To the Graduate Council:

I am submitting herewith a thesis written by Bill Clyde White entitled "Random Walk Population Sizing as a Model of Decision Making in a Deceptive Schema Partition." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Bruce Whitehead, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of the Graduate School

Random Walk Population Sizing as a Model of Decision Making in a Deceptive Schema Partition

A Thesis

Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Bill C. White

December 1999

Dedication

This thesis is dedicated to my parents and friends who provided endless encouragement and understanding.

Acknowledgments

I thank my advisor and thesis committee chairman, Dr. Bruce Whitehead, for the guidance and assistance he has provided. I also thank the other members of my committee, Dr. Dinesh Mehta and Dr. Kenneth Kimble. The merit of this work is due in no small part to their suggestions.

Abstract

Population-sizing models have been shown to be accurate predictors of genetic algorithm convergence. The random walk model incorporates and improves upon previous work in population sizing. This research investigates the ability of the random walk model to predict the underlying genetic mechanisms of schema processing. These mechanisms are evaluated by examining the decisions a genetic algorithm makes in deceptive schema partitions. Results of this research indicate that the random walk model is an accurate predictor of schema partition processing in deceptive partitions. Visualization of the ongoing competition among the schemata in a partition as vertices of a hypercube confirm the dynamics of schema processing predicted by the model. Taken together, the visualization of the process and the statistics of the results help to confirm the random walk model of schema processing at a more detailed level than previous studies of end results only.

Contents

1 Introduction 1

2 GA Population Sizing Models 4

2.1 Genetic Algorithms 4

2.2 Deception in Genetic Algorithms 8

2.3 Review of GA Population Sizing 9

2.3.1 Building Block Supply Models 9

2.3.2 Building Block Decision Models 11

3 Research Methods 27

3.1 Approach 27

3.2 GA Implementation 28

3.2.1 Random Number Generator 29

3.2.2 Binary Tournament Selection 29

3.2.3 Schema Visualization 31

3.2.4 GA Validation 33
3.3 Experimental Method 36
3.3.1 Fitness Function 37
3.3.2 Random Walk Model Evaluation Procedures 38
3.3.3 Schema Allocation Visualization 44

4 Results and Conclusions **45**
4.1 Experimental Results 45
4.1.1 Schema Visualization 49
4.2 Conclusions 65

Bibliography **67**

Appendix **71**

A Partiton Graphs **72**

Vita **89**

List of Figures

2.1 Single-point crossover. 6

2.2 A genetic algorithm. 7

2.3 Computing the probability of choosing incorrectly between the two best building blocks (schemata) with a single sample of each. . . . 13

2.4 Gambler's ruin problem. 20

3.1 Binary tournament algorithm. 30

3.2 A frame from a bit schema visualization animation. 31

3.3 Validation of *bsv* using F1. 34

3.4 Validation of *bsv* using F2. 34

3.5 Validation of *bsv* using F3. 35

3.6 Validation of *bsv* using F4. 35

3.7 Validation of *bsv* using F5. 36

3.8 f_{45} subfunction, 4-bit, fully deceptive. 37

3.9 Evaluation graph set. 40

3.10 Miller prediction of final convergence as a function of population size. 42

4.1 Miller prediction using small N, Equation 2.33. 49

4.2 Schema visualization of generation 0. 50

4.3 Schema visualization of generation 1. 51

4.4 Schema visualization of generation 2. 52

4.5 Schema visualization of generation 3. 53

4.6 Schema visualization of generation 4. 54

4.7 Schema visualization of generation 5. 55

4.8 Schema visualization of generation 6. 56

4.9 Schema visualization of generation 7. 57

4.10 Schema visualization of generation 8. 58

4.11 Schema visualization of generation 9. 59

4.12 Schema visualization of generation 10. 60

4.13 Schema visualization of generation 11. 61

4.14 Schema visualization of generation 12. 62

4.15 Schema visualization of generation 13. 63

4.16 Schema visualization of generation 14. 64

A.1 Population Size=21, Seed=71659920. 73

A.2 Population Size=21, Seed=1688918702. 74

A.3 Population Size=21, Seed=1507297010. 75

A.4 Population Size=21, Seed=1887561901. 76

A.5 Population Size=21, Seed=798798137. 77

A.6 Population Size=21, Seed=1934811071. 78

A.7 Population Size=21, Seed=1207155841. 79

A.8 Population Size=21, Seed=1450788425. 80

A.9 Population Size=127, Seed=71659920. 81

A.10 Population Size=127, Seed=1688918702. 82

A.11 Population Size=127, Seed=1507297010. 83

A.12 Population Size=127, Seed=1887561901. 84

A.13 Population Size=127, Seed=798798137. 85

A.14 Population Size=127, Seed=1934811071. 86

A.15 Population Size=127, Seed=1207155841. 87

A.16 Population Size=127, Seed=1450788425. 88

List of Tables

3.1 GA parameter constants. 29

3.2 Test suite from Goldberg et al. [5]. 33

3.3 Partition graph classification. 39

3.4 Classification prediction formulas. 43

4.1 Classification prediction for studied population sizes. 47

4.2 Experimental results for studied population sizes. 47

4.3 Prediction error in number expected cases. 47

4.4 Classification prediction for studied population sizes. New RWM prediction. 48

4.5 Prediction error in number of expected cases. New RWM prediction. 48

List of Symbols

d	Signal difference between best and second-best schema fitness averages in a partition
f_{s_1}	Average fitness of the best schema in a partition
f_{s_2}	Average fitness of the second-best schema in a partition
$f(\vec{x})$	Fitness function for string $\vec{x}$
f_{45}	4-bit, fully-deceptive fitness subfunction
g	Subscript for gambler's ruin variables
h	Arbitrary element in recursive derivation of gambler's ruin
i	Integer indexing the partitions in a problem
j	Summation index of partition fitness variances
k	Schema/partition order (building-block size)
ℓ	String length of individuals (chromosomes) in the population
m	Number of subfunctions in each string
n	Summation index of fitness function subfunctions
$o(s)$	Order of schema s
p	Probability of choosing correctly with a single sample of each schema
p_g	Probability of gambler's ruin particle moving toward N (winner absorbing state)

p_i Probability of choosing correctly with a single sample of each schema in partition i

q Probability of choosing incorrectly with a single sample of each schema

q_g Probability of gambler's ruin particle moving toward 0 (loser absorbing state)

q_i Probability of choosing incorrectly with a single sample of each schema in partition i

s A schema

s_1 Best schema within a partition

s_2 Second-best schema within a partition

t Arbitrary point in gambler's ruin sequence of trials

u Recursive function relating probability of ruin to change in capital in gambler's ruin derivation

w Number of GA runs per population size

$\vec{x}$ Arbitrary string to be evaluated for fitness

x_0 Estimate of the number of copies of a given building block in a randomly-initialized population

x_{0g} Gambler's starting capital in gambler's ruin problem

y Dummy variable representing starting capital in u

$z(q)$	Deviate value of the unit normal distribution which yields tail probability q
z_{norm}	Normalized result of a binary tournament
z_{result}	Result of a binary tournament
A	Number of cases where the globally-optimum building block is absent from the initial population
B	Number of cases where random walk reinforces correct initial decision and leads to takeover
C	Number of cases where random walk reverses correct initial decision and leads to ruin
D	Number of cases where random walk reinforces incorrect initial decision and leads to ruin
E	Number of cases where random walk reverses incorrect initial decision and leads to takeover
I	Index set for $\vec{x}$
$IDMM$	Predicted probability of correct initial decision
N	Population size
N_g	Total resources in the gambler's ruin problem
P_k	Probability that at least one string in the population has the optimum building block

P_N	Probability of obtaining all N copies of a given building block in the population
P_{N_g}	Probability of obtaining all N units in the gambler's ruin problem
Q_k	Probability that every string in the population lacks the optimum building block
RWM	Random walk model predicted probability of takeover by optimum building block
Φ	Cumulative distribution function for the unit normal distribution
Ψ	Failure rate for gambler's ruin population-sizing
σ_f^2	Variance of fitness function
σ_n^2	Variance of fitness function subfunction n
σ_{rms}^2	Root-mean-squared subfunction fitness variance
$\sigma_{s_1}^2$	Variance of fitness of the best schema in a partition
$\sigma_{s_2}^2$	Variance of fitness of the second-best schema in a partition
σ_F^2	Variance of fitness of the population
σ_M^2	Mean variance of fitness of competing building blocks
σ_i^2	Variance of partition i
σ_j^2	Variance of subfunction j

List of Abbreviations

bsv	Bit Schema Visualization
BB	Building block
GA	Genetic algorithm
S/N	Signal-to-noise ratio

Chapter 1

Introduction

The population in a genetic algorithm (GA) at a particular generation represents samples of the search space being explored; consequently, population size is closely related to the efficacy of GA processing. An initial, usually random, population supplies the GA with a collection of partial solutions (building blocks), which serves as the only source of diversity (assuming no mutation) during the run. This initial supply of building blocks must provide the necessary components for the selective-recombinative mechanisms of the GA to converge to the optimal solution [8]. During each generation of the run, the selection mechanism of the GA must choose correctly among competing building blocks to form the basis of the next generation.

Population sizing models useful for predicting GA convergence have been derived from schema decision models. Such models estimate fitness distributions

of competing schemata to determine the probability of selecting correctly in a binary tournament in the first generation [5], [6], [7], [10], [13], [14]. The random walk model, based on the classic gambler's ruin problem [2], has been related to previous population sizing research and incorporated into a unified model of GA convergence [8], [11], [12]. In all such models, the population size necessary for the GA to converge correctly at the end of the run with a specified confidence is computed. The assumption, based on the Schema Theorem [9], is that correct decisions in the first generation will lead to correct decisions in the intermediate stages of the run. Previous studies have validated this assumption only in terms of final results; evidence of correct decision making during the run has not previously been presented. This research provides evidence of the effectiveness of the random walk model to predict building block decisions in deceptive schema partitions during the run.

To ensure the GA used in this research produces results comparable to previous models, the implementation details of the research GA are presented. Variables and constants are defined to isolate the effects of experimental parameter adjustments. The implementation and limits of the random number generator are discussed. Schema visualization is introduced as a visual representation of schema processing during the run. Performance is compared to results and predictions from research by Goldberg, Deb, and Clark [5] and Miller [11].

The study of the decision-making process in deceptive partitions then exam-

ines GA simulations of the test function at several population sizes (confidence levels). Competitions in the schema partitions are classified into convergence categories, which correspond to decision making behavior during the entire run. The population sizing models are used to predict the expected number of cases in each category at each population size. Predicted and observed values are compared to evaluate the ability of the model to predict schema processing.

The random walk model is shown to improve upon previous models by more accurately reflecting the decisions taking place after the first generation. Results show that in partitions of a deceptive fitness function, the random walk model accurately predicts the decision-making process between competing schemata during the run. Since approximations are used in population sizing derivations, the definition and effect of small population size are considered and shown to influence the results of this study.

Chapter 2 provides background information on genetic algorithms and population-sizing models. The building-block supply and decision-making models are presented. A special form of the random walk, the gambler's ruin, is derived and adapted for GA use. Chapter 3 presents the methods used to evaluate the random walk model. Chapter 4 discusses the results and conclusions of the research.

Chapter 2

GA Population Sizing Models

Genetic algorithms are relatively simple procedures that exhibit complex behavior. This behavior has been analyzed from the theoretical viewpoints of schema processing and deception, which form the basis of population sizing models.

2.1 Genetic Algorithms

Genetic algorithms are stochastic search procedures based on Darwinian evolution, particularly natural selection, and modern genetics. More thorough introductions to genetic algorithms are presented in Goldberg [3] and Mitchell [15].

GAs work with a population of individuals, encoded as strings of length ℓ, called *chromosomes*. Chromosomes in this research are restricted to binary (bit) strings, though alphabets of other cardinalities have been studied. Each character location, or *locus*, identifies the position of a bit. Each of the letters in the

set of allowed characters ("0" and "1" here) is called an *allele*. Combinations of alleles encode domain-specific parameters: integers, real numbers, etc. During each iteration of the algorithm, called a *generation*, strings are selected for reproduction based on their *fitness*. Fitness is defined by a function that maps each bit string to a real number. The fitness function decodes the domain-specific parameters from the bit string, and it is the only part of the GA that depends on the problem domain. The GA knows nothing of what the bit strings mean, only that it is selecting and recombining them according to the fitness function given to it. Strings with higher fitness are expected to be selected more often, this is the evolutionary analog to "survival of the fittest," though selection operators are varied and can range from deterministic to random in their choice of parents for the next generation. In the reproduction phase, at a problem-defined probability, the parent chromosomes chosen in the selection phase exchange or *crossover* substrings (the process is also referred to as *recombination*) at a random point in the ℓ-bit string. In this research simple, single-point crossover is used (other types of crossover include n-point, uniform, etc). Figure 2.1 depicts a single-point crossover. Crossover is analogous to the exchange of genetic material during sexual reproduction in biological systems. Parents not chosen for crossover are simply put into the next generation unchanged. Though not used in this research, mutation is the last phase of the algorithm and the final link to evolutionary procedures employed in GAs. Mutation can be used to perturb bits at a very low frequency,

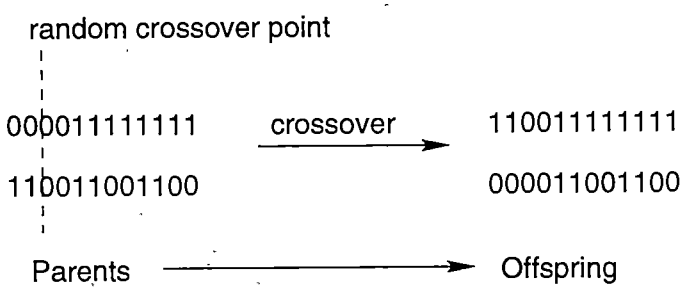


Figure 2.1: Single-point crossover.

thus providing an "insurance policy" against bits getting "stuck" [15]. The GA repeats until a specified number of generations have elapsed, or until a measure of convergence has been reached. The sequence from initialization to convergence is called a *run*. The basic form of a genetic algorithm is shown in Figure 2.2.

Since its introduction by Holland [9], GA behavior has been characterized in terms of schema processing. Schemata, also referred to as similarity templates, hyperplanes, or building blocks, represent subsets of the search space. Schemata are specified by strings with characters from the allele alphabet, called defined positions, and wildcard or "don't care" positions, represented by "*". Given the schema alphabet for bit strings, $\{0, 1, *\}$, has length three, and the string length is ℓ, there are 3^ℓ possible schemata. The number of defined positions is called the order of the schema. For any particular schema s, there are $2^{\ell - o(s)}$ possible string representatives of s, where $o(s)$ is the order of the schema. In other words, a schema s is a subset containing $2^{\ell - o(s)}$ of the 2^ℓ possible strings in the search space. The set of strings matching that schema in any given generation

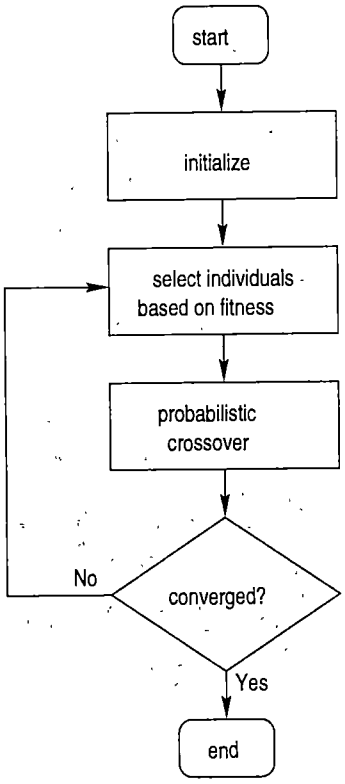


Figure 2.2: A genetic algorithm.

constitute a sample of the schema, whose fitnesses have a well-defined sample mean and sample variance. The Schema Theorem, introduced by Holland [9], claims that short (have small *defining length*, distance between defined positions in the schema), low-order, above-average schemata receive exponentially more samples in subsequent generations. This idea, considering schemata the units of genetic processing rather than the strings, is the basis for many theoretical analyses, including population sizing models. For this purpose schemata are grouped into *partitions.*

A partition is a set of schemata specified by a string composed of the charac-

ters "d" (defined positions) and "*" (wildcards); therefore, there are 2^ℓ possible schema partitions. A partition defines a division of the search space into $2^{o(s)}$ directly-competing schemata, where $o(s)$ is the order of the schemata in the partition; therefore, a partition can be thought of as a view into the population at a particular order. Viewed in this way, the Schema Theorem can be extended to competitions within schema partitions, rather than at the population level; that is, the best observed schema within a partition receives exponentially more samples in subsequent generations [15].

2.2 Deception in Genetic Algorithms

GA deception has been introduced as a way to make problems controllably hard for a GA to solve. The term is used to describe purposely misleading problems, that is, problems in which the fitness of low-order schemata say little or nothing about those of high-order schemata. Therefore, by sampling low-order schemata, the GA is misled about which schemata should receive more samples in subsequent generations. If the population size is too small, the GA will be led away from the global optimum toward the lower-order, deceptive, local optima. Results from GA simulations show that properly-sized populations overcome deception and converge to the optimum solution in a process predicted by the random walk model [11].

2.3 Review of GA Population Sizing

Population-sizing models predict the quality of GA convergence at the end of the run by adjusting the size of the population. Larger population sizes yield more diverse initial populations (more varied samples of the search space), facilitating a good mix of building blocks during selection and crossover, especially in the early stages. Unfortunately, processing more strings at each generation is computationally expensive and requires more processing time to reach convergence. Smaller populations converge faster; however, since a smaller population is likely to lose diversity quickly due to biased reallocation of samples at each generation, the probability of premature convergence to suboptimal solutions increases as the population size decreases. In the extreme case, if the population is small enough, none of the optimal building blocks will be present in the initial generation, and an optimal solution cannot be obtained [11]. To strike a balance between fast convergence and optimal solutions, mathematical models have been developed that attack the problem from two fronts: building block supply models and building block decision models.

2.3.1 Building Block Supply Models

In traditional implementations, the GA starts with a randomly- initialized population of strings. If the initial population does not contain enough optimal building

blocks, the GA is unlikely to converge to the optimal solution through mixing of suboptimal building blocks [8]. Building block supply models ensure the initial supply of building blocks contains at least one copy of the optimal building block. Since the probability that a particular building block of order k is present in a random string is $\frac{1}{2^k}$, the number of copies present in a randomly-generated population of size N is estimated:

$$x_0 = \frac{N}{2^k} \tag{2.1}$$

Setting x_0 to one and solving for N gives an estimate of the population size necessary to ensure at least one optimal building block is present:

$$N = 2^k \tag{2.2}$$

While these equations are useful for increasing convergence quality using small populations and determining the minimum population necessary for processing building blocks containing the optimum, more analysis is needed to ensure that the optimal building blocks present in the initial population lead to convergence to the optimal solution. Building block decision models address this issue.

2.3.2 Building Block Decision Models

Building block decisions are made by the GA selection mechanism. The decision quality of the selection, represented by the probability p of choosing correctly (or q for choosing incorrectly) between representatives of competing building blocks (strings), is used to derive population-sizing equations. Research has studied initial, first-generation selection as an exercise in statistical decision making [5]. To consider the decisions made at each generation during the run, a random walk model has been developed by Harik et al. [8] and studied by Miller [11]. The random walk model incorporates the building block supply and initial decision making models.

Initial Decision Making

According to Holland's Schema Theorem [9], while the GA selection mechanism is processing fitness values algorithmically, a deterministic choice, the GA is also making decisions at the building block level in each of the 2^ℓ building block partitions (this is sometimes referred to as *implicit parallelism*). The choice between two competing building blocks in a partition can be characterized in terms of statistical decision theory, since building block fitness averages and other derivative statistics are determined by sampling. By the central limit theorem, if enough samples are taken, the distributions will be approximately normal. Choosing correctly using these distributions has been posed as a decision-making problem

based on the detection of signal amongst noise [10]; that is, a discrimination of fitness difference (signal) amongst fitness variance (noise) of competing building blocks. From this viewpoint, the probability of choosing the correct building block can be calculated.

Goldberg et al. [5] consider a tournament between two randomly selected strings, representing two directly competing building blocks in partition i; that is, the strings represent the best and second-best building blocks in the partition. The best buiilding block (schema) s_1 has average fitness f_{s_1} with fitness variance σ^2_{s1}, and the second-best building block s_2 has average fitness f_{s_2} with fitness variance σ^2_{s2}. From these means and variances, and assuming normal distributions, the building block distributions can be estimated. Figure 2.3(a) shows how these distributions in the initial population might look graphically. The overlap of the two distributions represents the probability of a tournament choosing the wrong building block for further sampling (and thus reducing samples to the better of the two building blocks), since the overlapping tail of the second-best building block distribution represents the chance of choosing a string with higher fitness representing a suboptimal building block over a less fit representative of the better building block. A convolution is used to calculate this probability of an incorrect decision. The result of the convolution operation is a normal distribution with known properties: the mean of the convolution is the difference in fitness means of the two distributions and the variance of the convolution is the sum of the

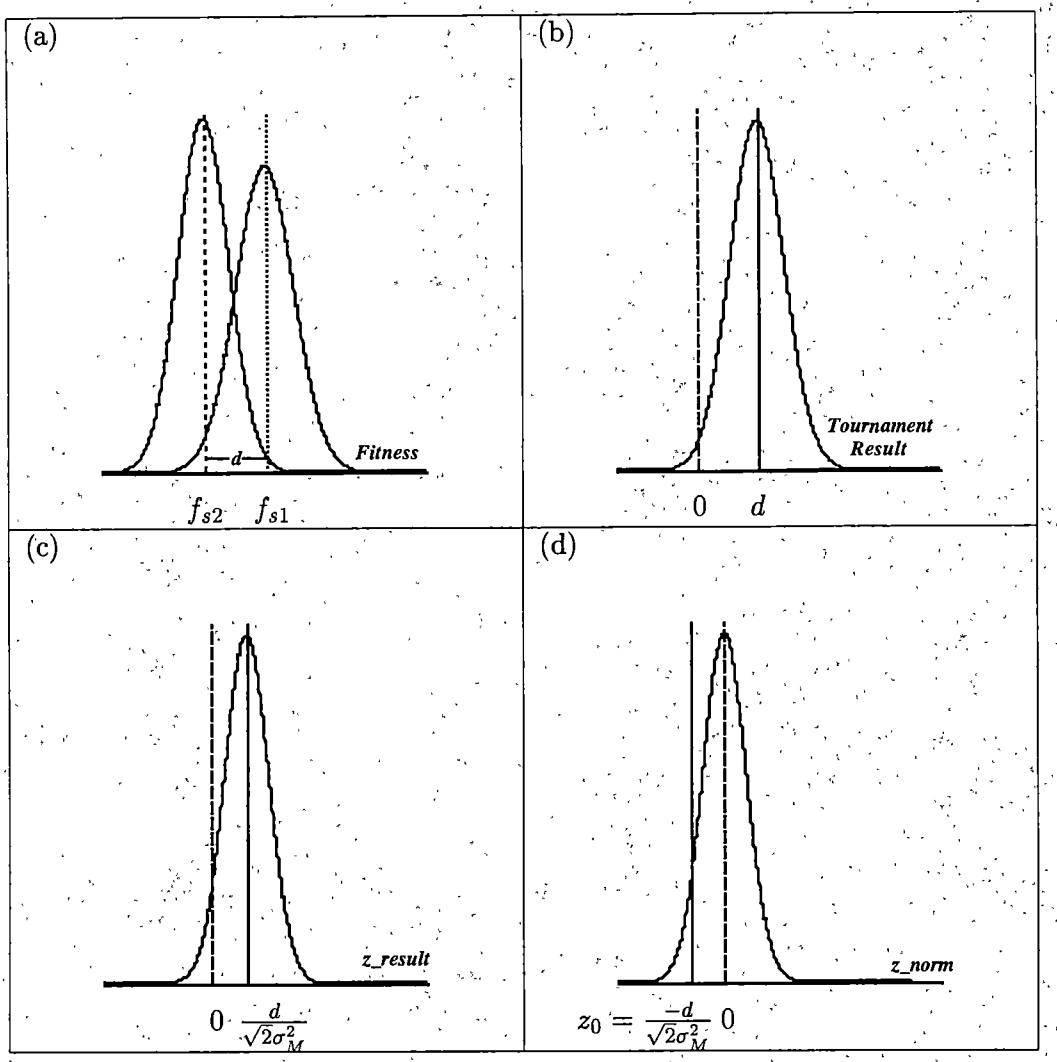


Figure 2.3: Computing the probability of choosing incorrectly between the two best building blocks (schemata) with a single sample of each. (a) Competition between two schemata fitness average distributions. (b) The result of the convolution between distributions in (a). (c) Convolution scaled to variance=1. (d) Convolution translated to mean=0.

distribution variances. This convolution gives the distribution function of the tournament result, where *tournament result* is defined to be the fitness of a string randomly selected from the sample of s_1 in the current population, minus the fitness of a string randomly selected from s_2. If this tournament result is greater than zero, a correct decision will be made; if less than zero, an incorrect decision will be made.

Defining signal difference as the difference in distribution means,

$$d = f_{s_1} - f_{s_2} \tag{2.3}$$

and mean variance as

$$\sigma_M^2 = \frac{\left(\sigma_{s_1}^2 + \sigma_{s_2}^2\right)}{2} \tag{2.4}$$

the mean of the resulting distribution of tournament results is d and the variance is $2\sigma_M^2$. Figure 2.3(b) shows the distribution of tournament results from the convolution of the original distributions. Here, a tournament result greater than zero means that the representative of s_1 randomly selected to participate in the tournament has higher fitness than the randomly selected representative of s_2. To determine the probability of making an incorrect decision (a tournament result less than zero), the result of the convolution must be normalized (mapped onto the unit normal distribution). The distribution is scaled to give a variance of one by dividing the tournament result by its standard deviation. Figure 2.3(c) shows

the result of this scaling, yielding a distribution mean of $\frac{d}{\sqrt{2}\sigma_M}$. Recall that the decision made by a binary tournament is incorrect when the tournament result is less than zero. This probability is represented by the area under the Gaussian distribution to the left of the vertical line where $z_{result} = 0$, where z_{result} is the tournament result. If we then translate this distribution to a mean of zero by defining $z_{norm} = z_{result} - \frac{d}{\sqrt{2}\sigma_M}$, then the vertical line where $z_{result} = 0$ becomes the vertical line where $z_{norm} = \frac{-d}{\sqrt{2}\sigma_M}$. Figure 2.3(d) shows the translation of the distribution. The probability of an incorrect decision is the area to the left of $z_0 = \frac{-d}{\sqrt{2}\sigma_M}$. This area is simply $\Phi(z_0)$, the cumulative normal distribution function of z_0. The absolute value of z_0 is called *signal-to-noise ratio* (S/N) [5].

For convenience, let z denote the inverse of the cumulative normal distribution function Φ. That is, given a probability q, $z(q)$ is the value whose cumulative normal distribution function is q. Using this terminology, the probability of an incorrect decision is:

$$q = \Phi\left(\frac{-d}{\sqrt{2}\sigma_M}\right) \tag{2.5}$$

or equivalently:

$$z\left(q\right) = \frac{-d}{\sqrt{2}\sigma_M} \tag{2.6}$$

Though not used explicitly in Goldberg's analysis, the probability of choosing correctly in the partition is important for analyses extending Goldberg's work.

Since $q = 1 - p$, Equation 2.6 is equivalent to:

$$z\,(1-p) = \frac{-d}{\sqrt{2}\sigma_M} \tag{2.7}$$

Solving for p gives the probability of choosing correctly with one sample of each schema:

$$\begin{aligned} 1 - p &= \Phi\left(\frac{-d}{\sqrt{2}\sigma_M}\right) \\ p &= \Phi\left(\frac{d}{\sqrt{2}\sigma_M}\right) \end{aligned} \tag{2.8}$$

where Φ is the cumulative distribution function for the unit normal distribution.

Equation 2.6 is stated in terms of the probability q of incorrectly deciding between two competing building blocks, considering only one sample of each. To include the standard-deviation reducing effects of larger samples, the equation is modified to reduce the variance (noise) by considering the likely number of samples allocated to each building block in a partition of order k. To incorporate the likely number of samples, Equation 2.6 is squared to use variance rather than standard deviation in the denominator:

$$z^2\,(q) = \frac{d^2}{2\sigma_M^2} \tag{2.9}$$

Although Equation 2.9 has both positive and negative roots, the negative root is

the one consistent with Equation 2.6.

Since variance of the sample mean is the variance of a single sample divided by the number of expected samples, substituting Equation 2.1 for the number of samples, the modified equation becomes:

$$z^2(q) = \frac{d^2}{(2\sigma_M^2)/(\frac{N}{2^k})} \tag{2.10}$$

Suppose we wish to set the probability of an incorrect decision q to some desired value. The population size N necessary to achieve this result is then given by solving Equation 2.10 for N:

$$N = 2z^2(q)\frac{\sigma_M^2}{d^2}2^k \tag{2.11}$$

To incorporate the size and difficulty of the problem, consider a fitness function composed of m independent subfunctions, each the size of the most deceptive partition k, yielding $m = \frac{\ell}{k}$ subfunctions. Each subfunction contributes to the total variance of the fitness function:

$$\sigma_F^2 = \sum_{n=1}^{m} \sigma_n^2 \tag{2.12}$$

The root mean squared (RMS) subfunction variance is:

$$\sigma^2_{rms} = \frac{\sigma^2_F}{m} \tag{2.13}$$

The variance of the average order-k schema is estimated by observing that the other $m-1$ subfunctions (the order-k function under consideration is fixed) are a source of noise to decision making (*collateral noise*):

$$\sigma^2_M = \sigma^2_{rms}\,(m-1) \tag{2.14}$$

Defining $m' = m-1$, the population sizing equation becomes:

$$N = 2z^2\,(q)\frac{\sigma^2_{rms}}{d^2}2^k m' \tag{2.15}$$

Equation 2.15 provides a conservative estimate of the population size necessary to ensure correct building block decision making, since it only considers the decision making in the first generation of the run. The underlying assumption is that if an initial move in the right direction in the search space is taken, the normal mechanics of GA selection and crossover will ensure the necessary mixing and retention of optimal building blocks. Likewise, if a step is taken in the wrong direction, the GA will "lose" building blocks and never regain the diversity necessary to converge to the optimum solution. Results in Goldberg et al. [5] show

that Equation 2.15 is conservative for a range of GA test functions. Research by Harik et al. [8] extends initial decision making by incorporating it into a random walk model that considers decisions made at each generation of the GA run.

Random Walk Model

The gambler's ruin is an example of a random walk stochastic process [2]. A stochastic process is one that is influenced by chance or random effects over a sequence of times. The random walk model considers a particle moving in a one-dimensional space (a line) with probabilities for moving along the line in a positive or negative direction at each time step. The gambler's ruin is a special form of the random walk in which the particle is allowed to move only one unit at a time and is subject to absorbing boundaries, at which the particle can no longer move. The model is usually thought of as a game between two players, a gambler and an opponent. The gambler starts with some amount x_{0g} of the fixed resources N_g. This leaves the opponent with $N_g - x_{0g}$ units. In each game, the gambler can win or lose one unit, with probabilities p_g and q_g respectively. Absorbing boundaries are at 0 (gambler is ruined) and N_g (opponent is ruined), and once the gambler reaches either boundary the game is over. The situation is shown graphically in Figure 2.4.

The analytical result of the model is an equation to compute the probability that the gambler will win all the units N_g. Let $f(x_{0g})$ represent the probability

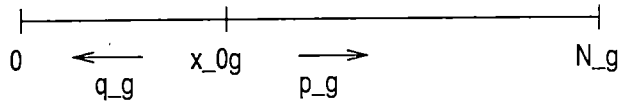


Figure 2.4: Gambler's ruin problem.

that the gambler will ruin (reach the 0 absorbing barrier) starting with x_{0g} units. On the first trial, the gambler can either win with probability p_g and proceed to lose $x_{0g}+1$ units or lose with probability q_g and proceed to lose $x_{0g}-1$ units. Expressed as a difference equation:

$$f(x_{0g}) = q_g f(x_{0g}-1) + p_g f(x_{0g}+1) \tag{2.16}$$

To express the difference equation as a recurrence relation on one less unit, recall $p_g + q_g = 1$; so Equation 2.16 becomes:

$$\begin{aligned}
(p_g+q_g)f(x_{0g}) &= q_g f(x_{0g}-1) + p_g f(x_{0g}+1) \\
p_g f(x_{0g}) + q_g f(x_{0g}) &= q_g f(x_{0g}-1) + p_g f(x_{0g}+1) \\
p_g\left(f(x_{0g}+1) - f(x_{0g})\right) &= q_g\left(f(x_{0g}) - f(x_{0g}-1)\right) \\
f(x_{0g}+1) - f(x_{0g}) &= \left(\frac{q_g}{p_g}\right)\left(f(x_{0g}) - f(x_{0g}-1)\right) \qquad (2.17)
\end{aligned}$$

provided x_{0g} is not at an absorbing boundary, i.e. $0 < x_{0g} < N_g$. The recurrence in this formula can be seen more clearly if we arbitrarily define a function u to elucidate the effect of a change in starting capital on the probability of ruin. More

precisely, for any nonnegative integer y, define

$$u(y) = f(y+1) - f(y) \tag{2.18}$$

Thus, u represents the change in probability of ultimate ruin that would result from one unit change in the starting capital.

Using this definition of u, Equation 2.17 becomes

$$u(x_{0g}) = \left(\frac{q_g}{p_g}\right)(u(x_{0g} - 1)) \tag{2.19}$$

which applied h times becomes

$$u(x_{0g}) = \left(\frac{q_g}{p_g}\right)^h (u(x_{0g} - h)) \tag{2.20}$$

in which the hth application depends upon not having reached an absorbing boundary after $h-1$ applications.

In other words, the hth application of Equation 2.19 assumes that $(x_{0g}-(h-1))$ is not an absorbing state, which is equivalent to assuming $0 < x_{0g} - (h-1) < N_g$. This means that Equation 2.20 is valid for all values of h less than $x_{0g}+1$, and in particular for $h = x_{0g}$, yielding

$$u(x_{0g}) = \left(\frac{q_g}{p_g}\right)^{x_{0g}} (u(x_{0g} - x_{0g})) \tag{2.21}$$

$$= \left(\frac{q_g}{p_g}\right)^{x_{0g}} (u(0)) \tag{2.22}$$

substituting our definition of u into this equation yields

$$f(x_{0g}+1) - f(x_{0g}) = \left(\frac{q_g}{p_g}\right)^{x_{0g}} (f(1) - f(0)) \tag{2.23}$$

Since the chance of ruin starting with 0 units is certain, $f(0) = 1$ and,

$$f(x_{0g}+1) - f(x_{0g}) = \left(\frac{q_g}{p_g}\right)^{x_{0g}} (f(1) - 1)) \tag{2.24}$$

provided $0 \leq x_{0g} < N_g$. Now consider the sequence from some starting point t to 0:

$$f(t) - f(0) = (f(t) - f(t-1)) + (f(t-1) - f(t-2)) + \cdots + (f(1) - f(0)) \tag{2.25}$$

Substituting Equation 2.24:

$$f(t) - f(0) =$$
$$\left[\left(\frac{p_g}{q_g}\right)^{t-1} (f(1) - 1) + \left(\frac{p_g}{q_g}\right)^{t-2} (f(1) - 1) + \cdots + (f(1) - 1)\right] (f(1) - 1)$$
$$f(t) - f(0) =$$
$$\left[\left(\frac{p_g}{q_g}\right)^{t-1} + \left(\frac{p_g}{q_g}\right)^{t-2} + \cdots + 1\right] (f(1) - 1) \tag{2.26}$$

Using the formula for the sum of a geometric series (provided $p_g \neq q_q$) and recalling $f(0) = 1$:

$$f(t) - 1 = \frac{1 - (q_g/p_g)^t}{1 - (q_g/p_g)} (f(1) - 1) \quad 0 \leq t < N_g \tag{2.27}$$

To find $f(t) - 1$, substitute $t = N_g$. Since the chance of ruin if starting with N_g is 0,

$$-1 = \frac{1 - (q_g/p_g)^{N_g}}{1 - (q_g/p_g)} (f(1) - 1) \tag{2.28}$$

or

$$f(1) - 1 = -\frac{1 - (q_g/p_g)}{1 - (q_g/p_g)^{N_g}} \tag{2.29}$$

Substituting Equation 2.29 into Equation 2.27:

$$f(t) - 1 = -\frac{1 - (q_g/p_g)^t}{1 - (q_g/p_g)^{N_g}} \tag{2.30}$$

Replacing t with x_{0g},

$$f(x_{0g}) = 1 - \frac{1 - (q_g/p_g)^{x_{0g}}}{1 - (q_g/p_g)^{N_g}} \tag{2.31}$$

This is the probability of reaching ruin starting with x_{0g}. To find the probability of winning all the units starting with x_{0g} units, find $P_{N_g} = 1 - f(x_{0g})$:

$$\begin{aligned} P_{N_g} &= 1 - \left(1 - \frac{1 - (q_g/p_g)^{x_{0g}}}{1 - (q_g/p_g)^{N_g}}\right) \\ P_{N_g} &= \frac{1 - (q_g/p_g)^{x_{0g}}}{1 - (q_g/p_g)^{N_g}} \end{aligned} \tag{2.32}$$

To connect the gambler's predicament with that of a GA, Miller [11] considers a GA gambling for all the building block representatives (the strings in the population). There is a total supply of $N_g = N$ strings, and the estimated initial supply of building blocks corresponds to gambler's starting capital $x_{0g} = x_0$. Substituting Equation 2.1 for x_{0g} in Equation 2.32 and p_i and $q_i = 1 - p_i$ for p_g and q_g, the probability that the GA will converge to all the correct building blocks is given by:

$$P_N = \frac{1 - ((1 - p_i)/p_i)^{N/2^k}}{1 - ((1 - p_i)/p_i)^N} \tag{2.33}$$

Since the denominator approaches zero faster than the numerator with increasing N, P_N can be approximated by:

$$P_N \approx 1 - \left(\frac{1 - p_i}{p_i}\right)^{\frac{N}{2^k}} \tag{2.34}$$

where p_i is the probability that the optimal, order-k building block is kept by tournament selection in partition i [11]. This probability is calculated as in Equation 2.8. Since total population variance is $\sigma_F^2 = \sum_j \sigma_j^2$ and the variance due to other partitions in the fitness function is the only source of noise being considered in the decision making of partition i, the variance in any schema belonging to partition i can be estimated as $\sigma_F^2 - \sigma_i^2$. Substituting $\sigma_F^2 - \sigma_i^2$ for σ_M^2 in Equation

2.8 yields:

$$p_i = \Phi\left(\frac{d}{\sqrt{2\left(\sigma_F^2 - \sigma_i^2\right)}}\right) \tag{2.35}$$

Defining the failure rate $\Psi = 1 - P_N$ and solving Equation 2.34 for population size N:

$$N = 2^k \ln(\Psi) / \ln\left(\frac{1-p_i}{p_i}\right) \tag{2.36}$$

To make calculation less cumbersome and to facilitate comparison with Equation 2.15, Equation 2.35 can be approximated using a standard approximation for the cumulative normal distribution [1]:

$$p_i = \frac{1}{2} + \frac{x}{\sqrt{2\pi}} \tag{2.37}$$

where $x = d/\sqrt{2(\sigma_F^2 - \sigma_i^2)}$. Equation 2.36 becomes:

$$N = 2^k \ln(\Psi) / \ln\left(\frac{1 - \frac{x\sqrt{2}}{\sqrt{\pi}}}{1 + \frac{x\sqrt{2}}{\sqrt{\pi}}}\right) \tag{2.38}$$

Since x is a small number, $\ln(1 \pm \frac{x\sqrt{2}}{\sqrt{\pi}})$ can be approximated by $\frac{x\sqrt{2}}{\sqrt{\pi}}$. Substituting the approximation and the value of x into Equation 2.38 yields:

$$N = -\frac{2^{k-1} \ln(\Psi)\sqrt{\pi}}{d}\sqrt{\sigma_F^2} \tag{2.39}$$

Equation 2.39 is an improvement over Equation 2.15 since in most cases it

reduces the population size necessary to achieve predictable convergence. Results from Miller [11] show Equation 2.39 is a more accurate, less conservative predictor of performance for a range of GA problems, including deceptive problems. These results suggest the random walk model is not only a good predictor of final convergence, but also a predictor of decision making in deceptive partitions during a GA run. Chapter 3 discusses the methods used to test this hypothesis.

Chapter 3

Research Methods

The methods used to test the hypothesis presented in Chapter 2 are outlined by first discussing the experimental approach. Next, the details of the research GA implementation are given, including simulation parameters, random number generation, tournament selection algorithm, fitness function, schema visualization, and validation of simulation output. Finally, the methods used to classify and predict simulation results are given.

3.1 Approach

The research presented in this thesis investigates the random walk model as a predictor of the decision-making process in deceptive schema partitions during a GA run. The primary goal of the research is to show that the gambler's ruin random walk model can be used to size populations for predictable partition decision-

making results. The approach is to run GA simulations with random number generator seeds at various population sizes, examining how well the model predicts decision making within the deceptive partitions. The results would support the proposition that the random walk model of schema processing is an accurate predictor of GA dynamics under deceptive conditions in which incorrect decisions are possible.

3.2 GA Implementation

The GA implemented in this research (referred to as *bsv* for Bit Schema Visualization) is based on GAlib, a library of C++ classes available from the Massachusetts Institute of Technology [17]. The simple GA implemented in the library is based on Goldberg's GA [3].

While GAs have many tunable parameters, the research conducted in this thesis restricts the domain of possibilities by setting some GA parameters to constant values. These parameters are summarized in Table 3.1. The variables for each experiment are thus restricted to population size and random number generator seed. The GA stops when the standard deviation in the fitness of the entire population falls below a threshold (0.01) or if the GA runs for over 100 generations.

Table 3.1: GA parameter constants.

Parameter	Value
string size	12
fitness function	3, 4-bit, fully-deceptive subfunctions
probability of crossover	0.9
probability of mutation	0.0
selection operator	binary tournament

3.2.1 Random Number Generator

A random number generator is needed for many operations in a genetic algorithm: probabilistic crossover, mutation and other genetic operators (binary tournament selection) and random population initialization. The random number generator used by GAlib [17] is based on *ran2* in [16]. The period of the generator is approximately 2.3×10^{18}. The return value is a uniform random deviate on $(0.0, 1.0)$. The GAlib implementation divides the *ran2* function into separate functions for initializing the seed, *sran2*, and getting the next random number in the sequence, *ran2*.

3.2.2 Binary Tournament Selection

Since the population sizing models used above assume tournament selection, *bsv* implements a binary tournament selection algorithm. The GAlib [17] class library includes a tournament "selector," but selection is based on proportionate fitness (roulette wheel) rather than randomly-selected individuals assumed in tournament

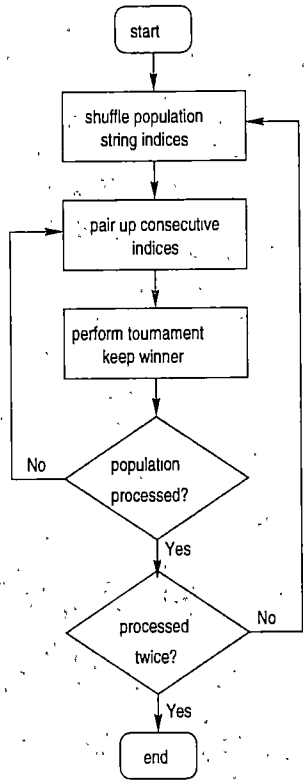


Figure 3.1: Binary tournament algorithm.

selection of other researchers [4], [15]. To include a random-competitor tournament selector, the GAlib classes were extended to include an algorithm as outlined in Figure 3.1. Binary tournament selection ensures each member of the population participates in at least two tournaments. In the special case of an odd-sized population, a randomly-generated index is used to provide a competitor for the last tournament.

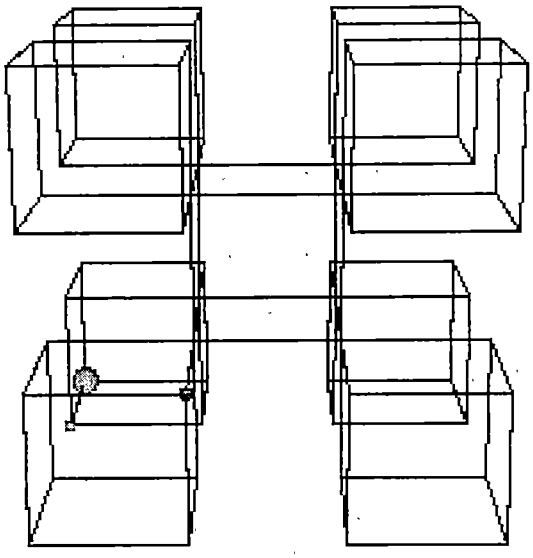


Figure 3.2: A frame from a bit schema visualization animation.

3.2.3 Schema Visualization

bsv includes a schema visualization component for observing the decision-making process in a partition during the GA run. The schemata in the partition are represented by the vertices of a six-dimensional hypercube as shown in Figure 3.2. The number of vertices used depends on the order of the partition to be visualized in a given run. An $o(s)$-order partition has $2^{o(s)}$ schemata. On a six-dimensional hypercube, up to $2^6 = 64$ schemata can be tracked; therefore partitions of up to order $o(s) = 6$ can be visualized. A sphere of time-varying radius at each vertex

of the hypercube represents the number of copies (samples) of the schema present in the population at each generation. The choice from bit-position to vertex is recursive starting from the big cube, each vertex of which is a little cube. Defined bits in the partition are chosen from left to right in the bit string. The left most bit (assumed "0" if $o(s) < 6$) chooses to the upper or lower levels of the big cube. The next bit to the right chooses to the left or right in the big cube. The next bit to the right chooses back to front in the big cube. The first three bits thus identify a little cube occupying one vertex of the big cube. The next three bits chose the vertex within the little cube. These bits are mapped like the first three in that they specify up/down, left/right, and back/front within the little cube. "0" bits map down, left and back. "1" bits map up, right, and front. For example, "000000" chooses to the bottom, left, back vertex; "111111" chooses to the top, right, front vertex.

Each vertex thus maps to a 6-bit string which specifies the defined positions of a given schema in the 6-bit partition being visualized. The radius of the sphere at that vertex represents the number of copies of the schema present in the population. Viewed over time, the dynamics of the schema decision making and reallocation process can be seen as expanding and contracting spheres. To visualize partitions containing schemata with less than six defined positions, the leftmost bits are set to zero. Figure 3.2 is an example frame from an animation of a GA run on an $o(s) = 4$ partition. Since $o(s) = 4$, the leading two bits are set

Table 3.2: Test suite from Goldberg et al. [5].

Function	Description	String Size ℓ	Figure
F1	uniformly scaled, linear	50	3.3
F2	non-uniformly scaled, linear	50	3.4
F3	uniformly scaled, linear with added noise	50	3.5
F4	uniformly scaled, nonlinear	40	3.6
F5	non-uniformly scaled, nonlinear	40	3.7

to zero, so only the bottom left position of the hypercube is used.

3.2.4 GA Validation

The test suite from Goldberg et al [5] is used to validate the performance of *bsv*. Table 3.2 summarizes each function. Figures 3.3 to 3.7 show *bsv* results for each function. The graphs show that *bsv* performs as well or better than the Goldberg predictions and matches closely the Miller predictions, indicating GA processing in *bsv* is comparable to previous GA studies. The Goldberg [5] prediction is the decision probability p in Equation 2.8. The Goldberg experimental results are estimated from graphs in [5]. The Miller prediction is derived from Equation 2.39. The Goldberg and the Miller predictions are shown in Figure 3.3 and Figure 3.6 since they are representative of the results reported routinely in GA research (the "OneMax" and bounded, deceptive problems).

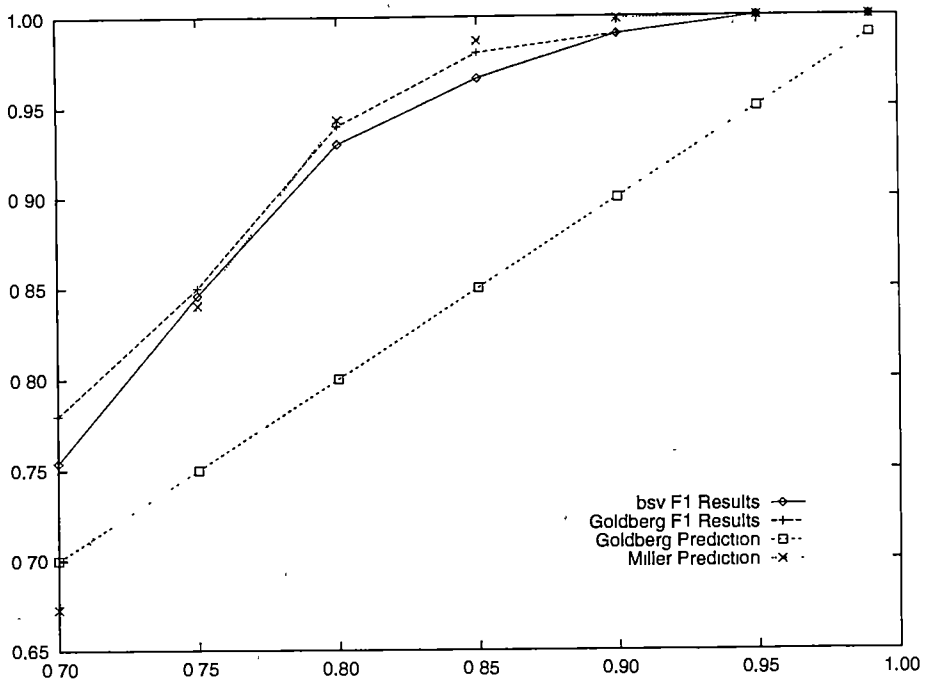


Figure 3.3: Validation of *bsv* using F1.

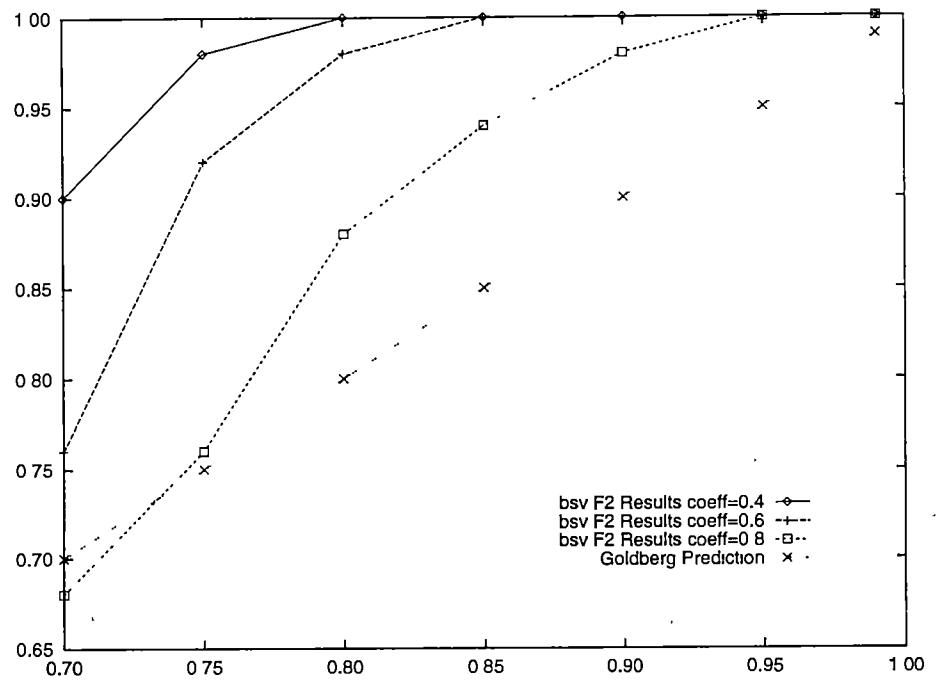


Figure 3.4: Validation of *bsv* using F2.

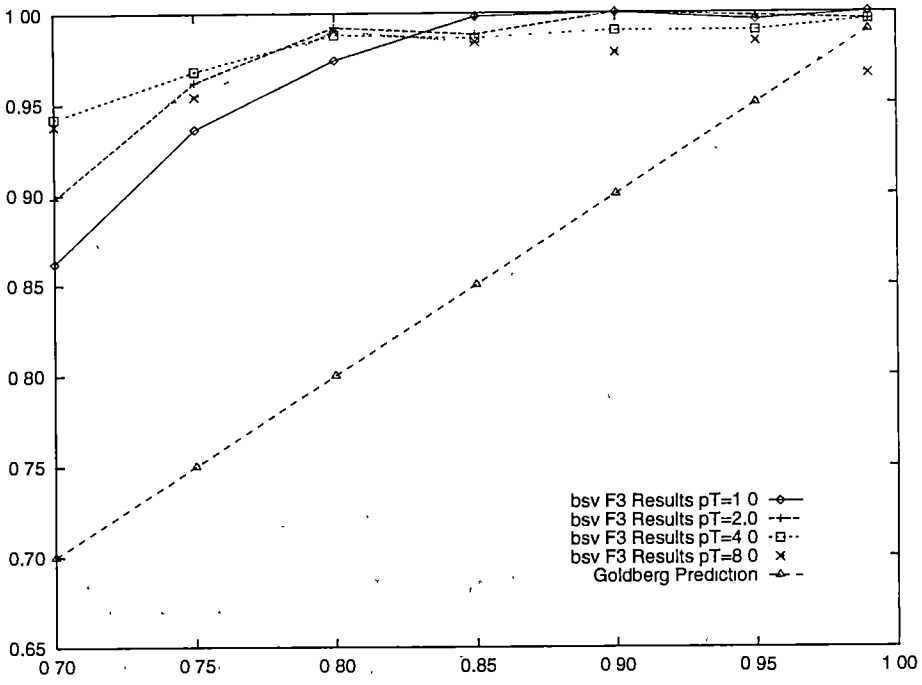


Figure 3.5: Validation of *bsv* using F3.

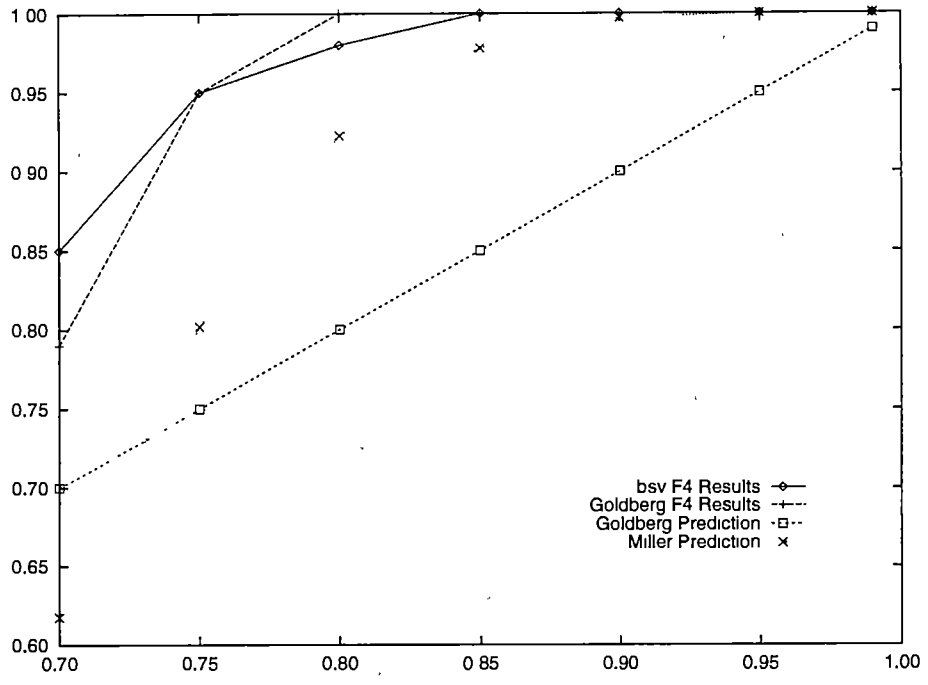


Figure 3.6: Validation of *bsv* using F4.

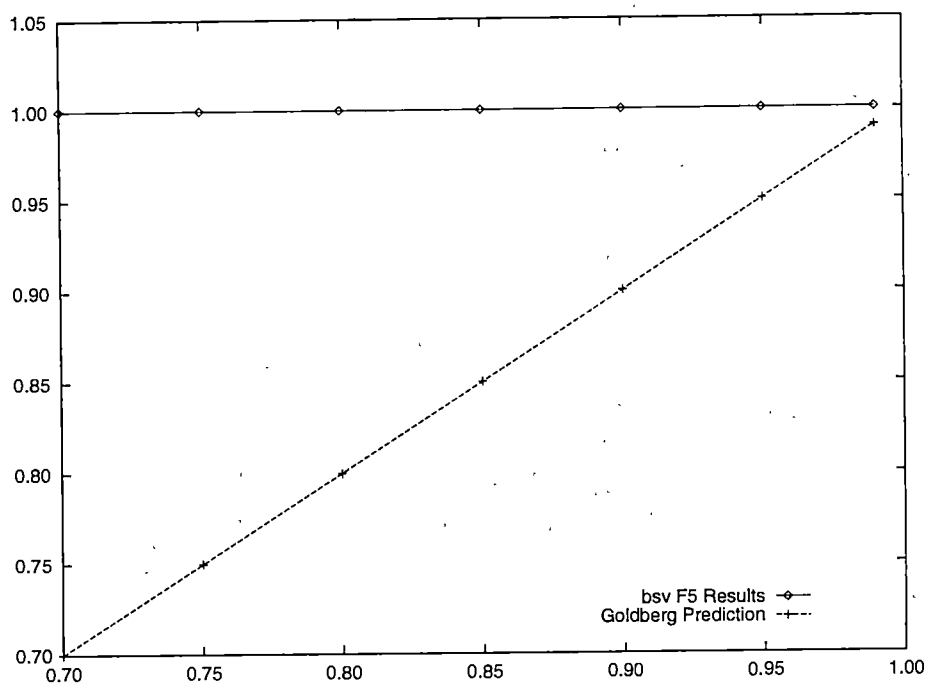


Figure 3.7: Validation of *bsv* using F5.

3.3 Experimental Method

The experimental method used to test the random walk as a model of building block processing begins by identifying the fitness function used in all simulations. Procedures used to evaluate the model are presented next. Evaluation is achieved by classifying schema processing during each run and comparing those classifications to predicted values. Finally, schema visualization is discussed, which provides an online, visual representation of schema allocation during the run.

3.3.1 Fitness Function

The fitness function used in the remainder of this research is a 12-bit version of the Goldberg F4 function in [5]:

$$f(\vec{x}) = \sum_{i=1}^{m} f_{45}(x_{I_i}) \tag{3.1}$$

$$I_1 = \{1, 2, 3, 4\} \qquad I_{i+1} = I_i + 4$$

where m is the number of subfunctions composing the entire problem. The 12-bit version of F4 has $o(s) = 4$ deceptive subfunctions f_{45}, therefore $m = 3$. The f_{45} function is shown in Figure 3.8. I is a set of indices into the string $\vec{x}$. Total fitness is the sum of each contiguous, 4-bit subfunction value.

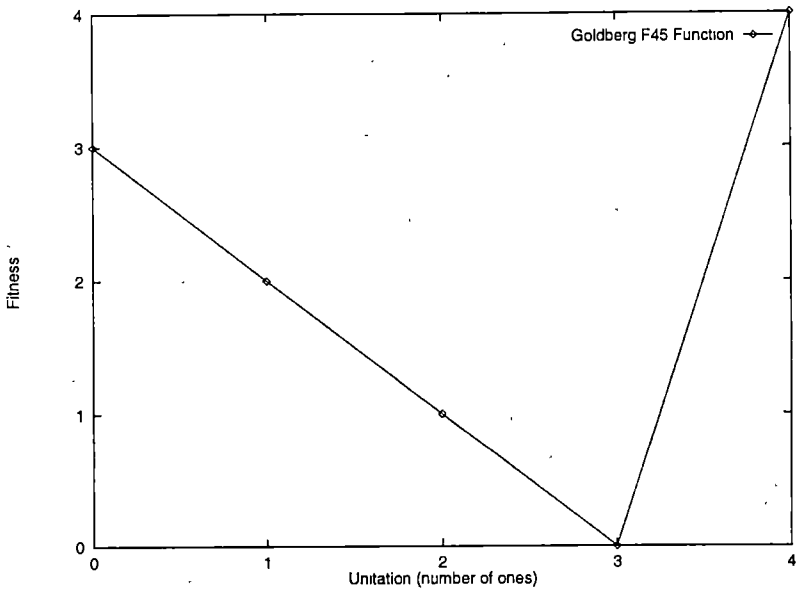


Figure 3.8: f_{45} subfunction, 4-bit, fully deceptive.

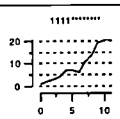
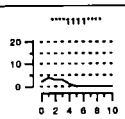
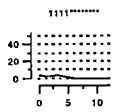
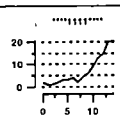
3.3.2 Random Walk Model Evaluation Procedures

The random walk model is evaluated by first classifying a number of GA runs of the 12-bit F4 function at different population sizes and random number seeds. The population sizes were chosen to reflect expected convergence quality, that is, predicted convergence at the end of the run. Confidence values of 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, and 0.99 were used, corresponding to population sizes 21, 35, 55, 84, 127, 212, and 422, which are computed using Equation 2.15. This is Goldberg's conservative estimate on the size of the population needed to achieve predictable convergence at the end of the run. If the random walk model is indeed a better predictor of schema processing, one would not only expect the GA to perform better than Goldberg's predictions, but also provide an explanation of why this is so. The remainder of this section discusses the methods used to find that explanation. The first step is classification.

Schema competitions are classified into five categories, corresponding to five different convergence behaviors. The categories are summarized in Table 3.3. In the special case where the initial decision leads to no increase or decrease in sample size, the competition is classified 0.5 to each (B)/(E) or (C)/(D), depending on the final outcome.

Figure 3.9 shows an example partition graph produced by plotting the GA simulation results of the F4 problem for the contiguous, order-4 partition in which

Table 3.3: Partition graph classification.

Behavior Classification	BB in initial population?	Initial decision correct?	Takeover/Ruin	Example
(A)	N	N	R	
(B)	Y	Y	T	
(C)	Y	Y	R	
(D)	Y	N	R	
(E)	Y	N	T	

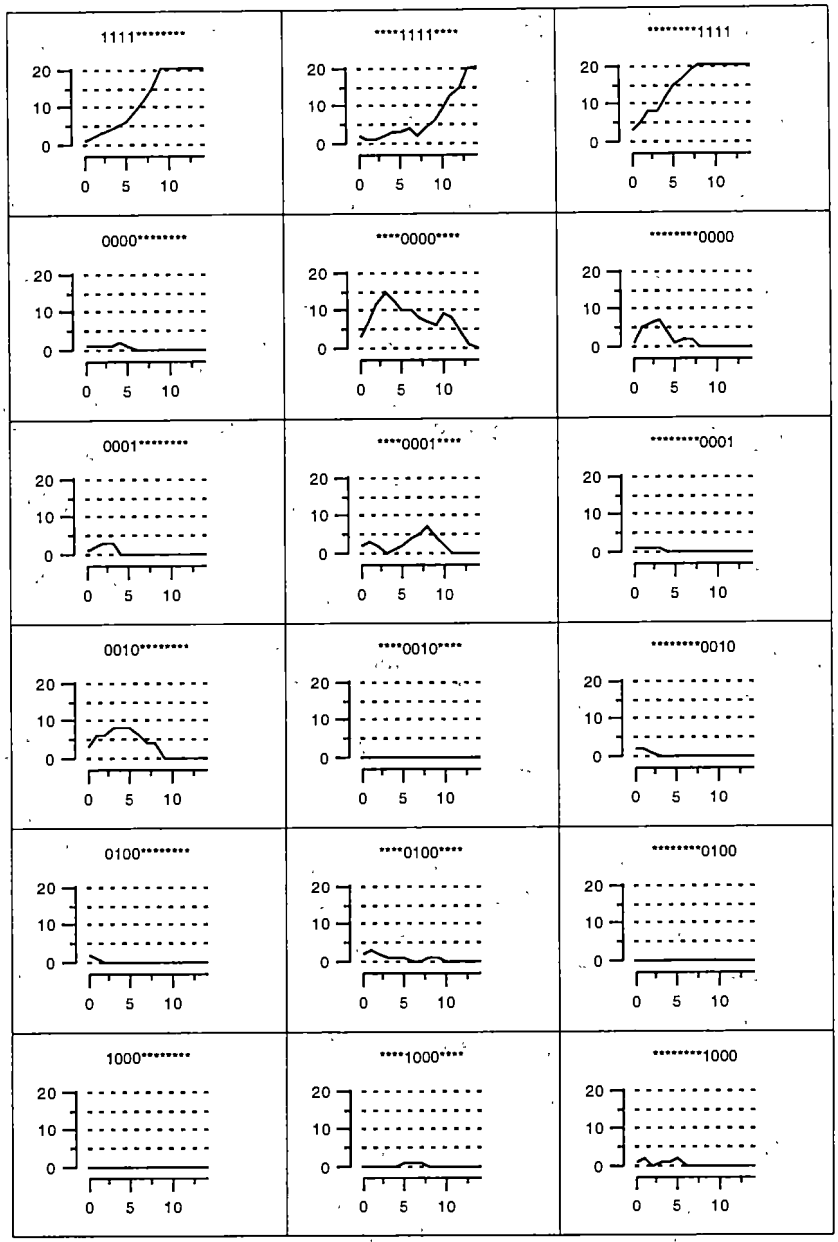


Figure 3.9: Evaluation graph set. Population Size = 21. Seed = 1887561901. Fitness function=Equation 3.1.

the globally-optimum and the locally-optimum, deceptive solutions (strings) compete. All the graphs plot count (number of copies of the schema) versus generation. There are three columns since in the 12-bit problem there are $m = 3$ partitions that make up the solutions. Each column represents a competition within the partition; therefore, each partition graph contains three competitions. Each row represents a particular schema. The top row represents the globally-optimal schema within each partition. The bottom four rows represent deceptive, locally-optimal schemata.

Having categorized the runs, a method for computing the expected numbers of instances of each category is developed. Eight random seeds were used for each population size, so the number of competitions per population size is $w = m \times 8 = 24$. The expected number of Behavior Classification (A) can be computed as follows. Since the probability that a given string contains the globally-optimal schema is $\frac{1}{2^k}$, the probability that the globally-optimal schema is absent from a given string is $1 - \frac{1}{2^k}$. Raising this quantity to the Nth power gives the probability that all the strings in the population lack the globally-optimal schema. Define Q_k to be the probability that every string in the population lacks the optimal schema:

$$Q_k = \left(1 - \left(\frac{1}{2^k}\right)\right)^N \tag{3.2}$$

Using an approximation for P_k by Harik et al. [8], Q_k becomes:

$$Q_k = 1 - P_k = \exp\left(\frac{-N}{2^k}\right) \tag{3.3}$$

so the predicted number of cases of Behavior Classification (A) is $w \times Q_k$.

To find the expected number of cases of (B) through (E), define *IDMM* (initial decision making model) to be the predicted probability of a correct initial decision, the Goldberg prediction of Equation 2.15. Also define *RWM* (random walk model) to be the predicted probability of takeover by the optimal schema, the Miller prediction. The Miller prediction for final convergence is based on the random walk population sizing relation in Equation 2.39. Figure 3.10 shows the predicted convergence probability $(1 - \Psi)$ versus population size for 12- and 40-

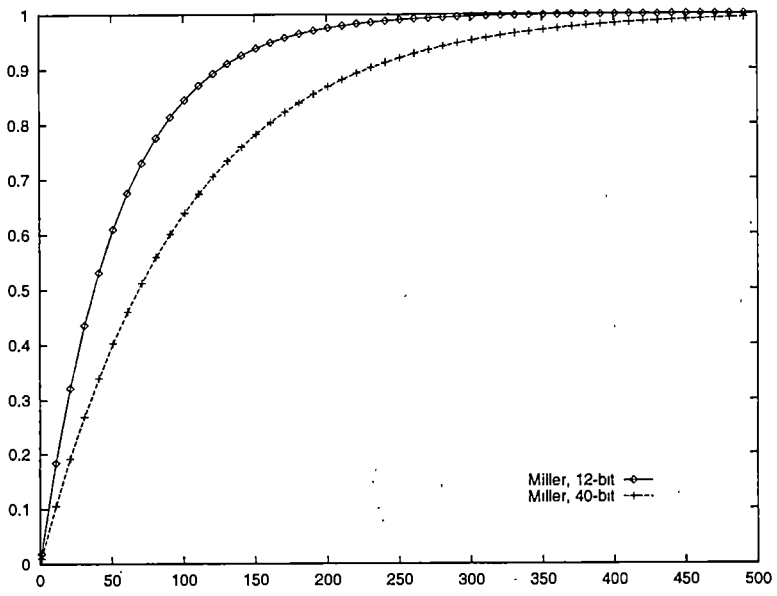


Figure 3.10: Miller prediction of final convergence as a function of population size.

Table 3.4: Classification prediction formulas.

Behavior Classification	Prediction formula
(A)	$w \times Q_k$
(B)	$IDMM \times w \times RWM$
(C)	$IDMM \times (w \times (1 - RWM) - (A))$
(D)	$(1 - IDMM) \times (w \times (1 - RWM) - (A))$
(E)	$(1 - IDMM) \times w \times RWM$

bit F4 function. As the figure shows, the 12-bit version is predicted to converge with higher probability up to a population size of approximately 400, where both problems reach the maximum convergence probability value. However, runs of the 12-bit version at confidence values lower than 0.85, corresponding to population sizes less than 100, have lower predicted convergence than the 40-bit version using the same decision probabilities. This is explained by the larger population sizes needed to achieve those confidence values in the 40-bit problem.

With these quantities and probabilities defined, we can predict the expected number of cases of each type (B)-(E) in Table 3.3. The prediction formulas are shown in Table 3.4. Note that $(A) + (B) + (C) + (D) + (E)$ adds up to w. The (C) and (D) predictions take into account the fact that the ruin cases predicted by Miller include (A), (C), and (D), so

$$w \times (1 - RWM) = (A) + (C) + (D)$$

$$w \times (1 - RWM) - A = (C) + (D)$$

The $(w \times (1 - RWM) - (A))$ cases are split between (C) and (D) as follows: (C) gets the IDMM fraction of them, and (D) gets the (1-IDMM) fraction of them. These predictions are compared to classification results to support the hypothesis that the random walk model better explains schema processing within partitions than previous end-result models.

3.3.3 Schema Allocation Visualization

The schema visualization technique described above is a useful tool for observing schema sample allocation as the GA runs. This is most useful when the GA is having difficulty or running for many generations. Online feedback reveals trends and patterns in processing that are difficult and time-consuming to ascertain using other methods. Frames captured from a schema visualization animation are shown to give an idea of this online, visual advantage.

Chapter 4

Results and Conclusions

Experimental results are presented to support the hypothesis that the random walk model is a not only a good predictor of end-of-run convergence, but also an accurate model of schema partition processing. Visualization of schema processing confirms decision making predicted by the random walk model.

4.1 Experimental Results

Partition graphs illustrate schema decision making during the run. The global and local optima are represented. Figure 3.9 shows a run using population size 21, random seed 1887561901. Appendix A contains the graphs for representative small and large population sizes with eight random seeds each used in this study. These graphs were used to classify the decision-making behavior of the globally-optimum schema in each partition as described in Chapter 3. Using $m = 3$ and

eight random seeds yields an experimental $w = 3 \times 8 = 24$. Q_k, the probability that the globally-optimum building block is absent from the initial population is computed for each population size. With these values for w and Q_k and for IDDM and RWM (explained in Chapter 3), we can predict the expected number of cases of each of the categories in Table 3.3 for each of the population sizes studied. The predicted values are show in Table 4.1. The experimental results are summarized in Table 4.2. The prediction error as the difference between predicted and experimentally obtained cases is shown in Table 4.3.

While the results are encouraging, they are not much better than IDMM. To test the effect of small population sizes, Equation 2.33 was used for the RWM prediction, since an approximation for large N is assumed for Equation 2.36, which is used in the Miller predictions in [11]. New predicted values are in Table 4.4; new prediction error values are in Table 4.5. These results show that the new RWM prediction for the probability of convergence is much more accurate than the previous prediction, suggesting that for population sizes less than 100, the small N equation should be used. Further studies are necessary to determine the generality of this conclusion.

Using the "small N" equation, prediction error is reduced to less than four expected cases in all predictions, with many cases less than one. These results support the hypothesis that the RWM is a predictor of schema processing during the run. A new validation graph for the 12-bit F4 function is shown in Figure 4.1.

Table 4.1: Classification prediction for studied population sizes.

Population Size	(A)	(B)	(C)	(D)	(E)
21	6.5	5.4	6.9	2.9	2.3
35	2.7	8.6	7.4	2.5	2.9
55	0.8	12.2	6.3	1.6	3.1
84	0.1	16.1	4.2	0.7	2.8
127	0.0	19.5	2.1	0.2	2.2
212	0.0	22.3	0.5	0.0	1.2
422	0.0	23.8	0.0	0.0	0.2

Table 4.2: Experimental results for studied population sizes.

Population Size	(A)	(B)	(C)	(D)	(E)
21	7.0	11.0	3.0	1.0	2.0
35	4.0	17.5	1.0	0.0	1.5
55	0.0	18.5	2.0	2.0	1.5
84	0.0	22.5	0.0	0.0	1.5
127	0.0	23.0	0.0	0.0	1.0
212	0.0	23.5	0.0	0.0	0.5
422	0.0	24.0	0.0	0.0	0.0

Table 4.3: Prediction error in number expected cases.

Population Size	(A)	(B)	(C)	(D)	(E)
21	-0.5	-5.6	3.9	1.9	0.3
35	-1.3	-8.9	6.4	2.5	1.4
55	0.8	-6.3	4.3	-0.4	1.6
84	0.1	-6.4	4.2	0.7	1.3
127	0.0	-3.5	2.1	0.2	1.2
212	0.0	-1.2	0.5	0.0	0.7
422	0.0	-0.2	0.0	0.0	0.2

Table 4.4: Classification prediction for studied population sizes. New RWM prediction.

Population Size	(A)	(B)	(C)	(D)	(E)
21	6.5	10.3	2.0	0.8	4.4
35	2.7	14.3	1.7	0.6	4.8
55	0.8	17.6	1.0	0.2	4.4
84	0.1	19.9	0.3	0.1	3.5
127	0.0	21.5	0.1	0.0	2.4
212	0.0	22.8	0.0	0.0	1.2
422	0.0	23.8	0.0	0.0	0.2

Table 4.5: Prediction error in number of expected cases. New RWM prediction.

Population Size	(A)	(B)	(C)	(D)	(E)
21	-0.5	-0.7	-1.0	-0.2	2.4
35	-1.3	-3.2	0.7	0.6	3.3
55	0.8	-0.9	-1.0	-1.8	2.9
84	0.1	-2.6	0.3	0.1	2.0
127	0.0	-1.5	0.1	0.0	1.4
212	0.0	-0.7	0.0	0.0	0.7
422	0.0	-0.2	0.0	0.0	0.2

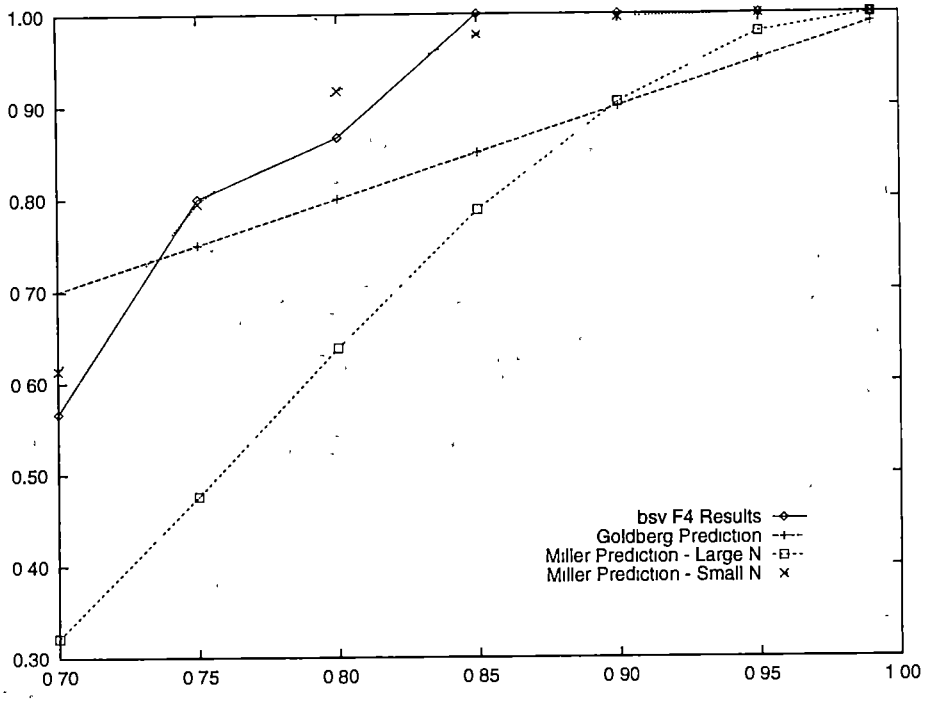


Figure 4.1: Miller prediction using small N, Equation 2.33.

4.1.1 Schema Visualization

Figures 4.2 to 4.16 illustrate the frames of a GA run using hypercube visualization. These frames represent competition among schemata in the 12-bit partition "****dddd****", as illustrated in the center column of the graphs in Table 3.9. The two largest spheres represent schemata "****0000****" (back, left, bottom vertex of the hypercube) and "****1111****" (front, right, top vertex of the hypercube). The visualization shows the GA initially making an incorrect decision about the optimal building block ("****1111****"), decreasing the number of

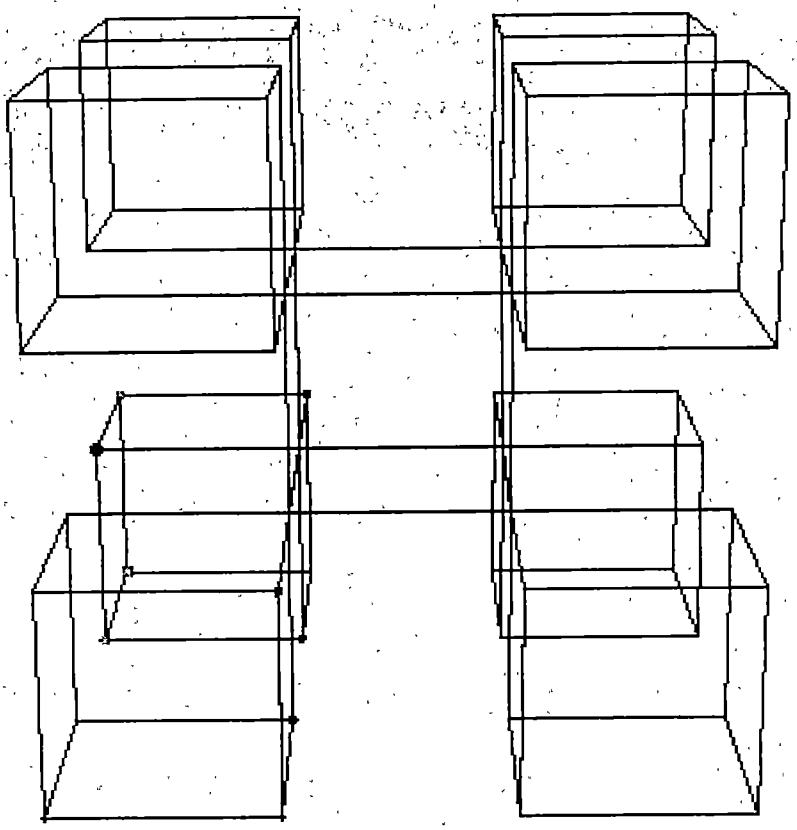


Figure 4.2: Schema visualization of generation 0.

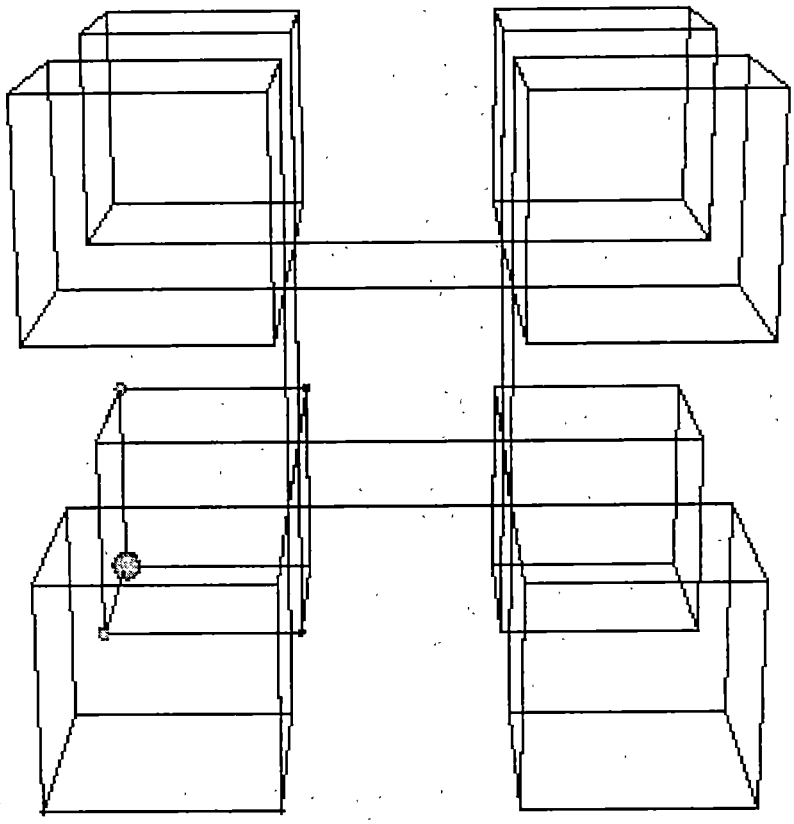


Figure 4.3: Schema visualization of generation 1.

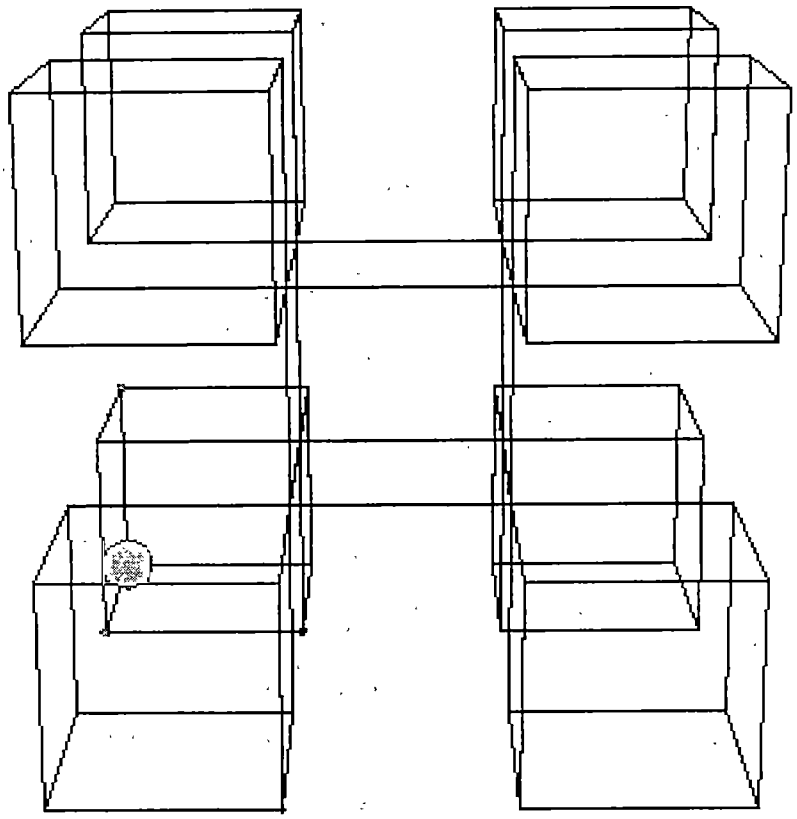


Figure 4.4: Schema visualization of generation 2.

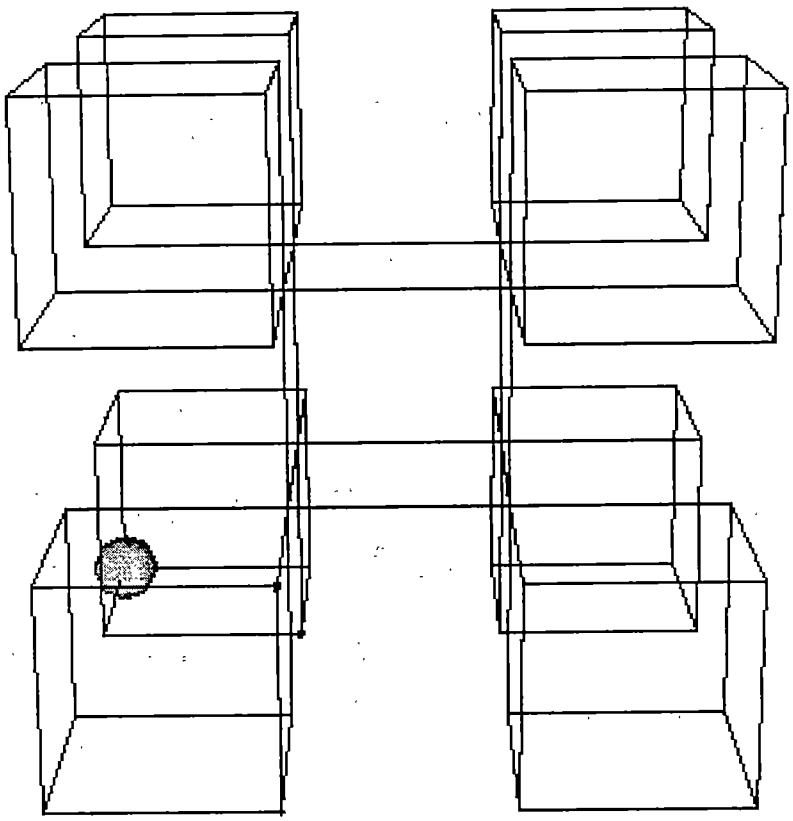


Figure 4.5: Schema visualization of generation 3.

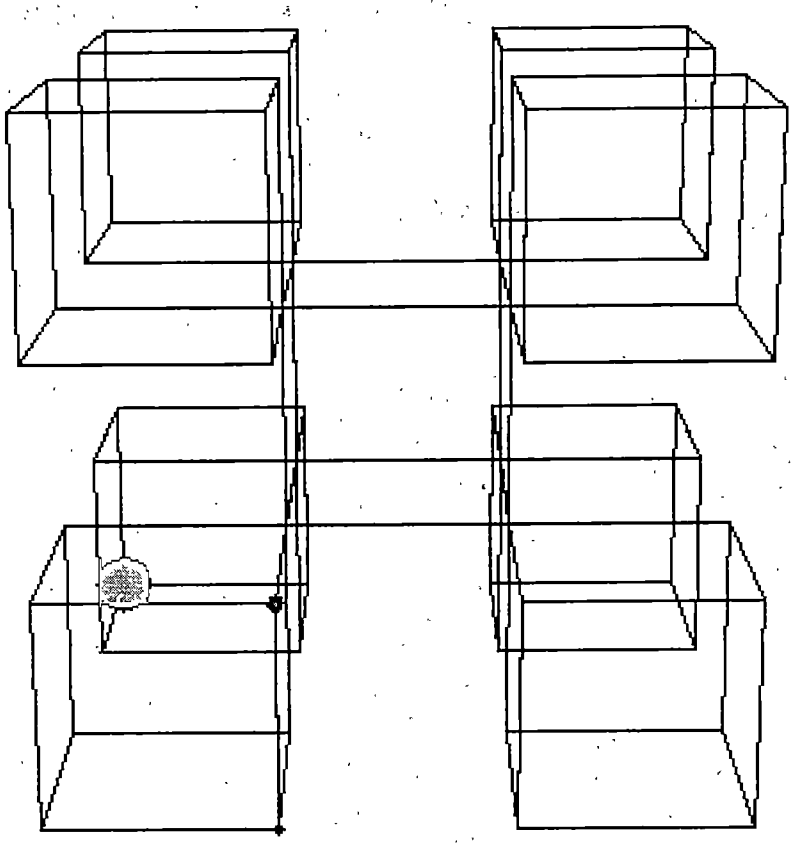


Figure 4.6: Schema visualization of generation 4.

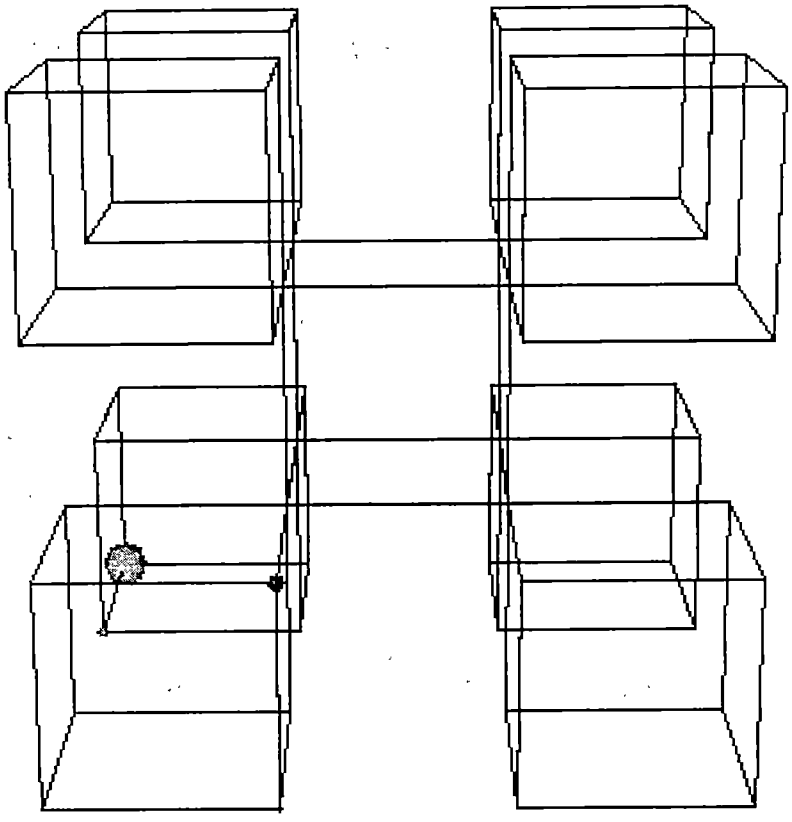


Figure 4.7: Schema visualization of generation 5.

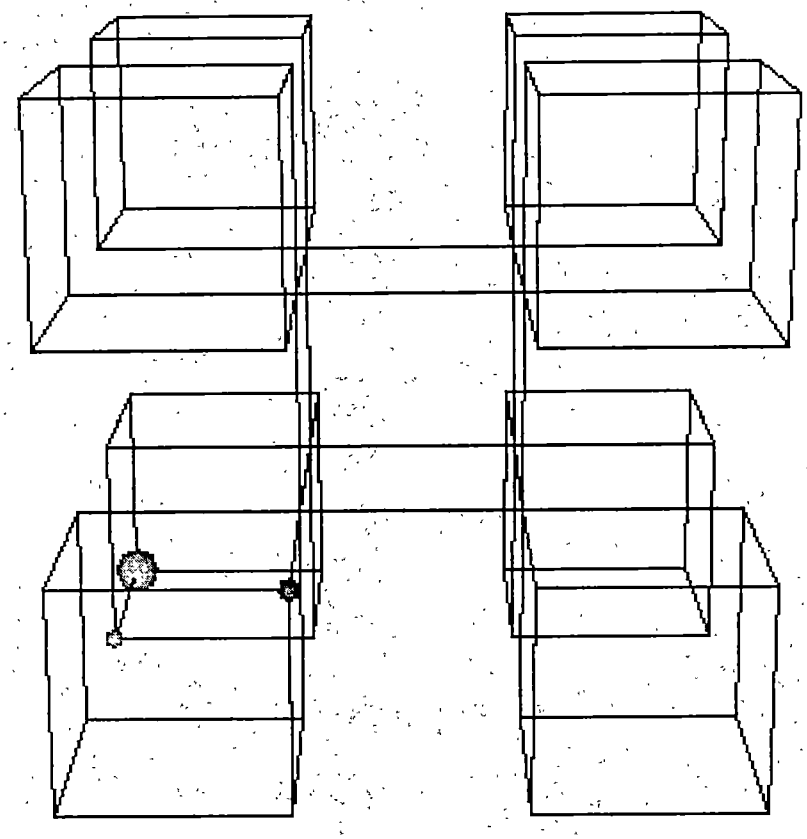


Figure 4.8: Schema visualization of generation 6.

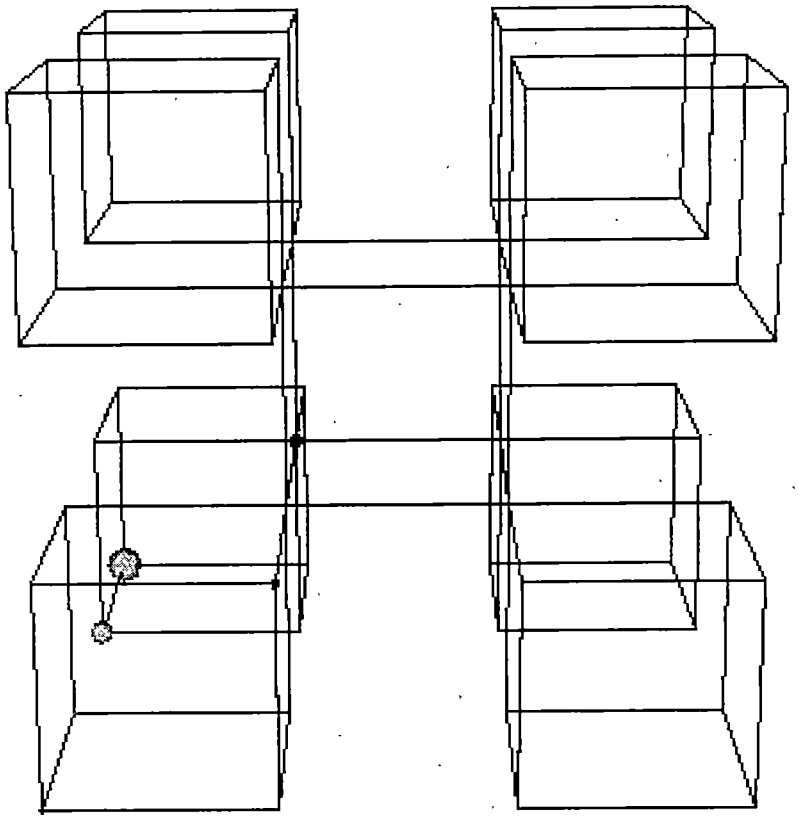


Figure 4.9: Schema visualization of generation 7.

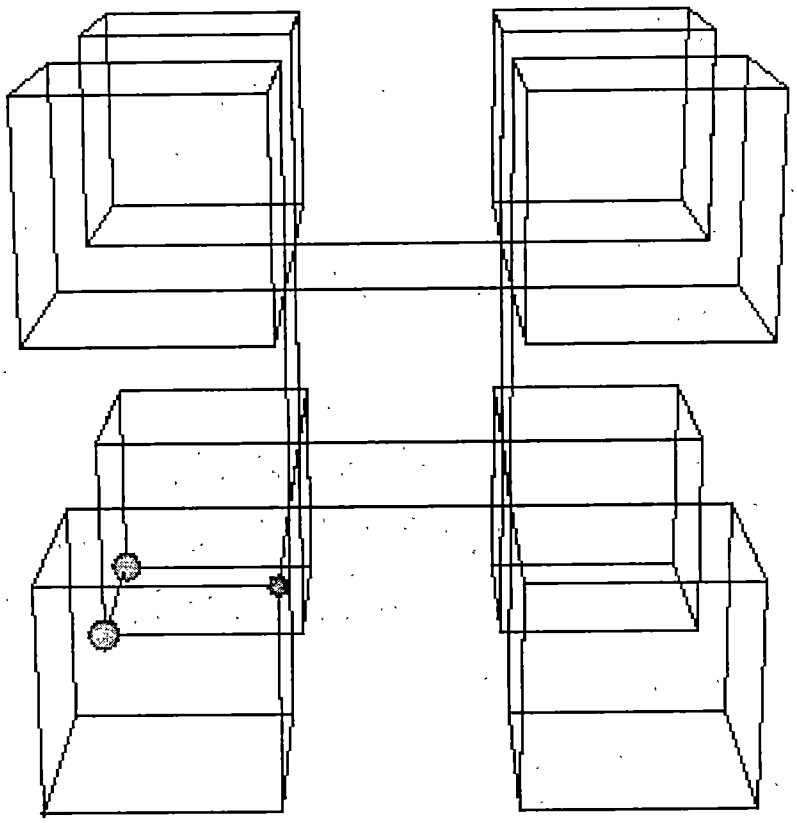


Figure 4.10: Schema visualization of generation 8.

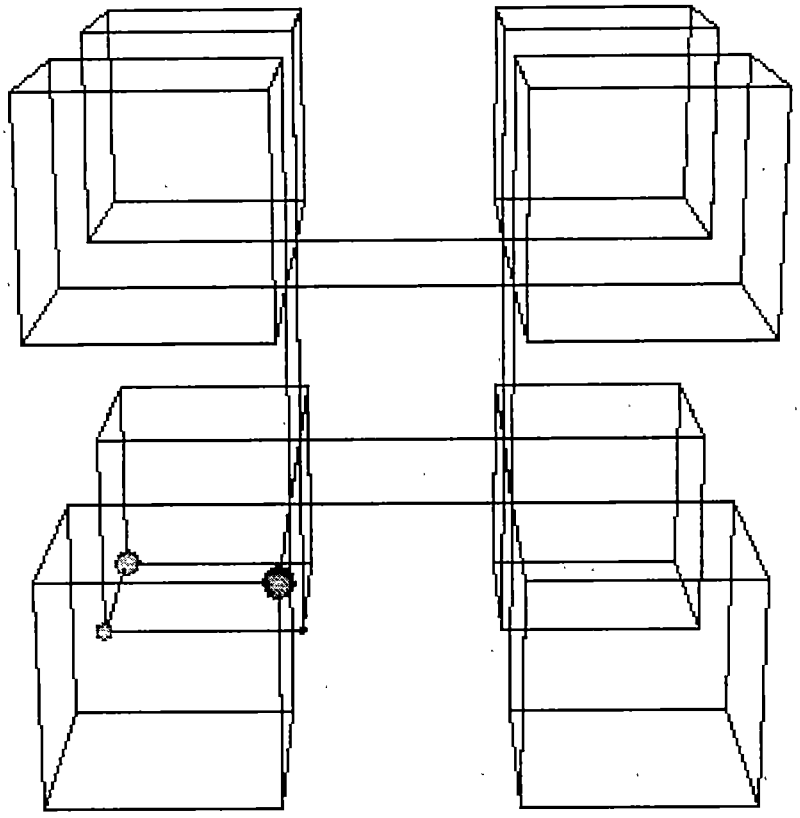


Figure 4.11: Schema visualization of generation 9.

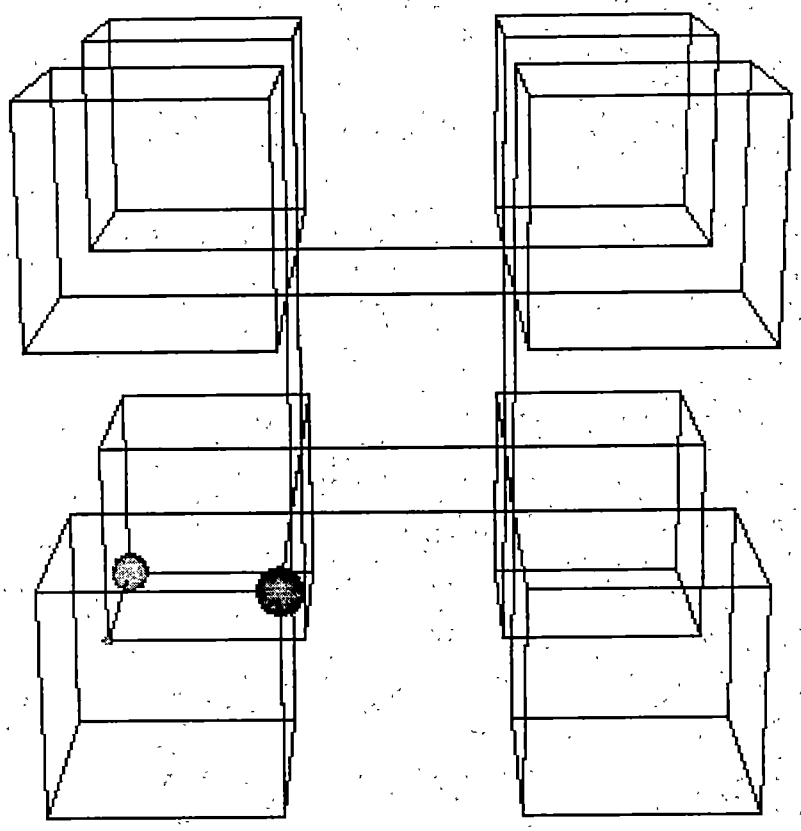


Figure 4.12: Schema visualization of generation 10.

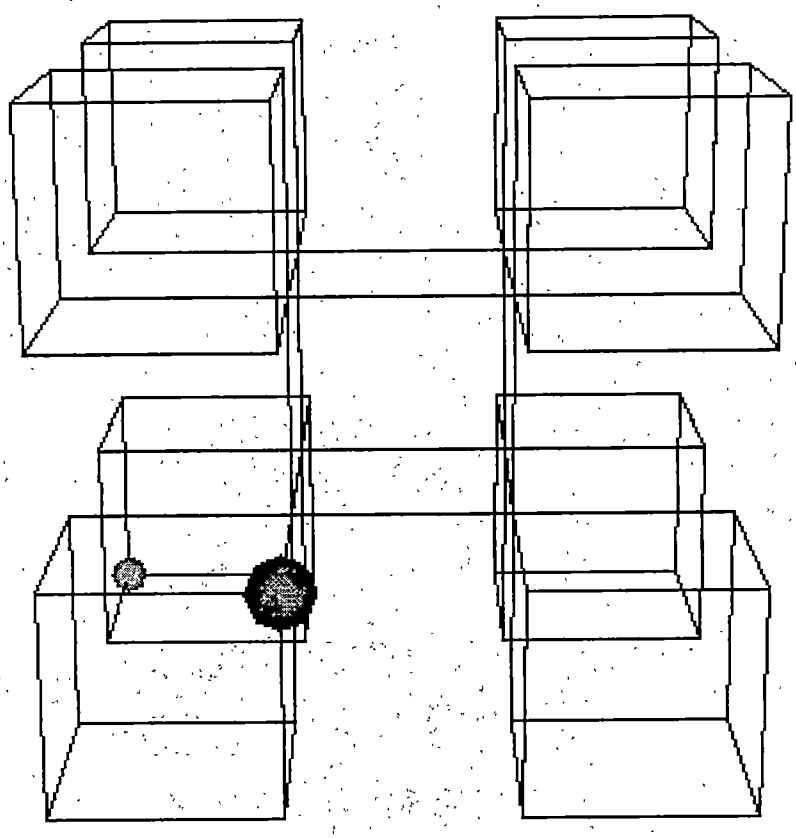


Figure 4.13: Schema visualization of generation 11.

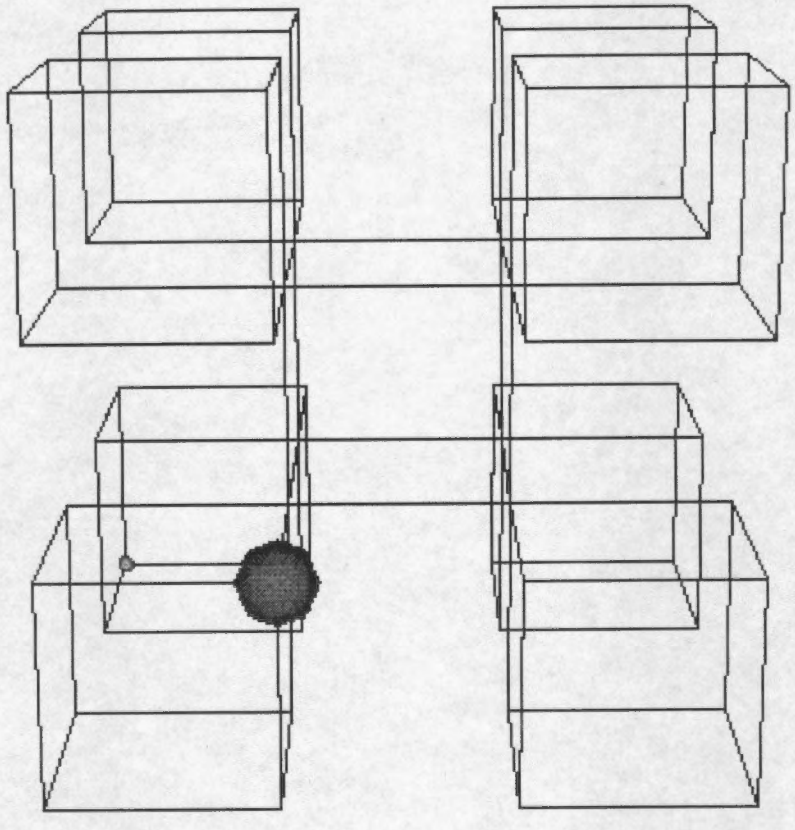


Figure 4.14: Schema visualization of generation 12.

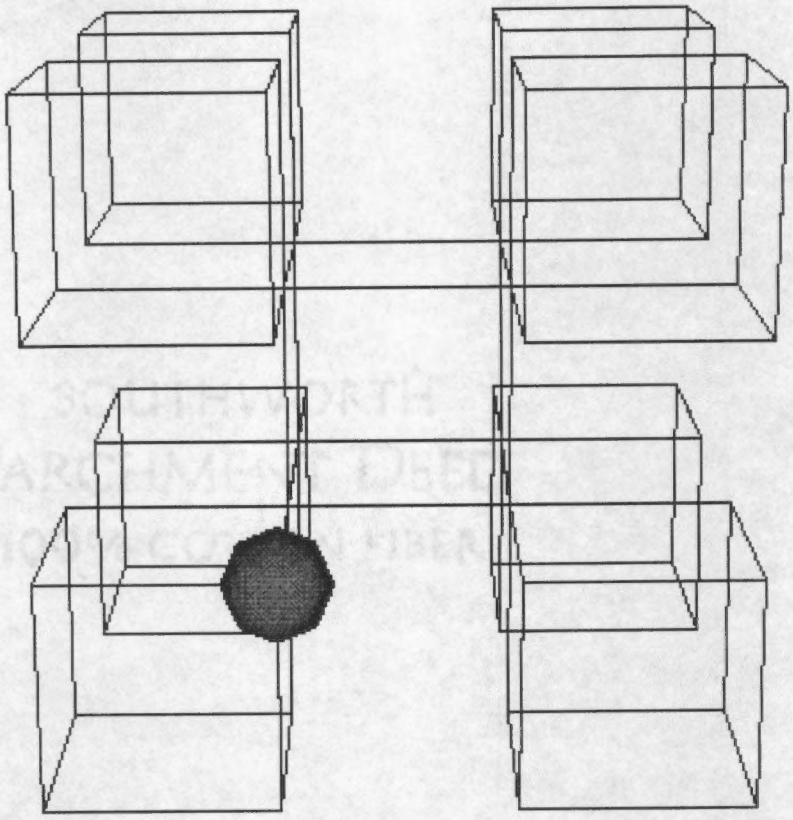


Figure 4.15: Schema visualization of generation 13.

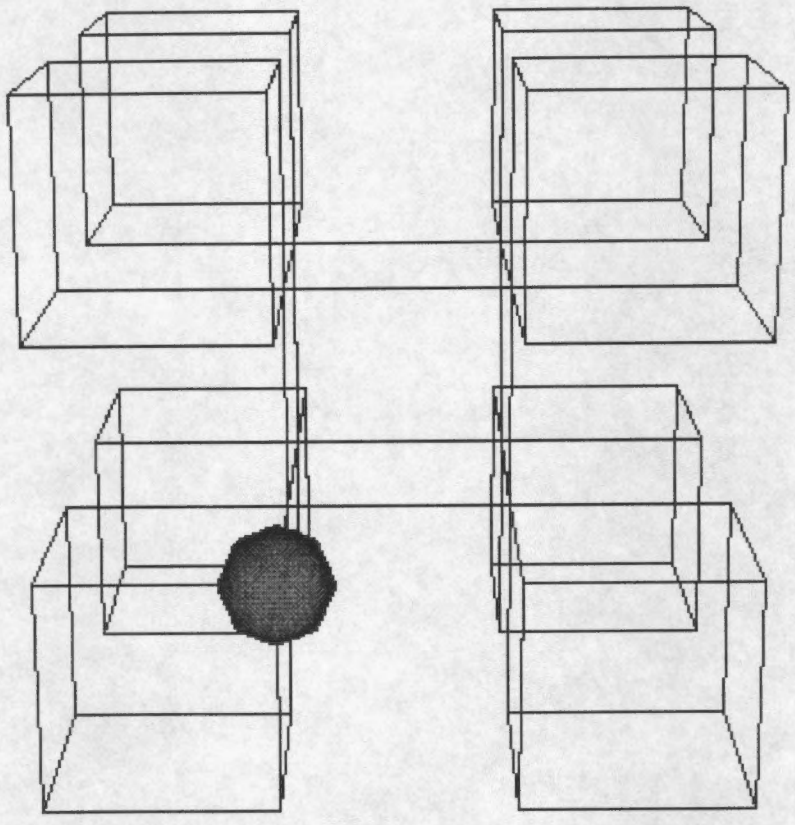


Figure 4.16: Schema visualization of generation 14.

representatives of that schema in subsequent generations (but not going to zero). In later stages, the GA decides correctly and eventually converges to the globally-optimum schema. This is consistent with the random walk model, which says the initial decision does not have as strong an influence on subsequent generations as the more conservative models assume.

4.2 Conclusions

Schema processing is the basis for many theoretical models of GA behavior, including population sizing models. These models are good predictors of final convergence, thus validating the theoretical perspective of schema thinking. However, these studies have not shown empirical evidence of this processing, assuming that the schema theorem is working in the generations following the initial one. The random walk model has proven to be useful in predicting final convergence, enough so to inspire the hypothesis that it is also a good predictor of schema processing within partitions during the run.

The random walk models helps explain the phenomena of recovering from incorrect decisions in competitions between schemata during the initial stage of the run. The initial decision making process is not an "all-or-nothing" proposition. The random walk considers each move toward convergence separately during the run, improving upon the conservative nature of previous models. Results from

this research support the use of random walk population sizing as a model of schema partition decision making during the GA run.

Bibliography

Bibliography

[1] M. Abramovitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover Publications, New York, 1972.

[2] A. Bruce Clarke and Ralph L. Disney. *Probability and Random Processes for Engineers and Scientists.* John Wiley & Sons, Inc., New York, 1970.

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

[4] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. Technical Report 90007, The Clearinghouse for Genetic Algorithms, Department of Engineering Mechanics, The University of Alabama, Tuscaloosa, AL 35487–2908, July 1990.

[5] David E. Goldberg, Kalyanmoy Deb, and James H. Clark. Genetic algorithms, noise and the sizing of populations. *Complex Systems*, 6(4):333–362,

August 1992.

[6] David E. Goldberg and Mike Rudnick. Genetic algorithms and the variance of fitness. Technical Report 91001, Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, January 1991.

[7] David E. Goldberg and Mike Rudnick. Genetic algorithms and the variance of fitness. *Complex Systems*, 5(3):265–278, June 1991.

[8] Georges Harik, Erick Cantu-Paz, David E. Goldberg, and Brad L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. Technical Report 96004, Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, July 1996.

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

[10] Hillol Kargupta and David E. Goldberg. Decision making in genetic algorithms: A signal-to-noise perspective. Technical Report 94004, Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, August 1994.

[11] Brad L. Miller. *Noise, Sampling, and Efficient Genetic Algorithms.* PhD thesis, The University of Illinois at Urbana-Champaign, 1997.

[12] Brad L. Miller. Noise, sampling, and efficient genetic algorithms. Technical Report 97001, Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, May 1997.

[13] Brad L. Miller and David E. Goldberg. Optimal sampling for genetic algorithms. Technical Report 96005, Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, August 1996.

[14] Brad L. Miller and David E. Goldberg. Optimal sampling for genetic algorithms. In *Proceedings of the Artificial Neural Networks in Engineering (ANNIE) Conference*, pages 291–297, New York, 1996. ASME Press.

[15] Melanie Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, Massachusetts, 1996.

[16] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brain P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, Massachusetts, 1992.

[17] Matthew Wall. Galib, a c++ library of genetic algorithm components, 1996. http://lancet.mit.edu/ga/galibdoc.pdf.

Appendix

Appendix A

Partiton Graphs

This appendix contains representative partition graphs, as mentioned in Sections 3.3.2 and 4.1, including small and large population sizes, 21 and 127 respectively. Each population size has runs for eight random number generator seeds.

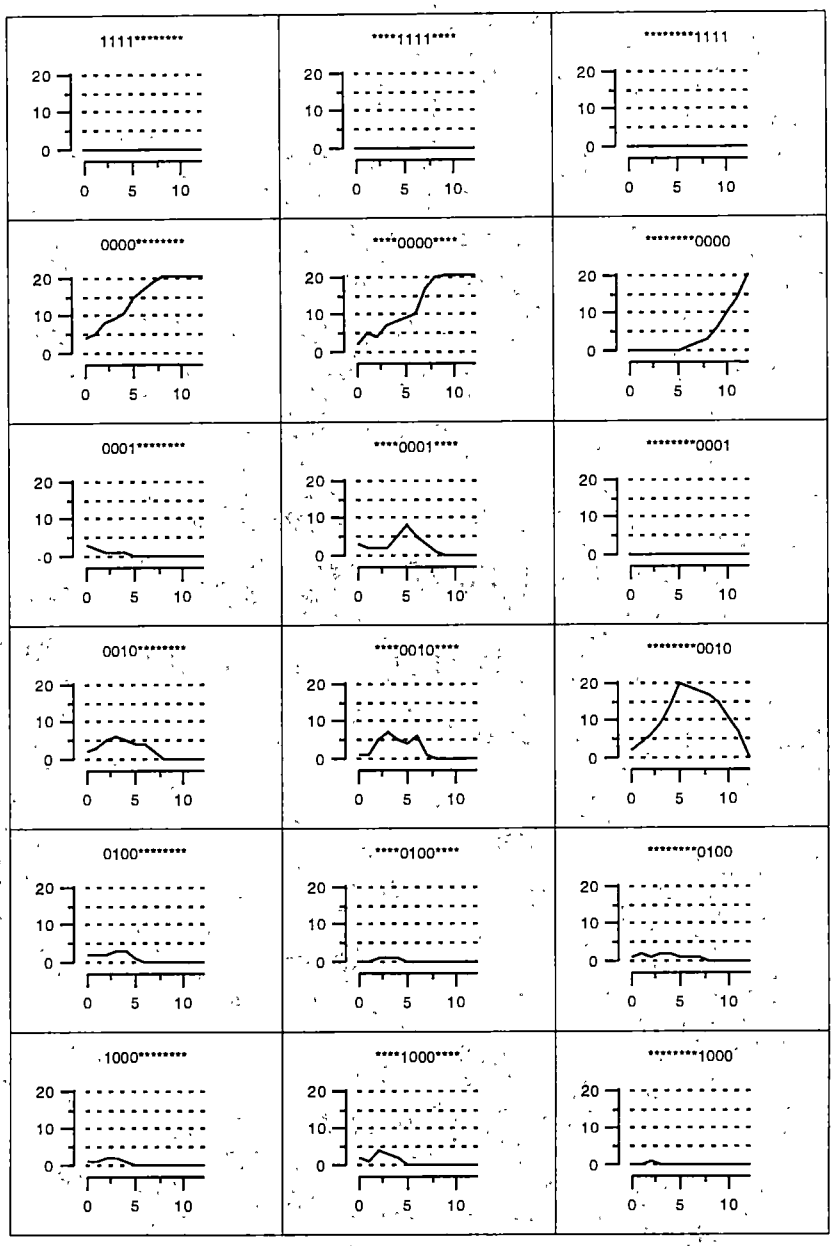


Figure A.1: Population Size=21. Seed=71659920. Fitness function=Equation 3.1.

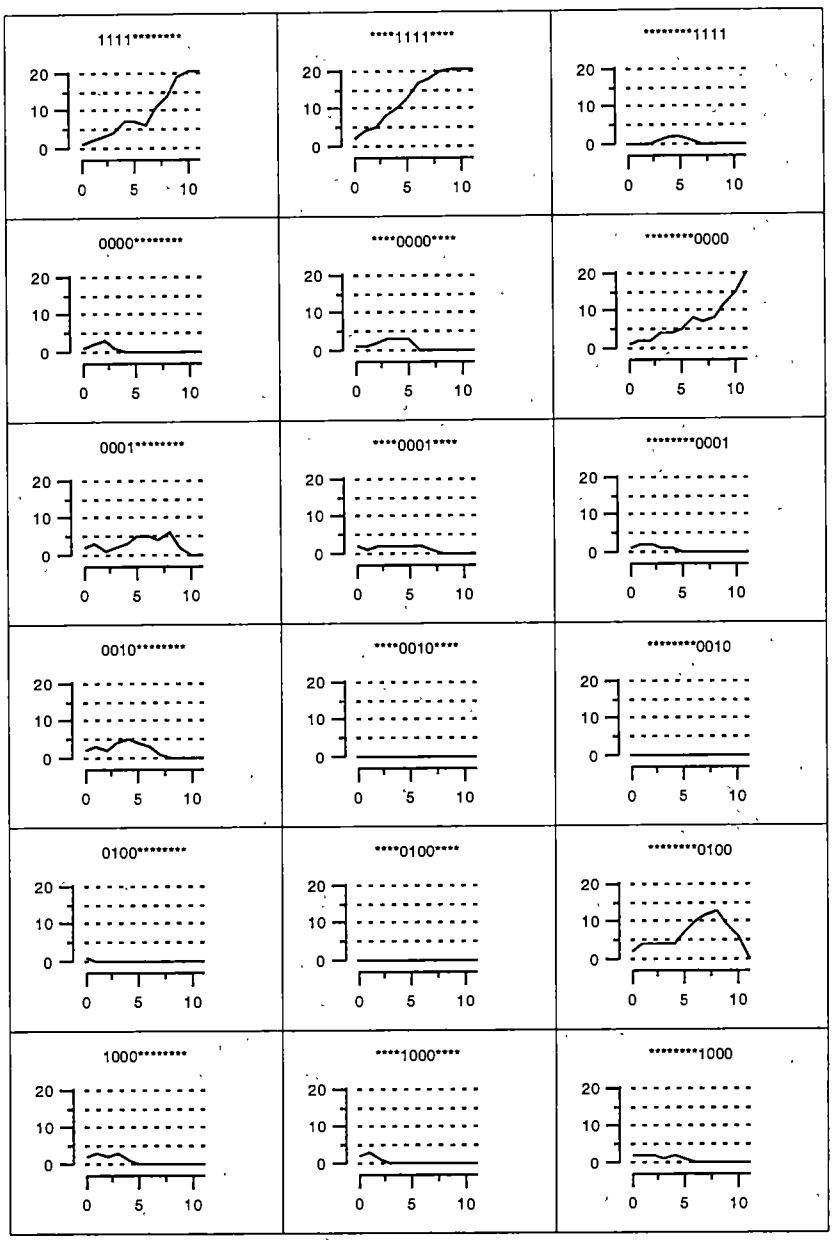


Figure A.2: Population Size=21. Seed=1688918702. Fitness function=Equation 3.1.

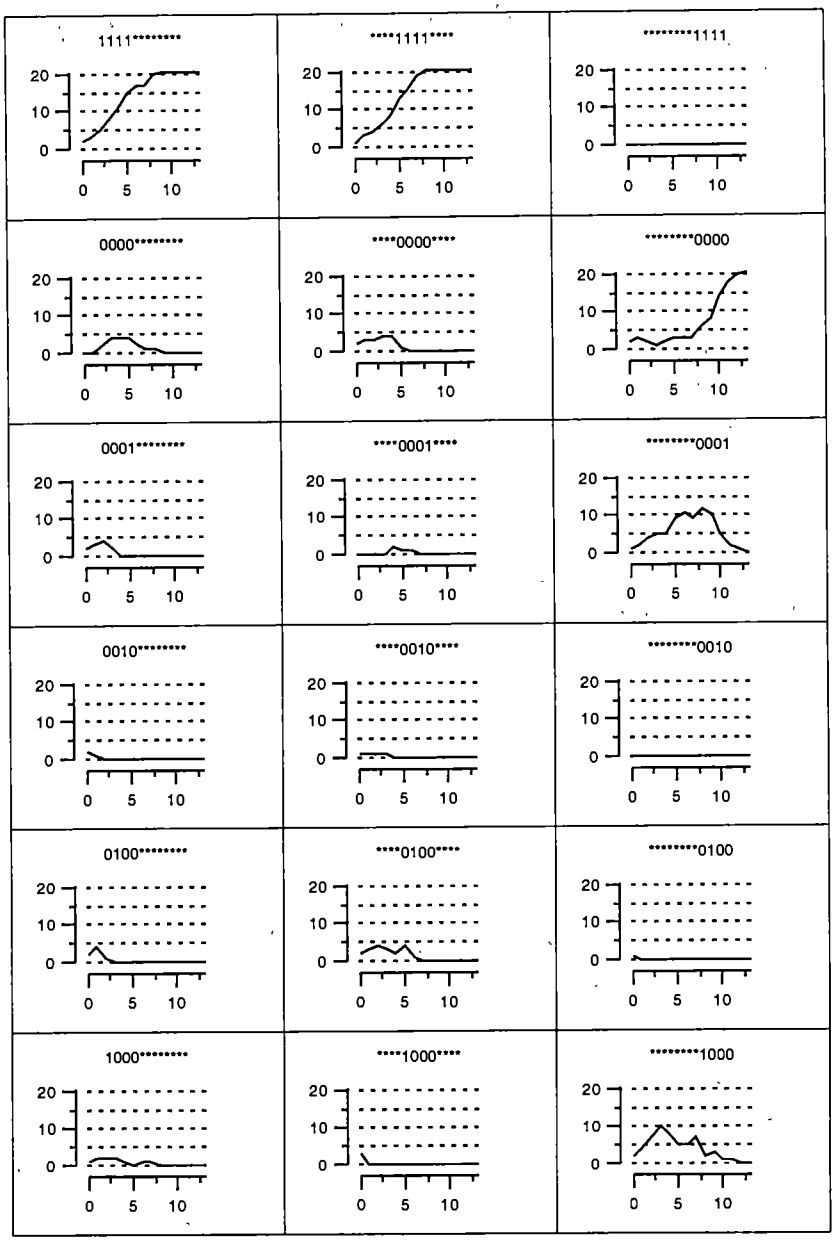


Figure A.3: Population Size=21. Seed=1507297010. Fitness function=Equation 3.1.

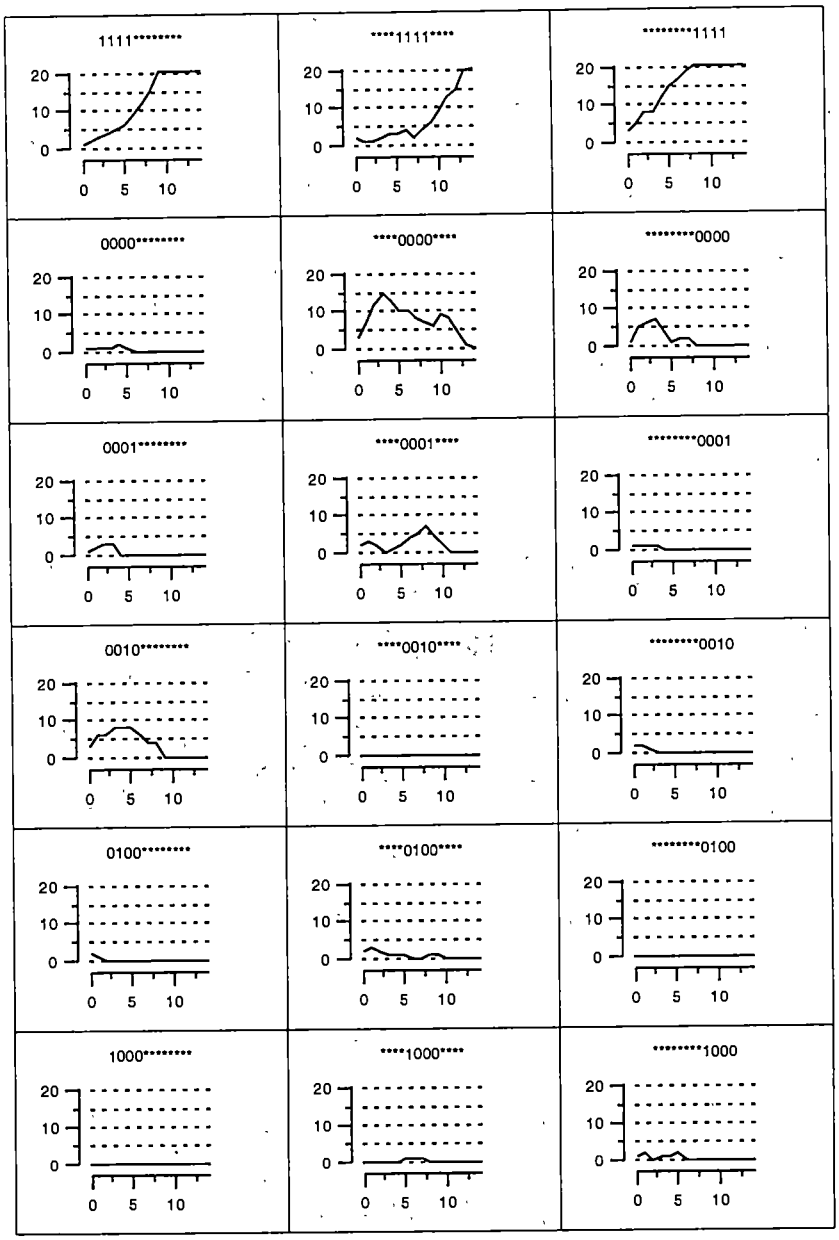


Figure A.4: Population Size=21. Seed=1887561901. Fitness function=Equation 3.1.

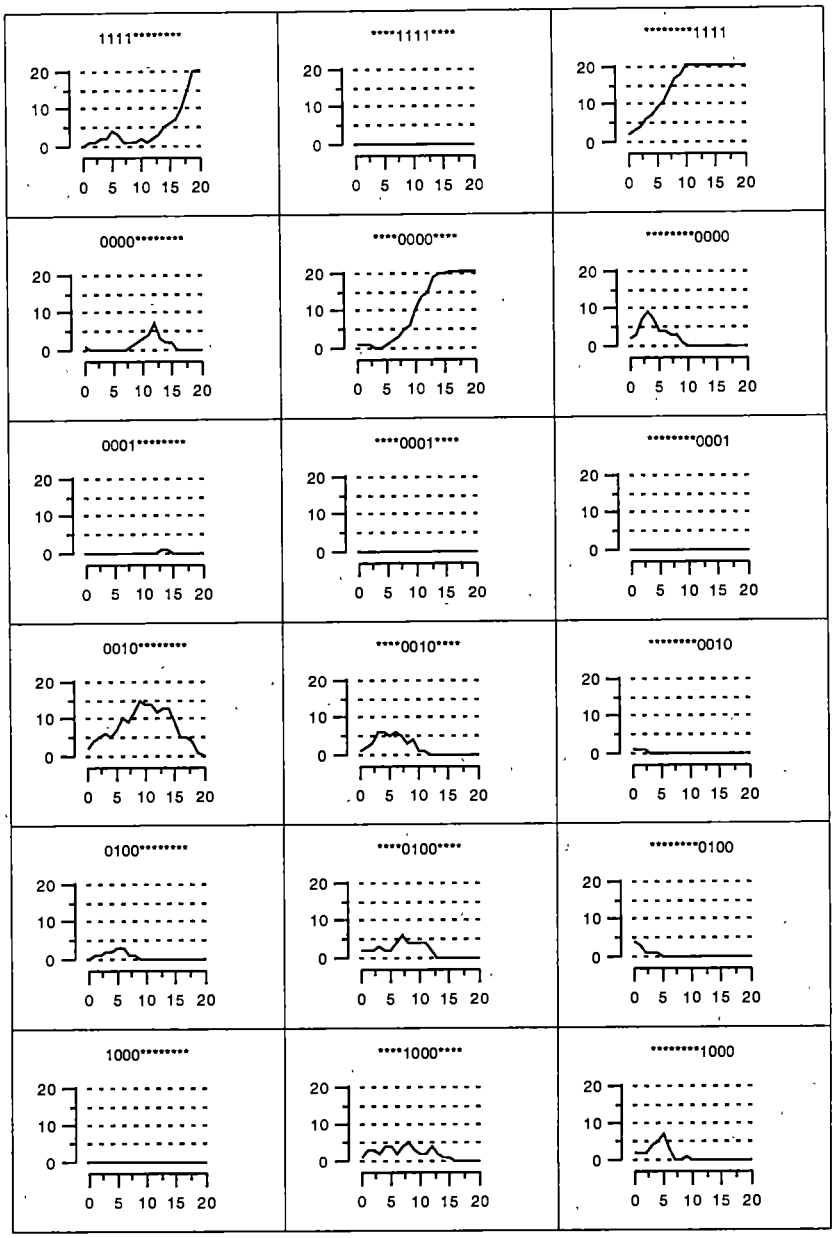


Figure A.5: Population Size=21. Seed=798798137. Fitness function=Equation 3.1.

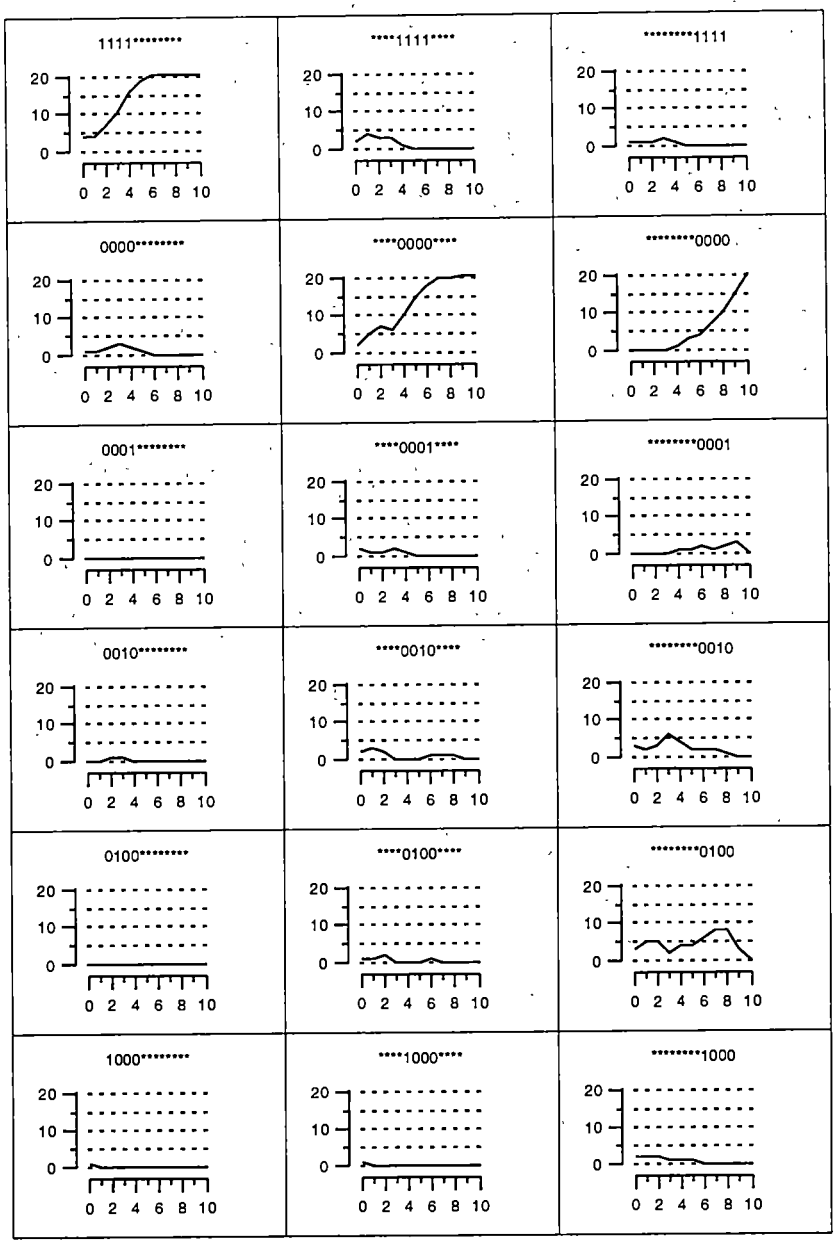


Figure A.6: Population Size=21. Seed=1934811071. Fitness function=Equation 3.1.

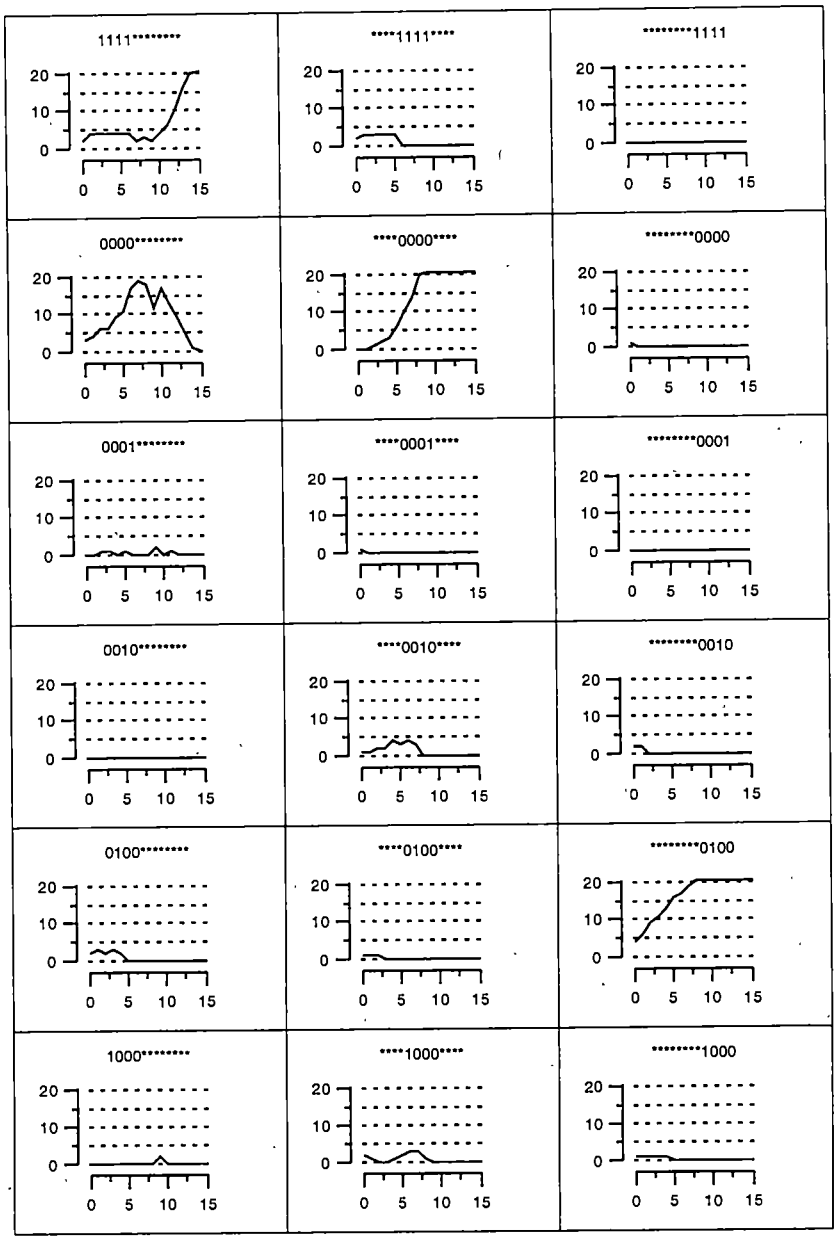


Figure A.7: Population Size=21. Seed=1207155841. Fitness function=Equation 3.1.

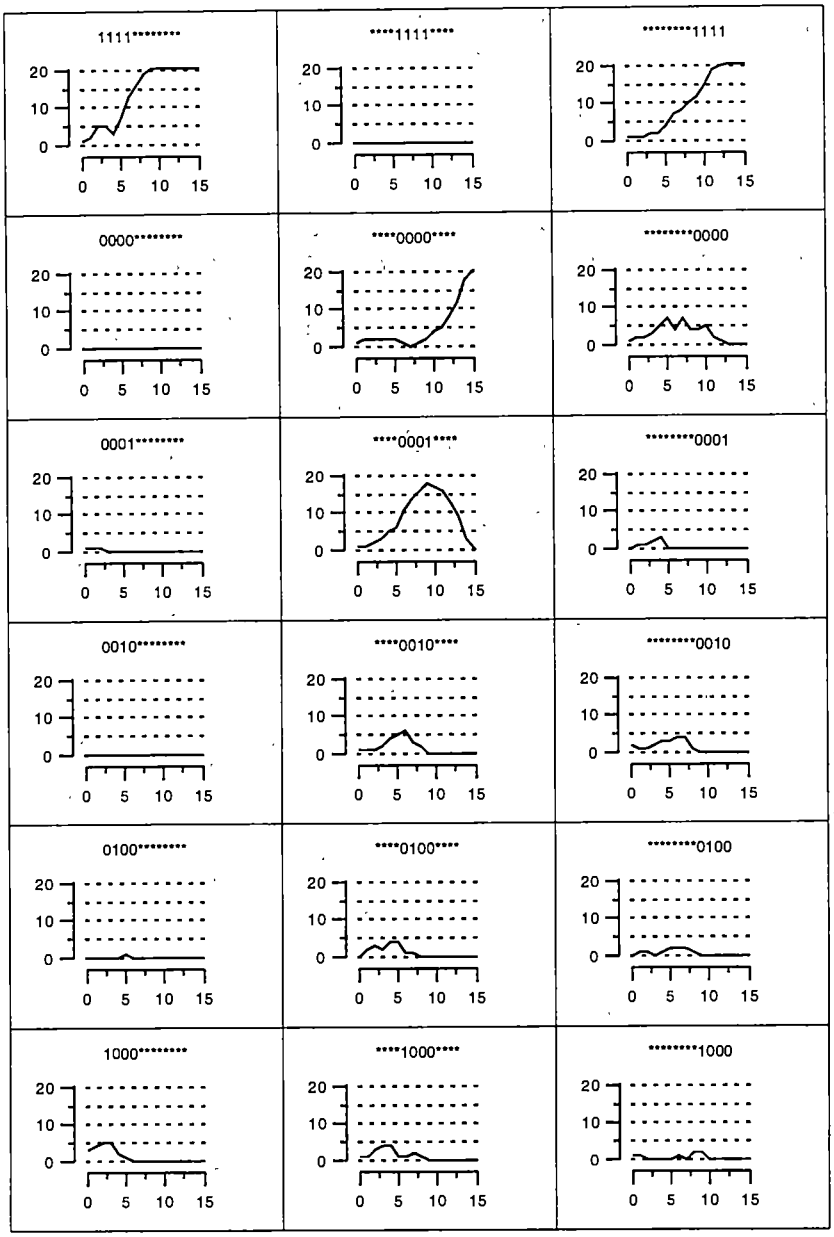


Figure A.8: Population Size=21. Seed=1450788425. Fitness function=Equation 3.1.

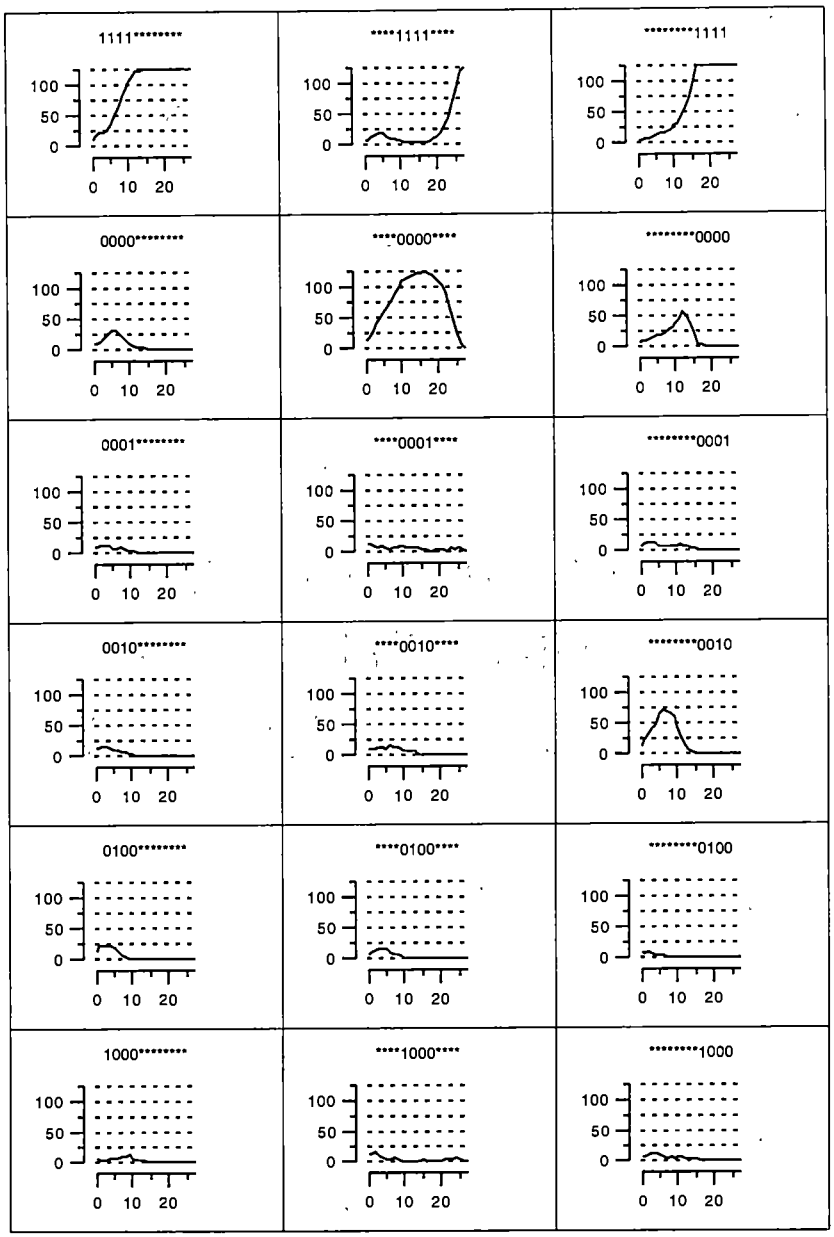


Figure A.9: Population Size=127. Seed=71659920. Fitness function=Equation 3.1.

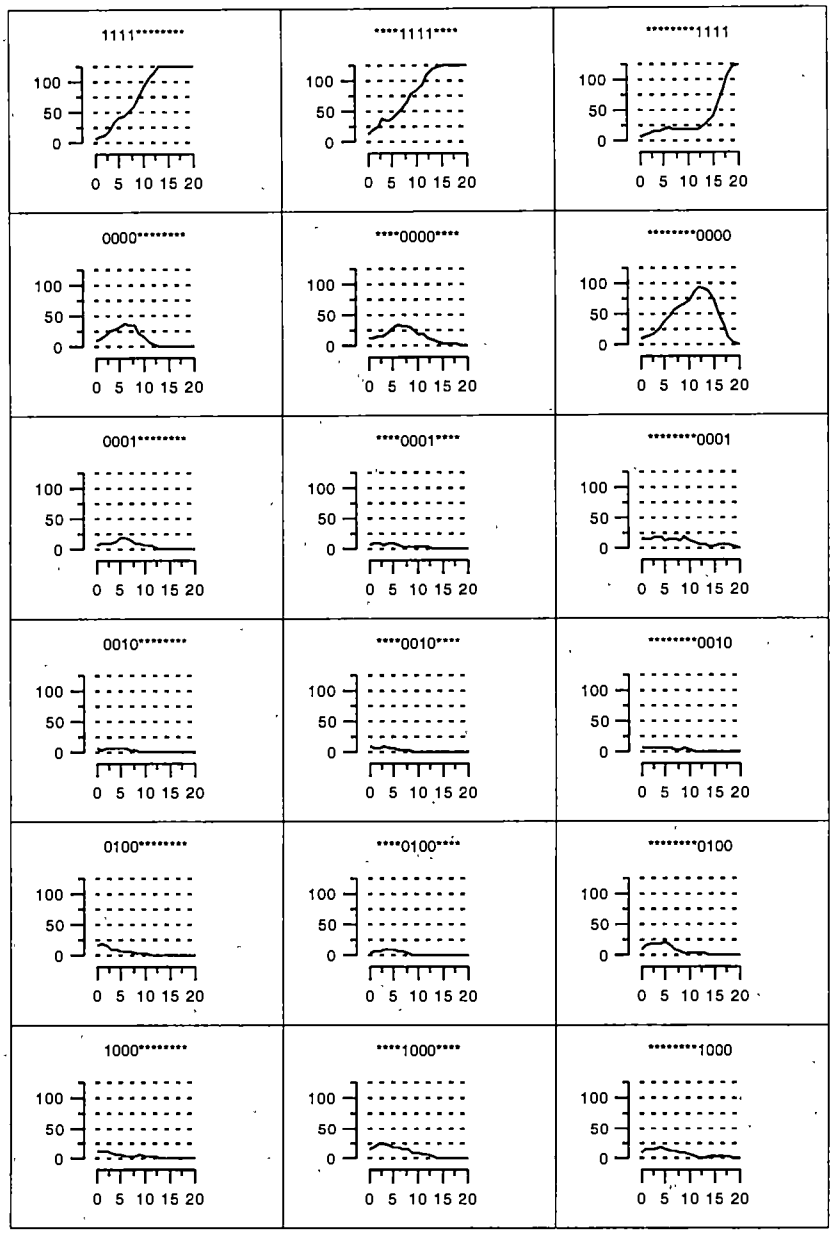


Figure A.10: Population Size=127. Seed=1688918702. Fitness function=Equation 3.1.

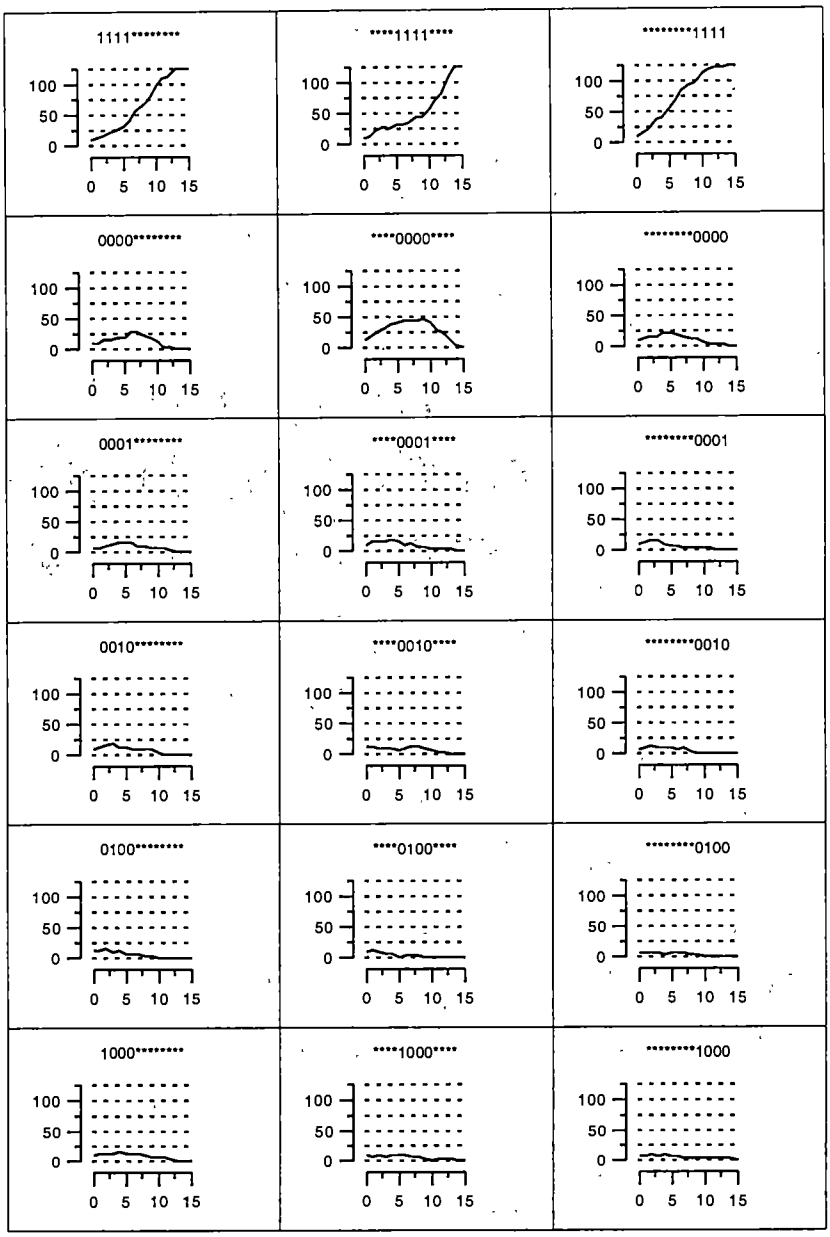


Figure A.11: Population Size=127. Seed=1507297010. Fitness function=Equation 3.1.

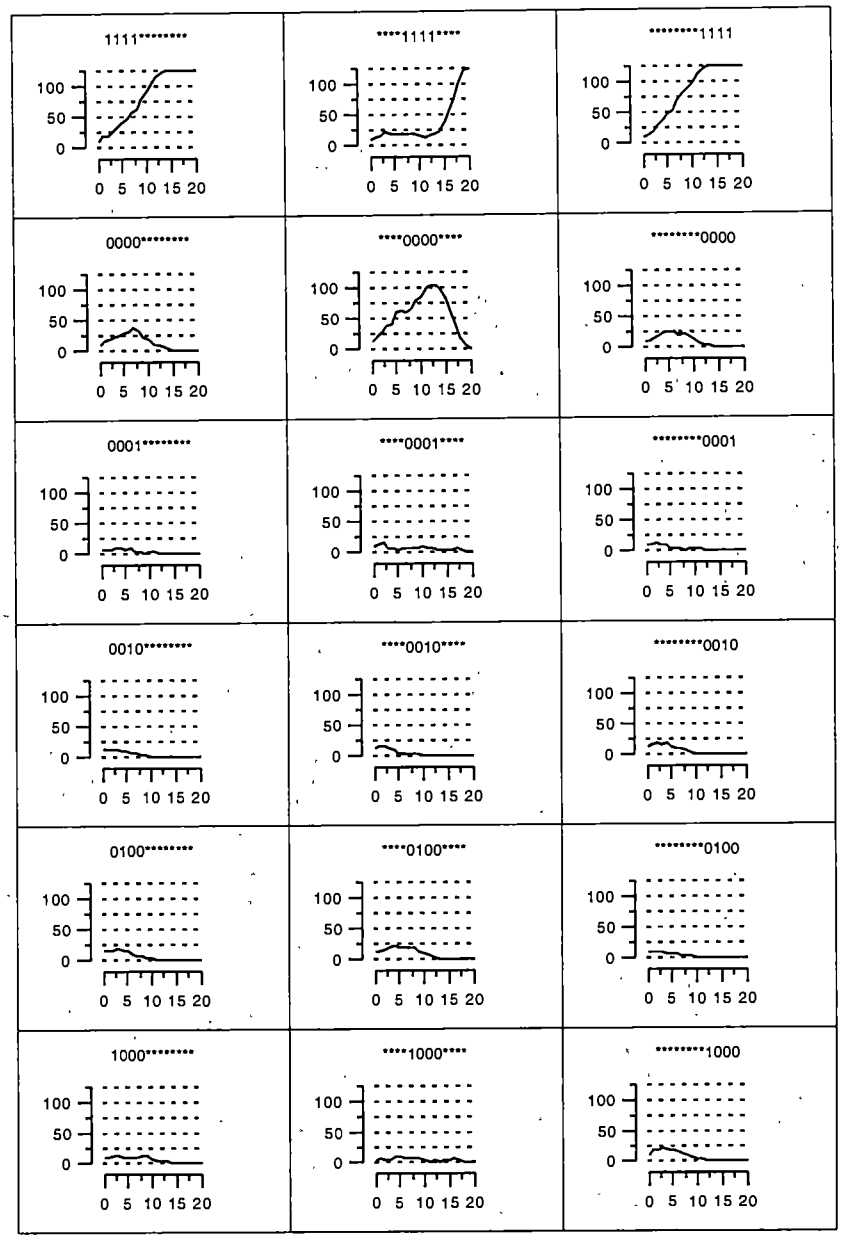


Figure A.12: Population Size=127. Seed=1887561901. Fitness function=Equation 3.1.

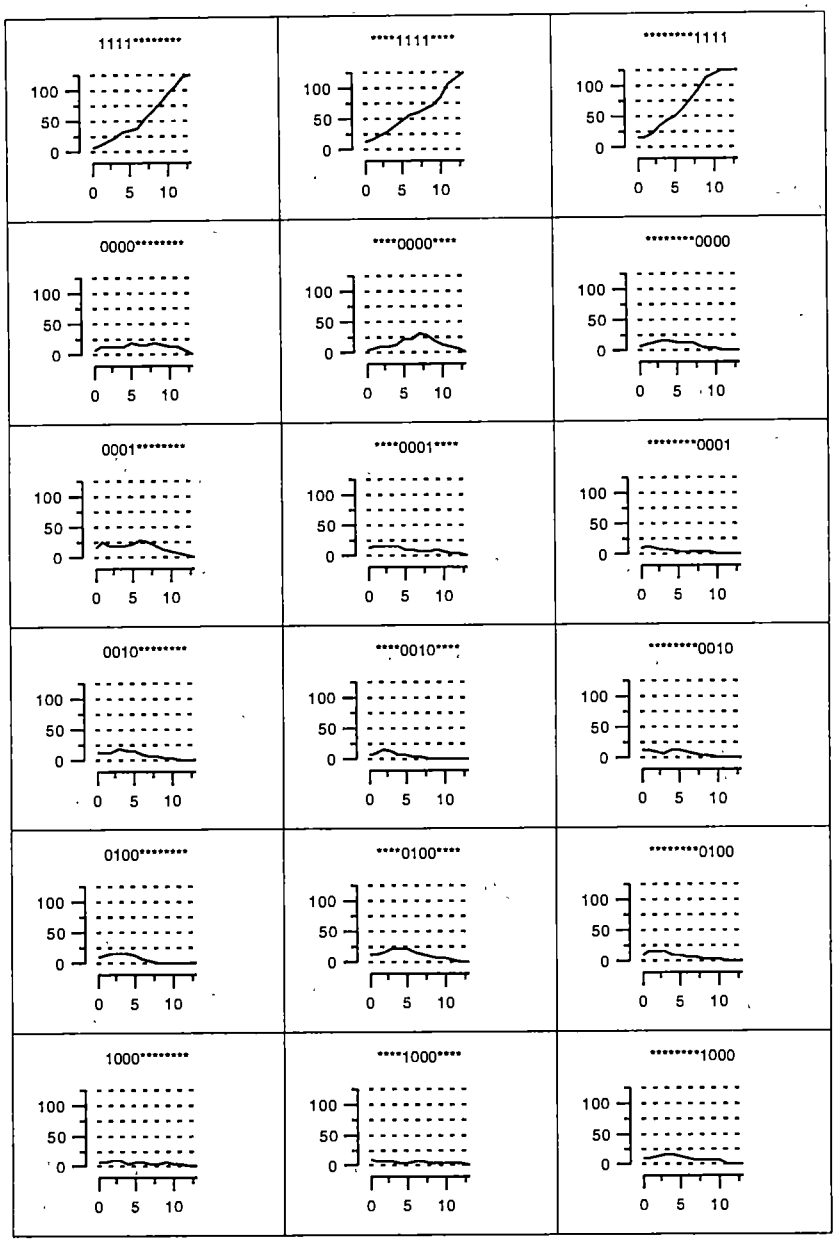


Figure A.13: Population Size=127. Seed=798798137. Fitness function=Equation 3.1.

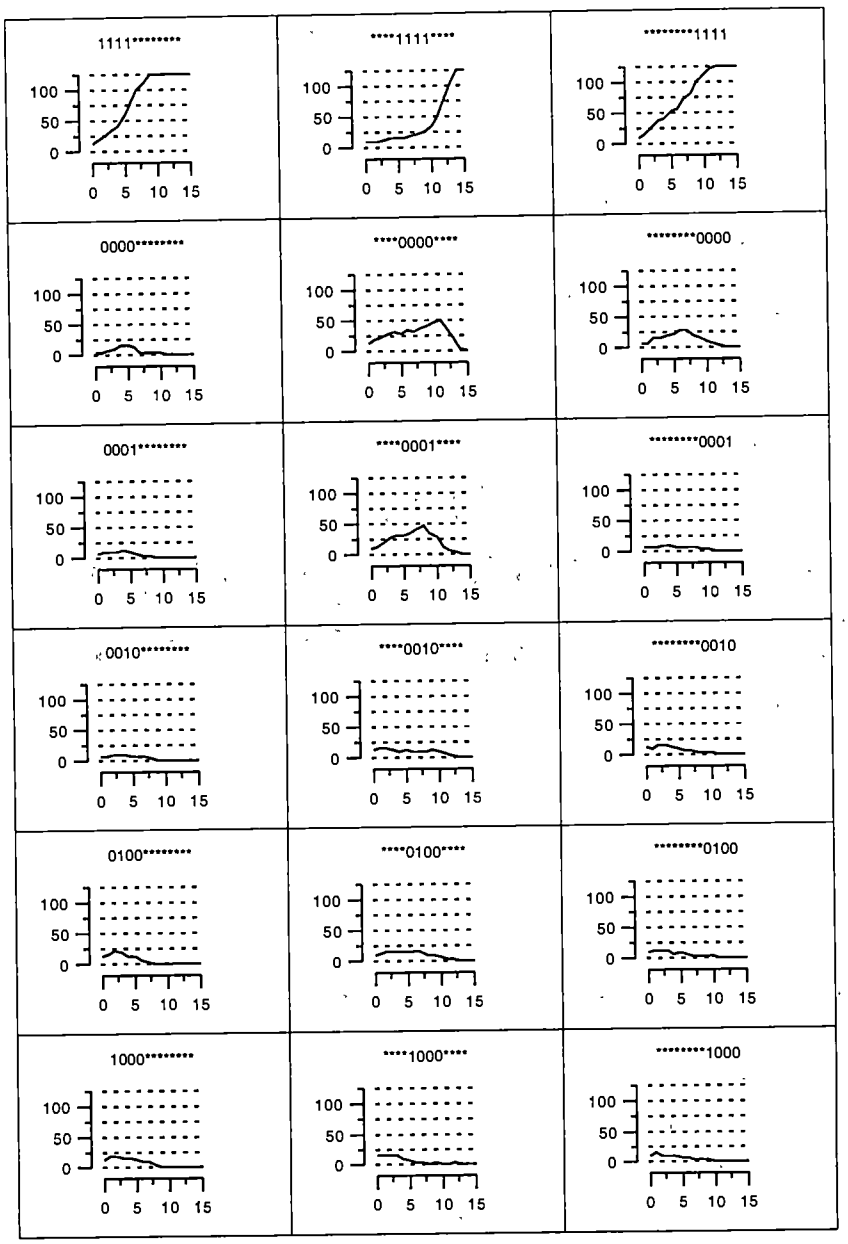


Figure A.14: Population Size=127. Seed=1934811071. Fitness function=Equation 3.1.

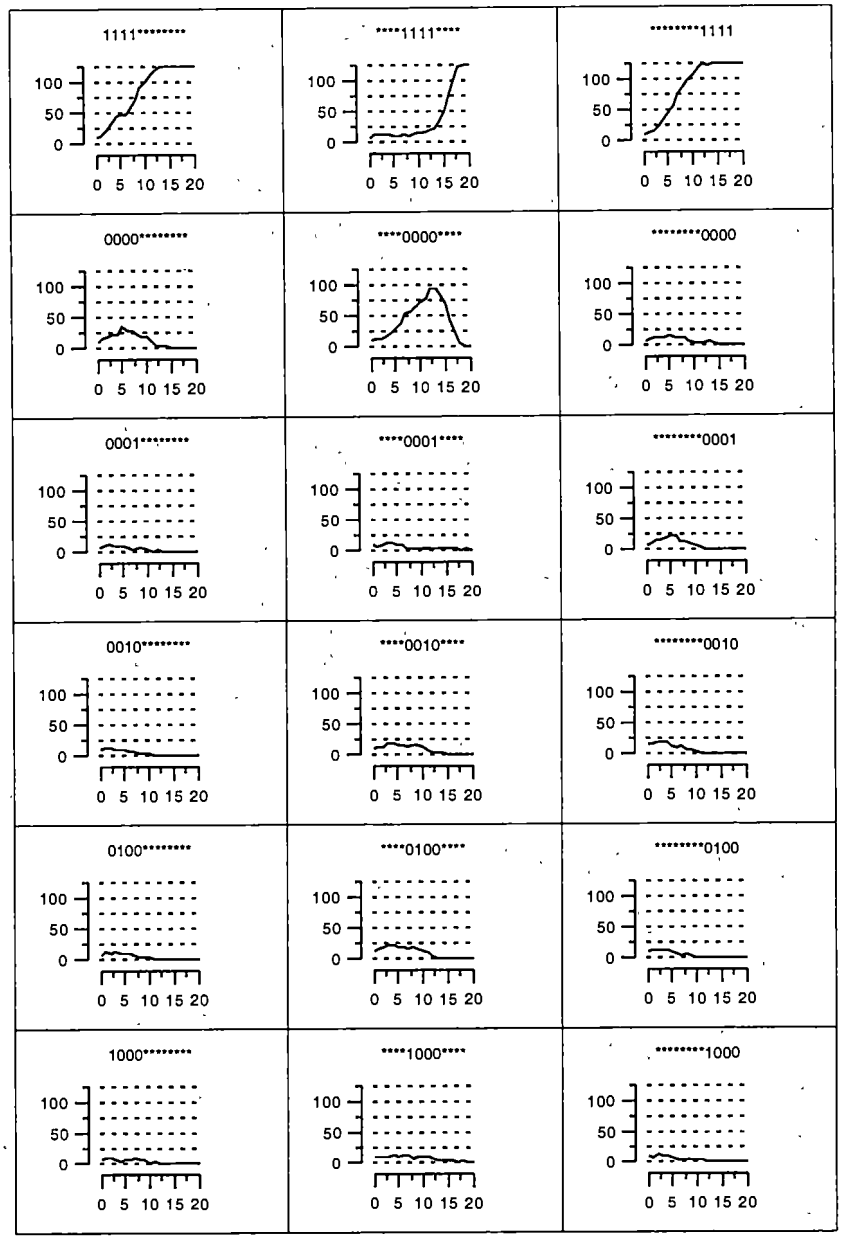


Figure A.15: Population Size=127. Seed=1207155841. Fitness function=Equation 3.1.

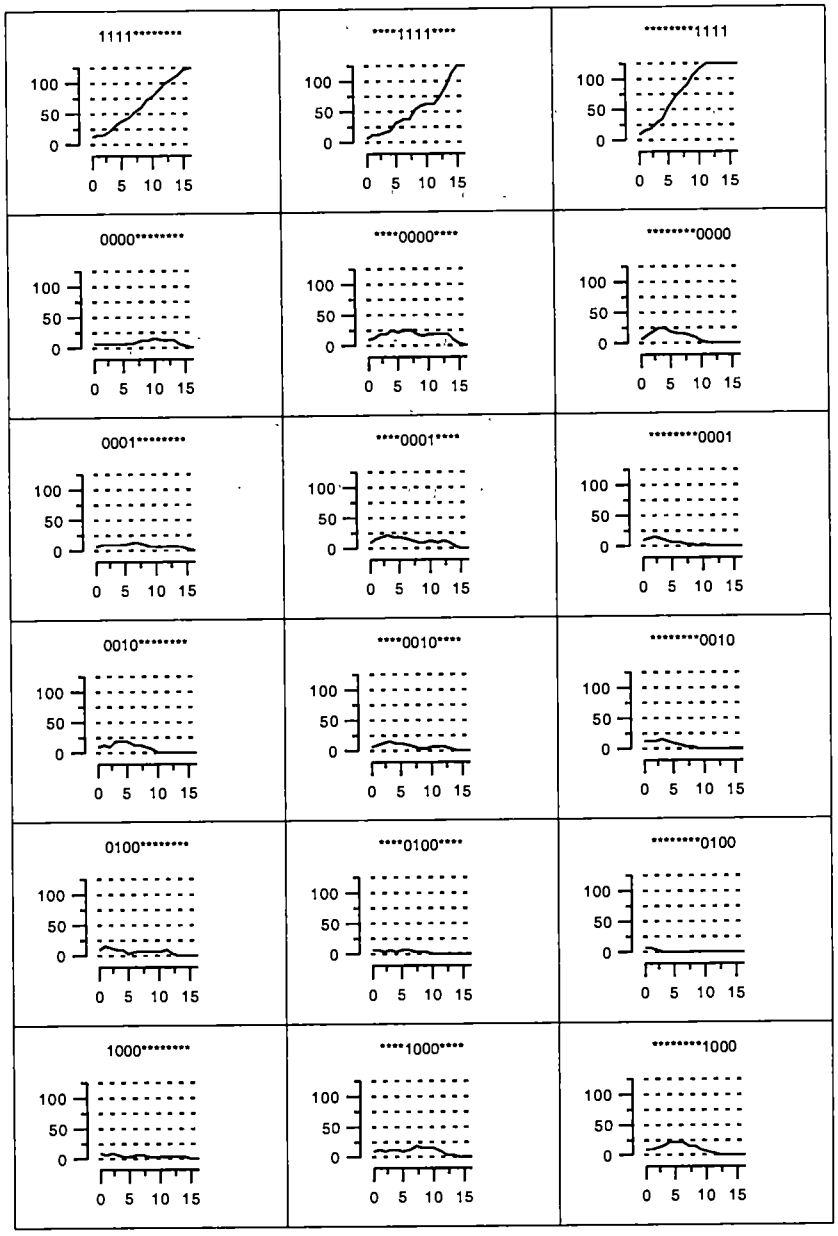


Figure A.16: Population Size=127. Seed=1450788425. Fitness function=Equation 3.1.