Behavior Of Variable-length Genetic Algorithms Under Random Selection

2007

Harold Stringer
University of Central Florida

BEHAVIOR OF VARIABLE-LENGTH GENETIC ALGORITHMS UNDER RANDOM SELECTION

by

HAROLD L. STRINGER
B.S. University of Florida, 1979

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2007

ABSTRACT

In this work, we show how a variable-length genetic algorithm naturally evolves populations whose mean chromosome length grows shorter over time. A reduction in chromosome length occurs when selection is absent from the GA. Specifically, we divide the mating space into five distinct areas and provide a probabilistic and empirical analysis of the ability of matings in each area to produce children whose size is shorter than the parent generation's average size. Diversity of size within a GA's population is shown to be a necessary condition for a reduction in mean chromosome length to take place. We show how a finite variable-length GA under random selection pressure uses 1) diversity of size within the population, 2) over-production of shorter than average individuals, and 3) the imperfect nature of random sampling during selection to naturally reduce the average size of individuals within a population from one generation to the next. In addition to our findings, this work provides GA researchers and practitioners with 1) a number of mathematical tools for analyzing possible size reductions for various matings and 2) new ideas to explore in the area of bloat control.

To my mother Emily Stringer who encouraged me to change and continue my education and to my father Harold Stringer who was with me the whole way and made sure I finished.

ACKNOWLEDGEMENTS

I would first like thank all of my Committee Members: Dr. Euripides Montagne, Dr. Ali Orooji, Dr. David Pratt and Dr. Annie S. Wu for their time, support and encouragement during my studies here at the University of Central Florida (UCF). My interest in genetic algorithms began with a research project involving Dr. Wu and Dr. Pratt. My thanks to both of them for letting me participate in that initial project and exposing me to the world of evolutionary computation. Special thanks to Dr. Wu for her patience, sound advice and feedback as I worked to complete my studies. My understanding of genetic algorithms, an appreciation for the research process, and improvements in my critical thinking and writing skills are all a direct result of her tireless efforts.

I would also like to thank Dr. Dutton, Dr. Lang, Dr. Montagne, Dr. Orooji, Dr. Workman and Dr. Wu for their help and guidance while teaching undergraduate courses at UCF during the past few years. That experience was as much a part of my education as any course work or research project. Your suggestions and encouragement were greatly appreciated.

Last but not least, I would like to thank my family and friends, in particular Kevin Miller and Connie Wightman, as well as fellow members of UCF's EC Lab for their support and assistance as I completed this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

ChGA	Chunking Genetic Algorithm
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolutionary Strategies
GA	Genetic Algorithm
GE	Grammatical Evolution
GP	Genetic Programming
LCS	Learning Classifier Systems
SGA	Simple Genetic Algorithm

CHAPTER 1: INTRODUCTION AND MOTIVATION

Genetic algorithms (Holland, 1975) or GAs consist of a family of search methods that have been applied to a wide variety of problem classes including optimization, classification, and machine learning. Within this family are numerous GA implementations which can be categorized in a variety of ways (generational vs. steady state, single vs. multi-objective, serial vs. parallel, fixed vs. variable-length, etc.).

Currently, fixed length representations dominate the field of genetic algorithms. Such representations are best suited to problems with a pre-determined solution size – a size known *a priori*. For such problems, the GA researcher or practitioner knows the number of variables for which to solve. Unfortunately not all problems are so simple or well understood.

Often the "size" of an optimal or acceptable solution cannot be determined in advance. Genetic programming or GP, a branch of evolutionary computation (EC) deals continuously with this type of problem. GP's goal is to develop systems (e.g., computer programs, finite state machines or logic circuits) that accomplish a specific objective but without advance knowledge of the quantity and quality of the system's components. Often these systems are coded using some "programmatic" representation (trees, LISP expressions) so that the phenotype can be executed to determine its fitness.

Not all complex open-ended problems require a "programmatic" solution or lend themselves to tree-like representations. For example, what number of rules will give the best results with a classifier system? Or closer to the real world, how many cell towers should be constructed to provide optimal coverage for a given area? These sample problems can use a simpler coding method with each rule or tower being represented by a gene within a single chromosome.

Both problems can be framed in a fixed-length format by determining quantities in advance – what are the 30 best rules or where should we locate 10 towers. In actuality, the best answer may require a number of rules or towers less than or greater than the number specified at the start.

This is where the variable-length GA can be applied. Using the cell tower example, a variable-length GA can be designed using individual genes to represent the location and type of each cell tower. The number of genes within a chromosome determines the total number of towers recommended. A single chromosome, therefore, represents the entire solution to the problem (number, location and type of towers). In theory, a variable-length GA would evolve both the quantity (number) and quality (type and location) of towers for a given area. Chromosomes within a population would grow in size adding more genes if more towers were needed. Smaller chromosomes would appear if fewer towers were best.

On a practical level this does not appear to happen. Chromosomes in a variable-length GA tend to grow in size indefinitely over the course of a GA run. Chromosome size increases, depending on the type of GA, by adding non-coding material or new genes which compete with or duplicate existing genes. The term "bloat" is used in EC literature to describe unrestricted growth in variable-length representations. The end result of bloat is that a good or acceptable solution may not be found by a variable-length GA. Chromosomes may grow past the size of the best possible solution, or a good solution may be present but unrecognizable since it is embedded in a longer chromosome full on non-coding or competing genes. Bloat is not unique to GAs. The phenomenon can be found in all variable-length representations in EC including genetic programming, grammatical evolution and learning classifier systems.

GA researchers (as well as researchers in other EC paradigms) have found it necessary to introduce mechanisms into their GAs to restrict bloating and keep variable-length individuals within a manageable size range. Simple truncation or adding length as a factor in fitness evaluation (parsimony pressure) are just two examples of such mechanisms. More extreme measures may be needed to actually cause average chromosome size within a population to shrink. These efforts add an additional layer of complexity to a variable-length GA and require additional computation time. They also require operators that might not be found in nature and move the variable-length GA farther from its source of biological inspiration.

In our opinion, bloat is the largest roadblock to the efficient application of variable-length GAs for solving complex open-ended problems.

Our past work (Stringer & Wu, 2004, 2005; Wu & Stringer, 2002) as well as that of other GA researchers (McPhee & Poli, 2001; Rowe & McPhee, 2001) shows that in the absence of selection pressure, variable-length structures, including GAs, do not grow indefinitely. Instead, they are able to naturally shrink chromosomes down to a smaller size without any explicit operators. Variable-length GAs under random selection are not affected by bloat.

The goal of this work is to determine the necessary conditions and mechanisms that cause this natural reduction in chromosome size. We begin with an overview of Genetic Algorithms in Chapter 2. This is followed by a survey of the current literature as it relates to variable-length GAs and changes in chromosome length under random selection in Chapter 3.

In Chapter 4 we examine the space of individual mating events and identify five regions of behavior. Each of these regions or areas makes a different contribution to a GA's ability to produce shorter than average children. By calculating these contributions in terms of probabilities, we provide insights into how selection of parents will affect the size of the

offspring generated in individual mating events. Chapter 5 demonstrates how a variable-length GA under random selection creates diversity in chromosome size and describes how size diversity is a necessary condition for production of shorter than average children.

The distribution probabilities for shorter than average children combined with 1) the imperfect nature of random sampling during selection and 2) the fact that random selection tends to increase size diversity causes finite population GAs under random selection to evolve ever decreasing mean chromosome lengths. We illustrate this combination of forces using the chunking genetic algorithm or ChGA (Wu & Stringer, 2002) as an example in which we can see this behavior in action. We believe that understanding this natural behavior unique to variable-length structures under random selection can lead to new methods for combating bloat in variable-length GAs and potentially other variable-length EC paradigms.

CHAPTER 2: OVERVIEW OF GENETIC ALGORITHMS

Genetic algorithms (Holland, 1975) or GAs consist of a family of search methods that have been applied to a wide variety of problem classes including optimization, classification, and machine learning. GAs are one branch of Evolutionary Computation (EC) which also includes among others:

- EP - Evolutionary Programming (Fogel, Owens & Walsh, 1966)
- ES - Evolutionary Strategies (Rechenberg, 1973; Schwefel, 1977),
- GE - Grammatical Evolution (Ryan, Collins & O'Neill, 1998),
- GP - Genetic Programming (Koza, 1992), and
- LCS - Learning Classifier Systems (Holland, 1986; Smith, 1983),

All of these branches of evolutionary computing share some common traits that allow them to solve computationally difficult problems (K. A. De Jong, 2003). Among those traits are 1) the concept of a population of individuals which represent possible solutions to a given problem, 2) a notion of fitness associated with each individual, 3) a birth-reproduction-death cycle repeated for some number of generations, and 4) the use of Darwinian-inspired operators or processes to create new individuals from selected members of the current population (reproduction). Each branch of EC has undergone substantial change in the particulars associated with each of above traits since initial introductions.

Holland first proposed the GA as part of his study of adaptation. His GA included a population of binary string chromosomes, each chromosome encoding a possible solution to the problem being solved. The GA moved from one population to the next through the use of selection, recombination, mutation and inversion. Selection mimics natural selection based on

survival of the fittest. More fit chromosomes are more likely to reproduce and pass on their partial solutions (chromosomal substrings) to future generations. Reproduction was performed by a recombination operator which swapped substrings from two chromosomes thereby creating two new and different chromosomes as offspring. Mutation and inversion operators allowed for changes to take place within a single chromosome.

Over time the concept of a "simple" genetic algorithm or "SGA" developed (De Jong, 1975; Goldberg, 1989), which most researchers view as a starting point for GA modifications. A simple GA begins by encoding values for problem variables as sets of bit substrings sometimes known as *genes*. These bit substrings or genes are then grouped together in a single *chromosome*. Each chromosome contains a set of genes or bit substrings sufficient to represent all variables in a problem and thus a possible solution to the problem being solved. A pre-specified number of chromosomes make up a GA's *population*. An initial population can be randomly generated or generated based on heuristics relevant to the problem at hand.

The SGA applies the concepts of natural selection and genetic reproduction to this initial population. Over a number of *generations* the SGA "evolves" one or more individuals that represent a good or acceptable solution to the problem. Evolution of solutions is accomplished by repetitive applications of the SGA's evaluate-select-reproduce-replace cycle. During that cycle, the SGA evaluates the current population to obtain some measure of fitness for each chromosome. The best individuals within each generation are selected for reproduction. Reproduction combines portions of chromosomes belonging to two different individuals (parents) to form two new chromosomes. These new chromosomes can be thought of as children of the two parents. At the close of the reproduction cycle the children replace the parents in the current generation thus creating a new and different population.

An optional mutation phase can be included in the SGA to modify bit strings, one bit at a time. Mutation helps preserve genotypic diversity in a population and allows the GA to generate new solutions that might not otherwise be created through selection and reproduction only.

A simple GA executes the evaluate-select-reproduce-replace cycle for a given number of generations or until a solution of acceptable quality is found. The following pseudo-code shows the steps normally performed by a SGA during the search process:

```
procedure SGA
     {
     initialize population;
     while (stopping condition not satisfied){

          // Evaluate Current Population
          for (j=1 to population size){
               evaluate fitness of individual j;
          }

          //  Select and Reproduce
          for (j=1 to population size / 2){
               select two parents for reproduction;
               crossover to produce two children;
          }

          //  Mutate Chromosomes of Children
          for (j=1 to population size){
               perform mutation on child j;
          }

          //  Replace Parents with Children
          copy children to next generation;
     }
end procedure SGA
```

The details of a SGA vary slightly from the above depending on the researcher. For example, De Jong's SGA (defined in his dissertation as R1) produces only one child for each crossover occurrence. This requires twice the number of select and reproduce operations than shown in the preceding pseudo code. Goldberg's SGA produces two children from two parents

but performs mutation on the parents before crossover. In general, a simple GA, regardless of coding specifics can be summarized as having the following key features:

- a single *population of individuals* composed of fixed-length bit strings which represent possible solutions to the problem to be solved. The size of the population is invariant from one generation to the next.
- an objective *fitness function* that evaluates the quality of the solution represented by an individual. For purposes of the SGA, this is usually a single-objective, stationary problem;
- a *selection function* based on fitness that chooses individuals for reproduction based on their fitness (most often fitness proportional);
- one or more *genetic operators* that recombine partial solutions (bit substrings) from two selected individuals during reproduction (e.g., crossover) or which modify individuals (e.g., mutation) in order to increase population diversity;
- a *single-processor computing environment* in which the SGA is executed. This feature may be relaxed in a limited fashion to allow the use of additional processors to perform fitness evaluations.

Given the SGA as a starting point, research in the field of genetic algorithms has taken off in numerous directions. These directions can be classified in three major categories: 1) extending the SGA operationally (mostly empirical studies), 2) understanding how the GA works (theoretical studies), and 3) application of the GA to real world problems. The goal of this thesis is not to provide a survey of GA history. Therefore, a general discussion with respect to extensions and applications of the GA are not provided herein. Readers interested in these topics

may wish to review (Harik, 1997) (contains good background on GAs) or (Mitchell, 1996) as well as recent proceedings from GA-related conferences.

CHAPTER 3: LITERATURE REVIEW

This thesis looks at changes in genome length under random selection for variable-length genetic algorithms. As one can imagine, the quantity of literature directly addressing this specific issue is limited, but there are ideas from other EC paradigms, specifically GP, which are helpful in understanding our main topic. We have, therefore, expanded and divided our literature search into four major areas: variable-length representations in EC, variable-length representations in GAs, bloat and bloat control in GA/GP, and the effects of random selection on chromosome length in GA/GP.

3.1 Variable-Length Representations in EC

Variable-length representations have been used and studied in evolutionary computation (EC) for some time. Within EC the popularity of variable-length structures, not to mention, quantity of associated literature, depends primarily on the EC branch being studied (see Figure 1).

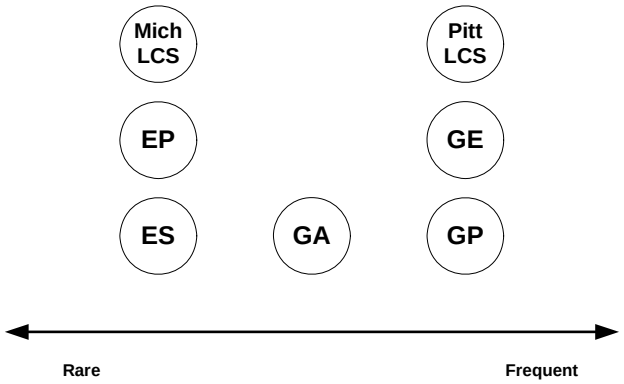

Figure 1: Use of Variable-length Representations by EC Paradigm

For example, EC paradigms such as GP and GE, which seek to evolve computer programs or hardware descriptions, almost always use variable-length representations. This result is due to the fact that these paradigms search for solutions to problems of open-ended complexity – the length of programs or the number of hardware components in an optimal solution can vary in size and is not known *a priori*. GP and GE have fully adopted variable-length representations into their implementations. Not surprisingly, these are also the areas of EC where bloat (uncontrolled growth in chromosome size) has been the greatest source of consternation and study.

At the other end of the spectrum are EP and ES. Typically these paradigms use chromosomes or vectors of numeric values (real or integer) which correspond to variables in an optimization problem. The number of variables in the problem is generally known in advance so representations in these paradigms are fixed-length in nature. This is not to say that variable-length representations are not possible.

In (Barone, While & Hingston, 2002), variable-length ES vectors were used to list two-dimensional coordinates for defining the cross-section of a crusher liner – the more coordinates in the list, the more complex the liner shape. This variable-length representation required the addition of two new mutation operators for increasing/decreasing the number of coordinates and thus redefining the liner shape.

Learning Classifier Systems or LCSs are a form of EC used in machine learning. Specific applications include development of rules for robotic control, data mining and autonomous agents. The goal of a classifier system is to discover a set of classifiers or rules which can be used to categorize data, direct the actions of an agent, or allow the LCS to interact with its environment through effectors given input from that environment by detectors. There

are two primary approaches to LCS rule discovery which are shown separately in Figure 1. The reasons for their appearance at opposite ends of the variable-length spectrum follows.

The *Michigan* approach (see (Wilson, 1995) for example) incorporates a population of classifiers with each individual in the population being a single rule. Classifiers are of fixed-length and consist of a left-hand side (condition) and right-hand side (action). A production system is required to evaluate the effectiveness of each classifier and determine its overall contribution to the rule set. At the heart of this LCS is a form of steady-state genetic algorithm which discovers new rules and passes them on to the production system. Although the population size (number of rules) may vary within a Michigan LCS, individuals in the population are of fixed size.

The *Pittsburgh* approach ((De Jong, 1988) and (De Jong & Spears, 1991)) to LCS focuses not on single rules but entire rule sets. Each member of the population contains a variable number of rules. Rules in this approach also have a left-hand side (disjunct) and a right-hand side (classification). A variable-length genetic algorithm is used to evolve successively better rules sets. The fitness function evaluates the quality of an entire set of rules at once. As a result, a production system like that used in Michigan-based systems is not necessary. A Pittsburgh LCS uses a population of fixed size consisting of variable-length individuals – an opposite representation from Michigan-based systems.

The partitioning of EC found in Figure 1 does not encompass all paradigms inspired by biology or evolution. There are other search algorithms that are gaining in popularity (e.g., particle swarms, immunological systems, ant colonies). In addition, there are many other areas of EC that deal with algorithm implementation (e.g., parallelization, co-evolution, multi-objective optimization) that interest many in the EC community. Discussions of these other paradigms or

areas are not included in this work as they begin to move us farther from our focus which is variable-length GAs.

3.2 Variable-Length Genetic Algorithms

As seen in Figure 1, GAs fall somewhere in the middle of our variable-length spectrum between ES/EP and GP/GE. Initial GA research used fixed-length binary strings. As the field has matured, numerous other representations have emerged (floating point, multi-character alphabets, etc.) In the last 10 to 15 years, variable-length GAs have been developed and studied in part due to the greater complexity of problem domains to which GA's have been applied.

In this section, we consider some of the different variable-length representations found in the GA literature and indicate which of these is related to our investigation of changes in genome length. Given the number of different representations, how should we go about organizing them for ease of understanding? It turns out that relating representations with the size of solutions they denote is a useful framework for categorizing variable-length GAs. From our perspective, solutions fall into one of three size categories: fixed, bounded, and unbounded solutions.

3.2.1 Representations for Fixed Solution Sizes

A fixed-sized solution is one that requires instantiation of a predetermined number of variables for completeness or optimality – too few variable values and the solution cannot be tested or the optimal solution cannot be found. The first GA's were developed with this type of problem in mind. Genes or bits on a GA chromosome corresponded to variable values for a potential solution based on their position within the chromosome.

3.2.1.1 The messyGA

The earliest and best known variable-length representation is the *messyGA* (Goldberg, Korb & Deb, 1989). This GA was developed to eliminate bit positional dependencies in a standard GA. The representation allows the GA to evolve both bit values and bit locations on a chromosome during a GA run. The goal of the messyGA is to find tight linkage between bits as well as good bit values simultaneously.

Each bit value in a messyGA chromosome is tagged with a name (an integer number indicating the position of the bit). Prior to fitness evaluation, the value bits are extracted from the chromosome and reordered based on their corresponding names. Bits within a genotypic string are no longer in fixed positions and can move around on a chromosome to develop better building blocks.

The messyGA was implemented using a cut-and-splice operator rather than standard one-point crossover. The operator produces variable-length chromosomes. Reproduction can sometimes result in children with multiple bit-values for a given name (overspecification) or the absence of bit values for a given name (underspecification). Instances of these two errors must be dealt with before a complete solution can be forwarded to the fitness function for evaluation.

Left-to-right precedence is used to eliminate overspecification due to duplicate or competing bit values for a given name. The bit value from the first messy gene with a given name is used during fitness evaluation. All other messy genes with the same name are then ignored and considered to be non-coding regions.

Underspecification occurs when no messy gene exists for a given bit required in the pre-test solution. This condition is handled by use of competitive templates. Bit values not found in the genome are pulled from a template, a string containing locally optimal bits from a previous

generation. The use of competitive templates allows the messy GA to build a complete solution for testing even when an individual chromosome does not encode a solution in its entirety.

The goal of the messyGA was not to address open-ended problems but to allow a GA to evolve faster and find better solutions to problems of fixed size. The limitation to the original messyGA is its focus on bits. A complete solution (assumes no under- or overspecification) for an n-variable problem with each variable needing l bits requires $l*n*(\text{floor}(\log_2(l*n-1))+1)$ additional bits in the genome to identify all its component bits. For very large problems, this amount of overhead can quickly exceed 90% of the bits in a chromosome.

3.2.1.2 Gene-Based Tagging for Fixed Solution Sizes

An extension of the messy GA is the identification or tagging of whole genes (building blocks) rather than individual bits. Here, a gene refers to any collection of adjacent bits which taken together code for a variable value. Assuming a fixed solution size, a gene-based approach to position independence requires only $n(\text{floor}(\log_2(n-1))+1)$ additional bits in the genome to identify genes for a complete solution.

Using tagged locus-independent genes in conjunction with fixed sized solutions has been studied previously in (Burke et al., 1998; Wu & Lindsay, 1996) albeit in the context of fixed-length rather than variable-length strings. Tagged position-independent genes can be used with variable-length GAs searching for solutions of fixed or bounded size.

A simple example of a GA problem with fixed-sized solution is the mxn MaxSum test function found in (Stringer & Wu, 2004). Each gene within a chromosome is viewed as a 2-tuple: $<m, n>$ where the first m-bits represent a positive integer index (a.k.a., tag, name, identifier) and the remaining n-bits represent a positive integer value. The function is the sum of

the largest value for each of the 2^m indices. For example, assume a 2x8 MaxSum, the best possible fitness of a chromosome is 1020 based on the presence of the following 2-tuples somewhere in the chromosome: <0,255>, <1,255>, <2,255> and <3,255>. In the case of overspecification, the function chooses only the largest value for a given index. Underspecification (no gene for a specific index) will result in a less than optimal solution. If we view each index as a variable, the function requires a fixed size solution that can only be optimized when all m+1 variables are instantiated.

3.2.1.3 The Virtual Virus

Another variable-length representation with fixed-size solutions is the *Virtual Virus* or *VIV* (Burke et al., 1998). The goal of this work was to create a genetic algorithm that more closely mirrored biology. The authors described a number of key features found in biological systems that may be absent from a standard GA including:

- Genomes which vary in length during evolution,
- Gene locations independent of position identified by stop and start codons,
- Presence of non-coding regions in a genome,
- Duplicative or competing genes on the same genome,
- Use of overlapping reading frames when retrieving genetic information,
- A multi-character alphabet (A, T, C & G representing nucleotides) and
- Degenerate mapping of nucleotide triplets (codons) to amino acids.

Each of these features was incorporated into VIV to some degree. Use of a multi-character alphabet and degenerative mapping allows for redundant genotype representations in

VIV. More than one codon (three alphabet characters) can translate into the same phenotypic trait. Incorporation of intergenic regions (non-coding information) and redundant genes (the same information can appear more than once) also allow VIV to more closely follow its biological inspiration.

To remove location restrictions, each gene in VIV is marked by a START codon consisting of three characters. This start tag is analogous to a promoter region in DNA. The end of a gene is specified by a corresponding STOP codon. The use of START and STOP codons allows 1) genes in VIV to be of variable-length, 2) non-coding regions to appear between STOP and START codons and 3) overlapping of genes due to three reading frames.

Processing of a chromosome begins by reading left to right across the genome. If a START codon is found at position *p*, any subsequent characters are read as genes until a STOP codon is reached. Reading of the genome picks back up at position *p*+1 looking for the next START codon. The processing of chromosomes in this manner allows VIV to contain overlapping genes within a chromosome. Overlap makes possible evolution of solutions in a very compact genome.

3.2.1.4 The Proportional Genetic Algorithm (PGA)

Another recent variable-length representation to appear in the literature is the *Proportional Genetic Algorithm* or *PGA* (Wu & Garibay, 2002). This representation evolves solutions of fixed length while taking locus-independence to the extreme. The PGA uses a multi-character alphabet. Each character is associated with a specific variable to be found in the solution. The number of each character appearing in the genome is used to compute a value that is then assigned to its corresponding variable.

In the case of a PGA1, values are determined by the equation:

$$P_{PGA1}(V_i) = \frac{number\ of\ characters\ (V_i)\ on\ genome}{length\ of\ individual}$$

where V_i is the ith variable in the solution. The values for all variables in a PGA1 chromosome sum to 1. As a result, PGA1 is well suited to use in resource allocation problems. The following gives an example of this representation:

Individual of Length 20:	ABBDAFEDEFADBAAABFDD		
Variable Values:	A = 6 / 20	or	0.30
	B = 4 / 20	or	0.20
	C = 0 / 20	or	0.00
	D = 5 / 20	or	0.25
	E = 2 / 20	or	0.10
	F = 3 / 20	or	0.15

The PGA uses alphabet symbols as atomic units in its representation rather than bits or genes. To quote from (Wu & Garibay, 2002), "The PGA representation is based on this idea that it is the content rather than the order of the encoded information that matters." The PGA allows evolution of the genome at the atomic character level rather than the gene level.

The PGA is our final example of a variable-length genotype that encodes a solution of fixed size. The number of characters in the PGA alphabet is determined at the start based on the number of variables in a complete solution. A downside of this representation is the need for large alphabets if problems have large numbers of variables.

A major benefit of the PGA is that over- and underspecification are no longer issues. All assigned characters in the genome are used to calculate proportions for each variable. Underspecification (absence of a character) is handled simply by setting the value associated with a missing character to "0". Another important benefit of this representation is the lack of any "overhead" information in the genome – no identifying tags or start/end tags are required

3.2.2 Representations for Bounded Solution Sizes

Bounded solutions, our second size category, are associated with problems where a maximum number of possible variables is stated or implied, but instantiation of all variables is not required to form a complete, testable solution.

The 0/1 Knapsack problem is a good example of a problem that can be represented with a bounded solution size. The problem is as follows: given a collection of *n* unique items, what combination of those items will maximize the value of the knapsack's contents but not exceed the knapsack's size/weight constraint?

This type of problem is easily represented using fixed size solutions. All that is needed is an *n*-bit string. Each bit in the string is associated by position to an item with a binary variable of value "1" (choose the item) or "0" (do not select the item) that can be used to calculate the knapsack's contents. The difficulty with such encodings arises out of crossover's position bias (Eshelman, Caruana & Schaffer, 1989); the order in which items are assigned to bits (or genes) in the encoding can affect the combinations of items that are likely to be tested.

The same problem can also be constructed using a variable-length representation which allows for a bounded solution size. Let us begin by assigning an identifier or tag to each item in the collection. Assume a collection of n=9 items. Let a GA chromosome contain a list of tags that together describe the entire contents to be loaded into the knapsack. The largest possible solution would be a string containing all nine tags, regardless of order, for example the string <9 1 3 5 6 2 4 7 8>. Obviously this solution would exceed the knapsack's capacity – otherwise there would be no problem to solve. A possibly better solution might be the chromosome <7 4 2 5>. This string encodes a potential solution that fills the knapsack with only four items.

This type of representation uses variable-length strings to encode a solution. So how do we handle over- and underspecification? For problems with bounded solution sizes, underspecification is really a non-issue – values for all variables are not required. A potential solution can be tested with fewer than the maximum number of variables. In some cases, a solution with fewer instantiated variables may actually be more fit.

Overspecification due to duplicate tags (same value for the same variable/tag) is also a non-issue since we can simply ignore additional copies. Only when competing genes arise do we have a problem. Competing genes occur when two or more tagged or named genes/bits in the same chromosome have different values associated with them – which one do you choose? In the case the 0/1 Knapsack problem described previously, competing genes do not occur. Representations using tag/value tuples (e.g., see GAs for MAV rule development below) do have competing genes, however, which must be addressed.

The simplest option for selecting between competing genes is to use left-to-right precedence similar to that found in the messyGA. Alternatively, one of several competing genes could be chosen randomly or based on some heuristic. Regardless of the method, only overspecification due to competing genes need be addressed for variable-length GAs applied to problems with bounded solution sizes.

3.2.2.1 GAs for MAV Rule Development

A well-known class of problems with a bounded solution size is the problem of evolving sets of condition/action rules for MAVs or micro air vehicles (Bassett & De Jong, 2000; Wu & Stringer, 2002; Wu, Schultz & Agah, 1999). GAs in these papers evolve rule sets for control and navigation of autonomous vehicles. Existing rules are matched against sensor inputs from a

simulated environment. Best matched rules are executed affecting the movement of the autonomous vehicle.

These problems have a bounded solution size because there is a theoretical maximum number of rules that can be generated assuming one rule for each and every possible condition. For example, each MAV in (Wu & Stringer, 2002) has a total of eight sensors that are in either an "on" or "off" state. A total of 2^8 possible environmental states or conditions can exist. The largest possible solution for this problem would contain 256 condition/action rules, assuming we associate one action with each possible sensor state combination. The number of possible conditions can be thought of as an upper bound on the number of variables for this problem.

However, not every condition requires its own specialized rule to create a valid and highly fit solution. The results of experiments in the above cited works show that a compact set of general rules (between 5 and 20 rules) is often sufficient to provide good results.

Over- and underspecification were not directly addressed by GAs working with this problem. Instead these situations were resolved by matching and rule selection algorithms within the MAV simulator. For underspecification (specific conditions not addressed), the MAV selects the rule that most closely matches the current environment. In the case of overspecification (multiple actions for the same condition), the MAV randomly chooses between competing rules to determine the action to take.

3.2.2.2 SAMUEL System

Another example of a GA incorporating bounded solutions sizes is the *SAMUEL System* (Grefenstette, Ramsey & Schultz, 1990). This system, a type of LCS, uses a variable-length GA at the heart of its learning module to learn rules for sequential decision making. Each individual

in the population represents a set of rules or a "tactical plan" to guide SAMUEL in its tasks. Plans consist of a variable number of rules that are loaded into SAMUEL's performance module for execution.

Rules are expressed in a high-level if/then representation rather than binary strings. Crossover occurs between rules and serves to combine good rules to form a better tactical plan. Creation of new rules is the function of mutation and a new specialization operator that creates a more specialized version of a pre-existing rule. Reduction in the number of rules is implied using operators for generalization and merging.

In (Grefenstette et al., 1990), SAMUEL learns to direct the actions of an airplane in order to avoid an oncoming missile. Actions to direct the planes course and speed are applied based on conditions defined by a suite of six different sensors tracking information about the plane's environment and the missile. Over 25 million possible conditions can occur and represent an upper bound on size of a complete tactical plan (solution). This number is large but does represent some bounding of the size. It should be noted that the authors of this work restricted solutions to ones with 32 or less rules – a number substantially smaller than the upper bound. No explanation is given for this restriction. Over- and underspecification are handled by a condition matching and selection algorithm located in SAMUEL's production system.

3.2.2.3 The Proportional GA Revisited

We add here another mention of the PGA. This representation can also be used for bounded solution sizes. Each possible variable in the solution must be assigned its own character(s) in the PGA's alphabet. If the PGA evolves away a particular character (its not included in the genome), then the variable is not used. The use of the PGA in this manner

assumes that the initial population contains at least one character for each possible variable in the problem. Otherwise, any solution would not be able to sample from the full range of possible solutions. An alternative would be the addition of an operator which could insert new alphabet symbols into the population from a pool of all possible characters/variables.

3.2.3 Representations for Unbounded Solution Sizes

Our third and final category of solutions are those that have unbounded size. These solutions address problems with open-ended complexity. (Harvey, 1992a) argues that genotypes must be unrestricted in length if we are to evolve "a structure with arbitrary and potentially unrestricted capabilities". Open-ended problems like those envisioned by Harvey are often found in artificial intelligence applications where the size and complexity of the solution cannot be determined at the start. The branches of EC most often addressing problems of this type are GP and GE. Both seek to evolve programs without any foreknowledge of their shape, size or complexity.

Variable-length GAs also have the ability to address these same kinds of problems. As an example we will refer to (Kavka & Schoenauer, 2003). In that work, the authors present a new approach to function identification. Rather than seek a global function to encompass the entire function domain, the function space is divided into Voronoi regions. A local approximator function is found which provides a good function approximation within a region. A local vector encodes parameters to identify a single Voronoi region and its local approximator. Each individual in the population is then a variable-length list made up of local vectors. The combination of all local vectors in a single individual covers the entire function space and provides a complete approximation.

The size of potential solutions for this new approach to function identification is unbounded and for two reasons. First, the size of the function domain itself is open ended possibly requiring large numbers of Voronoi regions to achieve a good approximation. Second, the quality of the solution may be dependent on the granularity of the regions themselves. Many small regions could provide a better solution than a few large ones. The size of an individual chromosome therefore must allow for unbounded growth in the number of local vectors in order to handle these two issues.

3.2.4 Fixed-Length GAs with Variable-Length Characteristics

The meaning of the term "variable-length genetic algorithm" is often in the eye of the beholder. Other representations have been cited by researchers as examples of variable-length GAs but these representations do not quite fit the definition we wish to convey in this work.

For example, the *Delta Coding Algorithm* (Mathias & Whitley, 1994) has been mentioned in the literature in conjunction with variable-length GAs. This algorithm begins by executing a standard GA until the population converges to a set of similar solutions. The best individual from this initial population is saved as an interim solution. A new population is them initialized using fewer bits for each variable. Fitness evaluation occurs after the genes from the new individuals are added to/subtracted from those in the interim solution. If a better solution is found, it becomes the new interim solution. This process is repeated for a number of *delta iterations* until the best possible solution is found or some other stopping condition is met.

Changing the number of bits per variable allows the delta coding algorithm to search different hyperspaces for solutions to a single problem. But the delta coding algorithm is really a fixed-position, fixed-length GA that uses different string lengths for all chromosomes in each run

(delta iteration). It does not vary string sizes within a run or allow variable string lengths within a population.

The *Promoter/Terminator Genetic Algorithm* or *ptGA* in (Mayer, 1998) is another example of a fixed length GA which is sometimes mislabeled as variable-length. This GA is designed to test the use of non-coding regions (called introns by the authors). Chromosomes consist of fixed-length bit strings. Individual genes are embedded within each chromosome, identified by tags, and surrounded by promoter and terminator bit sequences. The overall bit length (size) of the chromosome is larger than that needed to hold all genes. The GA evolves both the content of genes as well as their locations on the chromosome (without overlapping reading frames). Duplicate genes as well as non-coding bits are allowed, the amount of which depends on the overall size of the fixed-length chromosome – the larger the chromosome, the more space available for non-coding bits or duplicate genes. Mayer asserts that insertion of non-coding bits within a fixed-length GA accelerates the finding of good solutions.

We should also make mention here of modularity in genetic algorithms. Module acquisition (Angeline & Pollack, 1993) is the process whereby groups of atomic units in a genome are replaced by a single new unit that encapsulates or abstracts the meaning of the original atomic units. Module acquisition serves as a form of "compression" on two or more atomic units creating new types of building blocks. As a result, the original atomic units which make up the module are preserved and protected from disruption by reproduction operators such as crossover and mutation. Modularity can also be an effective aid in solving problems that contain hierarchical or repetitive properties (De Jong & Thierens, 2004).

Module acquisition originated in the GP community but has since become of interest to GA researchers. GAs can incorporate modularity operators to create or expand modules.

However, the use of modules does not automatically imply variable-length genomes. For example, De Jong's DevRep algorithm (E. D. De Jong, 2003) uses a fixed-length list of modules to represent a single solution to a variable-length problem. New modules are created by combining two existing and adjacent modules/units in the genome. These are then replaced with a new module and a null-module thereby preserving the assembly length. De Jong's work shows how variable-length problems (what we equate to unbounded solutions) can be addressed using a fixed-length list of modules using repetitive hierarchical module construction.

It is possible to have a variable-length GA with modularity assuming that no null modules are added to assemblies during module creation. The addition of an expansion operator (replacing a module with its components) would allow a chromosome to increase in size as well as shrink. Despite the fact that modular GAs can be represented using a variable-length genome, we do not include them in our prior list of representations for the following reason – the appearance of variable-length genomes in a modular GA is due to the implementation details of modularity operators rather than a quality inherent to modularity itself.

One final point concerning modularity – most GAs incorporating module acquisition are location dependent. Atomic units and modules are processed sequentially when constructing a potential solution. Position dependence permits easy identification of modules and repetitive atomic units within the genome. In contrast, most variable-length GAs in the literature are designed to relax or totally eliminate position dependence of bits or genes.

3.2.5 Discussion

The previous sections give a good illustration of the range of variable-length representations found within the GA paradigm. We have organized them based on solution size,

but common to all representations is the ability of individual chromosomes within a population to vary by size and to change size from one generation to the next. After that the differences can be summarized as follows:

- Encoding Alphabets – Chromosomes in these examples used a variety of encoding alphabets including bits (messyGA), rule languages (SAMUEL) and multi-character alphabets (proportional GA, VIV). The choice of alphabet does not seem to be a limiting factor in the use of variable-length representations. If anything, these examples show that variable-length representations can adapt to a variety of encoding methods.

- Crossover Base Units – Crossover in the PGA and VIV is allowed between any characters in the string. Crossover is simple and straightforward. Other representations like those used for developing MAV rule sets work with genes as a basic unit for determining crossover points with crossover typically occurring between genes. In these representations, each gene is associated with a given variable or rule in a final solution. "Intra-gene" crossover is possible but requires careful selection of crossover points to ensure that partial genes from each parent can form a valid whole gene after recombination. For gene-based GAs, mutation is the primary vehicle for generating new genetic material while crossover focuses more on new combinations of existing genes.

- Presence/Absence of Non-Coding Regions –VIV incorporates true non-coding regions. The use of START (promoter) and STOP (terminator) codons (tags) causes some information between genes to be ignored completely when decoding the genome. In other representations, this type of "true" non-coding region does not always occur. In the

messyGA, SAMUEL, and MAV rule development systems there can be genes that are not expressed due to overspecification by duplicate or competing genes. We do not consider this type of non-expressed material to be the same as non-coding material. Non-coding material, however, could be added into these systems by including genes/bits with tags or names that do not map to a variable in the problem. A similar approach could be used to add non-coding material into a PGA by adding characters not mapped to solution variables into the PGA's alphabet.

- Position Independence of Bits, Genes or Characters – In the traditional GA, bits or bit substrings are associated with specific problem variables based on position. The first *n* bits are used to determine the value for variable 1, the second *n* bits for variable 2 and so on. The messyGA relaxes that requirement and allows bits to float within a single chromosome. The goal is to improve building block development, evolving both bit values and positions in the genotype. Other variable-length GAs have also relaxed this requirement to varying degrees. Typically, identification tags or names facilitate the mapping between genome and variable.

Our objective for this section has been to establish a framework or loose classification system for variable-length GAs based on solution size. We have identified some of the more important characteristics and differences among these representations. Finally, we have provided examples from the literature which illustrate the ideas discussed herein.

Later, in Chapters 4 through 6, we will look at empirical and theoretical results that show how certain variable-length GAs can shrink in size under random (constant) selection pressure. Our interest in this area was spawned by earlier work on the chunking GA found in (Wu &

Stringer, 2002). We believe this type of size reduction is also applicable (and potentially useful) to other variable-length GAs with the following characteristics:

- bounded or unbounded solution size,
- base crossover units of non-changeable fixed size
- absence of non-coding regions, and
- position independent bits or genes,

We do not believe that our results will be affected by choice of alphabet or base crossover unit (bit or gene) as long as the alphabet remains constant and the size of base units is stable throughout a GA's execution.

3.3 Bloat Control in GA/GP

Most variable-length representations, regardless of the EC paradigm suffer from bloat to one degree or another. In general, bloat is an increase in genome length from one generation to the next, primarily due to growth in extraneous genotypic information. This information is extraneous since it is not necessary to form a valid and/or optimal solution.

For GAs, bloat may include bits which make up non-coding regions, duplicate genes or competing genes. Suggested causes of bloat within a GA include, but are not limited to 1) high mutation rates (Ramsey et al., 1998) and 2) the idea that additional genetic material is "free" (does not directly impact fitness) and provides more space to explore for better genes (Burke et al., 1998). GA researchers have taken a variety of approaches to handling bloat in their works. The following control methods are taken from papers cited previously:

- messyGA: No explicit bloat control mechanism; however, the authors suggest that use of competitive templates discourages duplicate genes.
- VIV: A linear length penalty is added to the fitness function. This penalty alters the initial raw fitness of each chromosome so that shorter individuals are favored assuming all raw fitness values are equal.
- PGA: Simple right truncation is applied after crossover to any child longer than a pre-specified limit. All characters/genes beyond that limit are deleted from the chromosome.
- SAMUEL: The crossover operator removes any duplicate rules from being passed to the same child. Specialization operator cannot create new rules if maximum number of rules per plan already reached.

A last GA bloat control method is more a philosophy than a mechanism – it is the *Species Adaptation Genetic Algorithm* or *SAGA* (Harvey, 1992a). In his work, Harvey argues that variable-length representations using random crossover tend to produce populations with individuals of widely disparate lengths. The result is a GA that must search through an undeterminably large hyperspace for a solution due to variations in genome size. This contrasts with the simpler search through a finite hypercube associated with a GA using fixed-length strings. The hyperspace represents an uncorrelated landscape if size variation is large within a population. The GA must first converge to a population of appropriate sized individuals before it can narrow its search for the optimal solution of that size.

Harvey proposes that genome size within a population be increased gradually over time (e.g., 1 gene per generation) rather than in large jumps. The smaller variations in size over time affords the GA the opportunity to search for solutions in more correlated landscapes as it works its way through a series of larger and larger hypercubes. Incremental size growth is

accomplished using the *SAGA Cross* (Harvey, 1992b). This reproduction operator carefully chooses a crossover point on the second parent that is complimentary to the crossover point chosen on the first parent. The SAGA cross causes a gradual increase in average genome size within a population from one generation to the next, yet allows small variations in individual length within the same generation. It should be noted that Harvey does not discuss a finite limit to the size of a variable-length structure. The SAGA Cross only limits growth to incremental steps until the appropriate size is reached.

Within EC, variable-length representations are most prominent in genetic programming. GP evolves parse trees represented by LISP programs which in turn are encoded as individual chromosomes. GP has been observed to evolve trees or programs that are much larger than necessary, containing one or more sections of unused code (Blickle & Thiele, 1994; Koza, 1992; Soule, Foster & Dickinson, 1996). Of all EC paradigms, GP has produced the largest quantity of bloat-related literature due to its abundant use of variable-length genomes.

Various causes have been espoused to account for increases in the size of GP chromosomes. Among these are theories based on hitchhiking of non-coding regions (Tackett, 1994), defense against crossover (Blickle & Thiele, 1994; Nordin & Banzhaf, 1995), removal bias (Soule & Foster, 1998), and the idea that fitness causes bloat (Langdon & Poli, 1997). These theories are summarized in (Luke, 2003) along with the introduction of a new theory which correlates GP bloat to the depth of modification points within a GP parse tree.

Methods for controlling bloat are equally numerous in the GP literature. Most methods fall into two categories: 1) parsimony pressure methods which incorporate some size penalty into the fitness function and 2) reproduction restrictions which eliminate or prevent children which exceed a given parse tree depth limit. Recent additions to the arsenal of anti-bloat methods

include use of size fair and homologous crossovers (Langdon, 2000), double and proportional tournaments (Luke & Panait, 2002), and the treatment of size and raw fitness as separate objectives in a multi-objective optimization scheme (Panait & Luke, 2004). Panait and Luke offer two additional methods (the Waiting Room, Death by Size) for use with steady-state GP systems.

Parsimony pressure appears to be the bloat control method common to both GAs and GP. Several of the newer GP methods just mentioned could easily be applied to variable-length GAs but to our knowledge have not yet been tried. Bloat control methods which take into account maximal tree depth are specific to GP and are not directly applicable to variable-length GAs.

As shown here, bloat control methods in GP are more numerous, varied, and often more complex than those described in the GA literature. This could be due to GP's popularity and larger body of work relative to that of variable-length GAs, or due to the more complex nature of GP problem representations.

3.4 Effects of Random Selection on Length in GA/GP

Later in this work, we investigate changes in chromosome length under random selection (no selection pressure). No mention of this specific topic was found in our search through the GA literature. The examples found to date have actually come from the GP community. Those examples are summarized below and a discussion of their relevance to GAs under similar conditions is provided.

The first reference concerning random selection on chromosome size was found in (Tackett, 1994). It was observed that under random selection bloating did not occur (Figure 7.1 of that work actually shows a slight reduction in tree size). Tackett theorized that bloat was

proportional to selection pressure – the greater the selection pressure caused by fitness, the faster the increase in chromosome size. It was suggested that bloat might be further strengthened by hitchhiking forces which carry non-coding regions along with useful material into next generation individuals.

Experiments performed in (Langdon & Poli, 1997) confirm some of the Tackett's findings. Bloat is shown to be related to selection pressure. The absence of selection pressure stops the growth of program size. Like Tackett, the authors found a "slow reduction in program size" under random selection. This reduction was attributed to biases in the crossover operator that require children to be less than a pre-specified length restriction.

A series of papers by Langdon, McPhee, Poli and Rowe as well as Stephens and Waelbroeck have culminated in new theory regarding variable-length structures that has direct bearing on our work. A summary follows of publications in that series which are directly relevant.

In (Poli & Langdon, 1997), a new schema theory is proposed for GP. This new theory is specific to GP using two new genetic operators: one-point crossover and point mutation. The new crossover operator works by selecting a point on both parents from a common region. The common region consists of those portions of both parents, starting from the root with the same arity of nodes and related edges. A single point is selected uniformly from edges within this region to serve as the crossover point. Subtrees below this point are then swapped between parents to produce two new children. This type of one-point crossover does not increase the depth of the children with respect to both parents but does encourage mixing of subtrees. The use of this specific form of crossover allowed Poli and Langdon to develop their new schema

theorem which mathematically models "the competitions between programs with different structure and programs with the same structure."

At about the same time, Stephens and Waelbroeck were developing a new exact schema theorem for GAs (Stephens & Waelbroeck, 1997, 1999). The primary contribution of these works was the addition of schema reconstruction via crossover and mutation to Holland's original, more pessimistic schema theorem. Another important finding in (Stephens & Waelbroeck, 1999) is the following:

> "We also showed that, generically, there is no preference for short, low-order schemata. In fact, if schema reconstruction dominates, the opposite is true – typically large schemata will be favored. Only in deceptive problems does it seem that short schemata will be favored, and then only in totally deceptive problems as the system will tend to seek out existing non-deceptive channels."

In this quote, the use of "short" and "large" refer to the defining length of schemata. This finding is important as a possible clue to the cause of bloat in variable-length GAs. Without some bloat control mechanism, such GAs grow rapidly in size during early generations as they work to combine building blocks into more complete and better solutions. The fact that large schemata can improve building block construction may explain why the GA grows so rapidly.

The idea of exact schema was applied to Poli and Langdon's new schema theory for GP in (Poli, 2000) to produce "a macroscopic exact schema theorem for genetic programming with one-point crossover." Included in that work was an example that calculated the total transmission probability and effective fitness for a specific schema in a limited population consisting of functions of singular or 1-arity.

The example just described served as inspiration for three subsequent works: (McPhee & Poli, 2001; Poli & McPhee, 2001; Rowe & McPhee, 2001). In these works, the authors develop an exact schema theory for linear structures (similar to a variable-length GA) consisting solely of

1-arity functions and a single terminal. Results were given for linear structures under both the previously described one-point crossover and standard crossover. In standard crossover, crossover points are chosen on both parents independently – no common region is used. This version of crossover is equivalent to that used in most variable-length GAs. We will not replicate here the details of Poli, McPhee and Rowe's papers but will summarize the points that are most applicable to our current work.

For one-point crossover, assuming an infinite population and constant fitness function (e.g., uniform or random selection), the average size of individuals over time remains constant – a fixed point equal to the average size of the initial population. In addition, the distribution of lengths within the population does not change over time. The distribution remains a fixed point also equal to the distribution of length within the initial population. One-point crossover has no effect on structure length.

The results under standard (GA-like) crossover are different. Again, assuming an infinite population and constant fitness function, the average size from one generation to the next remains constant and fixed at a point equal to the average size of the initial population. The distribution of lengths, however, do change over time. Poli and McPhee show that under constant fitness, shorter-than-average structures are sampled more often than larger ones. Over time, large individuals become larger but fewer in number while shorter individuals shrink but become more numerous. Thus, we see a change in length distribution but no change in mean length from one population to the next. They conclude their work in (Poli & McPhee, 2001) by showing that distribution of length under standard crossover is not a fixed point but a set of fixed points over time defined by a family of discrete gamma functions.

CHAPTER 4: OVER PRODUCTION OF SHORTER-THAN-AVERAGE CHILDREN

The literature survey in Chapter 3 shows that the idea of a variable-length GA or GP with linear structure over-producing shorter than average children is not new. That finding came from probability distributions in works by Poli, McPhee and Rowe as well as our own empirical studies. The unanswered question is how does over production happen? And why does a GA under random selection create shorter than average children at all?

Our approach to answering this question uses two matrices to calculate a probability estimate for the production of shorter than average children in a hypothetical GA population. Both matrices are of size nxn where n is a positive integer indicating chromosome size.

The first matrix denoted PrM contains the probabilities for matings occurring between any two individuals of size j and k where $j,k \leq n$. The sum of all cells in the matrix must equal 1. Since we are concerned with GAs using a random or uniform selection method, each cell in PrM will be of equal value. Specifically the value for each cell $\mathrm{PrM}_{j,k}$ will be

$$\mathrm{PrM}_{j,k} = 1/n^2 \tag{1}$$

assuming a uniform population distribution and uniform selection. Matrix PrM_n would look like the following:

	1	2	3	4		n
1	$1/n^2$	$1/n^2$	$1/n^2$	$1/n^2$	...	$1/n^2$
2	$1/n^2$	$1/n^2$	$1/n^2$	$1/n^2$	...	$1/n^2$
3	$1/n^2$	$1/n^2$	$1/n^2$	$1/n^2$	...	$1/n^2$
4	$1/n^2$	$1/n^2$	$1/n^2$	$1/n^2$	...	$1/n^2$
	...	...	...	...	...	...
n	$1/n^2$	$1/n^2$	$1/n^2$	$1/n^2$	...	$1/n^2$

Figure 2: Expansion of nxn mating probability matrix (PrM_n) for uniform selection.

Our second matrix is of the same size and shape but contains the percentage of possible shorter than average children from a single mating. The contents of each cell in this matrix denoted $pLT_{j,k}$ can be thought of as the percentage of shorter than average children which are produced by a one point crossover applied to two parents of length j and k for all possible crossover points X_j and X_k. The percentage can be calculated by dividing the number of less than average children possible by the total number of different children that could result from each mating.

As an example, assume we have a matrix of size $pLT_{n=5}$. The average size length for this matrix is $(n+1)/2$ or the value 3. To determine the percentage of shorter than average children, we must divide the expected number of shorter offspring by the total offspring in each cell. The expected number of total children in any give cell is the total number of crossover point combinations (X_j and X_k) times 2. In our approach, all matings result in two children. Equation 1 gives a formula for expected number of total children.

$$E_{j,k}(\text{children}) = 2jk \tag{2}$$

The number of shorter than average children is determined by looking at all crossover events possible given parents of length j and k. Figure 3 gives an example of how this is determined for a single cell $pLT_{3,4}$.

The table to the right of $pLT_{n=5}$ shows all possible crossover events for a mating between parents of size j=3 and k=4. Row and column headings indicate all crossover points for both parents. Intersections of crossover points indicate the two children C_l produced from the event where l equals the length of each child. Below the children is a number (or "none") indicating how many of the two children from each crossover event are less than average (<3) in size. For

this example, the total number of less than average children is 6 giving us a probability of 6/24. Thus $pLT_{3,4} = .25$.

$pLT_{n=5}$	1	2	3	4	5
1	2/2	4/4	4/6	4/8	4/10
2	4/4	6/8	6/12	6/16	6/20
3	4/6	6/12	6/18	6/24	6/30
4	4/8	6/16	6/24	6/32	6/40
5	4/10	6/20	6/30	6/40	6/50

	X_k=1	X_k=2	X_k=3	X_k=4
X_j=1	C_3, C_4 none	C_4, C_3 none	C_5, C_2 1	C_6, C_1 1
X_j=2	C_2, C_5 1	C_3, C_4 none	C_4, C_3 none	C_5, C_2 1
X_j=3	C_1, C_6 1	C_2, C_5 1	C_3, C_4 none	C_4, C_3 none

Figure 3: Single expanded cell from a 5x5 less-than percentage matrix ($pLT_{n=5}$) showing number of less than average size offspring possible from mating two parents of length j=3 and k=4.

On a simplistic level one might calculate the probability of producing less than average size children given $pLT_{n=5}$ by summing all shorter than average children then dividing by total possible offspring for the entire matrix. This approach would give us 130 / 450 or .289. However, this approach does not take into account the probability of mating. Assuming uniform mating probabilities, then a single short child produced from a (1,1) mating should weigh more than one produced by a (5,5) mating. For this reason, we must incorporate our first matrix $PrM_{n=5}$ (see Figure 4) into our calculations.

	1	2	3	4	5
1	.04	.04	.04	.04	.04
2	.04	.04	.04	.04	.04
3	.04	.04	.04	.04	.04
4	.04	.04	.04	.04	.04
5	.04	.04	.04	.04	.04

Figure 4: Expansion of $PrM_{n=5}$ mating probability matrix assuming uniform selection.

The following formula gives us a method for combining mating probability and less-than-average percentages to compute a weighted probability for the entire matrix.

$$\mathrm{PrLT_n} = \sum_{k=1}^{n}\sum_{j=1}^{n}\left(\mathrm{pLT}_{j,k} \cdot \mathrm{PrM}_{j,k}\right) \tag{3}$$

Figure 5 shows an expanded version of $\mathrm{PrLT_{n=5}}$ weighted by mating probabilities. Notice that cell (1,1) contributes more to the overall probability of the matrix than most other cells despite having the fewest number of shorter than average children. The total for $\mathrm{PrLT_{n=5}}$ is approximately 44%, much better than an unweighted probability of 28.9%

	1	**2**	**3**	**4**	**5**
1	0.0400	0.0400	0.0267	0.0200	0.0160
2	0.0400	0.0300	0.0200	0.0150	0.0120
3	0.0267	0.0200	0.0133	0.0100	0.0080
4	0.0200	0.0150	0.0100	0.0075	0.0060
5	0.0160	0.0120	0.0080	0.0060	0.0048

Figure 5: Expansion of $\mathrm{PrLT}_{n=5}$ matrix showing each cell's contribution to the overall probability of producing shorter than average children.

Before going on we should discuss how these matrices relate to a GA population. Our *n*x*n* matrices can be thought of as representing one of two different hypothetical populations. One option is to see each matrix as representing a population of *n* individuals of lengths 1 to *n*. Selection is made randomly from this population with replacement. A parent pair of the same length where *j*=*k* would therefore represent a form of asexual reproduction. Alternatively, we can view these matrices as representing a population of two of every size from 1 to *n*. In the first case our population size is *n*, in the second 2*n*. Later in this work, we will talk about how our findings relate to more realistic populations.

We also should discuss the idea of length. Given that both j and k correspond to the lengths of individuals in a GA population what do we really mean by "length". For purposes of this work, we assume these numbers represent crossover points rather than some bit length or unit of a representational alphabet. The use of crossover points as a measure of length allows our results to apply to any variable-length GA regardless of encoding form (e.g., integers, floating point numbers, multi-character alphabets).

Computing the probability for the $\text{PrLT}_{n=5}$ sample matrix was simple due to its size. For larger n's, counting the number of less than average children for each cell in pLT_n becomes more difficult. The remainder of this chapter provides equations for computing the various values needed to determine the less than average probability for any size matrix. We will show that the value of PrLT_n as n grows larger is approximately 55%

Computing PrLT_n requires that for each cell in the matrix we calculate the number of possible shorter than average children denoted as $\text{E}_{j,k}(\text{children}_{l<A})$ where l indicates length and A indicates the average. This number is divided by the total number of possible offspring $\text{E}_{j,k}(\text{children})$ to yield a percentage of crossover events $\text{pLN}_{j,k}$ for some mating between two parents j and k.

$$\text{pLT}_{j,k} = \frac{\text{E}_{j,k}\left(\text{children}_{l<A}\right)}{\text{E}_{j,k}\left(\text{children}\right)} \tag{4}$$

Substituting Equation 4 into Equation 3 gives us

$$\text{PrLT}_{\text{n}} = \sum_{k=1}^{n}\sum_{j=1}^{n}\left(\frac{\text{E}_{j,k}\left(\text{children}_{l<A}\right)}{\text{E}_{j,k}\left(\text{children}\right)} \cdot \text{PrM}_{j,k}\right) \tag{5}$$

In Equation 5, we know in advance how to compute two of the three terms. $E_{j,k}$(children) = 2*jk* from Equation 2. $PrM_{j,k}$ is assumed to always equal $1/n^2$ due to the uniformity of our matrices (or special characteristics of our hypothetical population). This leaves $E_{j,k}(\text{children}_{l<A})$, the number of shorter than average children the only term to finalize.

Unfortunately, there is no continuous function that will provide $E_{j,k}(\text{children}_{l<A})$ for all cells in a matrix. It becomes necessary to divide pLT_n into five regions or areas which each have unique equations for determining the expected number of shorter than average children. Figure 6 below shows how pLT_n must be partitioned.

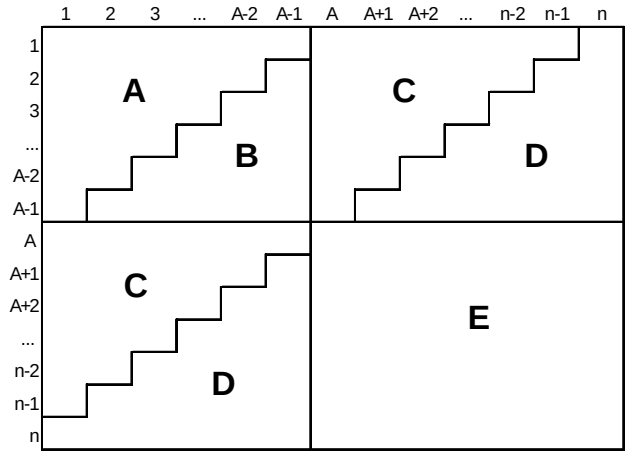

Figure 6: *n*x*n* matrix (pLT_n) divided into areas for determining the number of crossover events producing shorter than average children.

By dividing the matrix in this way, it becomes necessary to replace Equation 5 with a new equation that computes the probability for an entire matrix by summing the probabilities within each area:

$$\begin{aligned} \text{PrLT}_n &= \sum_{area=A}^{E} \text{PrLT}_{area} \\ &= \sum_{area=A}^{E} \sum_{k=1}^{n} \sum_{j=1}^{n} \left(\frac{\text{E}_{j,k \in area}\left(\text{children}_{l<A}\right)}{\text{E}_{j,k \in area}\left(\text{children}\right)} \cdot \text{PrM}_{j,k} \right) \end{aligned} \tag{6}$$

Table 1 summarizes these five areas and gives mathematical definitions for determining each area's boundaries.

Table 1: Definitions and descriptions of crossover results for each area in a partitioned pLT_n population matrix.

Area	Definition	Results of All Crossover Events
A	$j,k <$ Average, $j+k \leq$ Average	All Doubles
B	$j,k <$ Average, $j+k >$ Average	Mostly Doubles Few Singles
C	j *xor* $k <$ Average, $j+k >$ Average and $j+k <$ 2*Average	Mostly Singles Few Doubles
D	j *xor* $k <$ Average $j+k \geq$ 2*Average	Mostly Singles Few with None
E	$j, k \geq$ Average	Mostly with None Few with Singles

Another way to view these areas is in terms of the types of children produced by crossover events within each cell. For example, a mating between two individuals of length j and k in area A will always produce two children that are shorter than the average population size. Each such event is referred to as a "double" – both children are shorter than average. A mating in area E on the other hand cannot produce any doubles. Since both parents are greater than or equal to the average there is not combination of crossover points on j and k which would yield two shorter than average children. Most events in area E produce two longer than average children ("nones") and a few singles. A "single" event occurs when one child is shorter than average and the other child's length is greater than or equal to the average length.

As mentioned previously, each area has its own unique equation for determining the expected number of less than average offspring from all possible crossover events between to

parents of length j and k. These equations were determined from detailed analysis of crossover event behaviors within each area, then confirmed by experiments.

The equation for area A is simple due to the fact that all possible crossover events from any mating result in two shorter than average children (all are doubles). Given that the number of possible crossover events for any mating is $j*k$, we obtain the following:

$$\mathrm{E}_{j,k\in A}\left(\text{children}_{l<A}\right) = 2jk \tag{7}$$

Area B is dominated by matings which produce a high number of crossover events with two shorter than average children; however, matings in this area can also include some crossover events that produce only one shorter than average child. The equation for the expected number of shorter than average children reflects this by subtracting the number of children greater than or equal to the average from all possible crossover events.

$$\begin{aligned}\mathrm{E}_{j,k\in B}\left(\text{children}_{l<A}\right) &= 2jk - \left(2\cdot\sum_{i=1}^{j+k-A} i\right) \\ &= 2jk - \left(2\cdot\frac{(j+k-A)(j+k-A+1)}{2}\right) \\ &= 2jk - (j+k-A)(j+k-A+1)\end{aligned} \tag{8}$$

Matings in area C can potentially produce crossover events with two offspring shorter than average but this is a rare event. Most often mating occurrences are characterized by many single events (one child longer, one shorter). Like Equation 8, the following equation finds the number of children produced by all crossover events then subtracts out the one child for each single event.

$$
\begin{aligned}
\mathrm{E}_{j,k\in C}\left(\text{children}_{l<A}\right) &= 2jk - \left(2 \cdot \sum_{i=j+k-A-j+1}^{j+k-A} i\right) \\
&= 2jk - \left(2 \cdot \sum_{i=k-A+1}^{j+k-A} i\right) \\
&= 2jk - \left(2 \cdot \frac{\left(\left(j+k-A\right)-\left(k-A+1\right)+1\right)\cdot\left(\left(j+k-A\right)+\left(k-A+1\right)\right)}{2}\right) \\
&= 2jk - \left(j^2 + 2jk - 2jA + j\right) \\
&= 2jA - j^2 - j
\end{aligned}
\tag{9}
$$

Area D contains matings between individuals whose combined genetic material is too large for production of two under-average children. This area is dominated by matings with crossover events that produce a single below average size child. There are even a few crossover events for some matings which produce only children of length larger than the average. The equation for the number of less than average children in Area D is:

$$
\begin{aligned}
\mathrm{E}_{j,k\in D}\left(\text{children}_{l<A}\right) &= 2 \cdot \sum_{i=A-j}^{A-1} i \\
&= 2 \cdot \frac{\left(\left(A-1\right)-\left(A-j\right)+1\right)\cdot\left(\left(A-1\right)+\left(A-j\right)\right)}{2} \\
&= 2jA - j^2 - j
\end{aligned}
\tag{10}
$$

Notice that the equations for areas C and D simplify to the same expression. As a result, we will treat Areas C and D as a single area "C|D" going forward.

Last is area E which contains few singles and is primarily characterized by matings with crossover events which produce no shorter than average children – the length of the parents is just too long. The following equation gives the formula for the few shorter than average children expected from any mating in this area:

$$\begin{aligned} \mathrm{E}_{j,k\in E}\left(\text{children}_{l<A}\right) &= 2\cdot\sum_{i=1}^{A-1} i \\ &= 2\cdot\frac{(A-1)(A-1+1)}{2} \\ &= A^2 - A \end{aligned} \tag{11}$$

Referring back to Equation 4 we can compute the percentage of shorter than average offspring for any give cell in an area by dividing Equations 7, 8, 9, 10 or 11 by 2*jk* from Equation 2. We can then calculate the probability for production of shorter than average children for each cell within an area by multiplying the percentage by the probability of two parents mating from PrM_n. The next series of equations takes these steps to calculate the contribution each area makes to PrLT_n. Note that for each area, initial and terminating conditions for summations terms vary in order to match the borders defined for each in Table 1. Due to its complexity we leave solving for area B's contribution for last.

The contribution of area A to the overall matrix probability PrLT_n is:

$$\begin{aligned} \mathrm{PrLT}_A &= \sum_{k=1}^{A-1}\sum_{j=1}^{A-k}\left(\mathrm{pLT}_{j,k\in A}\cdot\mathrm{PrM}_{j,k}\right) \\ &= \sum_{k=1}^{A-1}\sum_{j=1}^{A-k}\left(\frac{\mathrm{E}_{j,k\in A}\left(\text{children}_{l<A}\right)}{\mathrm{E}_{j,k}\left(\text{children}\right)}\cdot\mathrm{PrM}_{j,k}\right) \\ &= \sum_{k=1}^{A-1}\sum_{j=1}^{A-k}\left(\frac{2jk}{2jk}\cdot\mathrm{PrM}_{j,k}\right) \end{aligned} \tag{12}$$

Given our hypothetical population, we have a uniform probability of selecting any two parents for mating. As a result the value for $\mathrm{PrM}_{j,k}$ is constant at $1/n^2$. Since n is twice the average we will use $1/(2A)^2$ where A is the average to help simplify our calculations. The fact that this value is constant for all cells in $\mathrm{PrM}_{j,k}$ also allows us to move this term from inside our summations to yield:

$$
\begin{aligned}
\mathrm{PrLT}_A &= \frac{1}{4A^2} \cdot \sum_{k=1}^{A-1} \sum_{j=1}^{A-k} \frac{2jk}{2jk} \\
&= \frac{1}{4A^2} \cdot \sum_{k=1}^{A-1} \sum_{j=1}^{A-k} 1 \\
&= \frac{1}{4A^2} \cdot \sum_{k=1}^{A-1} (A-k) \\
&= \frac{1}{4A^2} \cdot \left(\sum_{k=1}^{A-1} A - \sum_{k=1}^{A-1} k \right) \\
&= \frac{1}{4A^2} \cdot \left(A \cdot (A-1) - \left(\frac{(A-1)(A-1+1)}{2} \right) \right) \\
&= \frac{1}{4A^2} \cdot \frac{A^2 - A}{2} \\
&= \frac{1}{8} \cdot \frac{A^2 - A}{A^2}
\end{aligned}
\qquad (12) \text{ (continued)}
$$

The limit for Equation 12 as A approaches infinity is 1/8 or .125 Thus the overall contribution of cells in area A towards production of short than average children is 12.5%.

A similar process is followed to determine the contributions of areas C, D and E. Remembering that areas C and D are equivalent with respect to the expected number of less than average offspring we combine the calculation for both areas in Equation 13.

Regarding $\mathrm{PrLT}_{C|D}$ in Equation 13, note that the summation term is multiplied by 2. We actually calculate only the values in one quadrant of the matrix then multiply by two. This is possible due to the symmetry of the matrix. One other note regarding Equations 13 and 14, solutions for both of these equations require the introduction of the term H_n or H_A. Both terms indicate a Harmonic Series ($1/1 + 1/2 + 1/3 + \ldots + 1/n$). This series can be approximated by the natural log of n plus Euler's Constant or $\ln(n) + .57721$ (Euler's Constant to five decimal places).

$$
\begin{aligned}
\mathrm{PrLT}_{C|D} &= 2 \cdot \sum_{k=A}^{n} \sum_{j=1}^{A-1} \left(\mathrm{pLT}_{j,k \in C|D} \cdot \mathrm{PrM}_{j,k} \right) \\
&= 2 \cdot \sum_{k=A}^{n} \sum_{j=1}^{A-1} \left(\frac{\mathrm{E}_{j,k \in C|D}\left(\mathrm{children}_{l<A}\right)}{\mathrm{E}_{j,k}\left(\mathrm{children}\right)} \cdot \mathrm{PrM}_{j,k} \right) \\
&= \frac{2}{4A^2} \cdot \sum_{k=A}^{n} \sum_{j=1}^{A-1} \frac{2jk - j^2 - j}{2jk} \\
&= \frac{1}{2A^2} \cdot \sum_{k=A}^{n} \sum_{j=1}^{A-1} \frac{2k - j - 1}{2k} \\
&= \frac{1}{2A^2} \cdot \sum_{k=A}^{n} \frac{1}{2k} \cdot \left(\sum_{j=1}^{A-1} 2A - \sum_{j=1}^{A-1} j - \sum_{j=1}^{A-1} 1 \right) \\
&= \frac{1}{2A^2} \cdot \sum_{k=A}^{n} \frac{1}{2k} \cdot \left(2A(A-1) - \frac{(A-1)(A-1+1)}{2} - (A-1) \right) \\
&= \frac{1}{4A^2} \cdot \sum_{k=A}^{n} \frac{1}{k} \cdot \left(\frac{3A^2 - 5A + 2}{2} \right) \\
&= \frac{1}{4A^2} \cdot \frac{3A^2 - 5A + 2}{2} \cdot \sum_{k=A}^{n} \frac{1}{k} \\
&= \frac{3A^2 - 5A + 2}{8A^2} \cdot \left(\mathrm{H}_n - \mathrm{H}_{A-1} \right) \\
&\approx \frac{3A^2 - 5A + 2}{8A^2} \cdot \left(\mathrm{H}_n - \mathrm{H}_{\frac{n}{2}} \right) \\
&\approx \frac{3A^2 - 5A + 2}{8A^2} \cdot \left(\left(\ln(n) + .5772 \right) - \left(\ln(n/2) + .5772 \right) \right) \\
&\approx \frac{3A^2 - 5A + 2}{8A^2} \cdot \ln(2)
\end{aligned}
\tag{13}
$$

The limit for Equation 13 as A approaches infinity is 3/8*ln(2) or .375*.693147. Thus the overall contribution of cells in Area C|D towards production of short than average children is approximately 26%.

$$
\begin{aligned}
\mathrm{PrLT}_E &= \sum_{k=A}^{n}\sum_{j=A}^{n}\left(\mathrm{pLT}_{j,k\in E}\cdot \mathrm{PrM}_{j,k}\right) \\
&\sum_{k=A}^{n}\sum_{j=A}^{n1}\left(\frac{\mathrm{E}_{j,k\in E}\left(\mathrm{children}_{l<A}\right)}{\mathrm{E}_{j,k}\left(\mathrm{children}\right)}\cdot \mathrm{PrM}_{j,k}\right) \\
&= \frac{1}{4A^2}\cdot\sum_{k=A}^{n}\sum_{j=A}^{n}\frac{A^2-A}{2jk} \\
&= \frac{A^2-A}{8A^2}\cdot\sum_{k=A}^{n}\sum_{j=A}^{n}\frac{1}{jk} \\
&= \frac{A^2-A}{8A^2}\cdot\sum_{k=A}^{n}\frac{1}{k}\cdot\sum_{j=A}^{n}\frac{1}{j} \\
&= \frac{A^2-A}{8A^2}\cdot\left(\mathrm{H}_n-\mathrm{H}_{A-1}\right)\cdot\left(\mathrm{H}_n-\mathrm{H}_{A-1}\right) \\
&\approx \frac{A^2-A}{8A^2}\cdot\ln(2)\cdot\ln(2) \\
&\approx \frac{\left(\ln(2)\right)^2}{8}\cdot\frac{A^2-A}{A^2}
\end{aligned}
\tag{14}
$$

The limit for Equation 14 as A approaches infinity is $1/8*(\ln(2))^2$ or .125*.48045. Thus, the overall contribution of cells in area E towards production of short than average children is approximately 6%.

The previous computations leave only area B. As mentioned earlier, this computation was reserved for last due to its complexity. Calculating the contribution of area B to the overall matrix's production of shorter than average children would give us the following equation:

$$
\begin{aligned}
\mathrm{PrLT}_B &= \sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-k}\left(\mathrm{pLT}_{j,k\in B}\cdot \mathrm{PrM}_{j,k}\right) \\
&= \sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-k}\left(\frac{\mathrm{E}_{j,k\in B}\left(\mathrm{children}_{l<A}\right)}{\mathrm{E}_{j,k}\left(\mathrm{children}\right)}\cdot \mathrm{PrM}_{j,k}\right) \\
&= \frac{1}{4A^2}\cdot\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-k}\frac{2jk-\left(j+k-A\right)\left(j+k-A+1\right)}{2jk}
\end{aligned}
\tag{15}
$$

Unfortunately the use of k in the initial condition for the second summation introduces a Harmonic series which cannot be easily approximated or for which there is no know substitution. As a result, we cannot compute contributions on a cell by cell basis. Instead, we must calculate a contribution for the entire area. This approach is slightly less accurate since it weights all matings within area B equally but yields a result that is acceptably close to the simulations we will run later.

In order to determine the contribution of the entire B area, we use the following formula:

$$\begin{aligned} \mathrm{PrLT}_B &= \mathrm{pLT}_B \cdot \mathrm{PrM}_B \\ &= \left(\frac{\mathrm{E_B}\left(\mathrm{children}_{l<A}\right)}{\mathrm{E_B}\left(\mathrm{children}\right)} \right) \cdot \left(\frac{\text{Number of Matings in B}}{\text{Number of Matings in M}} \right) \end{aligned} \tag{16}$$

Each of the right hand terms are calculated below.

$$\mathrm{E_B}\left(\text{children}_{l<A}\right) = \sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}\left(2jk-\left(j+k-A\right)\left(j+k-A+1\right)\right) \tag{17}$$

$$= \sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}\left(-j^2+\left(2A-1\right)j-k^2+\left(2A-1\right)k-\left(A^2-A\right)\right)$$

$$= -\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}j^2\right)+\left(2A-1\right)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}j\right)-\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}k^2\right)+$$

$$\left(2A-1\right)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}k\right)-\left(A^2-A\right)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}1\right)$$

See Appendix for derivations of above terms.

$$= -\frac{9A^4-30A^3+27A^2-6A}{36}+\frac{24A^4-84A^3+84A^2-24A}{36}-$$

$$\frac{9A^4-30A^3+27A^2-6A}{36}+\frac{24A^4-84A^3+84A^2-24A}{36}-$$

$$\frac{18A^4-72A^3+90A^2-36A}{36}$$

$$= \frac{12A^4-36A^3+24A^2}{36}$$

$$
\begin{aligned}
\mathrm{E_B}(\text{children}) &= \sum_{k=2}^{A-1} \sum_{j=A-k+1}^{A-1} 2jk \\
&= \sum_{k=2}^{A-1} k \cdot \sum_{j=A-k+1}^{A-1} 2j \\
&= \sum_{k=2}^{A-1} k \cdot 2 \cdot \left(\frac{\big((A-1)-(A-k+1)-1\big)\big((A-1)+(A-k+1)\big)}{2} \right) \\
&= \sum_{k=2}^{A-1} k \cdot \left(2Ak - k^2 - 2A + k\right) \\
&= \sum_{k=2}^{A-1} \left(2Ak^2 - k^3 - 2A + k^2\right) \\
&= \sum_{k=2}^{A-1} \left((2A+1)k^2 - k^3 - 2Ak\right) \\
&= (2A+1) \cdot \sum_{k=2}^{A-1} k^2 - \sum_{k=2}^{A-1} k^3 - 2A \cdot \sum_{k=2}^{A-1} k \\
&= \left((2A+1) \cdot \sum_{k=1}^{A-1} k^2 - \sum_{k=1}^{A-1} k^3 - 2A \cdot \sum_{k=1}^{A-1} k \right) - \big((2A+1) - 1 - 2A\big) \\
&= (2A+1) \cdot \frac{(A-1)(A)(2A-1)}{6} - \left(\frac{(A-1)(A)}{2} \right)^2 - 2A \cdot \frac{(A-1)(A)}{2} \\
&= \frac{5A^4 - 14A^3 + 7A^2 + 2A}{12}
\end{aligned}
\tag{18}
$$

$$
\begin{aligned}
\text{Number of Matings in B} &= \sum_{k=2}^{A-1} \sum_{j=A-k+1}^{A-1} 1 \\
&= \sum_{k=2}^{A-1} (k-1) \\
&= \sum_{k=2}^{A-1} k - \sum_{k=2}^{A-1} 1 \\
&= \frac{\big((A-1)-2+1\big)(A-1+2)}{2} - (A-1-2+1) \\
&= \frac{A^2 - 3A + 2}{2}
\end{aligned}
\tag{19}
$$

$$\begin{aligned}\text{Number of Matings in M} &= (2A)^2 \\ &= 4A^2\end{aligned} \tag{20}$$

We now substitute Equations 17, 18, 19 and 20 into Equation 16 to find the contribution of area B.

$$\begin{aligned}\text{PrLT}_B &= \text{pLT}_B \cdot \text{PrM}_B \\ &= \left(\frac{\text{E}_\text{B}(\text{children}_{l<A})}{\text{E}_\text{B}(\text{children})}\right) \cdot \left(\frac{\text{Number of Matings in B}}{\text{Number of Matings in M}}\right)\end{aligned} \tag{21}$$

$$= \frac{\frac{12A^4 - 36A^3 + 24A^2}{36}}{\frac{5A^4 - 14A^3 + 7A^2 + 2A}{12}} \cdot \frac{\frac{A^2 - 3A + 2}{2}}{4A^2}$$

$$= \frac{4A^4 - 12A^3 + 8A^2}{5A^4 - 14A^3 + 7A^2 + 2A} \cdot \frac{A^2 - 3A + 2}{8A^2}$$

$$= \frac{4A^6 - 24A^5 + 52A^4 - 48A^3 + 24A^2}{40A^6 - 112A^5 + 56A^4 + 16A^3}$$

The limit for Equation 21 as A approaches infinity is 4/40 or .10. Thus, the overall contribution of all cells in area B towards production of short than average children is approximately 10%.

4.1 Experimental Confirmation

An experiment was conducted to determine if the equations found for estimating the contributions of each area to PrLT_n were valid. The experiment consisted of a number of nested loops which enumerated all possible crossover events for all possible matings for two large *n*x*n*

matrices. The children from this enumeration were counted and weighted with a $1/n^2$ mating probability. The results from the experiment are contained in Table 2.

Table 2: Probabilities of producing shorter than average children for each partition of a PrLT matrix. Includes estimated values as well as values from enumerative experiments.

Area of PrLT	Estimates from Equations	Enumeration Test n=499, Avg = 250	Enumeration Test n=500, Avg = 250.5
Area A	.125000	.124999	.124500
Area B	.100000	.105883	.105460
Area C	n/a	.158879	.15824
Area D	n/a	.100731	.101073
Area C|D	.259930	.259610	.259317
Area E	.060056	.060229	.060335
All Areas	.544986	.550723	.549614

The experimental results in column three (Avg = 250) closely mirror the expected values from our calculations. The major difference is due to the imprecise nature for estimating area B. For B, the probability of less than average children was computed for the area as a whole rather than the sum of contributions for each individual cell.

One other item to mention is a slight deviation depending on whether or not the average is an integer or real value. Column four shows the experimental results from a test with an average that is not an integer. In such cases, the probabilities are slightly more or less than those for populations/matrices with integer averages.

Our equations also confirm other experiments previously published in (Stringer & Wu, 2004). Figure 7 below is taken from that work and shows how the probability of less than average children approaches 55% as the size of *n* increases.

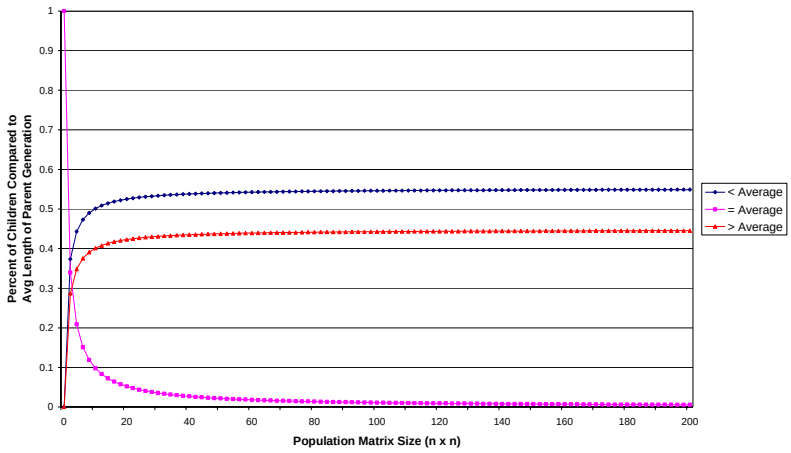

Figure 7: Percentages of children with size less than, equal to, or greater than the average size of their parent generation for an *n*x*n* probability matrices varying in size from *n*=1 to 200.

The experimental method used to produce Figure 7 actually tests 200 different matrices as *n* increases from 1 to 200. The average length in each matrix grows uniformly during program execution. A key finding here is that over-production of shorter than average children does not occur in populations with very small chromosomes (e.g., $n<20$).

A 0.55 probability that any child produced by our hypothetical GA will be shorter than average may not seem like much, but it may be enough given the GA's ability to compound the effect over time. To illustrate we run a simulation of a GA over multiple generations.

The simulation calculates the probabilities for individuals of sizes 1 to *n* over time using the following formula where Pr*(s,t*+1*)* is the probability of an individual of size *s* occurring in the population at time *t*+1 given known probabilities for selecting parents from the current generation of lengths *j* and *k* denoted as Pr(*j*,*t*) and Pr(*k*,*t*) respectively.

$$\Pr(s,t+1) = \sum_{k=1}^{n}\sum_{j=1}^{n}\left(\frac{\sum_{Xk=1}^{k}\sum_{Xj=1}^{j}\begin{cases}1 \text{ if } Xk+Kj=s\\ \text{else } 0\end{cases}}{2jk}\right)\cdot \Pr(j,t)\cdot \Pr(k,t) \qquad (22)$$

A computer program was written to apply the above computation for all sizes s, $1 \leq s \leq n$ for $n = 200$. The simulation was initialized using a uniform probability distribution of length values from 1 to 200. Children of length greater than 200 were right truncated (capped at 200). Results are shown in Figure 8.

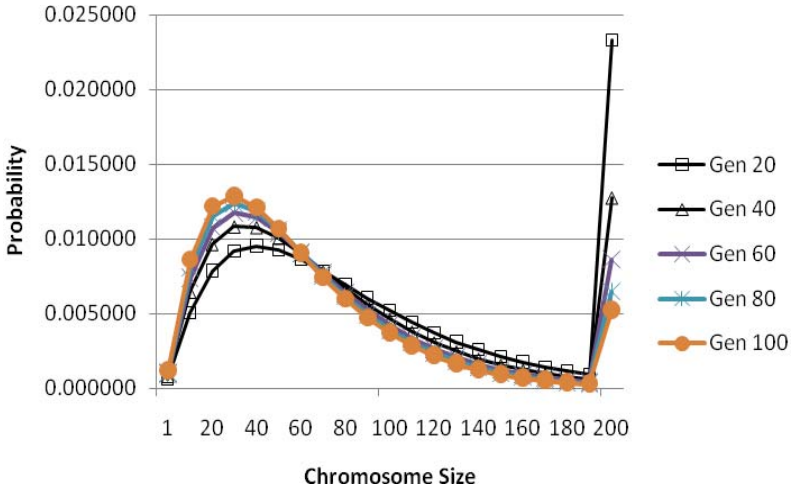

Figure 8: Probability of size distributions over time for a 200 member GA with an initial population uniformly distributed with respect to size. Any children of length > 200 created by crossover events are right trunctated.

The graph in Figure 8 shows how the probability of selecting shorter and shorter individuals increases over time. Plots of each generation are reminiscent of the gamma distributions described in previously cited works by Poli, McPhee and Rowe. The uptick at the end of these curves is caused by the truncation of overly long children to the maximum size of 200.

The next two figures show what happens to probability distributions if other methods for handling overly large children are used. In Figure 9, offspring larger than the size cap are set to the 1/2 the cap or $n/2$. In Figure 10, these children are reset to a random size between 1 and n.

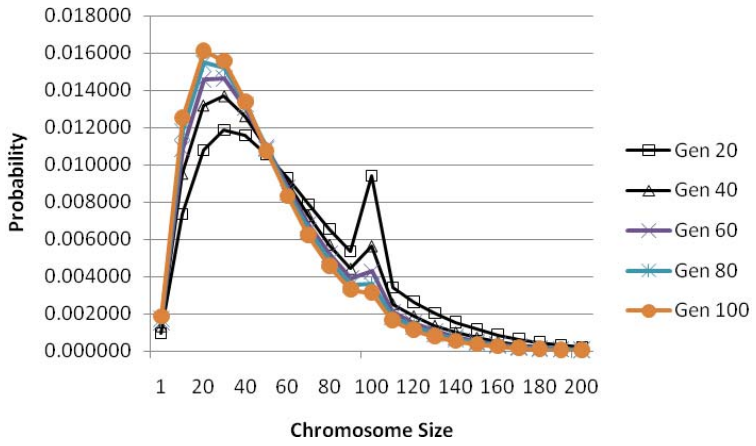

Figure 9: Probability of size distributions over time for a 200 member GA with an initial population uniformly distributed with respect to size. Any children of length > 200 created by crossover events are reset to a length of 100.

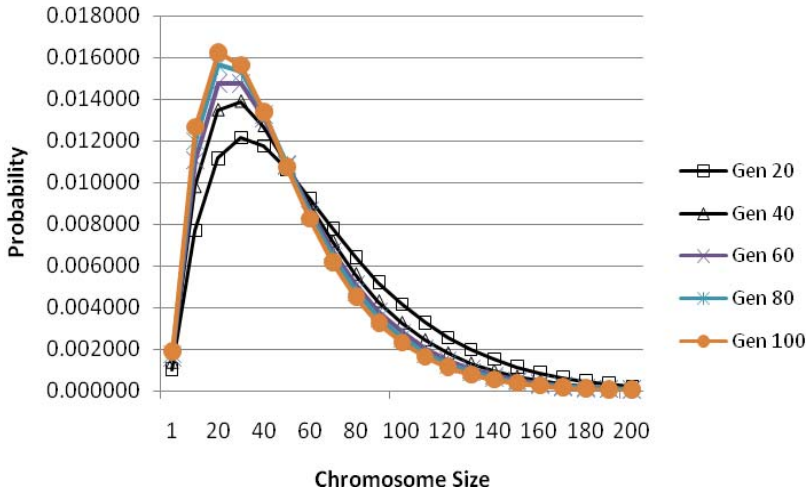

Figure 10: Probability of size distributions over time for a 200 member GA with an initial population uniformly distributed with respect to size. Any children of length > 200 created by crossover events are reset to a random length.

4.2 Discussion

The values listed in Table 2 reflect the probability that any child within a given area will be shorter than average for the entire matrix. Looking at these from a different perspective we

can see how important each area is relative to one another by determining the percentage of probability.

Area	$PrLT_{Area}$	% of Less Than Probability
A	0.12500	22.94%
B	0.10000	18.35%
C	0.15888	29.15%
D	0.10073	18.48%
E	0.06006	11.02%

Table 3: Percentage of probability for an area to produce shorter than average children.

For example, 29.14% of the opportunity to produce shorter than average children lies with matings that occur in Area C. If a GA never allows matings in this area due to either selection, fitness or some other population characteristic, then chances are a GA will never evolve shorter mean population size lengths over time. The following scenario serves to illustrate this point.

Assume we have some real-world variable-length GA which caps the length of chromosomes at 100. Over time chromosome sizes increase due to bloat and stabilize in the upper end of the size range with an average of 90. Overlaying the probability of selection with a partitioned $pLT_{n=180}$ matrix for the GA at this point in the GAs run might look something like the following:

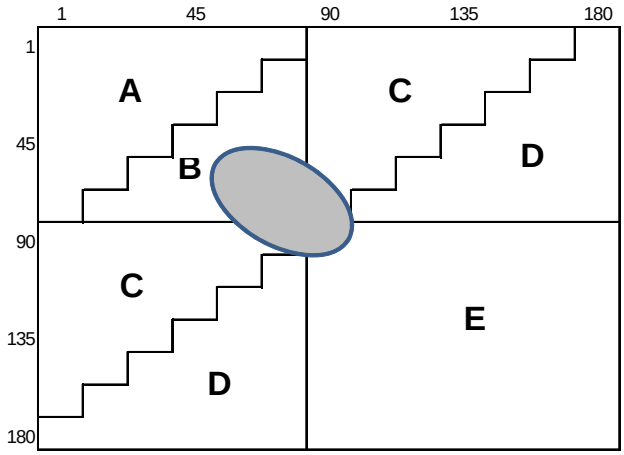

Figure 11: Overlay of GA population with only large sized chromosomes on top of a $pLT_{n=180}$ matrix. Shaded oval indicates where the highest probability of selection lies for the population.

Remember that the center of the matrix is the average size of the population, not 1/2 times the largest individual. As a result, a pLT matrix must be recalculated with each generation as the average size of the population changes.

The important idea from Figure 11 is that matings will only occur near the middle of the matrix (the shaded oval). There is little or no opportunity for mating to occur in areas A or C unless some odd individuals in the population lie outside the shaded oval. Areas A and C combined account for over half the opportunity of producing shorter than average children. Therefore, a GA in this situation will never reduce its overall length and will continue to create only larger size children from one generation to the next.

Part of the motivation for our research into the behavior of GAs under random selection was the hope of finding new ways to combat bloat in variable-length genetic algorithms. We believe the mathematical and empirical findings in this chapter have opened up many new avenues to explore. One can think of our matrices (PrM_n, pLT_n and $PrLT_n$) as road maps to new crossover operators that fight bloat.

We mentioned earlier that some GA researchers use parsimony pressure (factoring length into an individual's fitness) as one method for treating bloat. The problem with this approach is that if length is weighted too heavily, size of the chromosome becomes more important than the quality of the solution it represents.

An alternative approach would be to base selection of mating pairs based on the partition map for pLT_n found in Figure 6. For example, select the first parent at random from the current generation. Now choose the second parent in such a way as to increase the likelihood of producing shorter than average children. The simple approach would be to always choose a parent from area A, but assuming shorter individuals are not always the fittest, this approach would sacrifice quality for shorter children. Instead, select an individual from a region that increases the probability of shorter offspring but does not deterministically reduce the size of children relative to their parents. Some sample matches based on parents P1 and P2 might include:

- if P1 < .25*A* then choose a P2 such that mating is in area D or E),
- if .25*A* < P1 < .5*A* then choose a P2 such that mating is in area C or D),
- if .5*A* < P1 < .75*A* then choose a P2 such that mating is in area C or B),
- if P1 > .75*A* then choose a P2 such that mating is in area A or C)

This approach might be thought of as a kinder, gentler form of parsimony pressure.

Another alternative is to build a selection matrix after each generation *t* to determine the most productive crossovers for generation *t*+1. The selection matrix would combine the fitness of chromosomes with their potential to produce shorter than average children. For example calculate the percentage of less than average crossovers possible for all matings (generate a pLT_n matrix). Then fill the selection matrix using the following formula:

$$S_{j,k} = c_f \cdot (F(j) + F(k)) \cdot pLT_{j,k} \quad (23)$$

F represents the fitness function and c_f is a constant that can be used to adjust the weighting of fitness compared to $pLT_{j,k}$. After construction, the GA could parse through matrix S picking those matings which have the best combined fitness yet still have the highest probability of producing shorter than average children. This type of approach to selection changes the concept from choosing two good parents to picking the best match. The selection operator acts as a "match maker" for the population.

There are many other new operators or selection methods that can be created in light of this chapter's findings. Those mentioned above are just two examples with testing of these selection methods left for future study. They do, however, illustrate the idea of using selection strategies that attempt to balance size (probability of producing shorter children) with quality of solutions (assuming longer is better).

Another interesting finding in this chapter is related to truncation operators. Figures Figure 8, 9 and 10 show how different forms of truncation can affect probability distributions over time. If we plot the average population sizes in these same runs we obtain Figure 12. The graph in that figure serves to illustrate how a type of truncation operator might affect (slow down vs. speed up) reduction in mean population length.

Notice that right truncation definitely slows down reduction in mean population size. Resetting chromosomes whose length exceed some size cap to a smaller number randomly or deterministically (e.g., always 100) gives better results in terms of reduction speed. There is little difference between the two. Of course resetting size may result in poorer fitness for the resulting children than straight truncation.

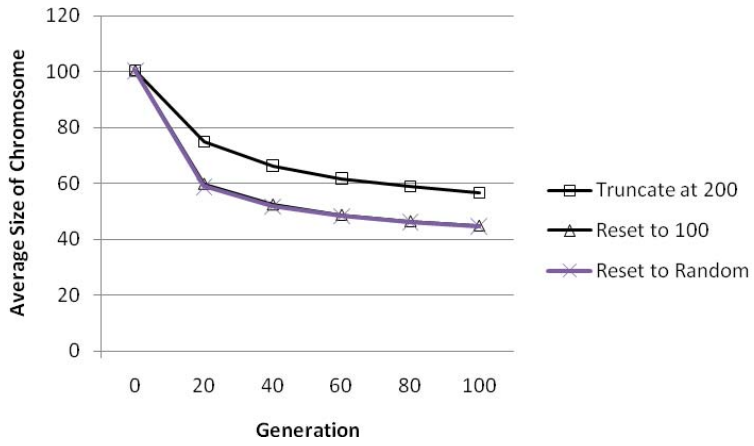

Figure 12: Effect of truncation method applied to chromosomes over 200 in length on average chromosome sizes over time. Data from probability distribution models using a 200 member GA with an initial population uniformly distributed with respect to size. Any children of length > 200 created by crossover events are right truncated, reset to a length of 100 or resent to random length.

One last point about randomly resetting lengths for chromosomes whose size exceeds the cap – this approach, while maybe not faster than using a specific number, does smooth out any bumps or upticks in the probability distribution curve (refer back to Figure 10). Of the three plots, this is the one that most closely resembles the types of curves seen with infinite population models.

CHAPTER 5: INCREASE IN SIZE DIVERSITY

This chapter explores the idea that a GA with random selection of parents creates a diverse population in terms of size. The motivation for this chapter is found in (Stringer & Wu, 2004). In that work, it was shown that reduction in a finite population's mean length did not occur until a large variation in chromosome size appeared within the population.

Figure 13 is taken from that paper and makes this more clear. The probability for production of shorter than average children was calculated for a series of matrices. In the test, the average size of the matrix was kept at 100. The variance was increased over time from 0 to 200. This method can be thought of as a *m*x*n* matrix expanding out from a fixed center point acting as the average. The values of *m* and *n* change over time – *m* equal to the average minus the variance, *n* equal to the average plus the variance.

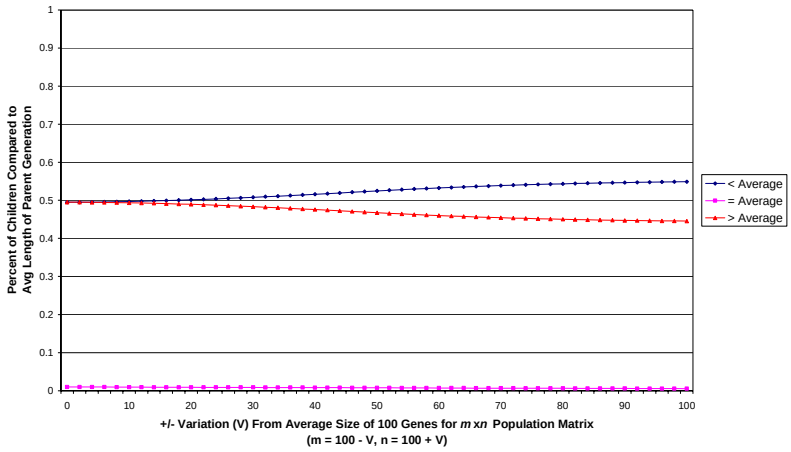

Figure 13: Effect of changes in variance (given a fixed average) on percentages of children whose size is less than the average size of the parent population for a *m*x*n* matrix.

Figure 13 clearly shows that as the variation in size increases so does the probability of creating shorter than average children. Note, however, that the growth is not infinite. At the end

of the experiment (200x200), the percentage of lesser children reaches 54.9% – close to the 55% probability derived in Chapter 4.

Figure 11 from the previous chapter may illustrate even better why a diverse range of chromosome sizes is needed in a population. A very small variation centered around the average does not allow any matings in areas with high probabilities of producing shorter than average children. It is not until the variation gets large enough that matings in areas A and C begin to have an effect on mean population size.

Based on our findings in Chapter 4 and empirical evidence from past works we now know that a wide variation in the population size is a necessary condition for the production of shorter than average children. GAs under random selection produce shorter than average children so they must possess some property that ensures a variety of sizes within a population. The remainder of this chapter determines how size diversity is maintained by a GA under random selection.

Our analysis begins by defining the different possible types of crossover events that can take place during a GA's reproduction cycle. Assuming one-point crossover, there are three possibilities where the size of offspring is concerned. A crossover event can produce 1) children of size equal to the parents (*equal* event); 2) one child longer than both parents and one child shorter than the shortest parent (*outside* event); or 3) both children shorter than the longest parent and longer than the shortest parent (*inside* event). Figure 14 illustrates these three distinct events.

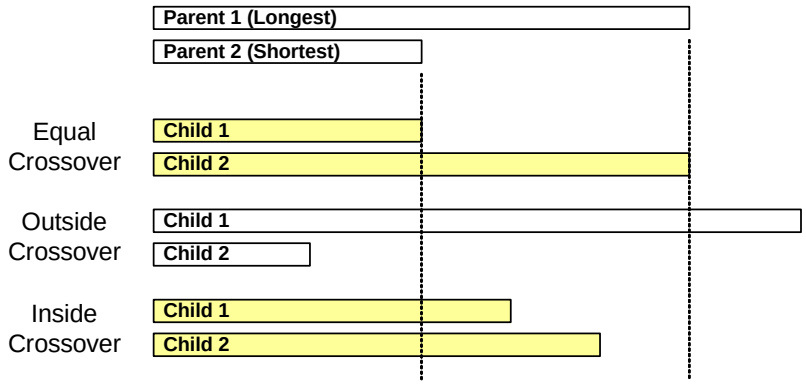

Figure 14: Illustration of crossover event categories for reproduction in variable-length GAs.

For the mating of any two parents, the total number of possible crossover events (regardless of type) can be computed using the following formula where the variables j and k are the number of crossover points contained in each parent.

$$\text{Crossovers}_{j,k} = jk \tag{24}$$

We now wish to determine the number of crossover events out of Equation 24 that fall into each of the three crossover type categories previously described. To do this, we look at two different possibilities. In the first possibility, the number of potential crossover points for both parents are equal ($j=k$). Alternatively, one parent (k) can have more crossover points that the other parent (j) due to differences in length and $j<k$. Table 4 gives the formulas used to determine the number of crossover types for both of these possibilities.

Table 4:
Formulas for computing quantity of possible crossover types resulting from 1-point crossover for any two individuals with j and k crossover points respectively.

	For $j=k$	**For $j<k$**
Equals events	j	$2j$
Inside events	0	$jk-j^2-j$
Outside events	j^2-j	j^2-j
Total events	j^2	jk

The percentage of possible crossover event types for a specific set of parents can be determined by using the formulas in Table 4 and dividing by Equation 24. For example, assume two parents with 5 and 12 crossover points respectively. The percentages of equal, inside and outside crossover events possible for these two parents are 16.67% (10/60), 50% (30/60), and 33.33% (20/60) respectively.

How do we calculate percentages of crossover types if j and k are not known in advance as is the case with a variable-length GA? More importantly, how can we predict percentages across an entire population rather than for two individuals? We turn again to our nxn matrix representing a hypothetical GA with a population of uniformly distributed lengths from 1 to n. We use the notation L_i where i is the number of crossover points to indicate the size of a given row or column. Each column or row corresponds to the length of an individual in our hypothetical population (see Figure 15, left).

	L_1	L_2	L_3	L_4	...	L_n
L_1	1	2	3	4	...	n
L_2	2	4	6	**8**	...	$2n$
L_3	3	6	9	12	...	$3n$
L_4	4	8	12	16	...	...
...	...	...	...	...	...	...
L_n	n	$2n$	$3n$	...		n^2

	X_1	**X_2**	**X_3**	**X_4**
X_1	**C_2, C_4 (E)**	**C_3, C_3 (I)**	**C_4, C_2 (E)**	**C_5, C_1 (O)**
X_2	**C_5, C_1 (O)**	**C_4, C_2 (E)**	**C_3, C_3 (I)**	**C_2, C_4 (E)**

Figure 15: Expansion of nxn population matrix cell to show length of children created by each possible crossover event. The number of rows and columns in the expanded matrix is determined by the number of crossover points in the parent genomes.

The number of possible crossover events for any given mating is the multiplication of the number of crossover points in each of the parents. For example, a total of 8 unique crossover

events is possible between a single mating of two parents, one with four crossover points and the other with two. We can fill each cell in this matrix with the number of possible crossover events (regardless of type) for two chromosomes of size j and k where $j,k \leq n$ by simply multiplying the row and column indexes. The matrix diagonal contains n elements representing crossovers that occur between parents of equal size ($j=k$). The remainder of the matrix (n^2-n elements) represents crossover events between parents of different sizes. Each cell of the left-hand matrix in Figure 15 shows the number of possible crossover events between parents with crossover point lengths of j and k up to n.

Given our nxn matrix, what are the probabilities for events by type (equal, outside, inside)? In the next two sections, we give the formulas for calculating these probabilities for our two possibilities: $j=k$ and $j<k$.

As an overview of this process, let us extend our initial matrix in Figure 15 by expanding a single cell to contain an inner matrix representing all possible crossover events for an L_2,L_4 mating. We use the notation X_i to indicate the different crossover points for both individuals where i is the location of the crossover point along the genome. Children produced by each possible crossover event are shown as C_l where l is the number of crossover points (length) in each child. For the mating of any L_2 and L_4 chromosome there are eight possible crossover events which produce children of various sizes.

By comparing the size of the children with that of the two parents, we can determine if the crossover event is of type inside (I), outside (O) or equal (E). As Figure 15 shows, the mating of an L_2 and L_4 parent results in eight possible crossover events – four of which are equal events, two are inside and two are outside events. This example shows the same results as those obtained by using formulas in Table 4 for $j<k$ with $j=2$ and $k=4$.

In order to determine the probabilities of crossover events for each cell, we will follow a process similar to that in Chapter 4. First, we calculate the percentage of crossover events by type for each mating. We then multiply these percentages by the probability of mating which in our uniformly distributed environment is defined in Equation 1 as $1/n^2$.

The formulas for counts of events from Table 4 are divided by the number of possible events for a single mating from Equation 24 or jk. These expressions are then reduced. For expressions in the column "For $j=k$", the variable j is substituted for k to simplify calculations later in this section.

Table 5:
Formulas representing the percentage of crossover types resulting from 1-point crossover for any two individuals with j and k crossover points respectively.

	For $j=k$	**For $j<k$**
Equals events	$\frac{j}{jk} = \frac{1}{k} = \frac{1}{j}$	$\frac{2j}{jk} = \frac{2}{k}$
Inside events	$\frac{0}{jk} = 0$	$\frac{jk - j^2 - j}{jk} = \frac{k - j - 1}{k}$
Outside events	$\frac{j^2 - j}{jk} = \frac{j-1}{k} = \frac{j-1}{j}$	$\frac{j^2 - j}{jk} = \frac{j-1}{k}$

The probabilities for each of the six scenarios in the above table can now be calculated for the entire matrix by multiplying the percentage of event types for each cell by the probability of being in that cell (mating), then summing the results.

5.1 Probability of Crossover Events by Type For $j=k$

The probability that an Equals event will occur when $j=k$ is equal to the percent of Equals events from Table 5 times the probability from $\mathrm{PrM}_{j,k}$ for each mating event. Since our objective

is to study GA behavior under uniform (random) selection, we use the expression $1/n^2$ for all $\mathrm{PrM}_{j,k}$.

$$\begin{aligned} \Pr(\text{Equals}_{j=k}) &= \sum_{j=1}^{n}\left(\frac{1}{j}\cdot \mathrm{PrM}_{j,k}\right) \\ &= \sum_{j=1}^{n}\left(\frac{1}{j}\cdot\frac{1}{n^2}\right) \\ &= \frac{1}{n^2}\cdot\sum_{j=1}^{n}\frac{1}{j} \\ &= \frac{\mathrm{H}_n}{n^2} \end{aligned} \tag{25}$$

A reminder that H_n in Equation 25 and subsequent equations indicates the Harmonic Series ($1/1 + 1/2 + 1/3 + \ldots + 1/n$). This series can be approximated by the natural log of n plus Euler's Constant or $\ln(n) + .57721$ (Euler's Constant to five decimal places).

From Table 2, we see that the percentage of inside crossovers when $j=k$ is 0. This again is due to the fact that there is no way to produce children of sizes in between two equal parents. For completeness, we provide Equation 26 below:

$$\begin{aligned} \Pr(\text{Inside}_{j=k}) &= \sum_{j=1}^{n}\left(\frac{0}{jk}\cdot \mathrm{PrM}_{j,k}\right) \\ &= \frac{1}{n^2}\cdot\sum_{j=1}^{n}0 \\ &= 0 \end{aligned} \tag{26}$$

The probability of outside crossover events where $j=k$ is calculated as follows:

$$\begin{aligned}\Pr(\text{Outside}_{j=k}) &= \sum_{j=1}^{n}\left(\frac{j-1}{j}\cdot \text{PrM}_{j,k}\right) \\ &= \frac{1}{n^2}\cdot\left(\sum_{j=1}^{n}\frac{j}{j}-\sum_{j=1}^{n}\frac{1}{j}\right) \\ &= \frac{1}{n^2}\cdot\left(n-\text{H}_n\right) \\ &= \frac{n-\text{H}_n}{n^2}\end{aligned} \tag{27}$$

5.2 Probability of Crossover Events by Type For $j<k$

We now wish to calculate the probability for each type of event that can occur when $j<k$ and $k\leq n$. The use of a square matrix as a representation for our hypothetical population provides us with some assistance. Since the matrix is square and symmetric, we know that a crossover between parents P1=j and P2=k is the same as a crossover between P1=k and p2=j. We merely swap the chromosomes (making j the shorter and k the longer chromosome) before applying the formulas from Table 5. We therefore only need to calculate probabilities for the bottom half of the matrix (less the diagonal) and multiply by 2 to get our results.

The following formula gives us the probability of crossover events that produce children of the same length as both parents for $j<k\leq n$:

$$\begin{aligned}\Pr(\text{Equals}_{j<k}) &= 2\cdot\sum_{k=2}^{n}\sum_{j=1}^{k-1}\left(\frac{2}{k}\cdot \text{PrM}_{j,k}\right) \\ &= 2\cdot\frac{1}{n^2}\cdot\sum_{k=2}^{n}\left(\frac{2}{k}\cdot\sum_{j=1}^{k-1}1\right) \\ &= \frac{2}{n^2}\cdot 2\cdot\sum_{k=2}^{n}\frac{k-1}{k} \\ &= \frac{4}{n^2}\cdot\left(\sum_{k=2}^{n}\frac{k}{k}-\sum_{k=2}^{n}\frac{1}{k}\right) \\ &= \frac{4}{n^2}\cdot\left((n-1)-(\text{H}_n-1)\right) \\ &= \frac{4n-4\text{H}_n}{n^2}\end{aligned} \tag{28}$$

The following formula gives us the probability of inside crossover events for $j<k\leq n$:

$$\begin{aligned}\Pr(\text{Inside}_{j<k}) &= 2\cdot\sum_{k=2}^{n}\sum_{j=1}^{k-1}\left(\frac{k-j-1}{k}\cdot \text{PrM}_{j,k}\right) \\ &= 2\cdot\frac{1}{n^2}\cdot\sum_{k=2}^{n}\left(\frac{1}{k}\cdot\left(\sum_{j=1}^{k-1}k-\sum_{j=1}^{k-1}j-\sum_{j=1}^{k-1}1\right)\right) \\ &= \frac{2}{n^2}\cdot\sum_{k=2}^{n}\left(\frac{1}{k}\cdot\left(k(k-1)-\frac{(k-1)(k-1+1)}{2}-(k-1)\right)\right) \\ &= \frac{2}{n^2}\cdot\sum_{k=2}^{n}\frac{k^2-3k+2}{2k} \\ &= \frac{2}{n^2}\cdot\left(\sum_{k=2}^{n}\frac{k^2}{2k}-\sum_{k=2}^{n}\frac{3k}{2k}+\sum_{k=2}^{n}\frac{2}{2k}\right) \\ &= \frac{2}{n^2}\cdot\left(2\cdot\sum_{k=2}^{n}k-\frac{3}{2}\cdot\sum_{k=2}^{n}1+\sum_{k=2}^{n}\frac{1}{k}\right) \\ &\frac{2}{n^2}\cdot\left(2\cdot\left(\frac{n(n+1)}{2}-1\right)-\frac{3}{2}\cdot(n-1)+(\text{H}_n-1)\right) \\ &= \frac{n^2-5n+4H_n}{2n^2}\end{aligned} \tag{29}$$

The following formula gives us the probability of outside crossover events for $j<k\leq n$:

$$\begin{aligned}
\Pr(\text{Outside}_{j<k}) &= 2\cdot\sum_{k=2}^{n}\sum_{j=1}^{k-1}\left(\frac{j-1}{k}\cdot \text{PrM}_{j,k}\right) \\
&= 2\cdot\frac{1}{n^2}\cdot\sum_{k=2}^{n}\left(\frac{1}{k}\cdot\left(\sum_{j=1}^{k-1} j-\sum_{j=1}^{k-1}1\right)\right) \\
&= \frac{2}{n^2}\cdot\sum_{k=2}^{n}\frac{1}{k}\cdot\left(\frac{(k-1)(k-1+1)}{2}-(k-1)\right) \\
&= \frac{2}{n^2}\cdot\sum_{k=2}^{n}\frac{k^2-3k+2}{2k} \\
&= \frac{2}{n^2}\cdot\left(\sum_{k=2}^{n}\frac{k^2}{2k}-\sum_{k=2}^{n}\frac{3k}{2k}+\sum_{k=2}^{n}\frac{2}{2k}\right) \\
&= \frac{2}{n^2}\cdot\left(\sum_{k=2}^{n}\frac{k}{2}-\sum_{k=2}^{n}\frac{3}{2}+\sum_{k=2}^{n}\frac{1}{k}\right) \\
&= \frac{2}{n^2}\cdot\left(\frac{1}{2}\cdot\frac{(n-2+1)(n+2)}{2}-\frac{3}{2}\cdot(n-1)+(\text{H}_n-1)\right) \\
&= \frac{n^2-5n+4\text{H}_n}{2n^2}
\end{aligned} \tag{30}$$

5.3 Probability of Crossover Events by Type for all *j,k*

We now combine the probabilities derived for situations $j=k$ and $j<k$ to determine crossover probabilities by type for the entire population matrix. The following equations provide us with our final results:

$$\begin{aligned}
\Pr(\text{Equals}_{j,k}) &= \Pr(\text{Equals}_{j=k})+\Pr(\text{Equals}_{j<k}) \\
&= \frac{H_n}{n^2}+\frac{4n-4\text{H}_n}{n^2} \\
&= \frac{4n-3\text{H}_n}{n^2}
\end{aligned} \tag{31}$$

$$\begin{aligned}\Pr(\text{Inside}_{j,k}) &= \Pr(\text{Inside}_{j=k}) + \Pr(\text{Inside}_{j<k}) \\ &= 0 + \frac{n^2 - 5n + 4\text{H}_n}{2n^2} \\ &= \frac{1}{2} \cdot \frac{n^2 - 5n + 4\text{H}_n}{n^2}\end{aligned} \tag{32}$$

$$\begin{aligned}\Pr(\text{Outside}_{j,k}) &= \Pr(\text{Outside}_{j=k}) + \Pr(\text{Outside}_{j<k}) \\ &= \frac{n - \text{H}_n}{n^2} + \frac{n^2 - 5n + 4\text{H}_n}{2n^2} \\ &= \frac{n^2 - 3n + 2\text{H}_n}{2n^2} \\ &= \frac{1}{2} \cdot \frac{n^2 - 3n + 2\text{H}_n}{2n^2}\end{aligned} \tag{33}$$

The limits for Equations 31, 32 and 33 as *n* approaches infinity are 0, 1/2, and 1/2 respectively.

5.4 Experimental Confirmation

An experiment was performed to verify the formulas obtained in the previous section. The experiment consisted of a number of nested loops which enumerate all possible crossover events for all possible matings for a series of *n*x*n* population matrices varying in size from *n*=1 to 200 with *n* an even number. All events were categorized, counted and weighted based on mating probability for each matrix by type of event. The results of this enumerative computation were then plotted in Figure 16.

As *n* increases in the experiment, the number of Equal crossover events diminished rapidly. Equal events were replaced by a growing number of Inside and Outside events which both approached 50% of all possible crossovers. These percentages match those obtained from limits to Equations 31, 32 and 33.

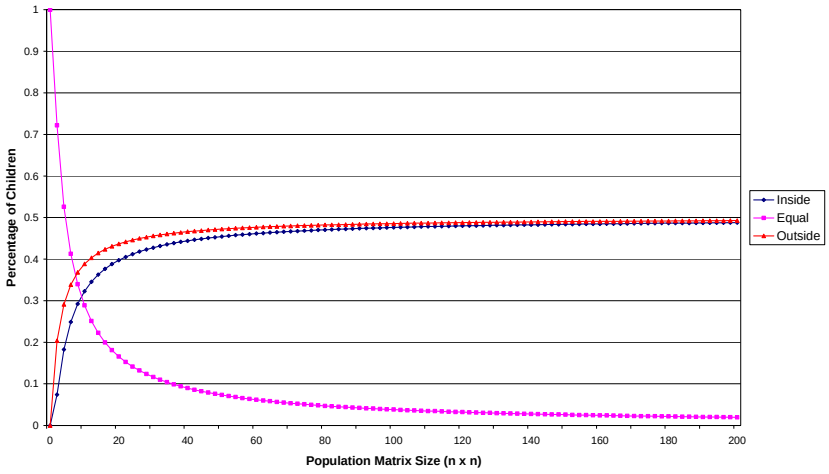

Figure 16: Probability of crossover events by type for *n*x*n* population matrices varying in size from *n*=1 to 200. Event probabilities weighted based on mating occurrences.

From the findings in this chapter, we can summarize our results in the following way: Given 1) two randomly selected individuals from a sufficiently large population of variable-length chromosomes of uniformly distributed size and 2) two randomly chosen crossover points within those individuals, the probability of an inside or outside crossover events approaches approximately 50% each and the probability that the two resulting children will be equal to the parents in length approaches 0.

5.5 Speed of Size Diversification

The probabilities found in Section 5.3 help us understand how the GA can process variable-length strings and create size diversity within a population. As the variation in size of individuals within population approaches uniform distribution, the probabilities of inside and outside crossovers approach 50%. The equality of these percentages shows that the GA does not inherently favor one type of event over another in a population with uniform size distribution.

What happens when size distribution within a population is not uniform and how rapidly does diversity appear? We answer these two questions with a series of experiments which show that regardless of distribution at time *t*, a GA under random selection will quickly diversify sizes of individuals within the population.

The first type of experiment is based on the one given in section 4.1 to calculate probability distributions over time. The second type of experiment consists of simple simulations of GA selection, crossover and reproduction using integer numbers (see Section 3.0, (Stringer & Wu, 2005) for details). In this second experiment, we use a simulation which models a GA with a population of 100 individuals and a maximum size cap of 200. Children produced by the model with lengths greater than 200 are right truncated (i.e., integer value for child is reset to 200). Selection of parents and crossover points is random.

Both experiments were conducted for four scenarios, each with a different initial population. The results from these experiments are described in the following paragraphs and show that chromosome size and the probability of size distribution within a population diversifies rapidly regardless of starting population when random selection is applied.

The first set of experiments assumed an initial population with a uniform size distribution. Figure 17 shows how the probability distribution changes over time. The initial generation or Gen 0 is represented by the straight line across the graph. All sizes from 1 to 200 have an equal chance of selection (.5%).

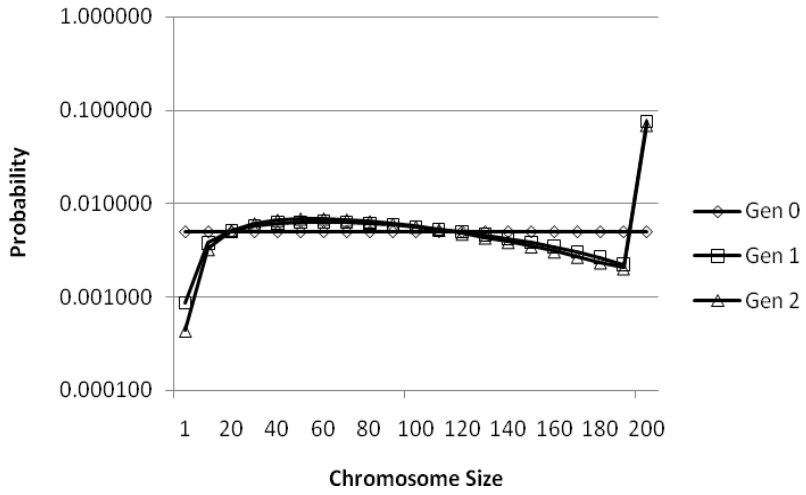

Figure 17: Probability of size distributions over time for a GA with an initial population (Gen 0) uniformly distributed with respect to size.

Plots for generations 1 and 2 show the beginnings of the gamma distribution described in previously cited works by Poli, McPhee and Rowe as well as those found in our own Figure 8, 9 and 10. The uptick at the end of curves for generations 1 and 2 is caused by the truncation of overly long children to the maximum size of 200.

Figure 18 shows a sample scatter plot from a single run of the GA model simulation. Each chromosome is identified on the X-axis by number for a total of 100 individuals. In the initial population each chromosome's length is the same as its identifier. This creates a uniform distribution of lengths indicated by the series of diagonal data points in the plot. After two generations, the length of individuals is fairly random though no longer neatly ordered.

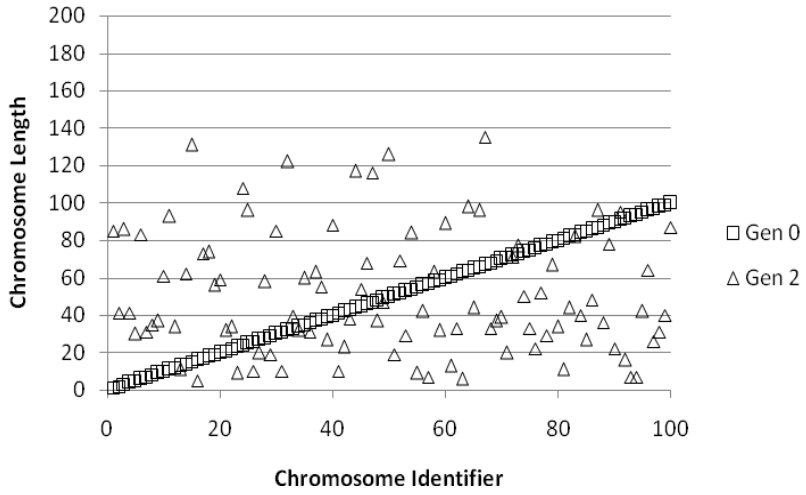

Figure 18: Scatter plot showing sizes of each individual in a modeled GA population after two generations. The initial population (Gen 0) contains one individual of each size to simulate uniform size distribution.

Additional experiments were performed assuming three different initial population conditions: Single Point, Two Point and Upper Region.

A *single point* distribution is often used by GA researchers who set the all individuals in the initial population to the same size. The state of the initial population can be easily seen in Figure 19 as the spike in the middle of the graph. All members of the starting population were set to a length of 100 (probability = 1.0)

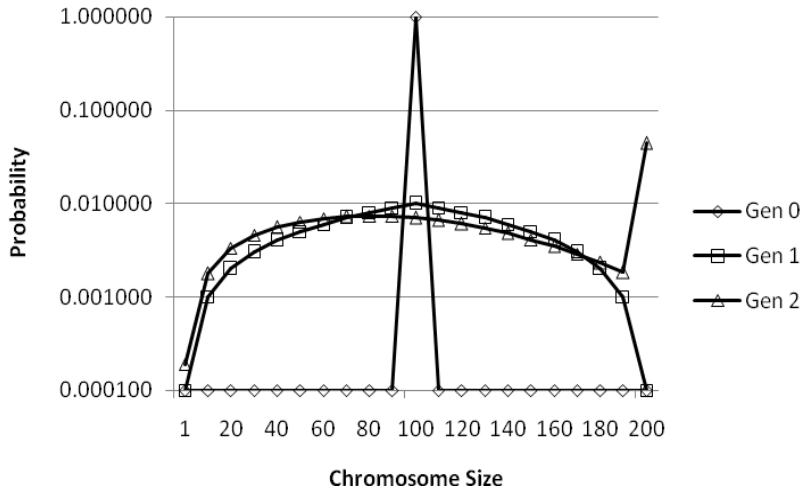

Figure 19: Probability of size distributions over time for a GA with an initial population (Gen 0) set to a single distribution point (100) with respect to length.

All crossover events using this initial population are of type j=k. We know from our equations in Table 5 that the probability of outside crossover events is $(j\text{-}1)/j$ or in our model .99. Even without this calculation, it is obvious that the GA will create children that fill in the gaps from sizes 0 to 100 and 100 to 200. During the next reproduction cycle, the GA further diversifies sizes within the population and begins to create the gamma distribution characteristic of GAs under random selection (and the uptick at 200 caused by truncation).

The figure below is a scatter plot showing lengths of each individual in the population at Generations 0 and 2. The straight line across graph is the result of our initial population where all chromosomes were set to 100.

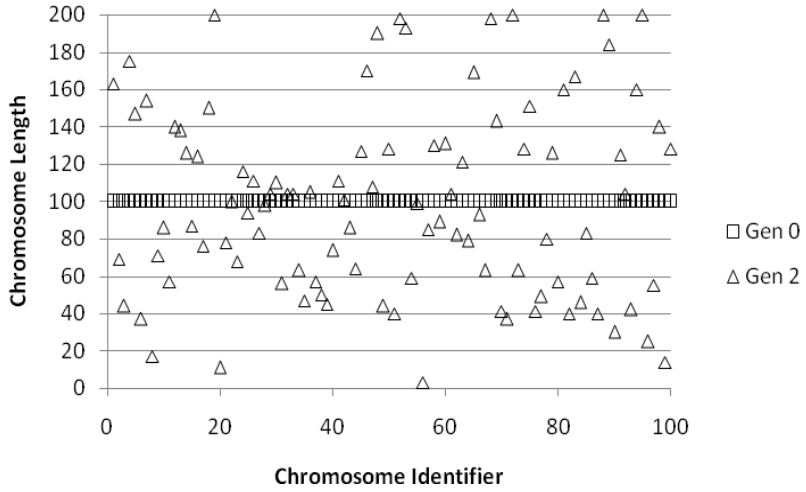

Figure 20: Scatter plot showing sizes of each individual in a modeled GA population after two generations. The initial population (Gen 0) contains only individuals of size 100.

Our next initial condition is a Two Point Distribution (see Figure 21 and Figure 22). Here the initial population was set to one of two lengths; 20 and 180. This type of distribution is not usually seen in GA practice but we show it here to illustrate how quickly the GA can diversify chromosome size in a population by filling in large gaps between groups of similar sized individuals.

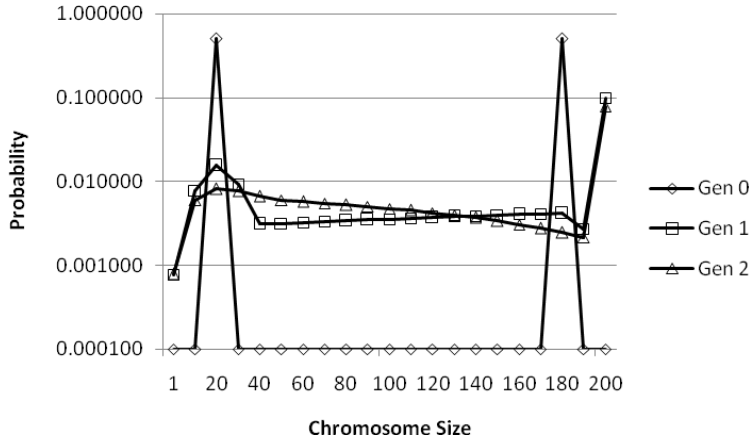

Figure 21: Probability of size distributions over time for a GA with an initial population (Gen 0) set to a two different distrubution points (20, 180) with respect to length.

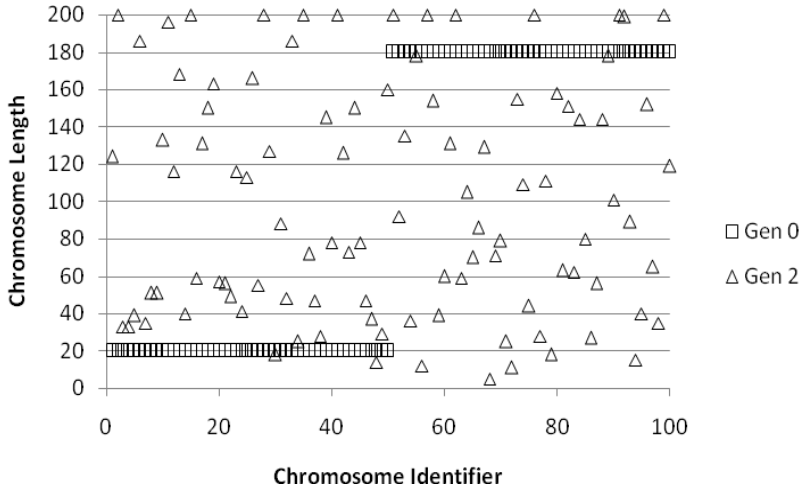

Figure 22: Scatter plot showing sizes of each individual in a modeled GA population after two generations. The initial population (Gen 0) contains equal numbers of individuals of size 20 or 180.

We call our last initial condition the "Upper Region" distribution (see Figure 23, 24 and 25). This distribution is not used in GA practice for starting populations; however, it is almost always seen in variable-length GAs when bloat occurs. Bloat causes the GA to evolve chromosomes with long lengths close to some upper size limit imposed by a truncation parameter or anti-bloat operator.

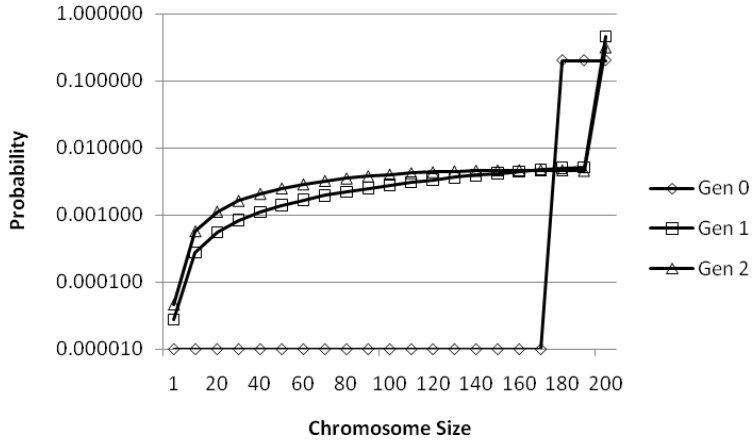

Figure 23: Probability of size distributions over time for a GA with an initial population (Gen 0) whose size distrubition includes only longer individuals (≥ 180).

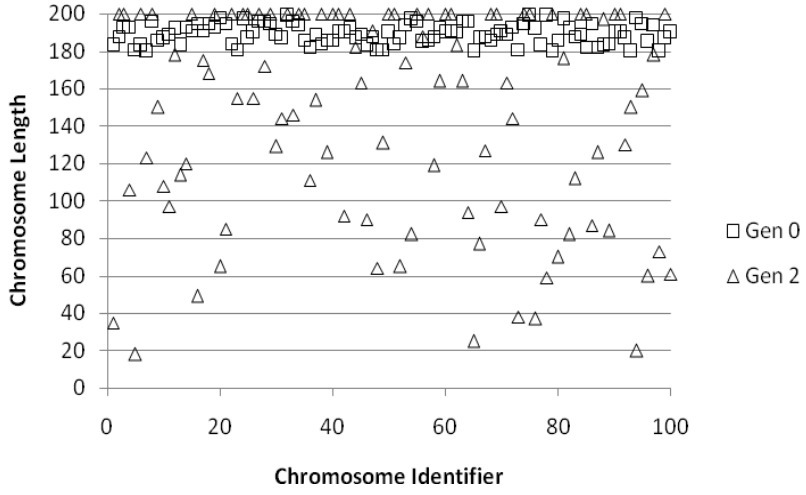

Figure 24: Scatter plot showing sizes of each individual in a modeled GA population after two generations. The initial population (Gen 0) contains only individuals of length greater than 180.

The previous two figures show that even with an upper region distribution a GA with random selection can begin to diversify chromosome size within a population. Unlike our other

initial conditions, however, the pace here is somewhat slower. Figure 24 illustrates this with a large number of chromosomes still in the upper region after two generations. Multiple runs of our model indicate that it takes from 5 to 10 generations before an upper region is fully distributed. Figure 25 shows the scatter plot for a run at generation 10. In that figure we see a more uniform size distribution has finally been reached.

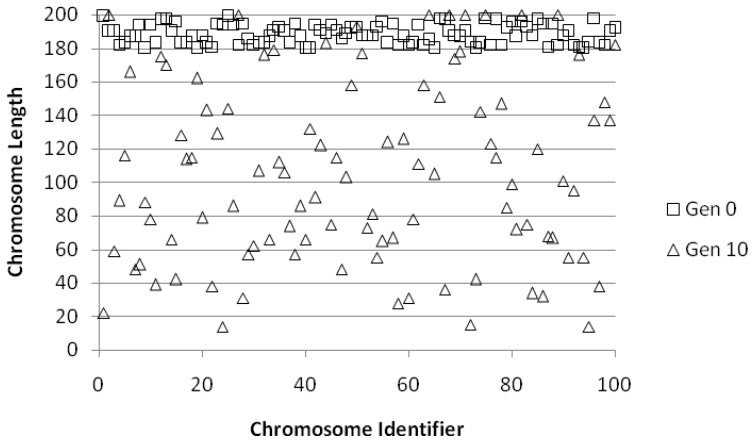

Figure 25: Scatter plot showing sizes of each individual in a modeled GA population after ten generations. The initial population (Gen 0) contains only individuals of length greater than 180.

5.6 Discussion

The equations and confirming experiments in this section prove what to many is intuitively obvious – that a GA under random selection will create and maintain diversity in size within a population. The surprising finding in this chapter is the speed at which this diversification takes place. For all but upper region starting conditions, the disbursement across sizes happens in only two or three generations. Extrapolating from our results, we present the following rules of thumb:

- The size distribution of a starting population is not an important factor in variable-length GAs under random selection unless all members are concentrated in an upper region.

Whether a single point, multiple points or a uniform distribution, the random selection GA will quickly fill in the gaps.

- The closer the initial distribution is to uniform (or concentrated in the middle of the populations allowable size range), the sooner the GA can begin reducing the mean size of the population.
- For all but upper region distributions the change in size distribution happens rapidly.

How can we apply these rules of thumb? One possibility is in the area of bloat control. One idea we have contemplated for some time is to turn off selection for short durations (5-10 generations) whenever a population converges near a size cap during the GA's run. Our thinking was to allow the GA to naturally shorten the mean size of a population through over production of shorter than average children. From our empirical studies, we now know that 5 to 10 generations is not enough time for this to occur in a population with sizes concentrated in an upper region. It takes that long just to diversify the chromosome sizes, much less begin producing shorter than average children.

Our rationale for this anti-bloat method is still reasonable but requires modification due to the slower diversification rate for an upper region starting point. One option is to leave random selection turned on for longer periods 20-30 generations once the mean population size reaches some upper region threshold. An alternative is to activate random selection early before the GA converges to an upper region size distribution allowing size diversity to be maintained or occur more quickly. Whatever the distribution at the time of early activation, the GA will quickly spread out the sizes and almost immediately begin production of shorter than average children. These two options can be thought of as "Longer Later" and Shorter, Sooner". We

leave investigation of these two anti-bloat methods for future research. It will be interesting to see which of the two has the lesser impact on overall population fitness.

CHAPTER 6: PUTTING IT ALL TOGETHER

Our work on the ChGA was where we first saw reduction in chromosome size occurring in a variable-length representation. We offer the following discussion of the ChGA's behavior to illustrate the roles played by size variation (Chapter 5) and over-production of shorter than average children (Chapter 4) in reducing the mean size of a finite population GA over time.

The ChGA combines a variable-length GA with a shared communal memory that allows evolution of both individual chromosomes and memory simultaneously. Memory can be accessed by individuals in the population to improve their solutions over the course of the ChGA run. At some point in the ChGA's execution, memory contains numerous high-value genes. Individuals begin accessing these memory genes more and more frequently. Over time, the contents of memory becomes the primary contributor to each individuals' fitness and the contents of base chromosomes becomes irrelevant. At this point the ChGA begins to evolve away base chromosomes as they are no longer needed to improve or maintain high fitness.

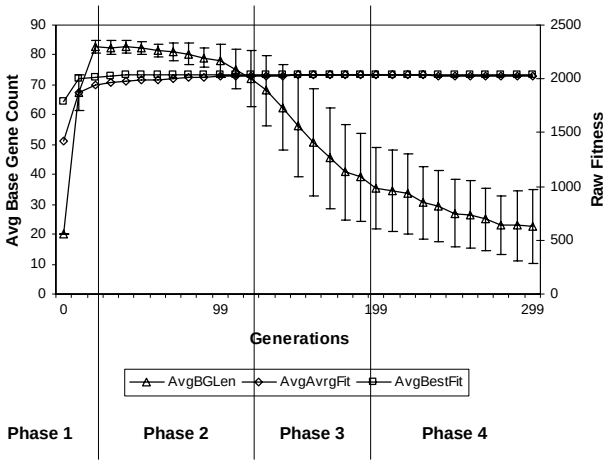

Figure 26: Partition of ChGA run for a 3x8 MaxSum Problem using 8 Memory Slots. Average Base Chromosome Length shown with +/- one standard deviation.

Figure 26 illustrates this process and the impact of random selection on the ChGA. During Phase 1, the average size of chromosomes grows very rapidly. The ChGA is exploring the search space looking for good genes as well as good gene combinations. In this phase, longer individuals are favored in order to construct better and larger gene building blocks (Stephens & Waelbroeck, 1997). Growth in chromosome length continues until a 100-gene cap is reached and an upper region distribution occurs. At that time, the mean population length stabilizes with minimum variation in size distribution. The lack of variation prevents the development of below average size children.

This continues through Phase 2 during which the ChGA identifies good genes and moves them into memory. Towards the end of Phase 2, the ChGA's memory contains most, if not all of the good genes required for the best possible solution. Individuals that reference these memory slots can now survive into the next generation regardless of base chromosome size. As more and more of the population references high valued memory, selection among individuals becomes constant and search moves to a flat fitness landscape.

Crossover events now begin to play an important role by diversifying the lengths of chromosomes within the population. Populations centered tightly around an average size are more likely to produce outside crossover events which increases size diversity. We see this occurring at the start of Phase 3 and continuing for the duration of the ChGA's run. As size diversity increases, the ChGA begins to over produce shorter-than-average sized chromosomes.

The number of these shorter individuals increases to the point where they dominate the population and are oversample by the selection process, often at the expense of larger siblings. Since selection is random, it is possible to select shorter individuals as parents in a proportion greater than their actual presence within the population at large. Over-selection of shorter

individuals leads to a speed up in the reduction of average population size over time. Under-selection on the other hand can slow down or even cancel the reduction. This explains those few ChGA runs where the average size of the population did not shrink or reduction occurred slowly.

Phase 3 clearly shows how this diversification/over-production/over-sampling process picks up steam and quickly shortens the mean length shown in Figure 26 by over half in 100 generations.

Phase 4 of the illustration is really an extension of Phase 3 with the exception that the variation in population size decreases along with a corresponding rate in over-production of shorter children. We, therefore, see a slowdown in the rate of reduction in genome length.

CHAPTER 7: CONCLUSIONS

The theoretical and empirical findings in this thesis are important on a number of levels. First and foremost, we have met our original objective by presenting a detailed analysis of the behavior of genetic algorithms under random selection. It has been known for some time that an EC-based algorithm with linear structures under random selection can reduce the mean population size. What is new in this work is an explanation of the mechanisms or processes that make reduction happen and how a large variation in size within a population is a precondition. The three forces identified as contributing to a reduction in average population size are:

- an increase in size diversity within the population;
- over production of shorter than average children; and
- the imperfect nature of random sampling during selection.

On a personal level it has been gratifying to see how our findings explain the behavior of the Chunking GA and its ability to produce compact solutions.

Equally important is the development of tools that can be used to analyze a GA, including those whose selection methods are other than random. Specifically, we see the following as important contributions to the GA practitioner's toolbox:

- A new classification system for variable-length GAs based on solution size (section 3.2);
- A partitioned map of the mating space with five distinct regions or areas (see pLT_n, Figure 6) for calculating numbers and percentages of shorter than average children;
- Equations for calculating the number of shorter than average children possible from any mating. Separate equations (Equations 7 through 11) are provided for each area in the partitioned mating space;

- A method for computing the probability of producing shorter than average children for any single mating ($pLT_{j,k}$ * $PrM_{j,k}$);
- The concept of a shorter-than-average probability matrix (PrLT) than can be calculated from pLT and PrM matrices giving the probability that an entire GA population will produce shorter than average children in the next generation;
- Equations for calculating percentages of possible crossover events by type (inside, outside and equals) for any single mating under one-point crossover (Table 5).

Improving our understanding of the impact of mating choices on the size of offspring may be the most important contribution of this work. Past research has shown that variable-length evolutionary algorithms are subject to bloat and tend to evolve overly long chromosomes in which useful information is interspersed with unused information. For problems where resources are limited, smaller solutions are quicker to process and more efficient. Fighting bloat is also important if the variable-length GA is ever to solve highly-complex open-ended problems with unbounded solution size. In this work we have offered new and specific ideas for combating bloat in variable-length GAs based on our findings in Chapters 4 and 5. More importantly, we believe the ideas presented in this thesis open up new avenues and new tools for exploring ways of restraining bloat in genetic algorithms as well as EC in general.

APPENDIX: SOLUTIONS TO PARTIAL TERMS IN EQUATION 17

Due to the number of terms in Equation 17 the complete solution is offered in this appendix.

$$\begin{aligned} \mathrm{E_B}\left(\text{children}_{l<A}\right) &= \sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}\left(2jk-\left(j+k-A\right)\left(j+k-A+1\right)\right) \\ &= \sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}\left(-j^2+\left(2A-1\right)j-k^2+\left(2A-1\right)k-\left(A^2-A\right)\right) \\ &= -\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}j^2\right)+\left(2A-1\right)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}j\right)-\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}k^2\right)+ \\ &\quad \left(2A-1\right)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}k\right)-\left(A^2-A\right)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}1\right) \end{aligned} \tag{17}$$

Each of the terms in the last expression are solved individually on the next five pages.

$$\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1} j^2\right) = \sum_{k=2}^{A-1}\left(\sum_{j=1}^{A-1} j^2 - \sum_{j=1}^{A-k+1-1} j^2\right)$$

$$= \sum_{k=2}^{A-1}\left(\frac{(A-1)(A)(2A-1)}{6} - \frac{(A-k)(A-k+1)(2A-2k+1)}{6}\right)$$

$$= \sum_{k=2}^{A-1}\left(\frac{6A^2k - 6Ak^2 + 6Ak + 2k^3 - 3k^2 + k - 6A^2}{6}\right)$$

$$= \frac{1}{6}\cdot\left(\sum_{k=2}^{A-1}\left(6A^2k + 6Ak + k\right) - \sum_{k=2}^{A-1}\left(6Ak^2 + 3k^2\right) + \sum_{k=2}^{A-1} 2k^3 - \sum_{k=2}^{A-1} 6A^2\right)$$

$$= \frac{1}{6}\cdot\left(\left(6A^2 + 6A + 1\right)\cdot\sum_{k=2}^{A-1} k - (6A+3)\cdot\sum_{k=2}^{A-1} k^2 + 2\cdot\sum_{k=2}^{A-1} k^3 - 6A^2\cdot\sum_{k=2}^{A-1} 1\right)$$

$$= \frac{1}{6}\cdot\left(\begin{array}{l}\left(6A^2 + 6A + 1\right)\cdot\dfrac{(A-1-2+1)(A-1+2)}{2} - (6A+3)\cdot\displaystyle\sum_{k=2}^{A-1} k^2 + \\ 2\cdot\displaystyle\sum_{k=2}^{A-1} k^3 - 6A^2\cdot\sum_{k=2}^{A-1} 1\end{array}\right)$$

$$= \frac{1}{6}\cdot\left(\begin{array}{l}\dfrac{6A^4 - 17A^2 - 13A - 2}{2} - (6A+3)\cdot\left(\dfrac{(A-1)(A)(2A-1)}{6} - 1\right) + \\ 2\cdot\displaystyle\sum_{k=2}^{A-1} k^3 - 6A^2\cdot\sum_{k=2}^{A-1} 1\end{array}\right)$$

$$= \frac{1}{6}\cdot\left(\begin{array}{l}\dfrac{6A^4 - 17A^2 - 13A - 2}{2} - \dfrac{12A^4 - 12A^3 - 3A^2 - 33A - 18}{6} + \\ 2\cdot\left(\left(\dfrac{(A-1)(A)}{2}\right)^2 - 1\right) - 6A^2\cdot\displaystyle\sum_{k=2}^{A-1} 1\end{array}\right)$$

$$= \frac{1}{6}\cdot\left(\begin{array}{l}\dfrac{6A^4 - 17A^2 - 13A - 2}{2} - \dfrac{12A^4 - 12A^3 - 3A^2 - 33A - 18}{6} + \\ \dfrac{A^4 - 2A^3 + A^2 - 4}{2} - 6A^2\cdot(A-1-2+1)\end{array}\right)$$

$$= \frac{1}{6}\cdot\left(\begin{array}{l}\dfrac{6A^4 - 17A^2 - 13A - 2}{2} - \dfrac{12A^4 - 12A^3 - 3A^2 - 33A - 18}{6} + \\ \dfrac{A^4 - 2A^3 + A^2 - 4}{2} - \left(6A^3 - 12A^2\right)\end{array}\right)$$

$$= \frac{9A^4 - 30A^3 + 27A^2 - 6A}{36}$$

$$(2A-1)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1} j\right) = (2A-1)\cdot\sum_{k=2}^{A-1}\frac{\big((A-1)-(A-k+1)+1\big)\big(A-1+A-k+1\big)}{2}$$

$$= (2A-1)\cdot\sum_{k=2}^{A-1}\frac{2Ak+k-k^2-2A}{2}$$

$$= \frac{2A-1}{2}\cdot\left(\sum_{k=2}^{A-1}(2Ak+k)-\sum_{k=2}^{A-1}k^2-\sum_{k=2}^{A-1}2A\right)$$

$$= \frac{2A-1}{2}\cdot\left((2A+1)\cdot\sum_{k=2}^{A-1}k-\sum_{k=2}^{A-1}k^2-2A\cdot\sum_{k=2}^{A-1}1\right)$$

$$= \frac{2A-1}{2}\cdot\left((2A+1)\cdot\sum_{k=2}^{A-1}k-\left(\sum_{k=1}^{A-1}k^2-1\right)-2A\cdot\sum_{k=2}^{A-1}1\right)$$

$$= \frac{2A-1}{2}\cdot\left((2A+1)\cdot\frac{\big(A-1-2+1\big)\big(A-1+2\big)}{2}-\left(\sum_{k=1}^{A-1}k^2-1\right)-2A\cdot\sum_{k=2}^{A-1}1\right)$$

$$= \frac{2A-1}{2}\cdot\left(\frac{2A^3-2A^2-4A+A^2-A-2}{2}-\left(\sum_{k=1}^{A-1}k^2-1\right)-2A\cdot\sum_{k=2}^{A-1}1\right)$$

$$= \frac{2A-1}{2}\cdot\left(\frac{2A^3-A^2-5A-2}{2}-\left(\frac{\big(A-1\big)\big(A\big)\big(2A-1\big)}{6}-1\right)-2A\cdot\sum_{k=2}^{A-1}1\right)$$

$$= \frac{2A-1}{2}\cdot\left(\frac{2A^3-A^2-5A-2}{2}-\frac{2A^3-3A^2+A-6}{6}-2A\cdot\big(A-1-2+1\big)\right)$$

$$= \frac{2A-1}{2}\cdot\left(\frac{2A^3-A^2-5A-2}{2}-\frac{2A^3-3A^2+A-6}{6}-\big(2A^2-4A\big)\right)$$

$$= \frac{2A-1}{2}\cdot\frac{4A^3-12A^2+8A}{6}$$

$$= \frac{8A^4-28A^3+28A^2-8A}{12}$$

$$= \frac{24A^4-84A^3+84A^2-24A}{36}$$

$$\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}k^2\right)=\sum_{k=2}^{A-1}k^2\sum_{j=A-k+1}^{A-1}1$$

$$=\sum_{k=2}^{A-1}\left(k^2\cdot\left((A-1)-(A-k+1)+1\right)\right)$$

$$=\sum_{k=2}^{A-1}\left(k^2\cdot(k-1)\right)$$

$$=\sum_{k=2}^{A-1}\left(k^3-k^2\right)$$

$$=\sum_{k=2}^{A-1}k^3-\sum_{k=2}^{A-1}k^2$$

$$=\left(\sum_{k=1}^{A-1}k^3-1\right)-\left(\sum_{k=1}^{A-1}k^2-1\right)$$

$$=\left(\frac{(A-1)(A-1+1)}{2}\right)^2-\frac{(A-1)(A)(2A-1)}{6}$$

$$=\frac{A^4-2A^3+A^2}{4}-\frac{2A^3-3A^2+A}{6}$$

$$=\frac{3A^4-10A^3+9A^2-2A}{12}$$

$$=\frac{9A^4-30A^3+27A^2-6A}{36}$$

$$
\begin{aligned}
(2A-1)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}k\right) &= (2A-1)\cdot\sum_{k=2}^{A-1}k\sum_{j=A-k+1}^{A-1}1 \\
&= (2A-1)\cdot\sum_{k=2}^{A-1}\left(k\cdot\left((A-1)-(A-k+1)+1\right)\right) \\
&= (2A-1)\cdot\sum_{k=2}^{A-1}\left(k\cdot(k-1)\right) \\
&= (2A-1)\cdot\sum_{k=2}^{A-1}\left(k^2-k\right) \\
&= (2A-1)\cdot\left(\sum_{k=2}^{A-1}k^2-\sum_{k=2}^{A-1}k\right) \\
&= (2A-1)\cdot\left(\left(\sum_{k=1}^{A-1}k^2-1\right)-\left(\sum_{k=1}^{A-1}k-1\right)\right) \\
&= (2A-1)\cdot\left(\frac{(A-1)(A)(2A-1)}{6}-\frac{(A-1)(A-1+1)}{2}\right) \\
&= (2A-1)\cdot\left(\frac{2A^3-3A^2+A}{6}-\frac{A^2-A}{2}\right) \\
&= (2A-1)\cdot\frac{2A^3-6A^2+4A}{6} \\
&= \frac{4A^4-14A^3+14A^2-4A}{6}
\end{aligned}
$$

$$
= \frac{24A^4-84A^3+84A^2-24A}{36}
$$

$$\left(A^2 - A\right)\cdot\left(\sum_{k=2}^{A-1}\sum_{j=A-k+1}^{A-1}1\right) = \left(A^2 - A\right)\cdot\sum_{k=2}^{A-1}\left(\left(A-1\right)-\left(A-k+1\right)+1\right)$$

$$= \left(A^2 - A\right)\cdot\sum_{k=2}^{A-1}\left(k-1\right)$$

$$= \left(A^2 - A\right)\cdot\left(\sum_{k=2}^{A-1}k - \sum_{k=2}^{A-1}1\right)$$

$$= \left(A^2 - A\right)\cdot\left(\frac{\left(A-1-2+1\right)\left(A-1+2\right)}{2} - \left(A-1-2+1\right)\right)$$

$$= \left(A^2 - A\right)\cdot\frac{A^2-3A+2}{2}$$

$$= \left(A^2 - A\right)\cdot\frac{A^2-3A+2}{2}$$

$$= \frac{A^4-4A^3+5A^2-2A}{2}$$

$$= \frac{18A^4-72A^3+90A^2-36A}{36}$$

Partial solutions for each term are then recombined to give a final solution below:

$$\mathrm{E_B}\left(\mathrm{children}_{l<A}\right) = -\frac{9A^4-30A^3+27A^2-6A}{36} + \frac{24A^4-84A^3+84A^2-24A}{36} -$$
$$\frac{9A^4-30A^3+27A^2-6A}{36} + \frac{24A^4-84A^3+84A^2-24A}{36} -$$
$$\frac{18A^4-72A^3+90A^2-36A}{36}$$

$$= \frac{12A^4-36A^3+24A^2}{36}$$

LIST OF REFERENCES

Angeline, P. J. & Pollack, J. B. (1993). Evolutionary module acquisition. In Fogel, D. and Atmar, W., editors, *Proceedings of the Second Annual Conference on Evolutionary Programming*, pages 154-163, La Jolla, California.

Barone, L., While, L. & Hingston, P. (2002). Designing crushers with a multi-objective evolutionary algorithm. In Langdon, W. B., Cantu-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E. and Jonoska, N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 995-1002, New York, New York. Morgan Kaufmann Publishers, San Francisco, CA.

Bassett, J. K. & De Jong, K. A. (2000). Evolving behaviors for cooperating agents. In *International Symposium on Methodologies for Intelligent Systems*, pages 157-165.

Blickle, T. & Thiele, L. (1994). Genetic programming and redundancy. In Hopf, J., editor, *Proceedings of Genetic Algorithms within the Framework of Evolutionary Computation*, pages 33-38, Workshop at KI-94, Saarbrucken. Max-Planck-Institut fur Informatik (MPI-I-94-241).

Burke, D. S., De Jong, K. A., Grefenstette, J. J., Ramsey, C. L. & Wu, A. S. (1998). Putting more genetics into genetic algorithms. *Evolutionary Computation*, 6(4):387-410.

De Jong, E. D. (2003). Representation development from pareto-coevolution. In Cantu-Paz, E., Foster, J. A., Deb, K., Davis, L. D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N. and Miller, J., editors, *GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference*, LNCS 2723, pages 262-273, Chicago, IL. Springer, Berlin, Germany.

De Jong, E. D. & Thierens, D. (2004). Exploiting modularity, hierarchy, and repetition in variable-length problems. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E., Darwen, P., Dasgupta, D., Floreano, D., Foster, J., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D. and Tyrrell, A., editors, *GECCO 2004: Proceedings of the Genetic and Evolutionary Computation Conference*, LNCS 3102, pages 1030-1041, Seattle, Washington. Springer, Berlin, Germany.

De Jong, K. A.(1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Doctoral Dissertation, University of Michigan, Computer and Communications Sciences.

De Jong, K. A. (1988). Using genetic algorithms to learn task programs: the Pitt Approach. *Machine Learning*, 3(2-3).

De Jong, K. A. (2003). Evolutionary Computation: A Unified Approach. In *Genetic and Evolutionary Computation Conference*, Tutorial Program, Chicago, IL.

De Jong, K. A. & Spears, W. M. (1991). Learning concept classification rules using genetic algorithms. In *Proceedings of the Twelfth International Conference on Artificial Intelligence (IJCAI-91)*, 2, pages 651-656, Sidney, Australia. Morgan Kaufmann.

Eshelman, L. J., Caruana, R. A. & Schaffer, J. D. (1989). Biases in the crossover landscape. In Schaffer, J. D., editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 10-19.

Fogel, L. J., Owens, A. J. & Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, New York, New York.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.

Goldberg, D. E., Korb, B. & Deb, K. (1989). Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, 3(5):493-530.

Grefenstette, J. J., Ramsey, C. L. & Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5:355-381.

Harik, G.(1997). *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms*. Doctoral Dissertation, University of Michigan, Computer Science and Engineering.

Harvey, I. (1992a). Species adaptation genetic algorithms: a basis for a continuing SAGA. In Tettamanzi, A., editor, *Proceedings of the First European Conference on Artificial Life. Toward a Practice of Autonomous Systems*, pages 346-354. MIT Press, Cambridge, MA.

Harvey, I. (1992b). The SAGA Cross: The mechanics of recombination for species with variable-length genotypes. In Manner, R. and Manderick, B., editors, *Proceedings of Parallel Problem Solving from Nature II*, pages 269-278. North-Holland.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Holland, J. (1986). *Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems*. Morgan Kaufmann, Los Altos, California.

Kavka, C. & Schoenauer, M. (2003). Voronoi diagrams based function identification. In Cantu-Paz, E., Foster, J. A., Deb, K., Davis, L. D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N. and Miller, J., editors, *GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference*, LNCS 2723, pages 1089-1100, Chicago, IL. Springer, Berlin, Germany.

Koza, J. R. (1992). *Genetic Programming*. MIT Press, Cambridge, MA.

Langdon, W. B. (2000). Size fair and homologous tree crossovers for tree genetic programming. *Genetic Programming and Evolvable Machines*, 1(2):95-119.

Langdon, W. B. & Poli, R. (1997). Fitness causes bloat. In Chawdhry, P. K., Roy, R. and Pan, R. K., editors, *Proceedings of the Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing*, pages 13-22. Springer-Verlag, London.

Luke, S. (2003). Modification point depth and genome growth in genetic programming. *Evolutionary Computation*, 11(1):67-106.

Luke, S. & Panait, L. (2002). Fighting bloat with nonparametric parsimony pressure. In Merelo Guervós, J. J., Adamidis, P., Beyer, H.-G., Fernández-Villacañas, J.-L. and Schwefel, H.-P., editors, *Proceedings of Parallel Problem Solving from Nature VII*, LNCS 2439, pages 411-420, Granada, Spain. Springer-Verlag, Heidelberg.

Mathias, K. E. & Whitley, D. (1994). Initial performance comparisons for the Delta Coding algorithm. In *International Conference on Evolutionary Computation*, pages 433-438.

Mayer, H. A. (1998). ptGAs--genetic algorithms evolving noncoding segments by means of promoter/terminator sequences. *Evolutionary Computation*, 6(4):361-386.

McPhee, N. F. & Poli, R. (2001). A schema theory analysis of the evolution of size in genetic programming with linear representations. In Miller, J. F., Tomassini, M., Lanzi, P. L., Ryan, C., Tettamanzi, A. and Langdon, W. B., editors, *Genetic Programming, Proceedings of EuroGP'2001*, 2038, pages 108-125. Springer-Verlag.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

Nordin, P. & Banzhaf, W. (1995). Complexity compression and evolution. In Eshelman, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA95)*, pages 310-317. Morgan Kaufmann.

Panait, L. & Luke, S. (2004). Alternative bloat control methods. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E., Darwen, P., Dasgupta, D., Floreano, D., Foster, J., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D. and Tyrrell, A., editors, *GECCO 2004: Proceedings of the Genetic and Evolutionary Computation Conference*, LNCS 3103, pages 630-641, Seattle, Washington. Springer, Berlin, Germany.

Poli, R. (2000). Exact schema theorem and effective fitness for GP with one-point crossover. In Whitley, D., Goldberg, D. E., Cantu-Paz, E., Spector, L., Parmee, I. and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 469-476. Morgan Kaufmann.

Poli, R. & Langdon, W. B. (1997). A new schema theory for genetic programming with one-point crossover and point mutation. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H. and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 278-285. Morgan Kaufmann.

Poli, R. & McPhee, N. F. (2001). Exact schema theorems for GP with one-point and standard crossover operating on linear structures and their application to the study of the evolution of size. In Miller, J. F., Tomassini, M., Lanzi, P. L., Ryan, C., Tettamanzi, A. and Langdon, W. B., editors, *Genetic Programming, Proceedings of EuroGP'2001*, 2038, pages 126-142. Springer-Verlag.

Ramsey, C. L., De Jong, K. A., Grefenstette, J. J., Wu, A. S. & Burke, D. S. (1998). Genome length as an evolutionary self-adaptation. In Eiben, A., Back, T., Schoenauer, M. and Schwefel, H.-P., editors, *Proceedings of Parallel Problem Solving from Nature V*, LNCS 1498, pages 345-353, Amsterdam, Netherlands. Springer-Verlag, Heidelberg.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog, Stuttgart, Germany.

Rowe, J. E. & McPhee, N. F. (2001). The effects of crossover and mutation operators on variable length linear structures. In Spector, L., E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors., *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001.*, pages 535-542, San Francisco, CA. Morgan Kaufmann.

Ryan, C., Collins, J. J. & O'Neill, M. (1998). Grammatical Evolution: Evolving programs for an arbitrary language. In Banzhaf, W., Poli, R., Schoenauer, M. and Fogarty, T. C., editors, *Proceedings of the First European Workshop on Genetic Programming*, 1391, pages 83-95. Springer-Verlag.

Schwefel, H. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhauser, Basel.

Smith, S. (1983). Flexible learning of problem solving heuristics through adaptive search. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI 83)*, pages 422-425, Karlsruhe, Germany.

Soule, T. & Foster, J. A. (1998). Removal Bias: a new cause of code growth in tree based evolutionary programming. In *Proceedings of 1998 IEEE International Conference on Evolutionary Computation*, pages 781-786. IEEE Press.

Soule, T., Foster, J. A. & Dickinson, J. (1996). Code growth in genetic programming. In Koza, J. R., Goldberg, D. E., Fogel, D. B. and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 215-223. MIT Press.

Stephens, C. R. & Waelbroeck, H. (1997). Effective degrees of freedom in genetic algorithms and the block hypothesis. In Back, T., editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, pages 34-40, Lansing, MI. Morgan Kaufmann.

Stephens, C. R. & Waelbroeck, H. (1999). Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109-124.

Stringer, H. & Wu, A. S. (2004). Winnowing wheat from chaff: The Chunking GA. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D. and Tyrrell, A. M., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, LNCS 3103, pages 198-209, Seattle, WA. Springer, Berlin.

Stringer, H. & Wu, A. S. (2005). Behavior of finite population variable-length genetic algorithms under random selection. In *Genetic and GECCO 2005: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1249-1255, Washington D.C.

Tackett, W. A.(1994). *Recombination, Selection, and the Genetic Construction of Computer Programs*. Doctoral Dissertation, University of Southern California, Department of Electrical Engineering Systems.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149-175.

Wu, A. S. & Garibay, I. (2002). The proportional genetic algorithm: Gene expression in a genetic algorithm. *Genetic Programming and Evolvable Hardware*, 3(2).

Wu, A. S. & Lindsay, R. K. (1996). A comparison of the fixed and floating building block representation in the genetic algorithm. *Evolutionary Computation*, 4(2):169-193.

Wu, A. S., Schultz, A. S. & Agah, A. (1999). Evolving control for distributed micro air vehicles. In *Proceedings of IEEE Computational Intelligence in Robotics and Automation Engineers Conference, 1999*.

Wu, A. S. & Stringer, H. (2002). Learning using chunking in evolutionary algorithms. In *Proceedings of the 11th Conference on Computer-Generated Forces and Behavior Representations*, pages 243-254, Orlando, FL.