require many more generations to converge when using one fitness function than when using another. To enable the experiments to run in reasonable time, a limit is imposed on the number of generations allowed in order to achieve convergence: If the IPM doesn't converge within 2000 generations, the original population is re-loaded, a different fitness vector is generated, and the IPM is re-started.

2.8. The SGA

The SGA must be run to convergence. The maximum generation number for the SGA (one of the parameters which can be varied) was set at 2000. (The actual number of generations required for convergence under the parameter sets used in this thesis seldom exceeded 500, and never reached 1500.)

2.9. Stability Calculation

The stability of the convergent population of the SGA (expressed in IPM population format) is calculated from Equation 2.13.

3. Verification of Correctness

It is difficult to verify a stochastic algorithm, since the answer is not deterministic. Also, small errors can be lost in the “noise” of the algorithm.

We are reasonably confident of the correctness of the operation of our computer program, having taken the following steps to uncover errors:

- Used a profiler to locate memory leaks.
- Implemented Fast Walsh Transformation on spreadsheet, and compared with
program results.

- Traced the stability calculation on a hand-calculator.

- Used a spreadsheet to compare the (desired) epistasis $\varepsilon$ used in the calculation of the proportionality constant $\alpha$ with the (calculated) epistasis $\varepsilon_f$ of the $\alpha$-linear fitness vector $f$ produced from $\alpha$.

- Analyzed a script of an SGA run in which all changes to variables were recorded, to verify that the SGA operated as specified.

- Compared with generation 3 of an exact Markov Chain model, using bit-string length 3 and population size 3 (initial population: 001, 011, and 110).

4. Experiments

As mentioned above, one experiment means one execution of the general algorithm with a particular $(\ell, r, \varepsilon)$ parameter set. Guidelines used for the ranges of string length $\ell$, population size $r$, epistasis $\varepsilon$, and the number of repetitions of the experiment are:

- String length is restricted to $\ell = 4, 6, 8$. The program allows $3 \leq \ell \leq 9$ for an experiment. For $\ell > 9$, only the SGA algorithm is executed.

- The choice of population sizes is based on string length $\ell$ in 3 ways:
  
  1. constant population size: $r_\ell = k \forall \ell$. We chose $k$ values (and hence $r$ values) of 16, 32, 64, and 128.

  2. population size proportional to the square root of the size of the search space: $r_\ell = k \sqrt{2^\ell}$. We chose $k = 2$, so $r_4 = 8, r_6 = 16, \text{ and } r_8 = 32$.

  3. population size proportional to string length: $r_\ell = k \ell$. We chose $k = 2, 4$: thus, $r_4 = 8, 16; r_6 = 12, 24; r_8 = 16, 32$. In addition, $r_8 = 512$ was chosen, to provide a large population for the longest string.
- Epistasis $\varepsilon$ varies between 0 (the fitness vector is a linear function of bit-positions in a string) and 1 (a fitness vector with no linearity) in increments of $\frac{1}{9}$, producing 10 levels of epistasis for analysis.

- The number of experiments per $(\ell, r, \varepsilon)$ parameter set was 50.

**Genetic Algorithm Experiments**

Experiments were grouped by string length, then sub-grouped by population size:

Experiment Group $\ell = 4$ has subgroups $r = 8, 16, 32, 64, 128$

Experiment Group $\ell = 6$ has subgroups $r = 12, 16, 24, 32, 64, 128$

Experiment Group $\ell = 8$ has subgroups $r = 16, 32, 64, 128, 512$

Within each subgroup 50 experiments were performed for each epistasis setting (i.e., for each $(\ell, r, \varepsilon)$ set).

**Fitness Function Experiment**

The same $\alpha$-linear fitness vector is used to produce fitness vectors for which epistasis increases from linear ($\alpha = 0$) to epistatic ($\alpha = 1$) by varying epistasis in increments of $\frac{1}{9}$.

The effect of increasing epistasis can be observed visually in a series of scatterplots of fitness (as a proportion of optimum fitness) vs hamming distance to the optimally fit string, and numerically as the correlation between fitness and hamming distance.
Chapter 5

Results

The raw data for each \((\ell, r, \varepsilon)\) setting was averaged over the 50 runs of that subgroup, producing average data for the three experimental groups. Ten sets of such data were averaged to produce the information presented in the charts which deal with conversion behavior (figures 5.1 to 5.46).

Data for the charts dealing with the composition of the fitness vector (figures 5.51 and 5.52) was obtained by setting the \(\alpha\) value of an \(\alpha\)-linear fitness vector to create each desired epistasis level, then saving each fitness vector so produced. The linear fitness vector and epistatic fitness vector used to generate all the data were obtained from the decomposition of a random fitness vector into its linear \((\mathcal{L})\) and epistatic \((\mathcal{E})\) components.

1. Charts

1. The frequency with which the SGA converges to a stable fixed point.

The proportion of SGA convergent populations which correspond to stable fixed points of the IPM (i.e., calculated stability is less than 1.0) is plotted against epistasis, for each population size. It is clear that the proportion that converge
to stable fixed points increases with population size. There is also a general trend for the smaller population sizes to show an increase in convergence to stable fixed points as epistasis increases. Note the opposite trend, however, for the largest population size, particularly for $\ell = 8$.

Figures 5.4, 5.5, and 5.6 show SGA convergence to stable fixed points by string length $\ell$, using the three methods for dealing with population size as a function of $\ell$ (as mentioned in the section “Experiments” in Chapter 4): in Figure 5.4, population size is constant for all $\ell$; in Figure 5.5, population size is proportional to the square root of the size of the search space; in Figure 5.6, population size is proportional to string length. (These three forms of comparison are used when the information is presented by string length rather than by population size.)

![Figure 5.1: SGA converges to Stable Fixed Point, $\ell = 4$](image)
Figure 5.2: SGA converges to Stable Fixed Point, $\ell = 6$
Figure 5.3: SGA converges to Stable Fixed Point, $\ell = 8$
Figure 5.4: SGA converges to Stable Fixed Point, \( r_\xi = 128 \)
Figure 5.5: SGA converges to Stable Fixed Point, $\tau_f = 8 \sqrt{2^f}$
Figure 5.6: SGA converges to Stable Fixed Point, \( r_\ell = 4\ell \)
2. The frequency with which the SGA converges to an almost-stable fixed point.

Here we plot the proportion of SGA convergent populations having stability less than 1.1 against epistasis, for each population size. As expected, the overall effect is to shift the results of Figures 5.1, 5.2, and 5.3 toward higher rates of occurrence. There is also some "flattening" of the rate of convergence to (almost) stable fixed points at the higher epistasis levels.

Figure 5.7: SGA converges to Almost-Stable Fixed Point, $\ell = 4$
Figure 5.8: SGA converges to Almost-Stable Fixed Point, $\ell = 6$
Figure 5.9: SGA converges to Almost-Stable Fixed Point, $\ell = 8$
Figure 5.10: SGA converges to Almost-Stable Fixed Point, $r_\ell = 128$
Figure 5.11: SGA converges to Almost-Stable Fixed Point, $\tau_\epsilon = 8\sqrt{2^l}$
Figure 5.12: SGA converges to Almost-Stable Fixed Point, $r_\ell = 4\ell$
3. The frequency with which the IPM and SGA converge to populations having the same fitnesses.

Considering the small fitness spaces used (the largest, \( \ell = 8 \), requires 256 fitnesses), converging to the same fitness is essentially equivalent to converging to the same population string (a case which is presented later). Therefore, if the fitnesses of the convergent SGA and IPM populations differ by less than 0.1 (the fitness tolerance), they are considered to be the same fitness for the purposes of the next three charts.

Population size is a rough indicator of the rate of convergence to populations of similar fitness (larger population, greater likelihood), while epistasis is not — except at full epistasis, where the rate of convergence generally drops markedly. Compare these charts to those depicting convergence to the same string (Figures 5.22, 5.23, and 5.24), to see the effects of the 0.1 fitness tolerance.
Figure 5.13: IPM and SGA converge to populations having the same fitness, $\ell = 4$
Figure 5.14: IPM and SGA converge to populations having the same fitness, $\ell = 6$
Figure 5.15: IPM and SGA converge to populations having the same fitness, $\ell = 8$
Figure 5.16: IPM and SGA converge to populations having the same fitness, $r_\ell = 128$
Figure 5.17: IPM and SGA converge to populations having the same fitness, \( \tau_t = 8\sqrt{2} \)
Figure 5.18: IPM and SGA converge to populations having the same fitness, \( r_\ell = 4\ell \)
4. A count of convergence to the optimum (500 experiments)

These charts represent the number of times the SGA and IPM converge to the optimum, without regard to epistasis. The data presented is the average of the results of 10 runs of each subgroup, in which 50 runs at each of the 10 epistasis settings were performed. (The results of the ten epistasis levels were grouped together in each subgroup.)

The initial population of the *Infinite Population* Model is based on a finite population of given size. This is the reason for the observed dependency of the results for the IPM on the population size.

The IPM results increase gradually with population size, appearing to approach some limiting value, which tends to decrease as \( \ell \) increases. The SGA results start considerably lower for small populations, and increase more dramatically with population size. For a constant population size, an increase in \( \ell \) reduces the rate of convergence to the optimum.
Figure 5.19: Convergence to Optimum vs Population Size, $\ell = 4$
Figure 5.20: Convergence to Optimum vs Population Size, $\ell = 6$
Figure 5.21: Convergence to Optimum vs Population Size, $\ell = 8$
5. The frequency with which the IPM and SGA converge to the same string.

This would seem to be the purest measure of the IPM as a predictor of SGA behavior. Convergence to the same string occurs more frequently with larger populations, but generally decreases with increasing epistasis. For the smaller populations, increasing epistasis doesn't seem to have a great effect. For the same population size, as string length increases, frequency of convergence to the same string goes down noticeably.

Figure 5.22: IPM and SGA converge to identical populations, \( \ell = 4 \)
Figure 5.23: IPM and SGA converge to identical populations, $l = 6$
Figure 5.24: IPM and SGA converge to identical populations, $\ell = 8$
Figure 5.25: IPM and SGA converge to identical populations, \( r_L = 128 \)
Figure 5.26: IPM and SGA converge to identical populations, $r_\ell = 8\sqrt{2^\ell}$
Figure 5.27: IPM and SGA converge to identical populations, $r_{\ell} = 4\ell$
6. The count of IPM convergence populations falling in each level of hamming distance to optimum.

These charts are based on the number of experiments in a subgroup (250 for \( \ell = 4 \) and \( \ell = 8 \), and 300 for \( \ell = 6 \)) at each epistasis level. As epistasis increases, we note two trends (which become more pronounced as \( \ell \) increases): convergence to the optimum decreases, and convergence to populations near the optimum (hamming distance 1) is rare. Note also how convergence to populations of hamming distance 2 from the optimum becomes less frequent at higher epistasis levels, falling below the level of occurrence of populations at hamming distances of 3, 4, and even 5, for \( \ell = 8 \).

Figure 5.28: Hamming Distance to Optimum String: IPM (250 experiments), \( \ell = 4 \)
Figure 5.29: Hamming Distance to Optimum String: IPM (300 experiments), $\ell = 6$
Figure 5.30: Hamming Distance to Optimum String: IPM (250 experiments), $\ell = 8$
Figure 5.31: Hamming Distance to Optimum String: IPM, $\ell = 4, 6, 8$
7. The frequency of occurrence of zero hamming distance to optimum, for each population size.

These charts present only zero hamming distance data. For the larger population sizes, the SGA converges to the optimum less frequently, in a manner similar to the IPM results. The smaller populations seem to be remain unchanged as epistasis increases.

Figure 5.32: SGA Zero Hamming Distance to Optimum (by Population Size), $\ell = 4$
Figure 5.33: SGA Zero Hamming Distance to Optimum (by Population Size), $\ell = 6$
Figure 5.34: SGA Zero Hamming Distance to Optimum (by Population Size), $\ell = 8$
Figure 5.35: SGA Zero Hamming Distance to Optimum (by String Length), $\tau_\ell = 128$
Figure 5.36: SGA Zero Hamming Distance to Optimum (by String Length), $r_L = 8\sqrt{2^L}$
Figure 5.37: SGA Zero Hamming Distance to Optimum (by String Length), $r_l = 4l$
8. The average number of SGA generations to convergence, as a function of epistasis.

The number of generations required to obtain convergence for the SGA generally decreases for the larger populations as epistasis increases. The smaller populations appear to be indifferent to epistasis levels.

Figure 5.38: SGA Generations to Convergence vs Epistasis, \( \ell = 4 \)
Figure 5.39: SGA Generations to Convergence vs Epistasis, $\ell = 6$
Figure 5.40: SGA Generations to Convergence vs Epistasis, $\ell = 8$
Figure 5.41: SGA Generations to Convergence vs Epistasis, $r_t = 128$
Figure 5.42: SGA Generations to Convergence vs Epistasis, $\tau_\ell = 8\sqrt{2^\ell}$
Figure 5.43: SGA Generations to Convergence vs Epistasis, $r_\ell = 4\ell$
9. The average number of IPM generations to convergence, as a function of epistasis.

Although the scale is different for different $\ell$ values, there is a striking similarity in the shapes of the graphs: a rapid decrease in number of generations when epistasis is introduced, followed by a slight decrease across the intermediate levels of epistasis, then a very rapid drop as total epistasis is reached.

![Figure 5.44: IPM Generations to Convergence vs Epistasis, $\ell = 4$](image-url)
Figure 5.45: IPM Generations to Convergence vs Epistasis, $\ell = 6$
Figure 5.46: IPM Generations to Convergence vs Epistasis, \( \ell = 8 \)
Figure 5.47: IPM Generations to Convergence vs Epistasis, $r_t = 128$
Figure 5.48: IPM Generations to Convergence vs Epistasis, $r_\ell = 8\sqrt{2^\ell}$
Figure 5.49: IPM Generations to Convergence vs Epistasis, $r_e = 4\ell$
10. Scatterplots depicting the distribution of fitnesses (as a fraction of the optimum fitness) vs hamming distance from the optimum string.

To generate the data for the scatterplots below, a random fitness vector was decomposed (in Walsh space) into its linear ($L$) and epistatic ($E$) components, which were used to create $\alpha$-linear fitness vectors of given epistasis by calculating and applying the appropriate $\alpha$ values.

The ratio of each string's fitness to the optimum fitness is plotted against the hamming distance to that string from the optimum string. Because we use the same generating pair (i.e., the purely linear and purely epistatic fitness vectors in Walsh space) to obtain the fitness vectors for the various epistasis levels, the overall change in correlation of the data is cleanly illustrated as epistasis is increased.

The charts are arranged by $\ell$ value (columns) and epistasis (rows). The first (linear) row shows a high correlation, which becomes progressively smaller as epistasis increases, until the coefficient of correlation is essentially zero at the last (purely epistatic) row.

Axis labels and titles were left off the charts, to leave more room for the data to be displayed. The sample chart below indicates the axis labels and the location of the coefficient of correlation for the data in that chart.
Figure 5.50: Sample Scatterplot
Figure 5.51: Fitness vs Hamming Distance to Optimum (Epistasis $\frac{0}{9}$ – $\frac{4}{9}$)
Figure 5.52: Fitness vs Hamming Distance to Optimum (Epistasis $\frac{5}{9} - \frac{9}{9}$)
2. Observations

Several general statements can be made about the charts.

2.1. IPM as Predictor

The value of the IPM as a predictor of SGA convergence behavior is summarized in the following ways:

Increasing $\ell$:

1. Compare Figure 5.6, which present convergence to a stable fixed point, with Figure 5.12, which present convergence to an almost-stable fixed point, both by string length $\ell$ (with population size $\tau = 4\ell$). The rate of convergence to stable fixed points decreases as $\ell$ increases, but the rate for convergence to an almost-stable fixed point increases as $\ell$ increases. Similar results occur for fitness: Figure 5.27 presents convergence to the same string (and hence the same fitness) and Figure 5.18 presents convergence to almost the same fitness (i.e., within the fitness tolerance).

This implies that as $\ell$ increases, the SGA finds a stable fixed point (or the IPM convergence string) less often, but finds points close to a stable fixed point (or the IPM convergence string) more often. Note this is based on small populations whose size is proportional to string length.

Comparisons based on constant population size or population size proportional to the square root of the search space size were not useful, generally because the "tolerances" were so large that almost every point qualified for inclusion (i.e., rate of occurrence equals 1), making it impossible to determine what occurred.
2. In general, performance degrades in two ways as $\ell$ increases. Figures 5.35, 5.36, 5.37, and 5.31, showing rate of convergence to the optimum string as $\ell$ varies, all indicate that increasing $\ell$ implies the optimum will be found less frequently. At the same time, Figures 5.41, 5.42, 5.43 for the SGA and Figures 5.47, 5.48, 5.49 for the IPM all indicate a larger $\ell$ will require more generations for convergence.

Thus, as $\ell$ increases the optimum is found less often, and the search for it takes longer.

**Increasing $r$:**

As expected, the resemblance of the SGA convergence behavior to that of the IPM improves as $r$ increases, but population size alone is inadequate as an indicator of similarity. For example, in the case of the SGA and IPM converging to the same string (Figures 5.22, 5.23, 5.24): As epistasis increases, the rate for the largest population falls below that of smaller populations with lower epistasis.

**Increasing $\varepsilon$:**

It appears that increasing epistasis does reduce the frequency of convergence to the optimum (Figures 5.19, 5.20, 5.21), but at the same time the number of generations required for convergence decreases (Figures 5.38, 5.39, 5.40). The combination of these two factors with the results for convergence to approximately the same fitness (Figures 5.13, 5.14, 5.15) could be interpreted to indicate that both algorithms tend to settle too soon on some nearly optimal fitness population when epistasis is high.

Although there does not seem to be an epistasis level beyond which increases in epistasis have no effect, there does seem to be a range of epistatic values (from about 3/9 to 7/9) within which changes in behavior are relatively small. The relationship
between $\alpha$ and epistasis (in Figure 5.53) provides an explanation: The change in $\alpha$ is flattest within that region, and flatter still as $\ell$ increases, which means $\alpha$-linear fitness functions within those levels of epistasis are more similar on average those taken from other regions.

**Convergence to Optimum**

Comparison of zero hamming distance to optimum for the IPM (Figures 5.28, 5.29, 5.30) and SGA (Figures 5.32, 5.33, 5.34) indicates the correspondence between the SGA and IPM results improves as population size increases.

A point of interest about convergence to optimum is the lack of occurrence of IPM convergence populations with hamming distance 1, and the decrease in the occurrence of hamming distance 2 results as epistasis increases (which is where the occurrence of larger hamming distance results becomes more frequent). For $\ell = 8$, the hamming distance 2 actually becomes less frequent than hamming distances 3, 4, and 5 (Figure 5.34). This seems to be an indication that if an IPM population gets near the optimum (in hamming distance), then it likely will find the optimum.

Another point of interest is how badly the SGA performs with linear fitness functions (Figures 5.1, 5.2, 5.3), in contrast to the IPM, which performs well with linear fitness functions (where the only stable fixed point is the optimum).

**2.2. Transition to Fully Epistatic**

A general feature of almost every chart is a distinct change as $\varepsilon$ increases from $\frac{8}{9}$ to $\frac{9}{9}$. A plot of $\alpha$ vs $\varepsilon$ (from Equation 2.11) shows a large change in $\alpha$ for the same region: For a given $\varepsilon$, the $\alpha$ is determined by the relative size of $\|L\|$ and $\|E\|$ (see Equation 2.10). Assuming $L$ and $E$ are derived from a random fitness vector, the expected values of $L^2$ and $E^2$ are proportional to the number of non-zero terms in $L$. 
In order to investigate $\alpha$ vs $\varepsilon$, we need the expected value of $||\mathbf{C}_2||$ for Equation 2.10. Figure 5.53 uses $\sqrt{\frac{E(\mathcal{L}^2)}{E(\varepsilon^2)}}$ as a plausible value.

A more reliable relationship was obtained by calculating $\alpha$ for 10,000 random fitness vectors (for each $\ell$ value), using Equation 2.10. The empirically obtained expected value of $\alpha$ for various epistasis values is plotted in Figure 5.54.

As $\ell$ increases, the $\alpha$ value for a given $\varepsilon$ decreases, as Figure 5.54 shows. Consider $\ell = 8$, where less than $\frac{1}{5}$ of the transition from linear to epistatic has been achieved at
the point where $\varepsilon = \frac{8}{9}$. In such a situation, the fitness vector can change drastically as $\varepsilon$ goes from $\frac{8}{9}$ to $\frac{9}{9}$. The effect becomes more pronounced as $\ell$ increases. This might explain the results seen at high epistasis levels.

A similar, but smaller, effect occurs at the low epistasis levels, and decreases as $\ell$ increases.

Figure 5.54: Alpha vs Epistasis (Empirical) for $\ell = 3, 4, 6, 8$
2.3. Random Fitness Vectors

For a fitness vector with components selected at random from [0, 1], $\alpha$ is $\frac{1}{2}$, and the epistasis is determined by using Equations 5.1 and 5.2, which depend on $\ell$. Thus, a random fitness vector for strings of length $\ell$ has a level of epistasis centered around $\varepsilon = \frac{2^\ell - 1 - \ell}{2^\ell - 1}$. In other words, random fitness vectors do not on average provide a diverse sampling of epistasis levels.

For example, consider the statistics based on 50 random fitness vectors for $\ell = 4$ and $\ell = 6$, shown in Table 5.1. The lowest and highest bounds (and average epistasis) are calculated from

\[
\text{lowest} = \frac{(\text{ave } ||\mathcal{E}|| - \text{stddev } ||\mathcal{E}||)^2}{(\text{ave } ||\mathcal{L}|| + \text{stddev } ||\mathcal{L}||)^2 + (\text{ave } ||\mathcal{E}|| - \text{stddev } ||\mathcal{E}||)^2}
\]

\[
\text{highest} = \frac{(\text{ave } ||\mathcal{E}|| + \text{stddev } ||\mathcal{E}||)^2}{(\text{ave } ||\mathcal{L}|| - \text{stddev } ||\mathcal{L}||)^2 + (\text{ave } ||\mathcal{E}|| + \text{stddev } ||\mathcal{E}||)^2}
\]

and shown in Table 5.2.
Chapter 6

Conclusion

Most of the points of comparison between the IPM and SGA which were mentioned in the objectives for this thesis were addressed in charts and discussion in Chapter 5: Results.

Here we discuss some of the more general issues.

1. How successful is the thesis at achieving its goals?

   (a) The search for a SGA population size at which the IPM is an effective predictor of convergence behavior produces ambiguous results. Population size is important, and a larger population implies the SGA convergence behavior will more closely approach the IPM behavior, but the population size we seek depends on what is meant by "effective predictor": How closely must the behaviors correspond?

   The epistasis level also has an effect: The IPM and SGA results have different degrees of correlation at zero epistasis and full epistasis.

   The best we can say is: As population size increases, the IPM becomes a better predictor of SGA convergence behavior.
(b) Some of the comparisons were hampered by the choice of large "tolerances" for stability (almost-stable fixed point) and fitness (convergence to almost the same fitness). This made it possible for too large a proportion of the data to qualify for inclusion, resulting in the same rate or frequency for all data (thus providing no information). Repeating calculations with smaller tolerances might make it possible to find some interesting relationships.

2. What further research might be done?

(a) The choice of string lengths was due largely to execution time constraints imposed by the IPM. But some of the results of interest depend only on the SGA. Running those experiments (such as convergence to stable fixed points) for longer string lengths ($\ell = 12, 16, 20$) would provide data to expand the analysis of the effect of string length on behavior.

(b) It is not unreasonable to suspect that the results of this thesis may be due more to how fitness vectors were obtained than to the nature of epistasis itself. Repeating the work with a different scheme for creating fitness vectors of given epistasis (such as Kaufmann’s NK Fitness Table) would provide a good "cross-check".

(c) Many of the graphs show an unexpected change in behavior between epistasis levels $\varepsilon = \frac{8}{9}$ and $\frac{9}{9}$, the region in which the proportion of epistasis ($\alpha$) in the $\alpha$-linear fitness vector undergoes its greatest change. In particular, compare Figures 5.32, 5.33, and 5.34, where the "upturn" at high epistasis is clear for $\ell = 4$ and $\ell = 6$, but not for $\ell = 8$. We postulate the effect for $\ell = 8$ does occur, but between $\varepsilon = \frac{8}{9}$ and $\frac{9}{9}$, where it is not visible, due to our choice of epistasis levels. This possibility should be verified by further investigation.
In fact, this line of reasoning suggests that similar unobserved changes may be occurring in the other charts, and a more detailed investigation of that range of epistasis values would likely produce interesting results.

3. What contributions does this thesis make to the general body of knowledge about genetic algorithms?

(a) Davidor [2] provided "a means to measure the nonlinearity" in a search space. His interest was in samples of the search space, and his epistatic variance was not a definition of epistasis.

For this thesis, we needed a numeric value for epistasis which took into account the proportion of the variance in the search space which was attributable to "nonlinearity". Transforming the fitness vector to Walsh space provided a simple way to separate linear and epistatic contributions to fitness, and calculate the proportion of fitness contributed by epistatic terms. Our definition of epistasis is equivalent to the ratio of Davidor's epistasis variance of the search space to the Aizawa [3] total variance of the search space, as was shown (Proposition 2.1).

But the question remains, whether we should maintain consistency with Davidor and Aizawa in the sense of a ratio of variances (or \( \text{length}^2 \) for us):

\[
\varepsilon = \frac{||\mathcal{E}||^2}{||\mathcal{L}||^2 + ||\mathcal{E}||^2},
\]

or instead use a ratio of lengths:

\[
\varepsilon = \frac{||\mathcal{E}||}{\sqrt{||\mathcal{L}||^2 + ||\mathcal{E}||^2}}.
\]

We see no compelling reason to prefer either choice.

(b) The ability to create a fitness vector of given epistasis from the linear and epistatic fitness vectors in the decomposition of a random fitness vector should be useful for those working with genetic algorithms.
Chapter 7

Appendix — Notation Dictionary

1. Thesis Notation

\( \ell \) — bit-string length

\( r \) — population size

\( f \) — fitness function

\( B \) — bit-string: \( B = b_1 b_2 \ldots b_\ell \)

\( \mathbb{R} \) — the real numbers

\( N \) — the number of possible bit-strings: \( N = 2^\ell \)

\( W \) — the (Vose) Walsh Transformation matrix

\( \mathbf{1} \) — the vector of all 1's: \( (1 1 \ldots 1)^T \)

\( \mathbf{0} \) — the vector of all 0's: \( (0 0 \ldots 0)^T \)

\( \mathbf{d} \) — the vector \( (2^0 2^1 \ldots 2^{\ell-1})^T \)

\( \mathbf{x} \) — an individual in the population: \( (x_0 x_1 \ldots x_{\ell-1})^T \)

\( x_i \) — the \( i^{th} \) component of \( \mathbf{x} \): a binary digit

\( \mathbf{x}^T \mathbf{d} \) — the number associated with the vector \( \mathbf{x} \)

\( S_z \) — the string \( x_\ell \ldots x_2 x_1 \) associated with the vector \( \mathbf{x} \)

\( A_k \) — the \( k^{th} \) column vector of matrix \( A \)
$X^*$ — the bipolar form of $X$: $X_j^* = (2^{\frac{j}{2}}) W_{2j}$

$X$ — the unipolar form of $X^*$: $X = \frac{1}{2} \left( 11^T - X^* \right)$.

The associated integers of the row vectors of $X$ are 0, 1, 2, \ldots, $2^\ell - 1$.

$f_i$ — the $i^{th}$ component of fitness vector $f$, a non-negative real number

$f_x$ — the fitness value of bitstring $S_x$ of length $\ell$

$f^\ell$ — a fitness vector for bitstrings of length $\ell$

$\hat{f}$ — the fitness vector $f$ in the Walsh basis

$c$ — the constant vector $(c_0, c_1, \ldots, c_{2l})^T$ for linear fitness functions,

$$f_x = c_0 + \hat{c}^T \hat{x}$$

$Y^\ell$ — a $2^\ell \times \ell$ matrix with entries $Y_{i,j} = 1$ if $i = 2^j$, and 0 otherwise.

$C_g$ — the constant term of the decomposition of fitness vector $g$ into constant, linear, and epistatic terms

$L_g$ — the linear term of the decomposition of fitness vector $g$ into constant, linear, and epistatic terms

$E_g$ — the epistatic term of the decomposition of fitness vector $g$ into constant, linear, and epistatic terms

$L$ — a purely linear fitness vector: $L = C_L + L_L$

$E$ — a purely epistatic fitness vector: $E = C_E + E_E$

$r_i$ — the $i^{th}$ component of vector $E_g$

$\varepsilon_g$ — the epistasis of fitness vector $g$

$C^2$ — alternate notation for $C^T C$

$L^2$ — alternate notation for $L^T L$

$E^2$ — alternate notation for $E^T E$

$\alpha$ — proportion of epistasis in an $\alpha$-linear fitness vector
2. Davidor Notation

\[ S \rightarrow \text{bit-string} \]
\[ v \rightarrow \text{fitness function} \]
\[ v(S) \rightarrow \text{fitness of string } S \]
\[ \overline{V} \rightarrow \text{average fitness of all strings: mean fitness} \]
\[ E_j(b) \rightarrow \text{the effect of bit } b \text{ at position } j \text{ in a string:} \]
\[ \text{the difference between the average fitness of strings with bit } b \in \{0, 1\} \text{ at position } j, \text{ and the average fitness of all strings} \]
\[ A(S) \rightarrow \text{genic fitness: a fitness calculated from } v(S) \text{ and } E_j(b) \]
\[ \sigma_E^2 \rightarrow \text{epistatic variance: a measure of the difference between } A(S) \text{ and } v(S) \]
\[ N \rightarrow \text{the number of strings} \]

3. Aizawa Notation

\[ L \rightarrow \text{bit-string length: the number of strings is } 2^L \]
\[ A \rightarrow \text{the solution space: all possible bit-strings of length } L \]
\[ F(x) \rightarrow \text{the fitness of string } x \]
\[ \mu_A \rightarrow \text{the mean fitness of strings in } A \]
\[ f_{(u_i)}(x^{(u_i)}) \rightarrow \text{Davidor's } E_j(b), \text{ in Aizawa's notation} \]

(For partitions other than that corresponding to the standard
(Cartesian) basis, this represents a generalization of Davidor's approach, in the sense that the effects of bitstrings can be handled, as well as single bit positions)

\( w_i \) — the \( i^{th} \) Walsh function (Paley)

4. Vose Notation

\( \Omega \) — domain of the SGA: the set of all bit-strings of length \( \ell \)
\( e_k \) — \( k^{th} \) column of the identity matrix \( I \): vertices of \( \Lambda \)
\( \oplus \) — bitwise exclusive-OR operation
\( \otimes \) — bitwise AND operation
\( \bar{x} \) — complement of \( x \): \( x \oplus 1 \ldots 1 \)
\([expr]\) — evaluate the predicate \( expr \): \([expr]\) is 1 if \( expr \) is true, else 0
\( \delta_{i,j} \) — \([i = j]\)
\( \sigma_k \) — \( n \times n \) permutation matrix: \((\sigma_k)_{i,j} = \delta_{i \oplus k,j}\)
\( \bar{x} \) — population vector: \( x_i \) is the proportion of the population represented by the string associated with the integer \( i \)
\( \mathbb{R} \) — the set of all real numbers
\( \Lambda \) — the unit simplex: \( \Lambda = \{(x_0, \ldots, x_{2^\ell-1})^T : \bar{x}^T\bar{1} = 1, x_j \geq 0\} \)
\( \mathcal{F} \) — the proportional selection operator (matrix)
\( F \) — diagonal \( 2^\ell \times 2^\ell \) matrix with entries \( F_{i,j} = \delta_{i,j}f_i \)
\( k \) — crossover mask: \( k \in \Omega \)
\( \chi_k \) — crossover (probability) distribution: the probability \( k \) will be the crossover mask
\( \mu \) — the mutation probability distribution
\( C \) — crossover rate
\( \mathcal{M} \) — mixing operator (matrix): performs crossover and mutation

\( M_{i,j} \) — probability that mixing \( i \) and \( j \) produces \( \tilde{0} \)

\( R \) — the bitwise mutation rate

\( G \) — the IPM genetic algorithm

\( U, V \) — neighborhoods about a point in the simplex

\( A^* \) — the "twist" of \( A \): \( A^*_{i,j} = A_{i\oplus j,i} \)

\( d\mathcal{M}_x \) — the differential of \( \mathcal{M} \) at \( x \)

\( dG_x \) — the differential of \( G \) at \( x \)

\( \text{spec}(dG_{e_k}) \) — spectrum (set of eigenvalues) of \( dG \) at vertex \( e_k \)

\( \rho(dG_x) \) — spectral radius: the maximum of the spectrum of \( dG \) at \( x \)
Chapter 8

References
Bibliography


