2008

The role of information flow in engineering optimization

Steven Michael Corns
Iowa State University

The role of information flow in engineering optimization

by

Steven Michael Corns

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Program of Study Committee:
Kenneth Mark Bryden, Major Professor
Daniel Ashlock
H. Scott Hurd
James Oliver
Tom I. P. Shih
Eliot Winer

Iowa State University

Ames, Iowa

2008

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to give my sincere thanks to all of those who helped make the completion of this dissertation possible. First I would like to thank my friend and advisor Kenneth Mark Bryden for his inspiration, insights and encouragement. His guidance in engineering, education, and professional service has proven invaluable in my research and professional pursuits.

I would also like to thank Dr. Daniel Ashlock, who shared with me his enthusiasm for computational intelligence and has helped me appreciate the breadth of problems it can be used to address. Dr. Ashlock has been a constant source of information and guidance in my chosen professional research. For his instruction, support, and patience, I would also like to thank Dr. H. Scott Hurd. Dr. Hurd helped me to broaden my work into the area of veterinarian medicine. To the faculty, staff, and students of the Virtual Reality Applications Center at Iowa State University I give my thanks for the many years or support and camaraderie.

I would like to thank my parents, Charles and Nancy Corns. Without their encouragement and assistance this work would not have been possible. Most importantly I would like to thank my wife Melissa, to whom this work is dedicated, for her love and understanding throughout my academic career.

ABSTRACT

Current optimization techniques work well for single components represented by a single model. However, many of the problems we face today are multi-disciplinary problems requiring the integration of complex models from different fields to gain a more complete understanding of the overall performance of a biological, engineering, or human system. One example is a modern automobile. Multiple systems (such as the power train and electronic engine control system) are designed and built from various assemblies and components, all of which are then integrated into one final product. This design process evokes a systems-of-systems concept that is also found in agricultural facilities, aircraft design, and many other industrial applications where multiple systems are orchestrated to achieve common goals. Optimization of these complex systems is challenging. Tight coupling between the various models, discontinuous search spaces, and long run times can quickly defeat traditional optimization techniques.

Evolutionary algorithms provide a way to approach optimization of these complex systems. Evolutionary algorithms blend the information contained in a population of solutions to answer problems that thwart many classical optimization methods, but loss of diversity in the evolving solutions is a critical issue. As this information is shared between the population members, the diversity in that population decreases as the solutions converge to a single answer. For many challenging engineering problems this loss of diversity occurs too rapidly for novel solutions to emerge. In addition, systems of systems optimization problems are often deceptive because the global optimum is composed of multiple building blocks, making the preservation of diversity crucial.

This work presents graph based evolutionary algorithms as a tool to control the rate at which information is spread throughout an evolving population and thereby limit diversity loss. Graph based evolutionary algorithms impose a computational geography on the evolving population, placing barriers to information flow to allow for the development of the building blocks required to assemble one or more superior solutions. Graph based evolutionary algorithms can be used to find new solutions and decrease the time to convergence to a global optimum for complex, deceptive problems. In addition, the performance of a problem on a set of graphs can be used as a taxonomical character to classify evolutionary computation problems. If comparisons can be made between classified problems and a new problem being examined, it would be possible to select a graph that matches the desired performance. This careful graph selection can provide solutions that are both novel and superior to those found by standard evolutionary algorithms. Successful examples can be found in a variety of disciplines, including the engineering design problem of optimizing cook stoves for use in the third world to biological systems-of-systems, such as the tailoring of antibiotic regimens for use in swine production.

1. INTRODUCTION

Optimization has long been a significant part of engineering. After a new solution is introduced, efforts are made to make it faster, less expensive, more efficient, or improve the design in some fashion. For most of engineering history there was no conscious practice of optimization; improvements to designs were typically done on a trial-and-error basis while the technologies were being developed and then by examination of performance when it was placed in service.

In the 1940's, optimization using mathematical models was introduced (Vanderplaats, 1999). These classical techniques, such as linear programming or Newtown's method, manipulated the mathematical models of components or systems using calculus or gradient methods. The maxima (or minima) found using these methods was typically sufficient for a single component, although there was no guarantee that it is the global optima.

In 1975, Holland put forth the idea that computer code could be manipulated to mimic natural selection (Holland, 1975). This was further explored by DeJong (1975), who proposed that this technique could be used to solve a wide variety of problems. These ideas have been merged with other concepts, such as evolution strategies, evolutionary programming and genetic programming to allow for a larger selection of representations, making it possible to successfully attack a much broader range of problems. These resulting evolutionary algorithms (EAs) provide an approach to solving and optimizing many mathematics, physics and engineering problems.

EAs are a valuable optimization tool. They have been used in the design of gas turbine blades (Martin and Dulikravich, 2002), airfoils (Jang and Lee, 2000), steam boilers (Vavak, Jukes and Fogarty, 1997), missile nozzle inlets for high-speed flow (Blaize, Knight and Rasheed, 1998) and heat exchangers (Fabbri, 1997). EAs are not as vulnerable to problems with early convergence as gradient search methods, and are able to find solutions to problems with discrete or discontinuous landscapes that are unsolvable by most other optimization techniques. They are also capable of solving high dimension problems that would thwart conventional methods. These algorithms work by blending different members of a solution population to generate new, novel and hopefully superior solutions through simulated evolution.

Combinatorial graphs have recently been combined with evolutionary algorithms to impose a spatial geography on the population of solutions. These graph based evolutionary algorithms (GBEAs) allow for a better control of diversity and time to convergence, preventing early termination of the algorithm when a sub-optimal solution to a deceptive problem is found. The members of the population are each placed on a vertex of a graph that is connected to a set number of other population members. When the program is started, population members are only allowed to mate with individuals that are connected to them by the graph's structure, controlling the rate that their information is spread.

Slowing the rate of information spread is vital for problems where the optimal answer is difficult to find and it is easy to find a good but sub-optimal answer. By imposing a spatial geography, sub-optimal answers are prevented from rapidly spreading across the population, destroying diversity. Additionally, as a sub-optimal answer spreads the average fitness increases. As a result it becomes more difficult for solutions that are significant different from the norm to survive. This creates a substantial barrier to maintaining a diverse population of solutions. Disparate individuals who mate with a creature nearer the average fitness are usually subject to replacement by their children, who are more similar to the sub-optimal leader.

The mechanisms and measures of information flow within a GBEA are not well understood. This research examines the dynamics involved in the controlling the spread of information across the solution set and develops a methodology for examining the rate of information spread. By understanding this transfer rate, an EA developer can select a graph that will give the best results depending on the problem at hand. In addition, this research investigates the interaction between population size and information spread in GBEAs.

1.1. Overview

Chapter Two provides an overview of evolutionary algorithms and the methodologies that led to their development. The parameters that affect the amount of information available initially and as the evolutionary proceeds are discussed, as well as methods to control diversity. The chapter concludes with a review of previous test suites and a

discussion of the No Free Lunch theorem, which states that there is no one best method to approach all evolutionary computation problems.

Chapter Three presents graph based evolutionary algorithms. A discussion of how GBEAs differ from standard evolutionary algorithms is given, followed by a description of several test problems used to investigate these algorithms. This is followed by a discussion of the development of a taxonomy of evolutionary computation problems by using information gained from GBEA research. This taxonomy makes it possible to apply a priori knowledge to select the graph that matches the desired outcome of the problem being solved. In addition, this taxonomy can be used to develop an effective test suite of evolutionary computation problems, making it possible to evaluate new techniques in the field and determine where their strengths lie.

Chapter Four investigates the rate at which information is transferred within a GBEA. First, the impact of population size on GBEAs is discussed. Of particular interest is the difference between initial diversity supplied by population size and the preservation of diversity achieved with GBEAs. Next, computational and empirical methods for finding the takeover times for GBEAs are discussed and the results are given, making it possible to determine the extent to which diversity is preserved for a particular graph. This chapter concludes with some recommendations on the type of graph to use for a given problem.

Chapter Five lists several applications in which the use of GBEAs has proven to be beneficial. While several applications are described, the use of GBEAs to develop an

antibiotic regimen for use in the swine industry is examined in detail. After presenting some background information on the use of antibiotics with swine and the effects of the antibiotic on bacteria, a review of existing bacteria models and swine growth modeling is given.

Chapter Six concludes the dissertation with a review of the benefits of using GBEAs, some qualitative rules for achieving the desired results with GBEAs in optimization problems. Chapter 7 lays out the direction of future work in this area. Appendix A gives an overview of graph theory to help in the understanding of this research and a description of the graphs used.

2. EVOLUTIONARY ALGORITHMS

Evolutionary algorithms attempt to mimic proposed natural evolution processes in computer code to develop novel and useful solutions to a variety of problems. Evolutionary algorithms encompass several earlier techniques that share similar attributes: evolution strategies, evolutionary programming, genetic algorithms (GAs), and genetic programming (GP) (Parmee, 2001). All of these methods share common traits, such as having several solutions being developed at once (population based) and using current solutions in the populations (parents) to develop newer solutions (children). As these techniques are the key components that form today's methodology, a brief overview of each is in order.

Evolution strategies (Rechenburg, 1984; Scwegel, 1975) and evolutionary programming (Fogel, Owens, and Walsh, 1966) are similar but independently developed methods for evolving solutions. Both use solutions of real valued strings, which could be compared to chromosomes, to form new solutions by changing one or more values in the string (mutation), typically using a normal distribution from the initial value. The major differences between the two are that evolutionary strategies use every member of the population to develop children that are compared to the parents, possibly using more than one parent to make the children (similar to recombination in genetics, where genetic material is contributed from more than one parent.) Evolutionary programming uses only the most fit of a group of population members to produce children for the next generation.

Genetic algorithms (DeJong, 1975; Holland, 1975) use a form of natural selection to determine which individuals in the population mate. Chromosomes are replaced with data arrays that can hold variables in an equation, instructions for controlling a virtual agent, or many other forms of information. These data arrays are normally made up of binary strings that represent real numbers or integers, although they can contain real values or program directions for controlling artificial agents. The solution set is then populated with different creatures made up of one or more chromosomes, usually constructed randomly. The major difference between GAs and the previous methods is the selection method used. Evolution strategies and evolutionary programming used only the most fit members of the population to produce offspring, while genetic algorithms allow for any member of the population to reproduce, although this is weighted to favor the more fit individuals. Those population members selected to breed then undergo one or more operators meant to simulate natural mating phenomenon, such as cross-over of parent chromosomes or random mutation of a data array values. In the simplest form, the children generated then replace the parents in the population, and the process is repeated until a satisfactory solution to the problem is achieved. The children may also be compared to all or part of the population using a fitness value that gives some indication of how well they solve the problem at hand, replacing inferior solutions if any are found.

Genetic programming (Koza, 1992) is significantly different from genetic algorithms, evolution strategies and evolutionary programming. Whereas the previous methods operated on numerical values, either binary or real strings, genetic programming is intended to evolve computer code. This is accomplished by using parse trees to store

formulas, with the internal nodes being operators, and the terminals being either constants or variables (terminals). Genetic programming is very similar to genetic algorithms in that it emulates natural selection in how parents are selected for breeding, and shares many of the same operators such as crossover and mutation. These operators, however, are necessarily altered slightly due to the differences in representation. When crossover is performed, care must be taken to ensure that the generated program is executable. This means that operands can only be crossed with operands, and terminals can only be switched with terminals. More information on genetic programming can be found in (Banzhaf, Nordin, Keller, and Francone, 1998; Kinnear, 1994; Koza, 1992).

Each of these methods has strengths and weaknesses. Evolution strategies and evolutionary programming have long run times, while crossover is often disruptive in genetic algorithms and even more so in genetic programming. There is also a problem with mutation having different impact on a binary string depending on the position at which the mutation is applied. Evolutionary algorithms can be thought of as using different aspects from these methods to develop a representation of the problem being studied that can avoid a particular method's weakness. Using this representation, it is then possible to choose parameters for the algorithm that properly explore the search space.

2.1. Evolutionary Algorithm Parameters

There have been several attempts to gain an insight into how different parameters in an evolutionary algorithm effect the time to convergence and/or the quality of the final

answer. Some of these parameters effect how the evolutionary algorithm is initialized, such as problem representation and population size. Other parameters such as local mating rules dictate how the algorithm simulates mating in nature. This includes selection method, crossover rate and type, mutation rate and type, and how the children are introduced into the population. To discuss the role of information flow when using evolutionary algorithms, it is first necessary to discuss the parameters that have the strongest impact on solution diversity.

2.1.1. Population Size

The population size is the number of creatures in the population available to breed. The first estimation for an optimal population size was the Schema Theorem for genetic algorithms introduced by Holland (1975), which estimated a population size of the order of magnitude of n^3 would ensure that there was a sufficient representation of possible combinations to solve the problem, where n is the length of the bit string being used. Grefenstette (1986) studied the effects of varying population size, as well as many other parameters, on the five problems introduced by DeJong in his doctorial dissertation (1975). He found that for the binary string problems he studied, a population size ranging from 30 to 100 provided sufficient diversity to solve these problems.

Goldberg has done the most extensive testing of population variations (Goldberg, 1989; Goldberg, Deb, and Clark, 1992; Goldberg, Sastry, and Latoza, 2002) along with Harik (1997.) Together they used the Schema Theorem (DeJong, 1975) to estimate the best

population sizes based on statistical analysis for the availability of necessary building blocks and probability of failure to select those blocks during mating. While this method was limited to specific section and crossover methods (Altenberg, 1995), it provides guidelines for population size based on the building blocks necessary to construct the final solution. Arabas, Michalewicz and Mulawka (1994) developed GAVaPS (genetic algorithm with varying population size), giving each member of the population a lifespan in generations based on their fitness, causing the more fit members to last longer and therefore produce more offspring as the algorithm progressed. Nimwegen and Crutchfield (2001) conducted research into population size by examining the epochal behavior of genetic algorithms. They found that finding population size and mutation rate combinations that prevent highly superior creatures from developing rapidly allow for faster convergence to a true optimal solution in all but the simplest cases.

All of these previous studies only examined populations of binary strings. While binary strings are useful in some applications, many real-world engineering problems require optimization of real valued functions. One example of the use of real valued functions is the work of Haupt and Haupt (1998; 2000), who performed experiments varying the population size and mutation rate for real valued functions. They found that for the real valued problems they studied a smaller population size and low mutation rate performed best, which is consistent with the observations made when evolutionary strategies are applied to similar problems (Rechenburg, 1984.) It should be noted that while the test problems used in their study (2000) are engineering problems, they are non-deceptive problems with simple fitness landscapes that can be solved with little difficulty. Real

values can be represented using a base two numerical system, but there are major drawbacks to this approach; it can require very long gene length as well as making the mutation operator unstable. A bit flip at different locations on the string could have drastically different impact on the value of the string, making crossover and mutation highly disruptive. This is commonly avoided by the use of gray coding which makes each bit flip have equal value (an implementation can be seen in Mathias and Whitley's work (1994)), but this creates unnecessary overhead in the algorithm, as real value coding could easily be used without as much added computational cost.

2.1.2. Selection Method

Selection method is normally associated with genetic algorithms and genetic programming. There are two major schemes to be chosen from when determining a selection method; whether to use a generational algorithm (DeJong, 1975) or a steady state algorithm (Reynolds, 1992; Syswerda, 1991; Whitley, 1989). The difference between these two being that a generational algorithm involves all members of the population while the steady state (originally termed the "GENITOR" algorithm by Whitley (1989)) selects individuals and performs mating on just the pair selected. After one of these mating schemes is chosen a method for pairing up individuals for the actual mating still needs to be determined. These are the methods in which GAs and genetic programming emulate natural selection. Some of the most popular are fitness proportionate selection (random selection of parent with higher fitness having a better chance of selection), rank selection (like fitness proportionate selection, but arranging the

solutions by fitness and then using this "rank" for selection) and tournament selection (the population is divided into groups, with the most fit members of the group reproducing) (Parmee, 2001). Closely related to selection method is the algorithm's replacement method, which can be absolute (child replaces a population member regardless of its fitness) or elite (child replaces a population member only if it is more fit).

2.1.3. Crossover

The crossover operator is how most evolutionary algorithms perform recombination to generate offspring. It is normally performed by randomly choosing one or more points on the selected parents' strings, then producing a child by copying the data first from one parent, then switching to the other whenever a crossover point is reached (called single or multiple point crossover). Another method is uniform crossover, where the donating parent is randomly determined for each locus. The chance of crossover occurring, normally referred to as the crossover rate, and how the crossover is accomplished have been studied by several researchers. One of the first attempts to determine what crossover rate should be used for an evolutionary algorithm was done by Grefenstette (1986). He also studied mutation rate and selection method in determining GA parameters. Although he found that genetic algorithms performed as well or better than other methods for a small problem set, no recommendations were made in his conclusions. DeJong and Spears (1990) compared multiple point crossover to uniform crossover, where they found that for small population sizes, uniform crossover performed

better. Researchers at UIUC (Goldberg, 1989; Goldberg and Deb, 1991; Goldberg, Deb, and Clark, 1992; Goldberg, Deb, and Thierens, 1993) studied the effects of crossover on the availability of building blocks. Wu, Lindsay and Riolo (1997) examined crossover and mutation as it occurred during a GA run, the results of which showed that diversity preservation was a major influence on time to convergence.

2.1.4. Mutation

Mutation changes the value at one or more location in the evolving solution, thereby making it possible to introduce an entirely new solution into the population. This is an effort to mimic mutations that occur naturally in living species. Mutation is normally conducted by randomly changing one or more values in a newly produced child before its fitness is evaluated. Much of the work done investigating evolutionary algorithms previously mentioned also explored the effect of varying mutation rate and the mutation itself (Goldberg, 1989; Greffenstette, 1986; Nimwegen and Crutchfield, 2001; Wu, Lindsay, and Riolo, 1997). One area of interest that has recently been studied is varying the mutation rate as the algorithm proceeds in an attempt to fine-tune the use of the operator (Smith and Fogarty, 1996). All research in genetic evolution can be considered studies in mutation, as this is the only operator used (Fogel, Owens, and Walsh, 1966).

2.2. Diversity and Convergence in Evolutionary Algorithms

Whenever an evolutionary algorithm is applied to a problem there is always a tradeoff between exploration of the search space and exploitation of the superior solutions already

found (Parmee, 2001). The exploration of a search space refers to how many different solutions are analyzed, and can be evaluated by taking inventory of the diversity of the members of the population. Diversity, as it applies to evolutionary algorithms, can be thought of as how many different solutions or partial solutions exist the in solution population. The exploitation of superior solutions can be seen as the population members utilize information already discovered by other members of the population to converge to a single and hopefully optimal solution.

The balance between exploration and exploitation required is problem specific, and is best highlighted by examining two extremes, a simple uni-modal evolutionary computation problem and a highly deceptive evolutionary computation problem. In a simple uni-modal problem, a change to the chromosome that leads to an improved fitness also leads to the global optimum. A highly deceptive problem in evolutionary computation is a problem in which any change in the chromosome that leads to an improved fitness causes the solution to converge to a sub-optimal answer. These two extremes require the maximum amount of exploitation and exploration, respectively.

For very simple and non-deceptive problems it is preferable to converge to a solution with little or no diversity required. As the fitness landscape grows more difficult (through increasing size, modality or deceptiveness) the need to preserve diversity increases (Parmee, 2001). Unfortunately, there is currently no method for determining beforehand how much diversity is necessary other than an experienced user's estimate.

Exploration and exploitation can be balanced by adjusting the various parameters of the algorithm being used. Initially, the amount of diversity is set by the population size used in the algorithm. In most evolutionary algorithms this diversity is then decreased by the selection methods and crossover, and increased by mutation. Other methods exist to control the rate at which diversity is destroyed, and will be discussed in Chapter 3.

There has been a significant amount of work done by Goldberg and Deb (1991) to determine the amount of time (measured in mating events) necessary for a single solution to dominate all others in the population. In its simplest form, his "takeover time" is based on separating solutions into different classes and then using probability theory to predict the number of members of that class there will be in the next generation. This was done for several selection schemes, including Whitley's GENITOR algorithm (1989). This gives a general idea of the number of mating events required for a method to converge to a solution, but does not take into account the modality of a problem. For this method to give a better insight into the true takeover time for an algorithm, other factors must be considered.

2.3. Evaluating Evolutionary Algorithms and the No Free Lunch Theorem

Whenever a new algorithm is developed, the creators need some method to evaluate its performance against other algorithms. The most common way to do this is to either find or develop a test problem or suite of test problems. There are several test problems and test suites available both on-line and in the literature (Ackley and Littman, 1992; DeJong,

1975; Mühlenbein, Schomisch, and Born, 1991; MDO Test Suite, 2002; Schaffer, Caruana, Eshelman, and Das, 1989). The authors of each test problem have applied their particular algorithm to the test problem as well as algorithms produced by others and made comparisons between the results. Unfortunately, there is often little emphasis placed on why a certain algorithm outperforms another and many of these test suites are unwittingly biased towards the new algorithm.

These issues along with several other issues related to test suite choice were discussed by Whitley, Mathias, Rana, and Dzubera (1996). In this work no test suite was proposed, but desirable characteristics for good test suites were proposed. These included: 1) the test suites should be resistant to hill-climbing techniques, since problems that can be solved with hill-climbers are solved faster and with better results using these methods, 2) test suites should also be nonlinear, non-separable and non-symmetric, as these types of problems can be decomposed and simplified into smaller parts that can be optimized on their own, 3) improving solutions with many interdependencies is one of the strengths of EAs, and should be stressed, and 4) test suites should be scalable in both the test function and the problem evaluation cost. Many real world problems need to be scalable in the number of variables of interest that are being manipulated. Also, when more variables are introduced into a problem the evaluation cost often increases at an exponential rate. An algorithm proposed for inclusion in a test suite needs to be able to manage this size increase without experiencing a degradation in performance.

Wolpert and Macready (1995; 1997) developed two theories to examine whether any algorithm was superior to another; one based on algorithms and one based on problem type. One of the key points of this work was that it can be proven that when compared to the entire population of problems available no algorithm is superior to another. This was contrary to the common opinion of the time, which held that some algorithms were intrinsically better than others, mainly due to the previously mentioned flaws in common test suites and problems. Called the "No Free Lunch Theorems," they showed that there is a relationship between problems and algorithms, indicating that there is no one algorithm that will outperform all others in a wide variety of problems. This makes comparisons of different algorithms difficult at best.

2.4. Diversity Control Techniques

There have been many theories as to why diversity is necessary and has not vanished in nature (Kimura and Crow, 1963; Wright, 1986). Many of these theories suggest that geographical obstacles and inherent mating rules (such as "mating dances") impose mating restrictions and hence preserve diversity. When obstacles such as these are applied to EAs, it helps to preserve the diversity and slow the time to convergence, which in turn helps keep the algorithm from getting stuck in local optima for deceptive problems (Ackley and Littman, 1992; Mühlenbein, 1991). There are several methods currently being employed to maintain, increase or otherwise control solution diversity in a population. These include the initial parameters of the algorithms, Niche Specialization, the Island Model (GENITOR II) and the EcoGA (Davidor, Yamada, and Nakano, 1993).

2.4.1. Population Size, Crossover, and Mutation

The first methods employed in evolutionary algorithms to control diversity were the population size, crossover and mutation. In fact, these methods were used in EAs before the necessity of diversity preservation was understood! An examination of these parameters shows how they contribute to diversity control. The larger the population size, the more "genetic materials" or building blocks (Goldberg, 1989) are available with which to build solutions. As the evolutionary process continues, crossover blends the solutions, making children that tend to match the higher fitness parent and thereby lowering diversity. Traditionally, the only way to re-introduce diversity was by applying a mutation operator. When a mutation occurs, part of the information in the solution is changed, which usually makes it different from other solutions it may have been similar to and increasing diversity.

2.4.2. Niche Specialization

Niche specialization (Goldberg, 1989) is a method that discourages solutions that are very similar by imposing fitness penalties to those individuals. The concept is similar to the biological concept that shares its name. It is an easily understood concept that usually yields good results; as more individuals find the same high performance solution, their fitness decreases, encouraging exploration of other areas of the search space. While the concept is rather simple, it can sometimes be difficult to employ since there needs to be some similarity measure to determine how different a solution is from others in the population. A hamming distance can easily be found for binary encoded problems, but if

there are a significant number of building blocks this could lower the methods effectiveness if there is no a priori knowledge of these blocks. For other problems, it is usually computationally expensive to calculate a similarity measure, if it is even possible at all. As the problem complexity increases, usually the difficulty in finding and computing the similarity measure increases as well.

2.4.3. GENITOR II

One approach to preserve diversity is the GENITOR II algorithm, also referred to as island Gas (Whitley and Starkweather, 1990), which simulates the natural separation of population members caused by land masses separated by bodies of water. In this approach, several populations are evolved separately for a given number of generations. Selected members are then copied from one of these "islands" onto another using a predetermined pattern in a process called migration, hopefully adding new and useful information into the receiving population. The number of members moved in a migration is termed the migration size, and the number of generations between migrations is the migration interval. These can be adjusted to control the rate at which information is shared between populations.

This method not only prevents early convergence for deceptive problems, it is also easily implemented on parallel computers, greatly lowering the amount of time required for results. A drawback of this method is that due to the inherit elitism, there are limits to how much diversity it can preserve.

2.4.4. EcoGA

The EcoGA (Davidor, Yamada, and Nakano, 1993) was another early GA to impose a geographic structure on a solution population to control diversity. This algorithm places the population of solutions onto a grid that wraps around to form a continuous space. Acting as a steady state algorithm (Reynolds, 1992; Syswerda, 1991; Whitley, 1989), members of the population are selected for mating, but only allowed to interact with neighboring solutions, creating a subpopulation of nine members for each event. This is very similar to graph based evolutionary algorithms, acting as a single type of graph with a degree of k=8. This work did show some promise, but was abruptly stopped for unknown reasons.

3. GBEAs AND THE EVALUATION OF EVOLUTIONARY COMPUTATION METHODS

To evaluate the role of information flow in engineering optimization, we first need a method to control that flow of information. For this research, graph based evolutionary algorithms (Bryden, et al., 2006) are employed. GBEAs use graph structures to control the rate at which information is spread through an evolving population. While exploring how these GBEAs can be used to control the flow of information, a large body of research data was collected, making it possible to construct both a taxonomy of evolutionary computation problems from those studied and to develop a test suite of problems that would allow an unbiased evaluation of newly proposed evolutionary computation methods. Together, these tools would make it possible to select a desirable level of information flow using a priori knowledge of the problem.

Research has shown that GBEAs can provide significant improvement in time to solution when the appropriate graph is selected for the problem being solved (Bryden, et al., 2006). The initial set of graphs used was chosen ad hoc and with little or no idea of which graph would perform well. As of yet, there is still only a limited understanding of what occurs within the population to make one graph perform better than another. By developing a better understanding of these interactions it would be possible to apply other graphs and develop new graphs that could be used to fine tune the rate of information flow.

With a wide variety of graphs to choose from, there is still a problem in determining which graph to use. Currently, the assignment of which graph would perform best is based largely on conjecture, with only modest accuracy. To improve the selection of preferred graph, it is necessary to not only gain a better understanding of the interactions that occur within a population of solutions in a GBEA but also to develop a means to classify evolutionary computation problems. While a classification of problems has been introduced (Ashlock, Bryden, Corns, and Schonfeld, 2006), more data needs to be incorporated into the hierarchy and the relationships between the problems examined before any guidelines could be established for graph usage. To do this, a variety of problems need to be examined.

In building the necessary information to make predictions about which graph would be best, it was also be possible to develop a preliminary test suite of evolutionary computation problems. To get a representative sample of the problems to which evolutionary computation would be applied, several types of test problems were examined. Using the taxonomical values found while classifying the problems, it is possible to determine which evolutionary computation problems are significantly different from one another. This would help to decrease the potential for bias in test problems, making it possible to develop a test suite to evaluate new evolutionary computation techniques and make an accurate comparison to existing methods.

3.1. Graph Based Evolutionary Algorithms

Graph based evolutionary algorithms are a method of preserving diversity by using graphs to impose an artificial geography on the population. In these graph based evolutionary algorithms (GBEAs), the population members are separated by limiting the number of other members they are allowed to interact with. Appendix A gives an overview of the graph theory used to design this geography. GBEAs control information flow, unlike the GENITOR II algorithm (Whitley, 1989) that prohibits information flow except during migration. This is done by assigning each member of the population to a vertex V(G) of the graph G, and if that vertex shares an edge E(G) with another vertex it is possible for it to mate with the individual assigned to the second vertex. A steady state evolutionary algorithm is then used, where the evolution occurs one mating event at a time. Mating is conducted by first randomly selecting a member of the population, and then selecting its mate by fitness proportional selection of the vertices it shares an edge with. This proceeds until one of the members of the population achieves a fitness level greater than a specified fitness goal or the algorithm times out. By varying the graph diameter and degree, the rate at which information spreads can be controlled, allowing a means for controlling diversity loss.

Initial research has been done using various types of graphs with a population size of 512 vertices (Bryden, Ashlock, Corns, and Willson, 2006), as described in Appendix A. Of the graphs used, the complete graph had the smallest diameter, followed by the hypercube graph. The toroidal graphs and the Petersen graphs had the next larger diameters, with the cycle graph having the largest diameter. Figure 1 shows examples of

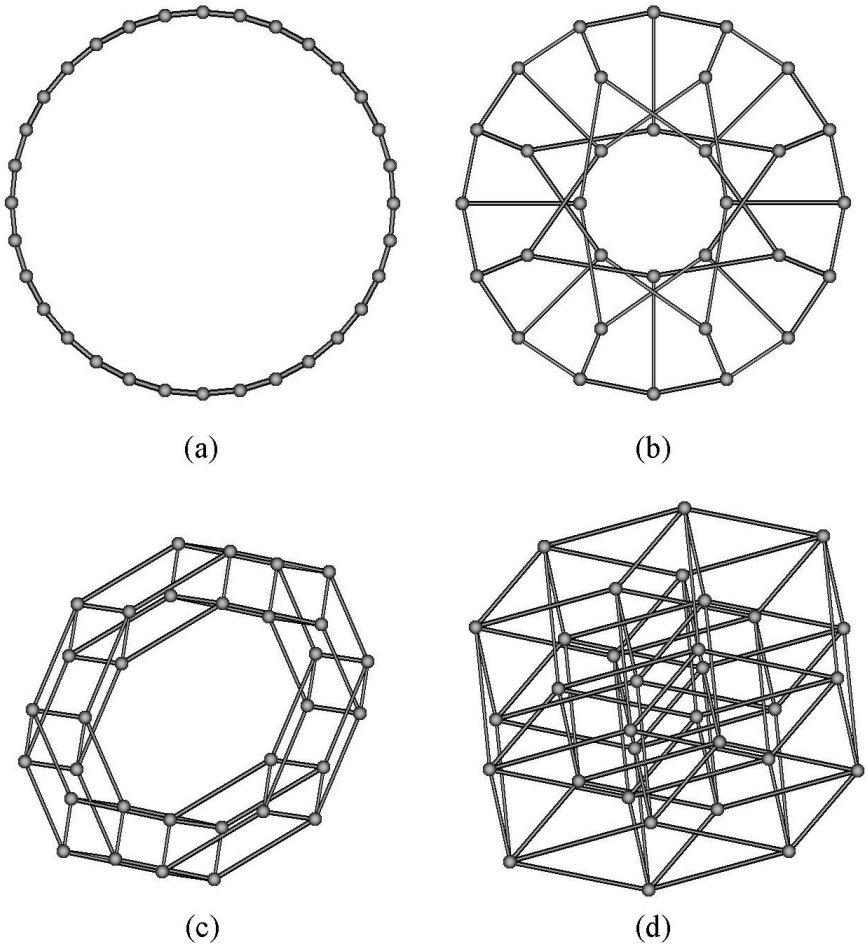

Figure 1, Graph types used in this study: Cycle (a), Petersen (b), Toroid (c) and Hypercube (d).

the cycle graph (a), the Petersen graph for k=3 (b), the toroid graph for m=4 (c) and the hypercube graph (d) for a population size of 32. Information on the graphs used in GBEA studies can be found in Appendix A. This initial study showed that problems with a simple fitness landscape are best solved by graphs with the smallest diameters, with the complete graph performing best (had the fewest mating events to solution.) For problems with a deceptive landscape, it was found that graphs with the largest diameters performed well, with the cycle graph performing best. This gave results indicating that the number of mating events to solution grouped together into different families (similar types of graphs). While these results were promising, more investigation is necessary to test the robustness of the method and to determine how best to apply these new tools.

3.2. Test Problems

To establish a taxonomy of evolutionary computation problems and develop a test suite, it is first necessary to examine a selection of problems. Intuitively, the problems selected should encompass the variety of areas of research found in evolutionary computation, including binary strings, real valued functions, genetic programming, and artificial life simulations. While no experimental evidence existed prior to this study, the inclusion of these diverse problems should guarantee adequate coverage of the problems being explored in the area of evolutionary computation.

The test problems within these research areas should also be significantly different from each other. Evaluation of a new evolutionary computation technique with similar problems at best increases the amount of time necessary to complete the experiments, and more likely will introduce bias in the results. The challenge is that there is no method for determining how similar two problems are without first performing experiments with them. With this in mind, several proposed test problems for evolutionary computation have been examined, and the following were selected for this study.

3.2.1. One-Max

The one-max problem is a simple string evolver. Chromosomes composed of a number of bits (20 were used when applied to GBEAs) are generated randomly and inserted into the population. Local mating rules for the algorithm are then applied, in the case of GBEAs single point crossover with a 10% chance of mutation (one of the bits is flipped) and elite replacement of the randomly selected parent. Fitness is calculated by summing the characters in the string. The algorithm was declared solved when the entire string is composed of ones, giving the largest fitness available.

3.2.2. Variable Dimension Surface (Keane Bump Test)

For the second test problem, a variable dimension surface problem (Eqns. 3-1 thru 3-4) as developed by Keane (1994) and used by Hacker, Eddy, and Kemper (1992) for a benchmark test of their hybrid genetic algorithm/hill climbers was used. It is designed to allow the user to adjust the degree of multi-modality, making the problem increasingly

difficult and deceptive. Figure 2 shows a graph of the function with two variables. The work by Hacker, Eddy, and Kemper used 2 and 10 design variables. When applied to GBEAs these values were used along with the addition of runs using 6 design variables to further investigate the equation. In this use of this test problem, the Keane Bump Test can be classified as a real valued string evolver using one point crossover and a 10% mutation rate. The mutation adds or subtracts a value ranging from 0.0 to 0.2 to the value stored in the selected string location. Elite replacement of the randomly selected parent was used and the algorithm was declared successful when a solution appeared that was within 1% of the true maximum value.

$$F(\bar{x}) = -\left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{3-1}$$

Subject to: $g_1(\bar{x}) \qquad 0.75 - \prod_{i=1}^{n} x_i \le 0$ (3-2)

$$g_2(\bar{x}) \qquad \sum_{i=1}^{n} x_i - \frac{15n}{2} \le 0 \tag{3-3}$$

$$0 \le x_i \le 10 \qquad i = 1, n \tag{3-4}$$

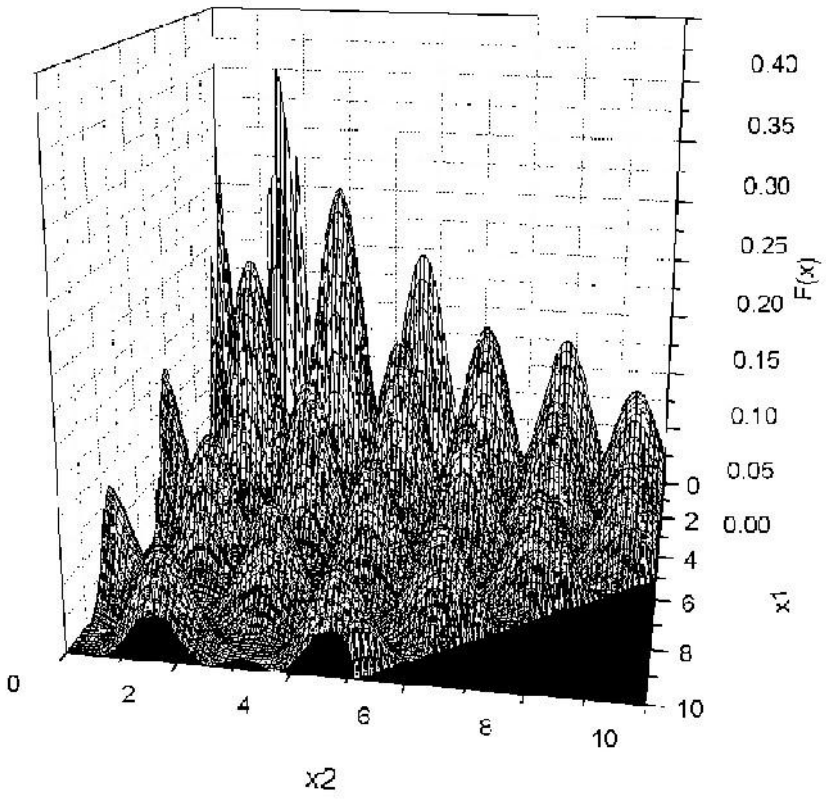

Figure 2, Keane bump test in two dimensions.

3.2.3. DeJong Test Functions

The DeJong test functions are described in detail in DeJong's doctorial dissertation (DeJong, 1975). The first function is a three-dimensional bowl and the second is a fourth-degree bivariate polynomial surface featuring a broad sub-optimal peak (also known as "Rosenbrock's Saddle"). The third function is a sum of integer parts of five independent variables creating what could be described as a six-dimensional ziggurat, being flat where it is not discontinuous. The fourth function is a fourth-order paraboloid in 30 dimensions with distinct diameters in different numbers of dimensions made more complex by adding Gaussian noise. The last function, often referred to as "Shekel's Foxholes", is a grid with many narrow local optima. These functions have been traditionally used as test problems in function optimization do not serve as a complete test suite (Whitley, et al., 1996). They are all classic GA problems in that they are composed of binary strings. Each of the variables in the solutions is represented by a string in the chromosome, which is subsequently transformed into Gray Code (Mathias and Whitley, 1994) before being used in the fitness evaluation. As used in previous GBEA studies the local mating rules were the same as those used in the one-max problem.

3.2.4. Greiwangk Function

The Greiwangk function is a sum of quadratic bowls, one per dimension, with cosine terms applied to them. These terms are subsequently translated to yield a positive function. It has a large number of local optima, making it a natural member of a test suite

(Mühlenbein, 1991.) Unfortunately, as the dimension of the Greiwangk function increases it approaches a uni-modal bowl. For this reason only five cases of relatively low dimension have been used for GBEA studies; N = 3, 4, 5, 6, and 7. As used for GBEAs, these problems share the same local mating rules as the one-max problem.

3.2.5. Plus-One-Recall-Store

The plus-one-recall-store (PORS) problem was originally developed to be included in a test suite for research conducted by Ashlock and Lathrop (1998a). It is a maximization problem that uses parse trees, which applies a basic form of genetic programming. The problem involves the efficient use of nodes so that when it is executed, the largest integer value result possible is generated when given a fixed maximum number of parse tree nodes. There are two operations (integer addition and store) and two terminals (one and recall from a memory position) in the language. Fitness for a parse tree is the number produced when it is executed. Maximum values are given in (Ashlock and Lathrop, 1998b). The difficulty of the PORS efficient node use varies depending on the congruence class (mod 3) of the maximum number of nodes. This problem has been studied in depth and the necessary building blocks for finding the solutions are well documented. For a solution to be successful, it must make use of a combination of four distinct building blocks given in (Ashlock, 2006): (+ 1 1), (+ (+ 1 1) 1), (+ Recall (Store T)) and (+ (+ Recall Recall) (Store T)). These are, respectively, 2, 3, times-2, and times-3. The 3 and times-3 blocks also have two equivalent forms: (+ 1 (+ 1 1)) and (+ Recall (+ Recall (Store T)).

In GBEA studies, experiments were run for a number of nodes equal to n=15, n=16 and n=17, the hardest, easiest and intermediate difficulty of the three classes respectively. Fitness for a parse tree was evaluated as the number produced when it was executed. The initial population for this experiment was of randomly generated trees with a number of nodes equal to the maximum. The trees then underwent crossover by randomly picking two nodes of the same type (either operation or terminal) and switching them with their corresponding sub-trees. If this resulted in a parse tree with more nodes than the maximum, a *chopping* operation was performed on the tree, which replaced the root node with one of its sub-trees until the number of nodes is equal to or less than the maximum. There was then a 10% chance that a mutation would occur, in which a new random sub-tree of the same size replaced a randomly chosen sub-tree. For all of the PORS experiments, local elite roulette mating was used.

3.2.6. North Wall Builder

The north wall builder problem uses computer-generated agents that are controlled by a genetic programming (Banzhaf, et al., 1998; Kinnear, 1994; Koza, 1992) structure called an ISAc (If Skip-Action) list (Ashlock, 2006.) An ISAc list is an array of four values [a, b, act, jump]. The first two values are indices to a data vector of the form [x1, x2, ... ,x8, 0, 1, 2], which relate to the eight grid locations surrounding the agent and the values they can contain for comparison purposes. Each of these can have a value of zero (location is empty), one (location is occupied by a block) or two (location is not on the grid). The third is an action that may be taken, with the choices being no action, jump, move

forward, turn right, or turn left. The do nothing command is inserted as a check, and should be rapidly weeded out of the population. The fourth is a location to jump to if the action is a jump command. This results in an evolvable programming language that is customizable to the problem being solved.

These agents are placed on a 7 x 7 grid, along the southern border and facing towards the spot marked 'X' on figure 3. Blocks are introduced into the grid at this marked location, with a new block appearing whenever that space is empty. The agents move on the grid as specified by the controlling ISAc list, pushing any blocks that are in front of them. This trial is ended after either 283 actions have taken place, or the bot "falls" off of the grid by moving outside of the grid's constraints. The fitness of an individual agent is evaluated by subtracting the number of grid blocks from the top (north wall) of the grid before a block is encountered from the total number of grid spaces (49). Figure 3 shows a configuration with a fitness of 42.

This fitness is then used for determining the second parent in the crossover of the ISAc structures, and for determining the finishing criteria. When applied to GBEAs, the crossover operator is used much like that in the one-max problem, with a section of the array being swapped between the two parents. The local mating rule for this problem is local elite, fitness proportional selection. This problem is of intermediate difficulty, with no mandatory building blocks as are found in the PORS 15 problem, but a high degree of solution interconnectivity as the early moves impact the effect of moves further along the ISAc list.

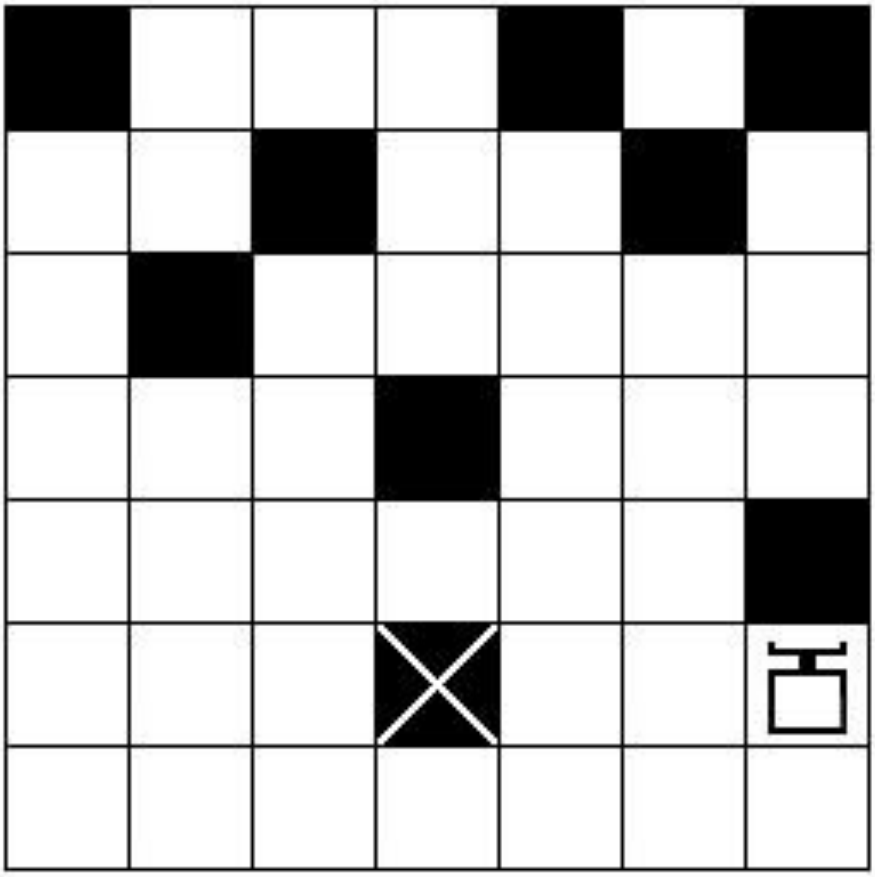

Figure 3, North wall builder sample board.

3.2.7. Self Avoiding Walk

The self avoiding walk (SAW) is a multi-modal problem with a linear string chromosome guiding an agent to efficiently cover a rectangular grid. The grids are denoted as number of columns by number of rows. The cases of the SAW problem treated in this study include 3x3, 3x4, 4x4, 4x5, 5x5, 5x6, and 6x6. For a given grid size NxM the chromosome will store NxM-1 moves, as the starting square is given to the agent for free. Each array value ranges from zero to three, representing agent movements of down, right, up, or left respectively. Starting in the lower left corner of the grid, the array dictates the path followed across the grid. The number of grid blocks the path travels through determines fitness, with the maximum (and stopping criteria) reached when all blocks are visited. Attempts to move off of the grid are ignored, but are implicitly penalized because a move is wasted. The agent is permitted to cross his own path, but this also wastes moves. The problem is called the self avoiding walk problem because the optimal solutions are self avoiding. As implemented for this study, crossover and mutation were performed in every mating event, with the crossover being two-point and the mutation operator being stochastic replacement of one array value with a new, randomly generated value.

3.2.8. DNA Barcode

DNA barcodes (Ashlock, Guo, and Qiu, 2002) are error correcting codes (McEliece, 1977) over the DNA alphabet $\{C, G, A, T\}$ that are able to correct errors relative to the *edit metric* (Gusfield, 1997). They are used as embedded markers in genetic constructs to

permit retention of source information when sequencing pooled genetic libraries. An example of their successful use to retrieve sequence source information appears here (Qiu, Guo, Wen, Ashlock, and Scnable, 2003). The algorithm used previously in GBEA studies searches for six-letter DNA words that are at a mutual distance of at least three. These are the parameters used for the wet lab testing of the technique in Qiu et al. (2003). Barcodes of this size and distance can correct one sequencing (edit) error.

3.2.9. Simple Differential Equation

Solving differential equations is a standard genetic programming problem. Modifying the usual technique, the algorithm used in GBEA studies extracts the derivatives symbolically when computing fitness. The differential equation solved was:

$$y'' - 5y' + 6y = 0 \tag{3-5}$$

which is a simple homogeneous equation with a two-dimensional solution space:

$$y = Ae^{2x} + Be^{3x} \tag{3-6}$$

for any constants A and B. A complete description and some solutions that appeared in the final population can be found in Bryden et al. (2006).

3.2.10. Steiner Systems

Steiner systems can be described as follows. For a set V of n objects, a Steiner k-tuple system on V is a set of k-subsets of V with the property that every pair of elements from V appears in one and only one of the k subsets. For the set {A;B;C;D;E; F;G} a Steiner triple system would be the set of 3-tuples: {{A,B,D}, {B,C,E}, {C,D,F}, {D,E,G}, {A,E, F}, {B, F,G}, {A,C,G}}. Notice that every pair of letters is present and each appears in exactly one triple. The example given is a Steiner triple system on seven points. Steiner systems are used in the statistical design of experiments. More information on Steiner triple systems can be found in Ashlock, Bryden, and Corns (2005). Steiner triples, quadruples, and quintuples have been examined to date.

3.2.11. Ordered Genes Problems

Ordered gene problems are those using a permutation ordered list as their representation. Two ordered gene problems were studied: sorting a list into order (the Order problem) and maximizing the period of a permutation. The *period* of a permutation is the smallest number of times it must be composed with itself to obtain the identity permutation. Both these problems are discussed in (Ashlock, 2006). Two variation operators were used. The crossover operator functions by choosing a crossover point uniformly at random. The entries of the list before the point are preserved. Those after the crossover point are retained but in the order they appear in the other permutation. The mutation operator exchanges two entries of the permutation chosen uniformly at random. Both these variation operators were applied, once each, for each mating event. The list ordering

problem was run for lists of length 8, 9, and 10 while the period maximization problem was run for lists of length 30, 32, 34, and 36. For both problems the sizes were selected so that the smallest size of a problem run was the first at which performance became significantly different on different graphs.

3.2.12. Parity Problems

Odd-parity is a boolean genetic programming (Banzhaf, et al., 1998) problem. The goal is to compute the truth value of the proposition "an odd number of the input variables are true." Two forms of genetic programming are used for parity in this study: simple parse trees and function stacks. Fitness for both representations is the number of cases of the parity problem computed correctly. Simple parse trees use no automatically defined functions (ADFs), and the variation operators are as for PORS. The operations used are logical and, or, nand, and nor; the terminals are the constants true and false and the input variables. Only the three-input version of the problem was done with simple parse trees – without ADFs the problem becomes exceedingly difficult. A *function stack* is a representation derived from *Cartesian Genetic Programming* (Miller and Thompson, 2002; Yu and Miller, 2002). The parse tree structure used in genetic programming is replaced with a directed acyclic graph. The vertices of this graph are stored in a linear chromosome. Each node specifies a binary Boolean operation, an initial output value for that operation, and two arguments for the operation. The available Boolean operations are: and, or, nand, and nor. The available arguments are: Boolean constants true and false, the input variables, and the output of any Boolean operation with a larger index in

the chromosome than the current one. Permitting references to the current output of nodes with larger indexes gives function stacks a *feed forward* topology. The binary variation operator used on function stacks is two-point crossover of the linear chromosome. The single point mutation operator chooses a random operation three-eighths of the time, a random argument half the time, and an initial value for a node's memory one-eighth of the time. If an operation is selected, then it is replaced with another operation selected uniformly at random. If an argument is selected, then it is replaced with a valid argument selected according to the scheme used in initialization. If an initial memory value is selected, it is inverted. The 3-, 4-, and 5-odd parity problems were run with function stacks.

3.2.13. Summary of Problems

The one-max problem was included as a baseline for comparison. There is a large amount of research already conducted on bit string evolvers, and inclusion of the one-max problem allows for a comparison to previous research. The DeJong functions and the Griewangk function are also bit string evolvers with a large amount of research, and so give a well known and representative sample from genetic algorithms. The Keane bump test was included because it is a real-valued optimization problem, giving a wider range to the types of problems being approached. The Plus-One-Recall-Store (PORS) problem is a genetic programming optimization problem that was shown to have results favoring the extremes in initial study (Bryden, et al., 2006). The north wall builder problem is a moderate difficulty agent based problem, and the self avoiding walk

problems are a scalable agent based problem that are interesting in that the fitness of the end values of the string are dependent on the earlier values. The DNA barcode, simple differential equation, Steiner systems, ordered gene, and parity problems are all application problems that have shown interesting results when GBEAs have been applied to them.

These problems give a good selection of problems that may be approached using evolutionary computation. They all have execution times that do not make multiple experiments require large amounts of computational resources, making comparisons between different methods manageable. The next step is to validate that there are indeed significant differences between the problems

3.3. A Taxonomy and Test Suite

As previously mentioned, there have been many different test problems or suites of problems introduced to evaluate different algorithms (Ackley and Littman, 1992; Mühlenbein, Schomisch, and Born, 1991; Schaffer et al., 1989). While these problems are necessary, most were written to test a specific method and tend to be biased because of that. To find what problems a method would work well on requires an unbiased test suite to show the method's performance over a wide range of problems. Before this test suite can be proposed, a method for differentiating problems needs to be devised. This can be done by developing a taxonomy of problems based on one or more aspect of the problem and its interaction with an evolutionary algorithm.

A taxonomy is a hierarchical classification of a set first established by Linnaeus to classify living organisms. This original taxonomy for living things assigned a kingdom, phylum, class, order, family, genus and species to each organism. This hierarchy gave a tree-like structure (known as a cladogram) to the taxonomy of all living creatures, showing the evolutionary relationship among various taxonomic groups. More information on developing taxonomies can be found in Mayr and Ashlock (1991). These concepts could be extended to other areas where classification would be beneficial. To construct a cladogram, it is necessary to extract taxonomic characters to perform the clustering of similar members of the set (in this case, evolutionary algorithms.) The choice of the taxonomic characters is critical for an accurate analysis. They must be unbiased, vary across the set of problems, and avoid arbitrary judgments to the greatest degree possible. One example of a bad selection would be using numbers to represent colors, as assigning numbers arbitrarily ranks some colors closer than others. This gives just a brief view of the difficulties that can be encountered in determining usable taxonomic characters.

GBEAs yield a source of taxonomic characters that are numerical and computable for any evolutionary computation problem that has a detectable solution or end point. These characters are objective in the sense that they do not favor any particular choice of representation or parameter setting. In outline, these characters are computed in the following fashion. The time-to-solution for a problem varies in a complex manner with the choice of graphical connection topology. This complexity is the genesis of our taxonomic characters. The taxonomic characters used to describe a problem are the

normalized mean solution times for the problem on each of a variety of graphs. While this presents a set of objective characters that enable automatic classification, these are not necessarily the "right" or only characters. Using results from the problems in Section 3.2, the mean number of mating events to solution were normalized to yield the taxonomic characters for the problems. Normalization consisted of subtracting the minimum average time from each average time and then dividing through by the maximum among the resulting reduced times. The taxonomic characters for each problem are thus numbers in the set [0, 1] for each graph. In this way a method for comparing the similarity of the problems was introduced with the problem's relative hardness removed.

Earlier work studying GBEAs (Bryden et al., 2006) has given a large amount of data on how different problems perform on different graphs. This normalized mean time to solution to find a satisfactory solution varies in a complex manner between graph type and the test problem being solved. This data is numerical in nature and objective, in that they do not have a preference to representation or parameter settings. This makes it possible to use this data to construct a cladogram for these different test problems to evaluate similarities in the desired amount of diversity preservation each problem requires. The study resulted in a cladogram (Fig. 4) constructed from 26 taxonomic characters for each of the problems investigated.

3.3.1. A Taxonomy of Evolutionary Computation Problems

It has been established that GBEAs provide a rich source of taxonomical characters, but to put these characters to use would require a broad assortment of problems to be analyzed. A taxonomy of evolutionary computation problems would provide researchers with a means to select a graph to apply to their problem of interest that would give a desired result in the shortest time. For computationally expensive problems previously studied, this could result in a speed up of weeks (Bryden, Ashlock, McCorkle, and Urban, 2002).

The authors of the initial work on GBEAs (Bryden, et al., 2006) have encouraged others to apply graphs to their problems to help build the taxonomy, as the addition of this information would serve to make the cladogram more representative of problems currently examined in evolutionary computation research. There are several problems available in the literature and on the web that would be useful additions that would improve this taxonomy. Other than those mentioned previously, there is also a repository maintained by William Spears (2006) that has contributions from various researchers. This is not so much a repository but more a listing of websites where researchers in evolutionary computation have posted some of the problems they have worked on. Adding these test problems to the taxonomy will increase the likelihood that a new

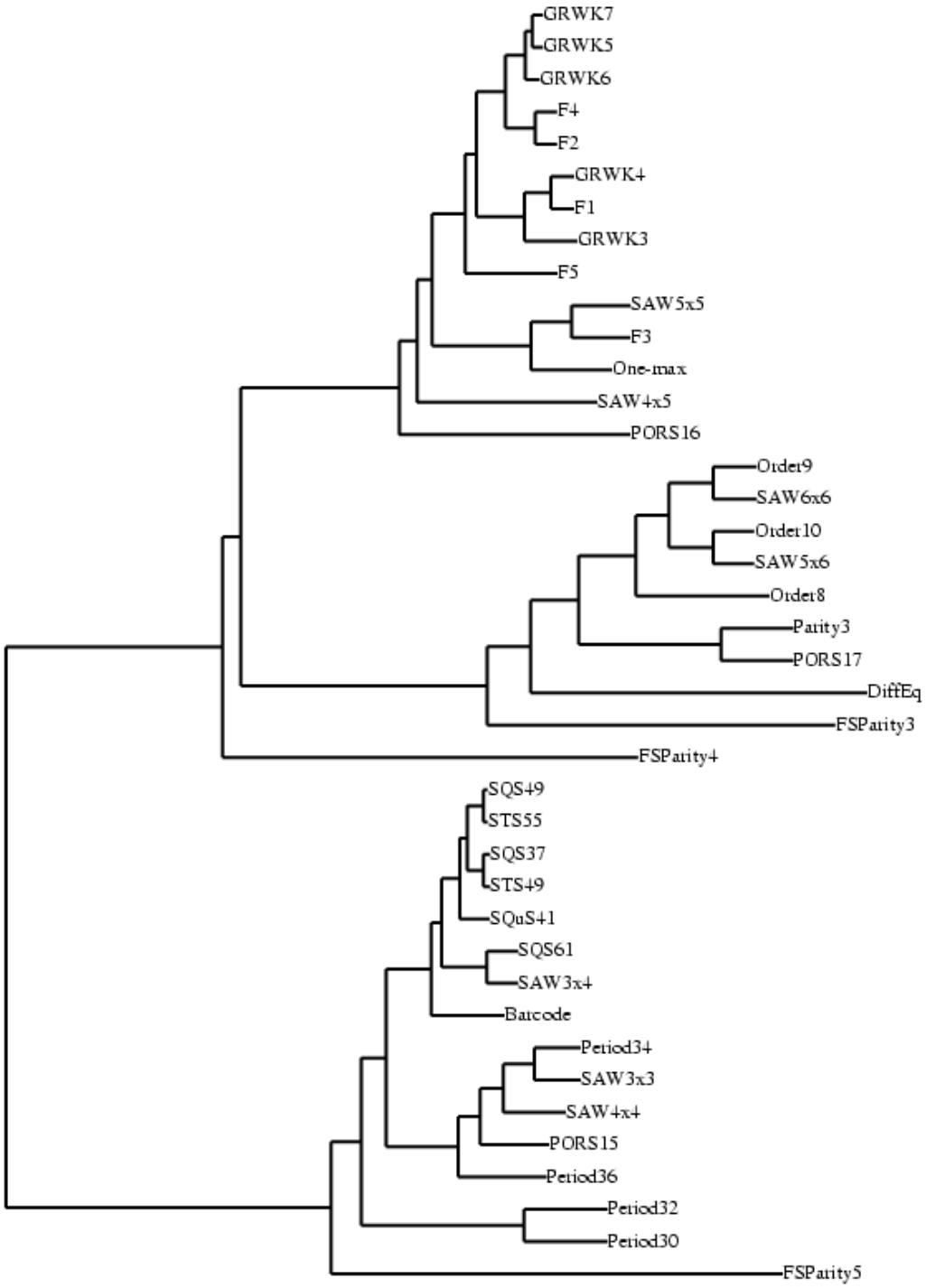

Figure 4, Cladogram of test problems based on solution times for various graphs.

problem will have a representation similar to one that already exists, giving the user a priori knowledge of how the problem can be approached. Also, if the representation can be changed to closely match one of the existing problems, it would seem likely that they would share the same preferred graph. Both of these would depend on whether there were any strong differences in problem deceptiveness, which may make a sparser graph preferable.

3.3.2. A Proposed Test Suite

With a method to differentiate problems in hand, a suite of test problems can now be developed. The first step to producing a test suite is to review those used by others. One of the most popular test suites used to determine the effectiveness of an evolutionary algorithm were the original five problems proposed in DeJong's doctorial dissertation (1975), as described in section 3.2.3. While these problems are solvable using evolutionary algorithms, they do not comprise a complete test suite and it has been questioned whether or not they were ever intended to be a test suite or just used as a proof of concept (Belew, 1992). Whitley et al. (1996) performed a review of the available test suites and found that many of the functions being used, such as the Rastrigin and Schwefel functions (Mühlenbein, 1991), are ill suited for use as test problems. NASA/MDO (2002) offers a test suite of engineering problems, but these are complex problems that do not lend themselves well to extensive study.

The work by Whitley et al. (1996) also described key traits that good test problems should have. This includes characteristics such as being non-separable, resistance to hill-climbing algorithms, being nonlinear, and having good scalability. It is apparent looking at the cladogram of evolutionary computation problems that graph preference should be added to this list. Keeping these guidelines in mind, it should be possible to develop a non-biased test suite based on the numerous problems evaluated to construct the cladogram (Fig. 4). First, it is necessary to examine the problems for the key traits of test problems and other factors.

The one-max problem is a simple uni-modal problem. While it meets none of the key traits, it was included as a baseline for comparison because of the speed at which it runs and the large amount of data readily available. For this same reason, the DeJong test functions are also being included. The Keane bump test is nonlinear and nonseperable, although the nature of the fitness function would likely favor a hill climbing algorithm, even if it were part of a memetic algorithm. While the scalability of the applied problems is questionable, the remaining problems all roughly fit the recommended principles for good test suite problems.

While it would be possible to use all of these problems as a test suite, the amount of computational resources to perform all 43 problems and build a large enough database to develop statistically significant results would become prohibitive. To further reduce the number of problems to be included in a test suite, the location of the problem in the taxonomy is used as the deciding factor. Because of varying levels of compute power

between users, two test suites are proposed, a smaller suite of seven problems to give a general performance of the proposed method(s) and a larger test suite of fifteen problems to evaluate across a wider range of possible problems that may be explored. The recommended problems for these test suites are given in Table 1. The first column lists the problems included in the smaller test suite, while the second column gives the problems that are added to the smaller test suite to build a more thorough problem set.

The problems in these test suites were sorted by their separation from the other problems. If two or more problems had similar performance on the graph set, only one would be selected. To determine which problem was retained, different considerations were used for the different test suites. For the smaller test suite, the general rule was to select problems that were similar in coding and easy to implement. For this reason, the PORS and function stack parity problems were chosen. The second DeJong function and the 3X4 Saw problem were included to cover the remaining areas of the cladogram. For the larger test suite, consideration was given to the run time of the problems and the make-up of the problem's fitness landscape. It was also desired to maintain a variety of evolutionary computation problems. As these are still only proposed test suites, ensuring that a representative sampling of the problems that are being solved with evolutionary computation techniques should allow for a more robust test suite.

Table 1, Proposed test suites of evolutionary computation problems.

Smaller Test Suite:	Larger Suite Also Includes:
DeJong function 2	One-max
PORS15	DeJong function 1
PORS16	DeJong function 5
PORS17	Griewangk Function in 5 dimensions
SAW 3X4	"Order" ordered gene problem 9
Function Stack Parity 4	Function Stack Parity 3
Function Stack Parity 5	Steiner Triple System 55
	North Wall Builder

3.4. Conclusion

The performance of different combinations of graphs and test problems shows that there is an abundance of data available for use in classifying both population structures and test problems. By continuing to compare graphs by using test problems and test problems using graphs, it is possible to explore the larger space of all evolutionary computation problems. The goal of this work is to one day have a large enough collection of data so that with a limited amount of a priori information it would be possible to select a graph that is tuned to the needs of the problem at hand. This information on the development of a test suite is a first step in making a recommendation of graph selection, although more information is needed on how information flow manages diversity. This is the topic of the following chapter.

4. CONTROLLING INFORMATION FLOW AND DIVERSITY

The control of information flow within an evolving population is the mechanism by which GBEAs achieve superior performance to standard evolutionary algorithms. While it has been shown that the use of graphs in evolutionary algorithms helps preserve diversity and allows for faster solution times for deceptive problems, it would be beneficial to have a deeper understanding of the dynamics of these GBEAs. Not only would knowing the optimal parameter settings allow for a more rapid solution of problems, it would be possible to select parameters that would promote the development of a more diverse population of solutions that are all satisfactory. This would allow for a more robust design tool, readily capable of providing another solution to the given problem if an unforeseen constraint were imposed on the problem.

In this chapter we examine diversity in graph based evolutionary algorithms and how diversity can be controlled. The amount of diversity present initially in any evolutionary algorithm is dictated by the population size. While diversity can be added using mutation, generally the amount of diversity present when the algorithm is started is the most that will be present. The question that arises is whether all of the necessary information is available to find a superior answer and how easy is it to bring the necessary pieces together. By comparing the affects of population size and graph choice, it is possible to gain some insight into diversity in a problem, when diversity needs to be preserved and when there is too much diversity.

4.1. Population Size in Graph Based Evolutionary Algorithms

The initial study of GBEAs investigated using various types of graphs with a fixed population size of 512 vertices (Bryden, et al., 2006.) This investigation showed that the selection of a graph significantly impacts the time to solution for many types of problems. In addition, selection of the optimum graph is specific to the problem and, in general, simpler fitness landscapes perform better as the graphs in which the rate of information spread was faster, e.g. graphs with smaller diameters. Conversely, those problems with more complex fitness landscapes perform better on graphs in which the rate of information spread was slower, e.g. with larger diameters. In the earlier study (Bryden et al., 2006), the population size was fixed, and the connectivity of the graph was only changed by changing the graph. Another mechanism for revising graph connectivity is to change the population size. This section investigates the role of population size in GBEAs and the interaction between graph type and population size in time to find a satisfactory solution.

To investigate the effects of varying the population size using different families of graphs, five test problems were selected and 5000 simulations were performed for each problem on each graph. The number of mating events required to find a solution in each of these simulations was recorded. A separate collection of simulations was run for each population size selected for the experiment, ranging from 32 vertices to 4096 vertices. While all population sizes of 256 or greater vertices had 22 graphs, some families of graphs become identical or else do not exist for smaller population sizes. This results in

there being only 21 graphs for population sizes 64 and 128, and only 19 graphs for population size 32.

Previous work such as that done by Goldberg, Deb, and Clark (1989; 1992) and Grefenstette (1986) has concentrated mainly on bit-string evolvers, real-string evolvers and genetic programming are also of interest and are examined in this study. The test problems used were designed to cover as broad a range of problem types as possible while still being unbiased to a particular algorithm (Whitley, et al., 1996). To perform a useful comparison of graph performance, it was also necessary to select problems with a know solutions. While this is true of all the test problems selected, the stopping criterion for the real-string evolver was a small range around the solution to account for the continuous nature of the solution space.

The goal of this computational experiment was to investigate the impact of varying the population size and graph on the mean number of mating events to solution. For each of the problems, 5000 evolutionary runs were performed on each graph and the mean time to solution was computed. A 95% confidence interval was used to compare performance across the set of graphs. Only results in which the confidence intervals calculated did not overlap were considered to be statistically significant and usable in comparing graph performance. To determine the utility of GBEAs, the performance of the preferred graph was compared to the complete graph, which resembles a standard genetic algorithm and so is used as a baseline. In the initial study (Bryden et al., 2006) it was found that graphs of the same family generally performed similarly, as shown in Figure 5. Based on the

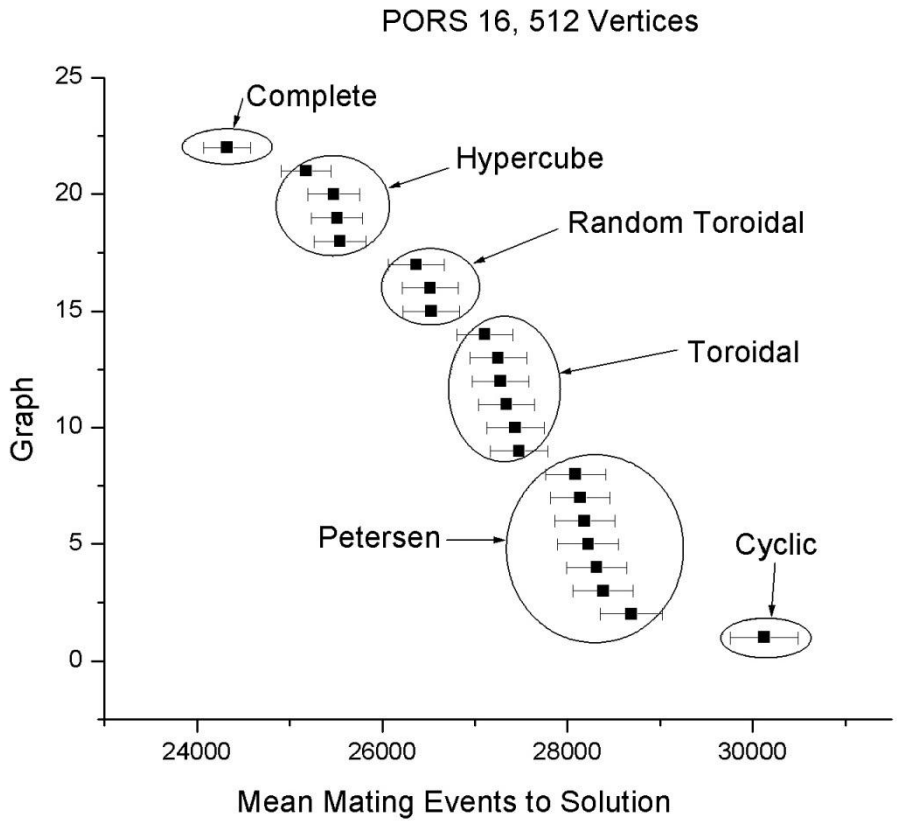

Figure 5, The average number of mating events to solution as a function of graph for the PORS 16 problem, 512 vertices.

similarities between families of graphs for a given population size, each of the graph groups were averaged together and presented as a "graph family," as shown in Figure 5. These graph family results are then plotted as a function of population size for each of the problems examined. Table 2 gives the average percent speed for each problem and population size combination studied here.

To explain these results in this study the term "critical point" is introduced. For each graph or graph family there is a population size where it performs best for a particular problem. When that graph or graph family is also the best choice at that population size, this is referred to as a critical point for the problem being examined. These are the combinations of population size and graph type that work together to provide and maintain the proper balance of diversity to solve the problem most efficiently.

4.1.1. Plus-One-Recall-Store

For the PORS 16 problem, increasing the population size from 32 to 64 resulted in a sharp improvement for all graphs, with critical points found using the cycle graph with 64 vertices and the complete graph with 128 vertices (Figure 6). For population sizes of 32 and 64, using the cycle graph works best with a speed up over the baseline of 311% and 152% respectively (Table 2), while using the complete graph results in the poorest performance. At a population size of 128 the usefulness of the graphs reverses order so

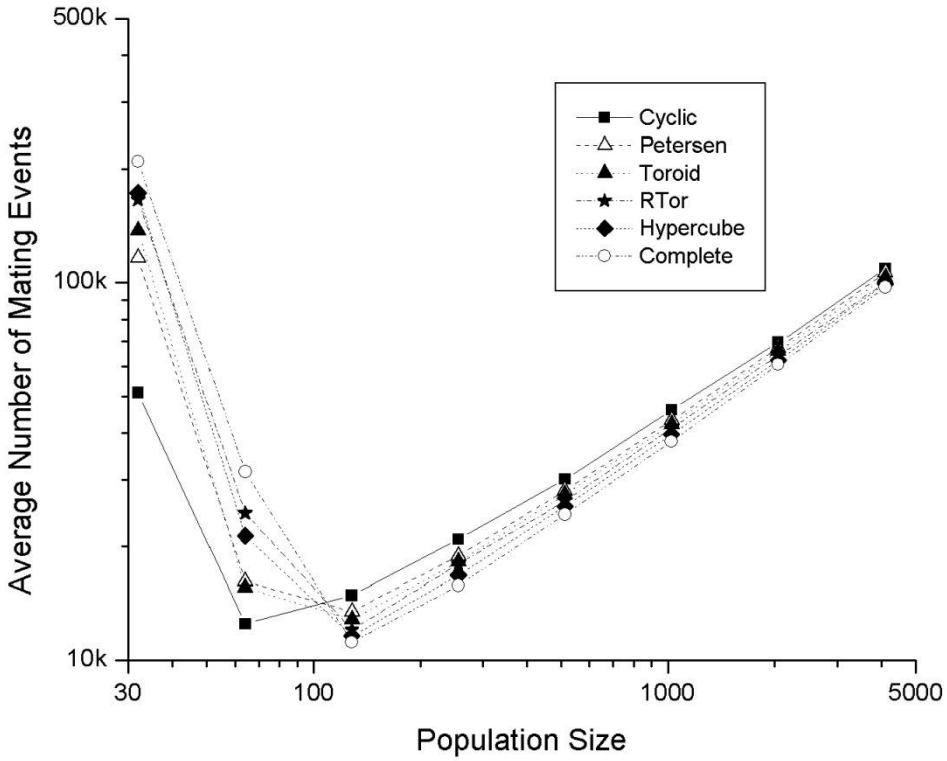

Figure 6, The average number of mating events to solution as a function of population size and graph for the PORS 16 problem.

Table 2, Performance Increase from Using Preferred Graph.

	32	64	128	256	512	1024	2048	4096
Onemax	0%	0%	0%	0%	0%	0%	0%	0%
PORS 15	39%	144%	432%	2395%	1146%	96%	10%	18%
PORS 16	311%	152%	0%	0%	0%	0%	0%	0%
n=6	--	45%	152%	239%	62%	0%	0%	0%
n=10	--	29%	241%	497%	51%	0%	7%	11%
NWB	175%	131%	64%	8%	8%	7%	6%	5%
3x3	11%	16%	15%	13%	1%	11%	13%	11%
3x4	41%	17%	20%	22%	22%	21%	22%	21%
4x4	--	12%	8%	25%	32%	40%	37%	36%

that the complete graph yields the best performance and the cycle graph the worst. This proves to be the trend for the remainder of the population sizes. The cycle graph (the graph with the slowest rate of information transfer) is the first graph to reach its maximum performance at a population size of 64, followed by the complete graph with 128 vertices (the graph with the fastest rate of information transfer). Once the graphs' rank ceases to change, the time to solution for all graphs displays asymptotic convergence to increasing parallel lines.

The PORS 16 problem is a relatively simple genetic programming problem, and as expected the optimal population size for the problem was small (cycle graph at 64 and complete graph at 128). It is interesting to note that there is a shift in which graph performs best that is related to population size. To satisfactorily solve the PORS 16 problem there needs to be a sufficient supply of the building blocks to reach one of the 24 solutions. When the population size is insufficient to provide the necessary supply of building blocks either initially or early in the search process (less than 128 members in this study), the population is dominated by sub-optimal solutions and it then becomes necessary to introduce diversity through mutation to find a satisfactory solution. Using a diversity preserving graph allows for the assembly of compatible building blocks before a solution dominates the search space by limiting the spread of information within the population. When the population size increases, there is a sufficient supply of building blocks either initially or early in the search process to assemble one of the correct solutions and so a diversity preserving graph is no longer required. In fact, once the necessary pieces are available, the restrictions on the spread of information imposed by a

diversity preserving graph slows the time to find a satisfactory solution by limiting the transfer of the building blocks through the population of solutions.

The PORS 15 problem, a more difficult, deceptive problem, also shows a significant improvement as the population size reached certain values but shows a wider separation of graph performance (Figure 7). For population sizes of 32, 64, 128, and 256, the graph rankings are the same as for the PORS 16 problem with a population size of 64 with performance increases ranging from 39% to 2,395% (Table 2). However, there is an increasing separation between the graphs and graph families. The Petersen graph family ranks second to the cycle graph for a population size of 256, and is followed by the toroid family. For a population size of 512, the cycle graph outperforms all other graphs, showing the best performance and reaching a critical point. At a population size of 1024, the only graph that has a statistically significant difference from the others is the complete graph, which has the poorest performance. This ranking continues for population sizes 2048 and 4096, with the complete graph performing closer, but still inferior to, all other graphs. As with the PORS 16 problem, once the graphs' ranking ceases to change, the time to solution using all graphs displays asymptotic convergence to parallel lines.

The preferred graph for the PORS 15 problem follows the same general trend as that seen in the PORS 16 problem, but with more diversity required. This diversity comes from both the size of population desired and the selection of graph, although a diversity

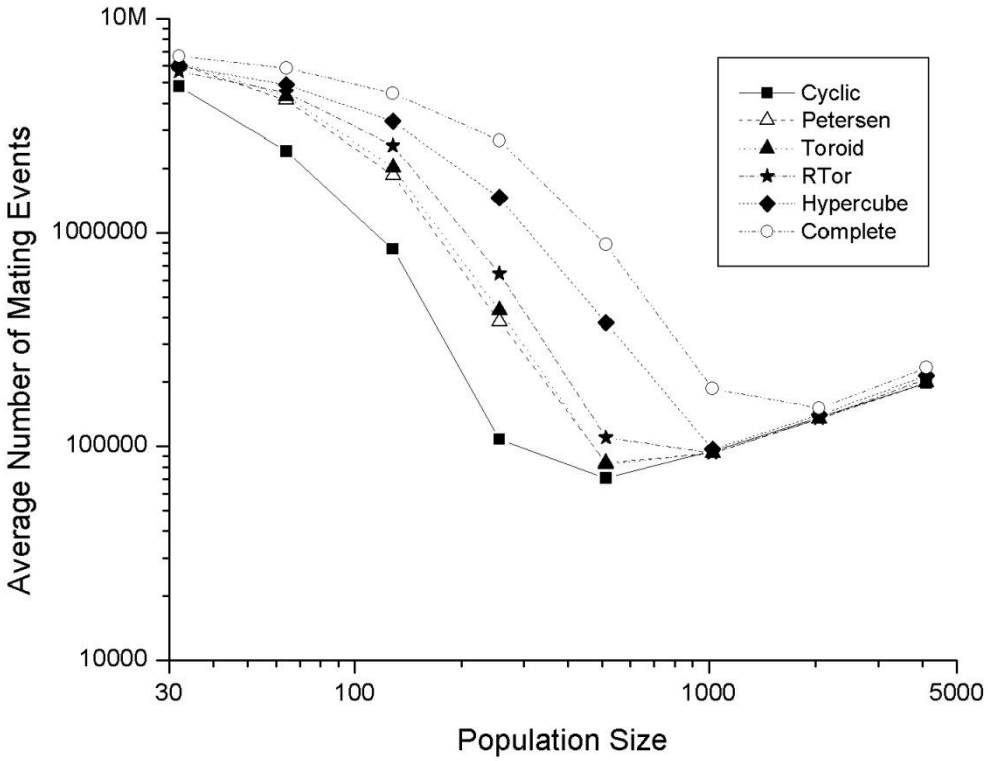

Figure 7, The average number of mating events to solution as a function of population size and graph for the PORS 15 problem.

preserving graph is preferred in all the trials conducted. It is interesting to note that this is the only problem studied in which the preferred graph never changed: the cycle graph was always the preferred graph. This is most likely due to the deceptive nature of the problem. Five building blocks composed of three nodes are required to find the solution, but it is easy for the algorithm to find blocks of five nodes that improve the fitness but prevent convergence to the true optima. By limiting the rate at which these larger building blocks are shared in the solution population, the algorithm is able to assemble the smaller building blocks to find superior solutions. For the algorithm to benefit from a higher rate of information transfer, a larger amount of initial diversity is required to provide a sufficient number of these building blocks to prevent them from being lost before they are put to use.

4.1.2. One-max

The one-max problem is a simple uni-modal problem that requires very little diversity for efficient solution (Bryden et al., 2006). Since this problem is a binary string evolver, schema theory can be applied to calculate an optimal population size for a simple genetic algorithm. Using formula's derived by Goldberg (1989) and a building block size of 1; the optimal population size without using a graph is approximately 4. Because this optimum is so low no critical point can be seen in the results (Figure 8). It is also impossible to construct many of the graphs used in this study with this population size, so no experiments were done using these parameters. The results show that graphs with a higher level of connectivity and a small population size work best; for small populations,

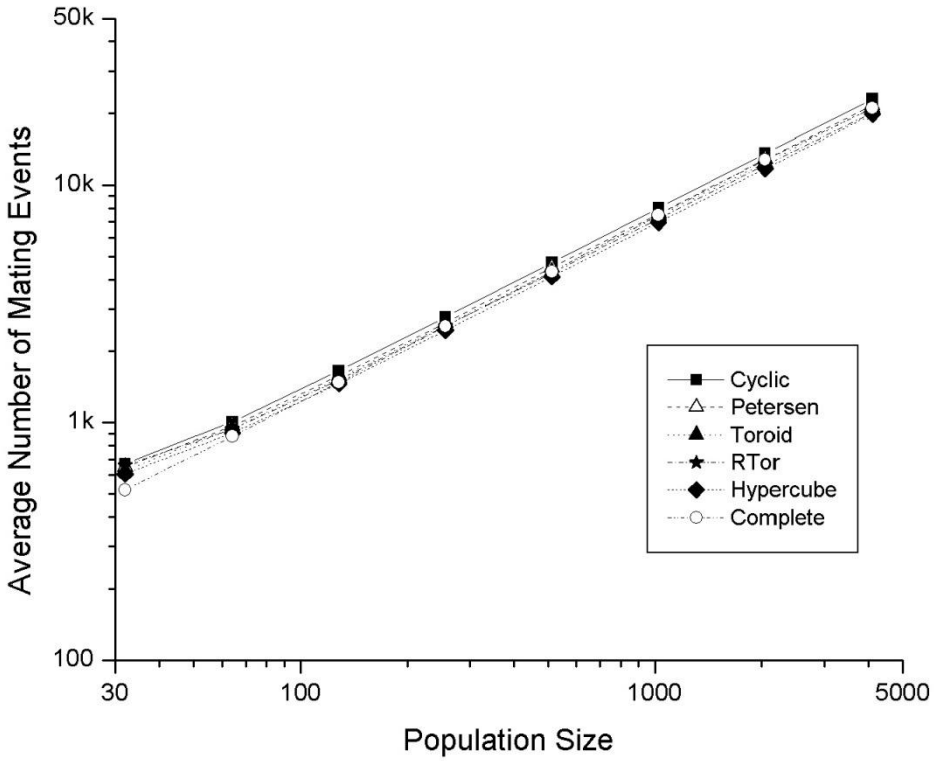

Figure 8, The average number of mating events to solution as a function of population size and graph for the one-max problem.

the algorithm using the complete graph converges to a solution fastest and using a cycle graph converges slowest. The optimal performance was observed with a population size of 32 (the smallest population examined) using the complete graph. For population sizes greater than 64, using the hypercube works best, and when the population size exceeded 1000 the toroid graph family outperforms the complete graph although the performance increase is modest (Table 2). Increasing the population size for this problem causes an almost exponential increase in time to solution, and population size has a much more significant effect than graph type.

4.1.3. North Wall Builder

For the north wall builder (NWB) problem, there is again a population size at which each graph makes a sharp improvement in performance, but the preferred graph for this problem changes several times as the population size increases (Figure 9). For the smallest population size, the cycle graph performed best. The Petersen graphs were preferred when the population was changed to 64. Population size 128 is the optimal population size, with the best performer being in the toroid family. When the population size is increased to 256 the hypercube family and complete graph performed best. For a population size of 512 the hypercube family performed best, followed by the complete graph. As the population size increases, graph performance exhibited more separation and shows the same general trend towards identical behavior at 512 vertices and higher with the time to solution for all graphs converging asymptotically. The largest benefit to

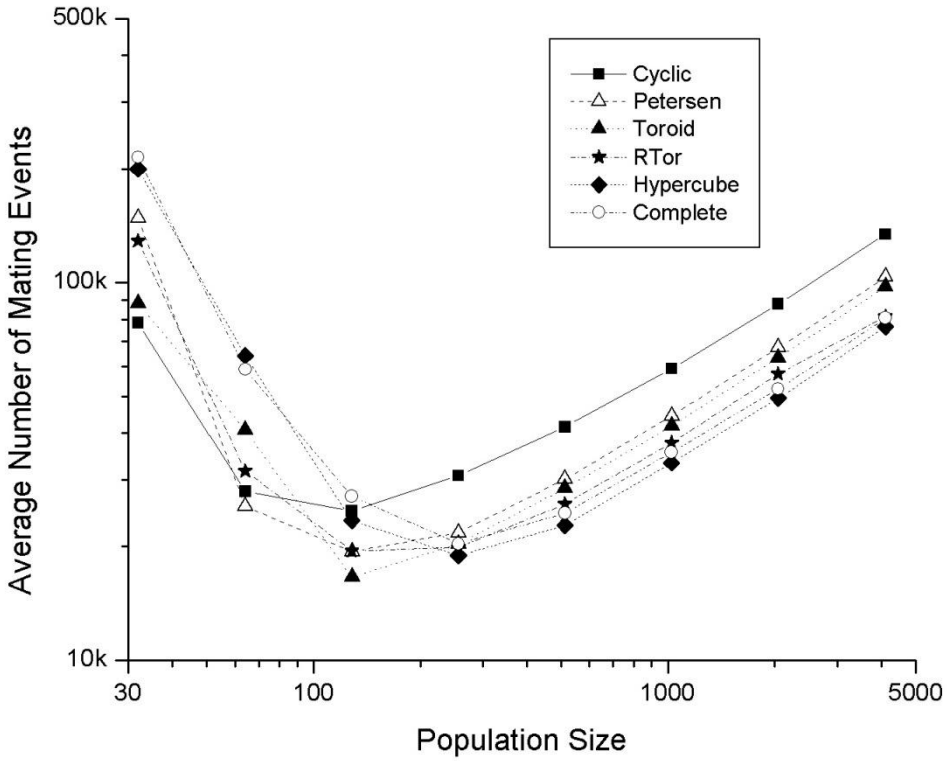

Figure 9, The average number of mating events to solution as a function of population size and graph for the North Wall Builder problem.

using a GBEA is seen for smaller population sizes, starting with a 175% performance increase for a population size of 32 and decreasing quickly to 8% at a population size of 256 (Table 2.)

As previously noted, the NWB problem is an intermediate difficulty problem and these results show that an intermediate amount of diversity is required for optimal performance. An interesting difference in this problem is that unlike the PORS 15 and 16 problems, the preferred graph makes a more gradual shift from one that preserves the most diversity to a more connected graph. As the population size increases, more connected graphs are preferred, but there is no abrupt change from a very sparse graph to a very connected graph as seen in the other problems. While the number of mating events to solution for the NWB problem is comparable to the PORS 16 problem, the NWB problem has a much larger selection of acceptable solutions that may or may not have interchangeable building blocks. A diversity-preserving graph allows a more effective search of the solution space so that one of these solutions may be found, as opposed to a larger population size that provides more diversity initially, but does not preserve it.

4.1.4. Keane Bump Test

For the Keane Bump Test with six variables, the graph families show only modest separation statistically. There is no statistically significant difference between the graphs with a population size of 32, and only the performance of the cycle graph can be distinguished from the other graphs for a population size of 64 (Figure 10.) All graphs

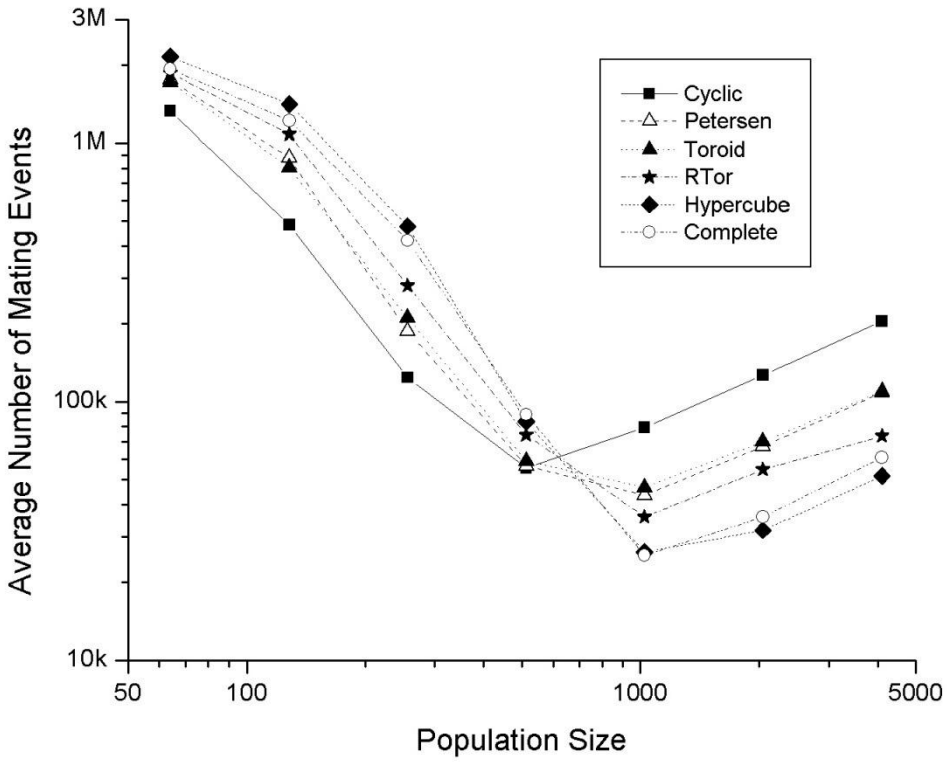

Figure 10, The average number of mating events to solution as a function of population size and graph for the Keane Bump Test, n=6.

show improved performance at population sizes of 128 and 256, with the cycle graph still performing best with a performance increase of 152% and 239% respectively (Table 2.) When the population size is increased to 512, the cycle graph's performance is not significantly different from the toroid and Petersen families, with these graphs performing best. When the population size is increased to 1024, all graphs but the cycle graph show improved performance. As the population increases from 512 to 1024, the cycle graph switches from the best to worst performer, and the complete graph or a graph in the hypercube family becomes the best performer. This trend continues as the population size increases, but with all performances decreasing as population size is increased beyond 1024.

The results for the 10-variable problem are strikingly similar to those for the 6-variable problem (Figure 11), although graph choice had more of an impact on time to solution as the population size changes. Again there is difficulty separating the graphs at population size 64, although the cycle graph performs best. The graphs' rankings are the same for population sizes 128 and 256, with performance increases of 241% and 497% respectively. At population size 512, the cycle graph flips from best performer to worst with the toroid graphs, the Petersen graphs, and the random toroid graphs grouped together as best. When the population size is increased to 1024, the complete and hypercube family of graphs perform best. This is the trend as the population size is increased to 2048 and 4096. While the superiority is not statistically significant, the random toroid graph shows some signs of outperforming the toroid graphs and Petersen graphs at the highest population size.

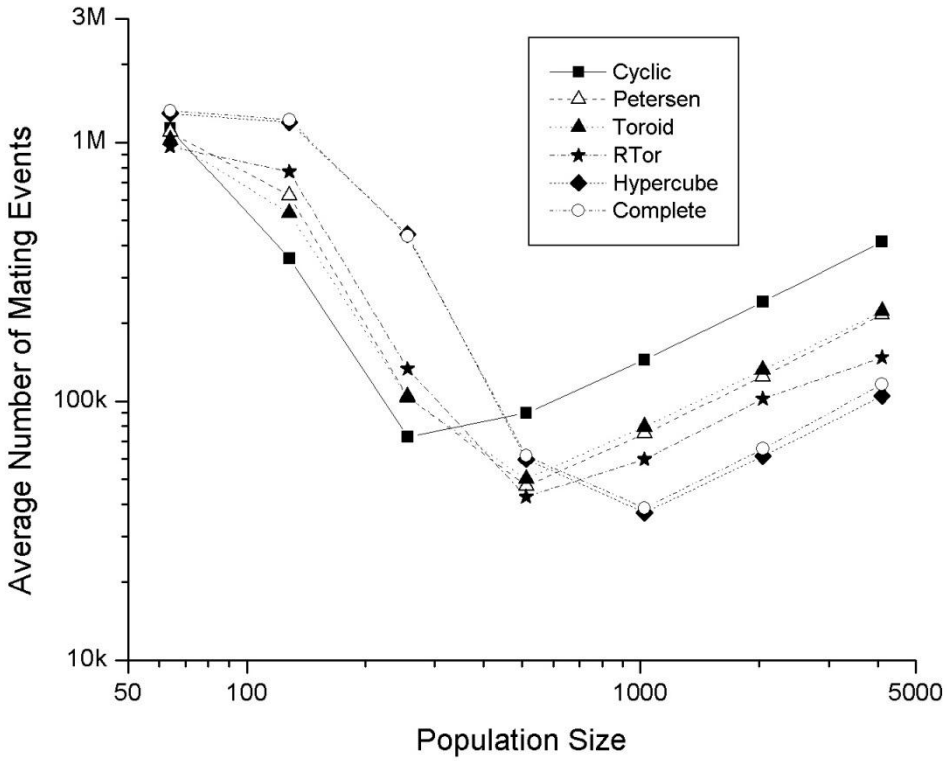

Figure 11, The average number of mating events to solution as a function of population size and graph for the Keane Bump Test, n=10.

There are three points in Figure 11 where the preferred graph changes; one for the cycle graph when it switches from best to worst at population size 512, one for graphs with intermediate information transfer rate (Petersen, toroid and random toroid) when they become best at population size 512, and one for graphs with high levels of information transfer (hypercube graphs and complete graph) when they become best at population size 1024. This indicates again that with a smaller population size and a corresponding smaller initial diversity, the need for diversity preservation is higher.

Note that, with higher dimensionality, graph choice becomes more important. When the population size for this problem is small, a graph with a slow rate of information transfer is preferred, indicating that there is some need for diversity. As the population size increases, the corresponding increase in initial diversity makes the need to preserve diversity less important, so graphs with intermediate amounts of information transfer are preferred. At 1024 vertices, there is enough diversity in the initial population to allow graphs with the fastest information transfer rate to solve the problem most efficiently. This is due to the increased number of building blocks necessary to find a satisfactory solution. The building blocks correspond to the correct solution in each dimension, which have to be found and then assembled. By isolating these pieces as the solutions develop, the graph structures allows all of these blocks to be found, preventing sub-optimal solutions with other building blocks from dominating the population and preventing this development. Larger population sizes allow for a more diverse population making it less likely that a solution with one or more correct building blocks will be replaced.

4.1.5. Self-Avoiding Walk

The smaller population sizes perform best for the 3x3 grid (the smallest grid studied) with more mating events required as the population size increased to 1024 vertices (Fig. 12.) This is a fairly simple instance of this problem, and all graphs are statistically indistinguishable at every population size, with the exception of the complete graph, which performs the worst for every population size. Performance gains from using GBEAs ranged from 11% to 16% (Table 2.) Graph performance at 2048 vertices is roughly the same as at 1024, and with 4096 vertices some improvement is seen. There appeared to be some need for diversity preservation to develop solutions, but a small population size has sufficient initial diversity to solve the problem.

The decrease in the number of mating events necessary from population sizes 2048 to 4096 warrants some extra analysis. The 3x3 SAW problem is a length 8 string evolver with a four-character alphabet. This gives 4^8 or 65,536 different combinations of strings available. There are 8 different possible solutions to the 3x3 SAW problem, meaning that there is a 1 in 8192 probability that each string generated when the algorithm is initialized will be a correct solution to the problem. This explains why the rate at which the number of mating events to solution increases starts to slow when the population size reaches 512, where there is a 6% probability that a solution will be in the initial population. There is also an increase in the confidence interval due to more correct solutions appearing in the initial population.

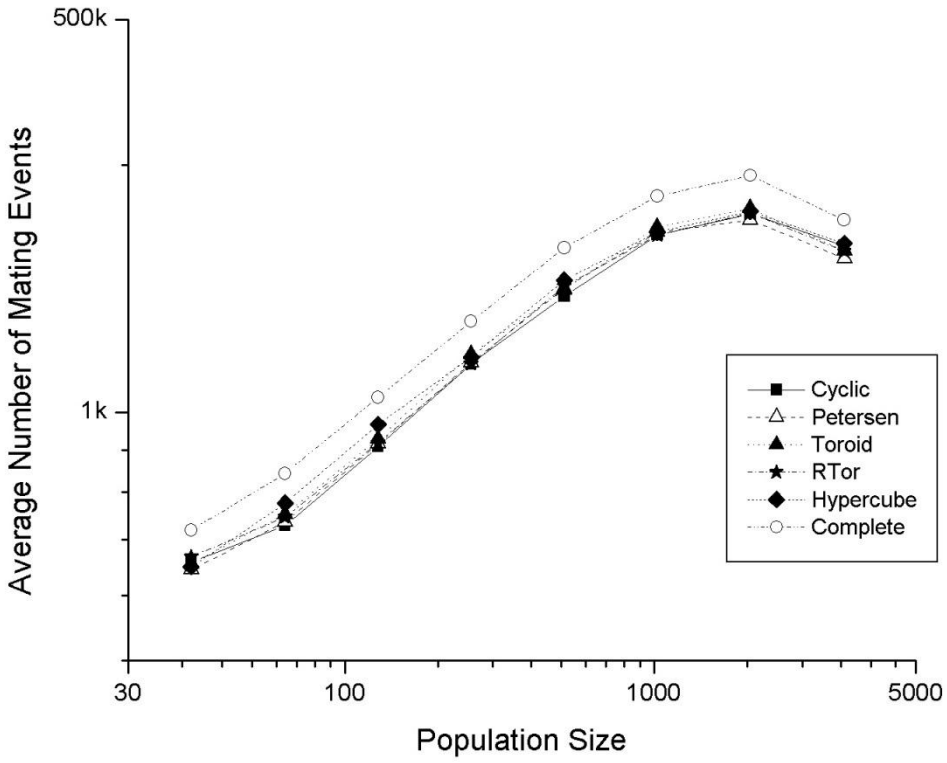

Figure 12, The average number of mating events to solution as a function of population size and graph for the Self-Avoiding Walk, 3x3 grid.

For grid size 3x4 and 4x4, a similar trend to other problems is seen with all graphs improving as population size increases to an optimal point, and then performance lessens as the population size increases past the critical point (Fig. 13.) When the grid size was increased to 4x4, graph of population size 32 ceased to have more than 95% of the runs find satisfactory results within the ten million mating events allowed. Because of this and the large confidence intervals calculated for this population size the results considered uninformative and are not given. As with the 3x3 SAW problem, there is little statistically significant difference between the graphs for population sizes up to 512 other than the complete graph, which consistently performs worst. Performance gains from using GBEAs ranged from 8% to 41% (Table 2.) It is interesting to note that the preferred graph did not change when the problem difficulty is increased, but the optimal population size and the ability of the graphs to solve the problem as a function of population size changes significantly.

As the grid size is increased to 4x5 and higher, graphs of population size 64 cease to have more that 95% of the runs find satisfactory results and the best population size is still 128. All of the graphs perform similarly at this optimal population size with the exception of one of the random toroid graphs and the complete graph that fails to consistently solve the problem. When the population size is increased to 1024, the hypercube family starts to perform worse than all the other graphs except the complete graph, which is still the worst performer. All graphs take more time to converge as population size is increased beyond 128. When the grid size is further increased to a 5x5 grid, most of the graphs

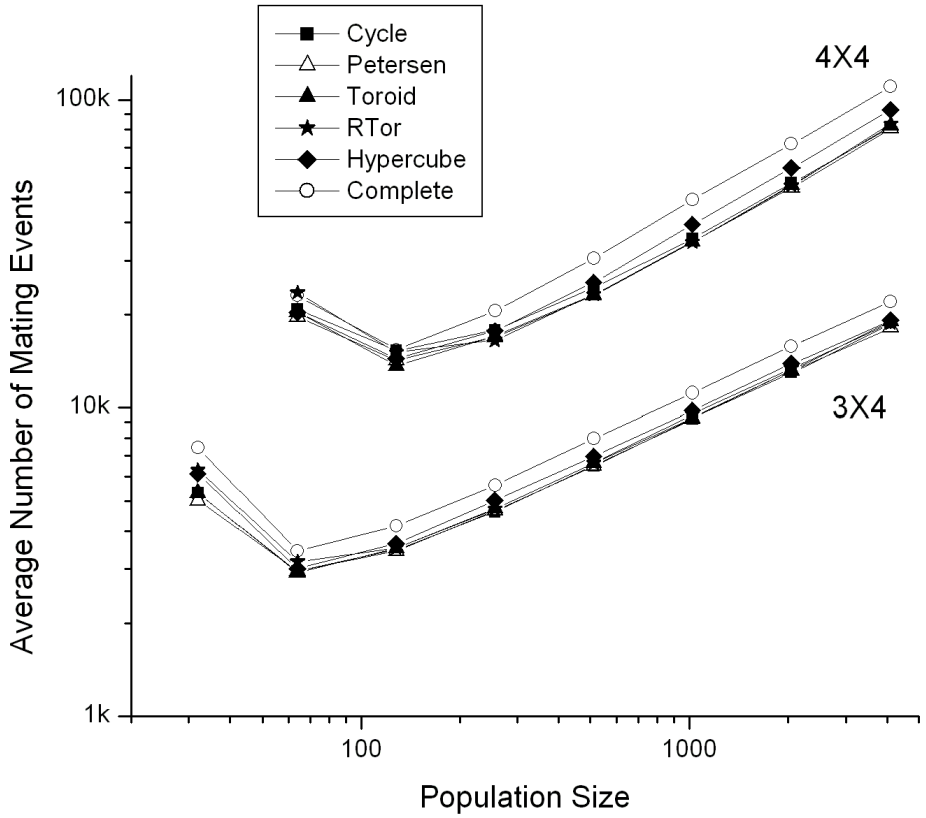

Figure 13. The average number of mating events to solution as a function of population size and graph for the Self-Avoiding Walk, 3x4 and 4x4 grids.

could not consistently find a solution until the population size is increased to at least 512. The trends shown by the experiments that did successfully solve the problem are the same as those exhibited by the 3x4 and 4x4 versions of the SAW problem.

This study investigates the effects of population size on the performance of graph based evolutionary algorithms. Five test problems were examined and the number of mating events required to find a satisfactory solution were determined for 5000 runs of each problem on each of the graphs. Over 80% of the problem and population size combinations showed some improvement when a graph based evolutionary algorithm was used, with nearly 20% finding the solution at least twice as fast.

As every graph's population increases, there is a point at which the required number of mating events to solve the problem decreases drastically, which is then followed by a slow increase in the necessary number of mating events as the population size increases. The trend of the results suggests that for each graph applied to a problem, there is a population size where the optimum performance is achieved, as shown in Table 3. The least difficult problem (one-max) and the least difficult SAW problem (3x3) do not show this behavior, but it would seem intuitive that there is also a preferred population size for these problems. For the one-max problem that size would be less than 32. As noted, the optimal population size for the one-max problem is 4.

Table 3, Critical points for test problems by graph family. Best graph family denoted with an asterisk (*).

Problem	Cycle	Complete	Hypercube	Petersen	Toroid	Random Toroid
Onemax	none	none*	none	none	none	none
PORS 16	64	128*	128	128	128	128
PORS 15	512	2048	1024	512*	512	1024
NWB	128	256	256	128	128*	128
Keane n=6	512	1024*	1024	512	512	1024
Keane n=10	256	1024	1024*	512	512	512
SAW 3x3	none	none	none	none*	none	none
SAW 3x4	64	64	64	64	64*	64
SAW 4x4	128	128	128	128	128*	128
SAW 4x5	none	none	none	none	none*	none
SAW 5x5	none	none	none	none*	none	none

Based on the results of the computational experiments, there is an apparent progression as the population size increases. As noted earlier, the results of the all the experiments in this study followed the same trends. In some experiments only a portion of this progression is shown based on the population sizes explored, but in each case the section observed follows this pattern. Before the critical point, the population is diversity starved, and so a graph that preserves the available diversity has a large impact on the performance. In this region sparser graphs (e.g. the cycle graph) are superior as they preserve the available diversity. This diversity starved region is labeled A in Figure 14. Region B is the optimal performance region, where the amount of diversity is best for evolutionary optimization to find single solutions fastest. This region contains the critical point for the problem and a satisfactory solution can be found in the fewest mating events. Region C, the excess diversity region, represents the population size where the diversity is rich, making it possible for the algorithm to find multiple solutions but at a substantial cost in added time. A large population with sufficient diversity to ensure that global solutions will evolve is also so large that it takes a long time to simply evaluate the fitness of each member of the population. In addition that algorithm often has competing solutions in the population. This competition between different solutions also slows convergence. In Region C the best graph tend to be a highly connected graph that can burn off the excess diversity as fast as possible. However as the population size grows the initial diversity overwhelms the ability of a graph to burn off diversity and all graphs tend to look the same. In the problems examined the exceptions to this were the PORS 15 problem and the self-avoiding walks. In these cases it appears that the solution is built from building blocks that must be first found and then assembled correctly. As a result

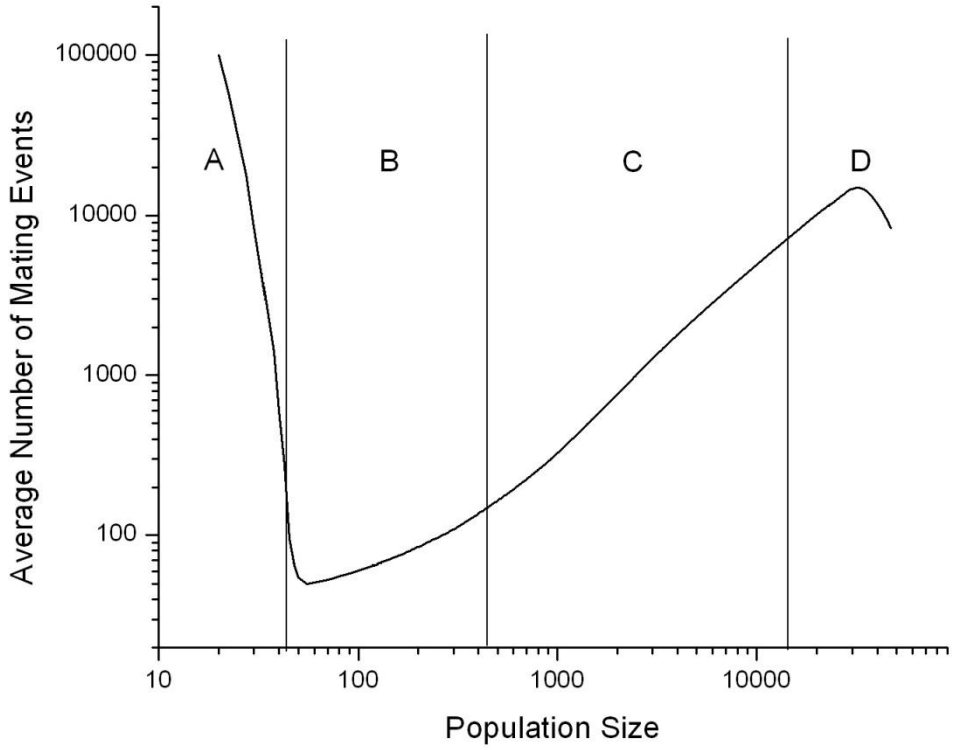

Figure 14, Population Size Regions (Log vs. Log scale). A – diversity starved, B – optimization, C – excess diversity, D – saturation.

graphs that maintain some diversity have a slight edge. The saturation region (Region D) is where the population size has increased sufficiently so that global solutions start to have a positive probability of appearing in the initial population of solutions. At this point the time cost of simply evaluating all the creatures in the initial population dominates time to solution. While there is a reduction in the average number of mating events in Region D over Region C, the reduction flattens out as the algorithm becomes equivalent to an algorithm that simply tests random examples until an optimal solution is found. The improvement in time-to-solution observed in Region D never reaches the low level of the critical point in Region B.

The use of a GBEA has several impacts on the population size at which the evolutionary algorithm enters the different regions. For smaller population sizes, GBEAs using sparser graphs tend to move Region B to the left on Figure 14, as the preservation of diversity makes up for lack of initial diversity. For more difficult problems, using a sparser graph allows the algorithm to function in Region B at a smaller population size and so find an optimal solution in fewer mating events. We theorize that preserving population diversity permits that algorithm to situate in more basins of attractions of the fitness landscape. This, in turn, makes it possible for necessary building blocks to be assembled when they are present in a given problem and, at least, permits that algorithm to climb and compare multiple hills in the fitness landscape. As the population size increases into Region C there is a gradual shift in the best performer from sparser graphs to graphs with a higher connectivity. As the population size increases, excess diversity inhibits the progress of the algorithm, and so population structures that increase the rate of information spread

start to outperform those that preserve diversity. When the population size enters Region D, graph choice begins to have no significant effect on the algorithm performance. Since search in Region D is equivalent to examining random examples until a solution is found the structure of the graph has no leverage to affect performance.

A second phenomenon of interest is a shift in the best graph as the population size increases for some problems. This can be seen with the PORS 16 problem (Fig. 6), where initially the cycle graph performs best, followed by the Petersen and toroidal graphs. This shifts to the complete graph performing best, followed by the hypercube, as the population size increases to 128. A similar shifting occurs with the north wall builder problem, although it occurs more slowly as a function of population size. This slower shift also shows that every graph was preferred at some population size, except for the complete graph. The most prominent case for this is seen in the variable dimension surface. It appears that for these problems, a certain amount of diversity is required to arrive at a solution, and when that level of diversity can be achieved, the problem then is benefited most by a faster sharing of that information, as shown by the graph families with higher connectivity being preferred in the Keane Bump Test problem (n=6 and n=10.) The fact that the shift in preferred graph occurs at a different rate for different problems indicates that the diversity preservation achieved by using a graph affects the evolving solutions differently than initializing the population with more diversity by using a larger population size.

As the population size is increased beyond the critical points for the problems, two trends were observed. These trends appear to be due to the differences in the type of problems investigated. For the problems that have building blocks (PORS and Keane Bump Test), there is a shift in the preferred graph that indicates that the complete graph is always the preferred graph when the population size increases to a very high level. This population size would correspond to the point when all of the necessary building blocks are available in sufficient supply to assemble the final solution. At this point, the algorithm benefits from having the highest amount of information sharing. In contrast, the NWB and SAW problems have constructs that are more tightly coupled to the solution they appeared in (i.e. the last 4 characters may contribute four to the fitness in one string, but when crossover is performed with another string, these characters may no longer make any contribution to fitness). In addition, it is likely that there are multiple competing optimal solutions evolving within the population that produce less fit children when they mate. For these reasons, there is less of a need for information sharing to speed the finding of a satisfactory solution, and so the complete graph does not become the best performer.

An increase in population size provides a larger amount of diversity (or necessary building blocks) in the initial population, as shown by the research of Goldberg (1989) and Grefenstette (1986), among others. For the problems examined in this study, there is a given amount of diversity required to find the solution. This required diversity can be achieved using just a larger population size, but by using a graph based evolutionary algorithm to limit the rate at which this diversity is destroyed during the evolutionary process by superior yet sub-optimal solutions, a smaller population can be used to arrive

at an optimal solution faster than with standard techniques. This results in a decrease in computational expense and faster answers to the problem being solved, especially those in which the cost of initializing the population is high compared to future evaluations, such as using neural networks to control computational fluid dynamic calls (McCorkle, Bryden and Carmichael, 2003). Future experiments could indicate if diversity preservation using graph based evolutionary algorithms instead of larger populations is universally superior.

While the combined effect of using graphs together with the optimal population size has been seen to give results as positive as a 94x increase in time to solution, as occurred in the PORS 15 problem, where the complete graph (which is equivalent to a standard EA) requires approximately 6.6 million mating events to find the solution with a population size of 32, while using a cycle graph with a population size of 512 requires just over 70,000. If applied to a similarly deceptive problem with a fitness evaluation taking just 1 second per evaluation, the cycle graph would require a day as compared to two weeks for the complete graph with the same population size.

4.2. Takeover Times for Graph Based Evolutionary Algorithms

Previous studies (Bryden, et al., 2006) have shown that using a properly selected graph decreases the number of mating events to find a solution for difficult and/or deceptive problems. This is thought to be because GBEAs allow the user to control the rate at which information is spread through that population. By decreasing the rate that

information about the solution space is shared by population members, a higher level of solution diversity is maintained but the time required for the population to converge to a solution is also increased. In this research we calculate the takeover time for various graphs to verify the information transfer rates. In addition, experiments are conducted to empirically determine the solution diversity present in the population after a set number of mating events. By comparing these results, it is possible to observe the balance between exploitation and exploration in the search space and how GBEAs can be used to adjust that balance.

Takeover times were first introduced by Goldberg and Deb and Thierens (1993) as a means for comparing the impact of varying selection pressures. Takeover time for an evolutionary algorithm (EA) is defined as the time (in generations or mating events) necessary for a superior solution to spread through the entire population. These takeover times are a common indicator of the amount of diversity preservation an algorithm yields in a population of solutions, with smaller takeover time algorithms being exploitive while larger takeover times are more nearly explorative. For most if not all algorithms, a larger takeover time generally leads to a more diverse final population.

There is a larger body of work concerning takeover times following the derivation of theoretical takeover times by Deb and Goldberg. This initial work was performed on standard genetic algorithms (SGAs) using standard selection methods. This was extended to spatially structured EAs empirically by Sarma and DeJong (1996), where it was found that grid based EAs have a growth curve that is logistic in nature. Rudolph (2000b) and

Giacobini et al. (Giacobini, Tomassini, Tettamanzi, and Alba, 2005) performed analysis on spatially structured EAs and cellular EAs respectively. These studies found that for a 2-D grid the growth curve is quadratic and for a ring structure the growth curve is linear.

This work introduces both analytical and empirical results for takeover times in GBEAs. The majority of work in the literature deals with generational algorithms and so cannot be assumed to be comparable to GBEAs, which are steady-state algorithms. However, there should be some correlation between synchronous cellular evolutionary algorithms using a ring of radius=1 (Giacobini, et al., 2005) and the cycle graph, which should be helpful in validation of these results.

4.2.1. Takeover Times

Takeover times are a method for determining how long it takes for the population of solutions to become completely taken over by a single solution, usually measured in generations or number of mating events. In this study, two methods were employed to determine the takeover times: expected value calculations and empirical testing. Expected value calculations are an analytical solution for takeover times that give highly accurate approximations, with only the stochastic nature of the algorithms providing variation. While accurate, they are also exceedingly difficult to calculate for most graphs. For this reason, empirical tests were conducted for the graph set, with the empirical results compared to the analytical results when possible.

4.2.2. Expected Value Calculations

To find the takeover times for these two graphs, it is first necessary to examine the interactions of population members as the solutions evolve, beginning with complete graph. There are two possible outcomes at the beginning of a mating event: either a superior member is selected or an inferior member is selected. The probability of a superior member being initially selected is:

$$P_s = \frac{x}{n} \tag{4-1}$$

and the probability of selecting an inferior member is:

$$P_i = \frac{n-x}{n} \tag{4-2}$$

Where x is the number of superior solutions in the population and n is the population size. Next the probability of selecting an appropriate neighbor using fitness proportional selection is considered. For this experiment, only outcomes mating a superior individual to an inferior are of interest as this is how the superior solutions spread through the population. This depends on whether a superior solution was initially selected and also depends on the fitness ratio of the superior and inferior solutions. Treating the complete graph as a size n GBEA mating event, the probability of selecting an inferior co-parent when a superior parent was selected is:

$$P_{cop,s} = \frac{n - x}{n + rx - r - x} \tag{4-3}$$

and the probability of selecting a superior co-parent when an inferior parent was selected is:

$$P_{cop,i} = \frac{rx}{n + rx - x - 1} \tag{4-4}$$

where r is the fitness ratio of the superior solution to the inferior solution. The probability that a superior parent will be randomly selected and then an inferior co-parent is selected is the product of equations 4-1 and 4-3:

$$P_{s*cop,s} = \frac{nx - x^2}{n^2 + rnx - rn - nx} \tag{4-5}$$

And the probability that an inferior parent will mate with a superior co-parent is the product of equations 4-2 and 4-4:

$$P_{i*cop,i} = \frac{rnx - rx^2}{n^2 + rnx - nx - n} \tag{4-6}$$

Adding these two probabilities together gives the chance that an inferior and a superior population member interact, leading to the spread of the superior solution (eqn4-7).

$$P_K = \frac{(nx - x^2)(n^2 + rnx - nx - n) + (rnx - rx^2)(n^2 + rnx - rn - nx)}{(n^2 + rnx - rn - nx)(n^2 rnx - nx - n)} \tag{4-7}$$

For the cycle graph a different approach is necessary to determine the probability of a superior solution interacting with an inferior solution due to the lowered connectivity of the graph. When there is only a single superior solution, either the superior solution must be selected as a parent ($P_S = 1/N$), or one of the inferior solutions adjacent to it must be selected as parent and the superior solution selected by fitness proportional selection:

$$P_i = \left(\frac{2}{n}\right)\left(\frac{r}{r+1}\right) \tag{4-8}$$

For a total probability of:

$$P_{C-1} = \frac{1}{n} + \left(\frac{2}{n}\right)\left(\frac{r}{r+1}\right) = \frac{3r+1}{nr+n} \tag{4-9}$$

After a second population member has a superior fitness and until there is only one inferior solution left, there are only two edges on the graph where there is a possibility of the solution spreading, referred to here as “active” edges. These are the edges that

connect a superior solution to an inferior solution. The probability of the randomly selected parent being either a superior or an inferior solution on one of these edges is the same:

$$P_{parent} = \frac{2}{n} \tag{4-10}$$

The co-parent is then selected using fitness proportional selection, with the probability of a superior neighbor being selected of:

$$P_i = \frac{r}{r+1} \tag{4-11}$$

and a probability of an inferior neighbor being selected of:

$$P_s = \frac{1}{r+1} \tag{4-12}$$

Adding equation 4-11 and 4-12 and multiplying by equation 4-10 gives the probability that the superior solution will spread to another vertex:

$$P_C = \frac{2r+2}{nr+n} \tag{4-13}$$

Reducing equation 4-13 gives:

$$P_C = \frac{2r+2}{nr+n} = \frac{2}{n}\left(\frac{r+1}{r+1}\right) = \frac{2}{n} \tag{4-14}$$

Equation 4-14 merits some extra discussion. The results indicate that with the exception of the first and last successful mating the fitness ratio has no influence on the probability a solution will spread and thereby the takeover time for a cycle graph. At first glance this would seem counter-intuitive, but as progress is only made on the "active edge" of the graph, there are two superior and two inferior population members of interest. A closer examination of equations 4-11 and 4-12 shows that as they are always sum to one, so the only effect the fitness ratio has on the spread of information is on how much more likely it is that information spread results from an inferior parent mating with a superior co-parent.

Using these probabilities the number of mating events required for the superior solution to takeover the graph are found using expected values. Letting *m* denote the number of mating events and *j* be a running counter, the takeover time can be found by as follows:

Set $x = 1$ and $m = j = 0$

While $x \langle n$ {

Increment m

Calculate $j = j + P$

Set $x = \lfloor j \rfloor$

}

At the end of this iterative process the value of j gives the takeover time in number of mating events for the corresponding graph probability. As previously discussed, this probability is a function of the number of population members (n), the fitness ratio (r), and the number of superior solutions present (x). To validate the curves generated using this method, empirical experiments were done to create takeover time curves for these and other commonly used graphs.

4.2.3. Empirical Takeover Time Experiments

To further investigate the rate of information transfer within the various combinatorial graphs used in GBEAs, a series of numerical experiments were performed, each experiments consisting of 1000 runs. These experiments were conducted for each population size and graph type used, with a comparison of the cycle and complete graphs to the expected value solutions used as validation. For each run, the graph is populated with a candidate solution at each vertex, assigned a fitness of one. An individual is then inserted into a random vertex and a mating event involving that vertex is performed to

start the run. A mating event consists of selecting of a population member (randomly after the introduction of the high fitness individual), and then selecting a mate using roulette selection of the available neighbors. If there is a difference in fitness between the two population members selected, the lower fitness individual is replaced by a copy of the higher fitness individual. After initialization, mating continues until the entire population has this higher fitness, recording the number of mating events after initialization required to reach whole number percentages.

Letting *r* be the fitness ratio between the superior individual and the initial population, runs were performed with fitness ratios of 1.5, 1.75, 2, and 3. In this way, the rate at which the superior solution spread across the graph can be tracked. The data is reported as the number of mating events required as a function of percentage spread, and then as the number of mating events required divided by the population size as a function of percentage spread, in an attempt to normalize the results.

4.2.4. Diversity Measurement Experiments

To test what effect different graphs have on the number of different solutions obtained, two problems were used: a real-valued multiple-sinusoid function and the PORS efficient node usage problem, a maximization problem in genetic programming.

4.2.4.1. Sinusoidal Function

The first set of experiments investigating the diversity enhancement enabled by a GBEA used a multiple-sinusoid function. The optima are known, and the function is highly multi-modal. A population of 512 individuals was used with the gene length (or problem dimension *d*) ranging from 3 to 9 real values. Values for each locus were randomly generated from 0.0 to 2.0 to initialize the populations. Fitness was then calculated using the equation:

$$f = \sum_{i=1}^{d} \sin\left(10\pi x_i\right) \tag{4-15}$$

where x_i is the value of the i^{th} locus. The search for a solution was said to be complete when the fitness evaluated in Eq. 4-15 was equal to the dimension of the problem within six decimal places. The variation operators used were single point crossover and mutation of one of the gene's values at random. For each of these experiments, there are 10^d different solutions, giving a large multi-modal landscape to explore. Since all solutions are expected to be equivalent, no attempt was made to distinguish different forms of the solution.

4.2.4.2. PORS Problem

The Plus-One-Recall-Store (PORS) problem is a standard genetic programming test problem, as explained in section 3.2.5. For this study, the number of nodes was set to 16

(designated PORS16.) PORS16 has 24 distinct solutions. While each of the 24 solutions is distinct from the others, the combination of nodes results in 4 permutations of 6 different combinations of the available building blocks (since two of the building blocks have two equivalent forms). Because of this, we examined both the number of different solutions (24 total) and the number of building block combinations (6 total) that were found in each experiment.

The sinusoid function and the PORS problem both have known solutions and so are useful for testing the impact of GBEAs on diversity. For the various trials of the sinusoid function and the PORS16 trials, 5000 runs were performed on each graph. For each of these simulations, the number of different solutions was calculated once 90% of the population had found an acceptable answer. These results were used to find a mean number of different solutions produced by each graph and a 95% confidence interval. It should be noted that since there are many solutions to these problems, the number of solutions found in each individual run was used to perform statistical analysis. Since the solutions are equivalent and come from random initialization, there should be no statistically significant difference when the data is examined in this way.

4.2.5. Takeover Time Experiment Results

This experiment was designed to investigate the spread of information in a GBEA, as well as the effect the difference in fitness has on this rate of spread. Figure 15 shows the information spread (percentage of vertices with higher fitness value) as a function of

mating event for the cycle; Petersen n,1; Petersen n,3; and toroid graphs with a population size of 512. Because of the scaling of Fig.15, the performance of the additional graphs is shown, but not identified. These additional graphs are shown in Figure 16. Figures 15 and 16 show that the general relationship between the number of mating events and the extent of the information spread is initially slow. This occurs because the number of high fitness members in the population is small and the randomly selected individual and its neighbors will not include a high fitness member each time. As the number of high fitness members grows, the extent of the information spread grows nearly linearly, until the population is composed of primarily high fitness members. At this point, the rate of information spread slows due to the difficulty of finding and selecting low fitness members to replace. The results for the graphs with higher connectivity in Figure 16 behave similarly to a logistic curve. As the connectivity of the graphs decrease, the results behave in a more linear fashion, although even the cycle graph displays the S-shape found in a logistic curve.

In the case of the cycle graph, the rate of information spread is constant for the entire experiment until only one low fitness individual is left, as there are always two edges where change can occur. There is a curve at the end of the run, when the final low fitness member must be found. For all other graphs, the rate of information spread is slow to begin with, increases to a nearly linear rate, and then slows again as the last low fitness members are found. As the rate of information spread within the graph increases, the S-shape of the mating event curves becomes more pronounced and more closely resembles a logistic curve.

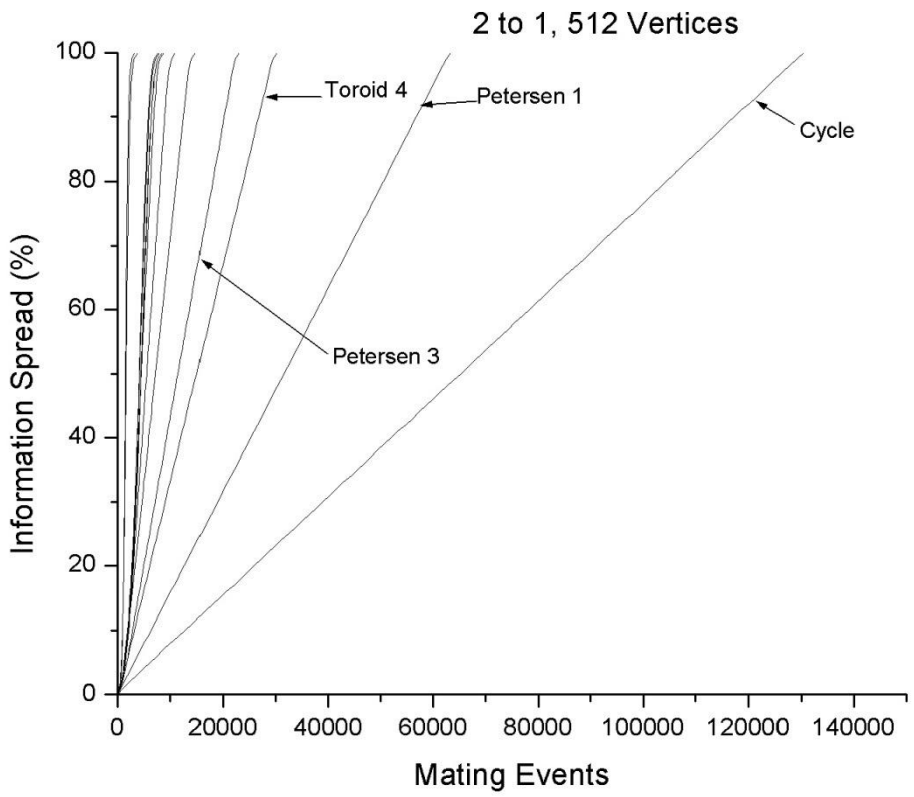

Figure 15, Information spread as a function of mating events and graph, 512 vertices, r= 2.

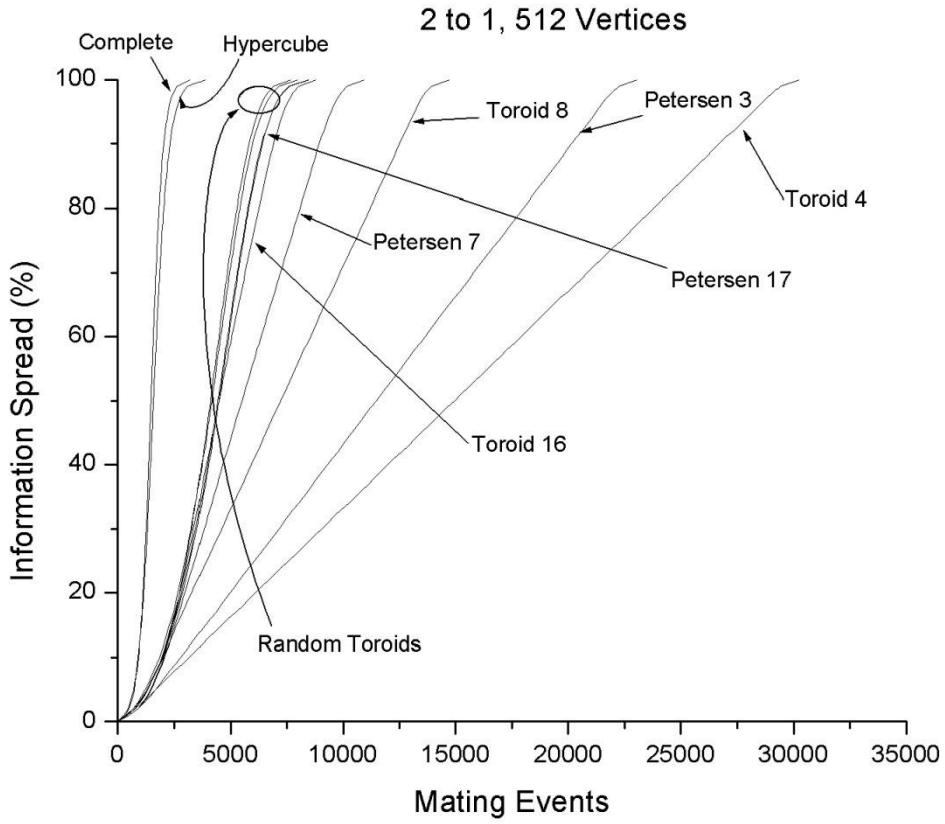

Figure 16, Information spread as a function of mating events and graph, 512 vertices, r= 2.

The ranking of the graphs by slowest to fastest rate of information spread is cycle, Petersen and toroid, random toroid, hypercube, and complete (Fig. 16.) Figures 17 and 18 show the rate of information spread for a population of 4096. As shown, the general shape of the curves and ranking of the graphs remains the same for all population sizes. The toroid and Petersen graphs were interspersed between the other graph families, with higher diameter graphs (Tables 4-7) having a slower spread rate, and lower diameter graphs having faster spread rates (Figs. 17 and 18). This indicates that the takeover times increase roughly the same for any GBEA as the population size increases. This is also true for a decrease in population size, although the effect is diminished for very small population sizes as the differences in the graphs decrease with a decrease in population size.

Another result was that for all population sizes and most graph types, the rate of spread was proportional to *r*. The rankings of the graphs did not change as the fitness ratio was changed (Fig. 19), but there was more of an increase in takeover time for the highly connected graphs when the fitness ratio was increased than for the sparser graphs (Figs 20 and 21.) For the cycle graph, it was found that changing the fitness ratio had little effect on the takeover time.

The number of mating events required for a solution to takeover the population was divided by the population size for each graph in an effort to make comparisons to generational schemes. When the number of mating events is divided by the population

Table 4, Graph Diameters and κ for population size 32 to 256, r = 2.

	32		64		128		256	
	Diameter	κ	Diameter	κ	Diameter	κ	Diameter	κ
Complete	1	3.39	1	4.17	1	4.87	1	5.64
Cyclic	16	15.10	32	31.22	64	63.03	128	127.08
H-n	5	4.27	6	5.58	7	6.36	8	7.09
Pn_1	9	8.07	17	14.24	33	27.39	65	53.81
Pn_3	6	5.31	8	8.96	14	13.41	24	22.41
Pn_7	5	5.34	6	8.28	10	10.57	14	13.96
Pn_17	N/A	N/A	9	9.02	10	10.53	10	12.37
T4_(n/4)	6	5.13	8	10.55	18	19.81	34	38.28
T8_(n/8)	N/A	N/A	10	6.86	12	10.81	20	18.88
T16_(n/16)	N/A	N/A	N/A	N/A	N/A	N/A	16	12.32
RAND3_1	6	5.99	7	8.12	8	10.15	9	11.74
RAND3_2	6	5.89	7	8.46	8	10.16	9	11.88
RAND3_3	6	5.78	7	8.38	8	9.91	10	11.81
RAND4_1	4	4.58	5	6.23	6	7.48	6	8.64
RAND4_2	4	4.73	5	6.22	6	7.40	7	8.58
RAND4_3	4	5.03	5	6.27	6	7.47	6	8.68
RAND9_1	3	4.32	4	5.19	4	5.91	4	6.61
RAND9_2	3	4.24	4	5.21	4	5.93	4	6.60
RAND9_3	3	4.18	4	5.28	4	5.91	4	6.61
RTor_1	4	5.24	8	7.08	10	8.86	13	16.23
RTor_2	4	6.02	6	6.82	9	9.39	26	16.73
RTor_3	6	5.36	5	6.59	8	9.54	17	15.91

Table 5, Graph Diameters and κ for population size 512 to 4096, r = 2.

	512		1024		2048		4096	
	Diameter	κ	Diameter	κ	Diameter	κ	Diameter	κ
Complete	1	6.32	1	7.05	1	7.71	1	8.37
Cyclic	256	255.60	512	512.22	1,024	1,022.36	2,048	2,047.17
H-n	9	7.83	10	8.56	11	9.23	12	9.72
Pn_1	129	106.74	257	211.80	513	423.21	1,025	986.54
Pn_3	46	40.37	88	76.34	174	147.90	344	340.85
Pn_7	22	21.04	42	35.28	78	63.89	150	137.21
Pn_17	18	15.97	25	21.11	34	32.07	67	59.50
T4_(n/4)	66	75.33	130	149.48	258	297.62	514	461.22
T8_(n/8)	36	34.88	68	66.96	132	131.01	260	208.36
T16_(n/16)	24	19.65	40	34.60	72	64.70	136	102.38
RAND3_1	11	13.49	12	15.20	12	17.15	15	16.70
RAND3_2	10	13.44	12	15.18	13	17.05	14	16.60
RAND3_3	10	13.45	11	15.15	13	17.15	15	16.63
RAND4_1	8	9.80	8	10.89	9	12.04	10	12.53
RAND4_2	7	9.75	8	10.91	9	12.06	9	12.48
RAND4_3	7	9.78	8	10.89	9	12.09	10	12.45
RAND9_1	4	7.30	5	8.00	5	8.68	5	9.27
RAND9_2	4	7.32	4	8.00	5	8.66	5	9.31
RAND9_3	4	7.31	4	7.99	5	8.70	5	9.34
RTor_1	19	17.15	23	20.70	38	35.04	30	26.75
RTor_2	20	17.08	29	23.44	47	29.70	50	26.40
RTor_3	16	15.99	25	21.95	29	30.30	40	26.65

Table 6, Graph Diameters and κ for population size 32 to 256, r = 3.

	32		64		128		256	
	Diameter	κ	Diameter	κ	Diameter	κ	Diameter	κ
Complete	1	2.85	1	4.17	1	4.89	1	5.62
Cyclic	16	14.58	32	31.15	64	63.05	128	126.73
H-n	5	3.18	6	5.21	7	5.94	8	6.77
Pn_1	9	7.17	17	15.17	33	29.94	65	58.91
Pn_3	6	4.71	8	7.90	14	12.96	24	22.87
Pn_7	5	4.69	6	6.69	10	8.92	14	12.76
Pn_17	N/A	N/A	9	8.40	10	9.19	10	10.87
T4_(n/4)	6	4.17	8	8.35	18	15.00	34	28.47
T8_(n/8)	N/A	N/A	10	6.41	12	9.21	20	15.04
T16_(n/16)	N/A	N/A	N/A	N/A	N/A	N/A	16	11.28
RAND3_1	6	5.32	7	7.28	8	8.91	9	10.22
RAND3_2	6	5.31	7	7.40	8	8.92	9	10.45
RAND3_3	6	5.25	7	7.52	8	8.73	10	10.44
RAND4_1	4	3.65	5	5.92	6	6.96	6	8.10
RAND4_2	4	3.94	5	5.79	6	6.94	7	8.00
RAND4_3	4	4.07	5	5.85	6	7.06	6	8.11
RAND9_1	3	3.26	4	5.09	4	5.81	4	6.56
RAND9_2	3	3.17	4	5.07	4	5.83	4	6.55
RAND9_3	3	3.15	4	5.13	4	5.79	4	6.50
RTor_1	4	4.35	8	6.73	10	8.42	13	14.78
RTor_2	4	4.94	6	6.50	9	8.91	26	15.28
RTor_3	6	4.52	5	6.29	8	8.58	17	14.31

Table 7, Graph Diameters and κ for population size 512 to 4096, r = 3.

	512		1024		2048		4096	
	Diameter	κ	Diameter	κ	Diameter	κ	Diameter	κ
Complete	1	6.30	1	6.98	1	7.71	1	8.37
Cyclic	256	255.28	512	512.23	1,024	1025.16	2,048	2047.59
H-n	9	7.51	10	8.22	11	9.01	12	9.69
Pn_1	129	117.90	257	235.46	513	470.61	1,025	938.17
Pn_3	46	42.98	88	82.80	174	162.92	344	322.81
Pn_7	22	20.54	42	36.07	78	67.43	150	129.82
Pn_17	18	14.63	25	20.08	34	32.13	67	56.50
T4_(n/4)	66	54.81	130	107.99	258	214.44	514	427.35
T8_(n/8)	36	26.94	68	50.57	132	97.96	260	192.61
T16_(n/16)	24	16.35	40	27.42	72	49.76	136	94.57
RAND3_1	11	11.77	12	13.27	12	14.68	15	16.43
RAND3_2	10	11.79	12	13.22	13	14.74	14	16.37
RAND3_3	10	11.76	11	13.16	13	14.69	15	16.39
RAND4_1	8	9.10	8	10.14	9	11.23	10	12.34
RAND4_2	7	9.13	8	10.19	9	11.22	9	12.30
RAND4_3	7	9.06	8	10.17	9	11.28	10	12.33
RAND9_1	4	7.27	5	7.91	5	8.65	5	9.33
RAND9_2	4	7.23	4	7.94	5	8.65	5	9.32
RAND9_3	4	7.28	4	7.95	5	8.67	5	9.37
RTor_1	19	15.58	23	18.81	38	26.79	30	24.26
RTor_2	20	15.61	29	21.00	47	26.82	50	23.99
RTor_3	16	14.77	25	20.06	29	29.77	40	24.22

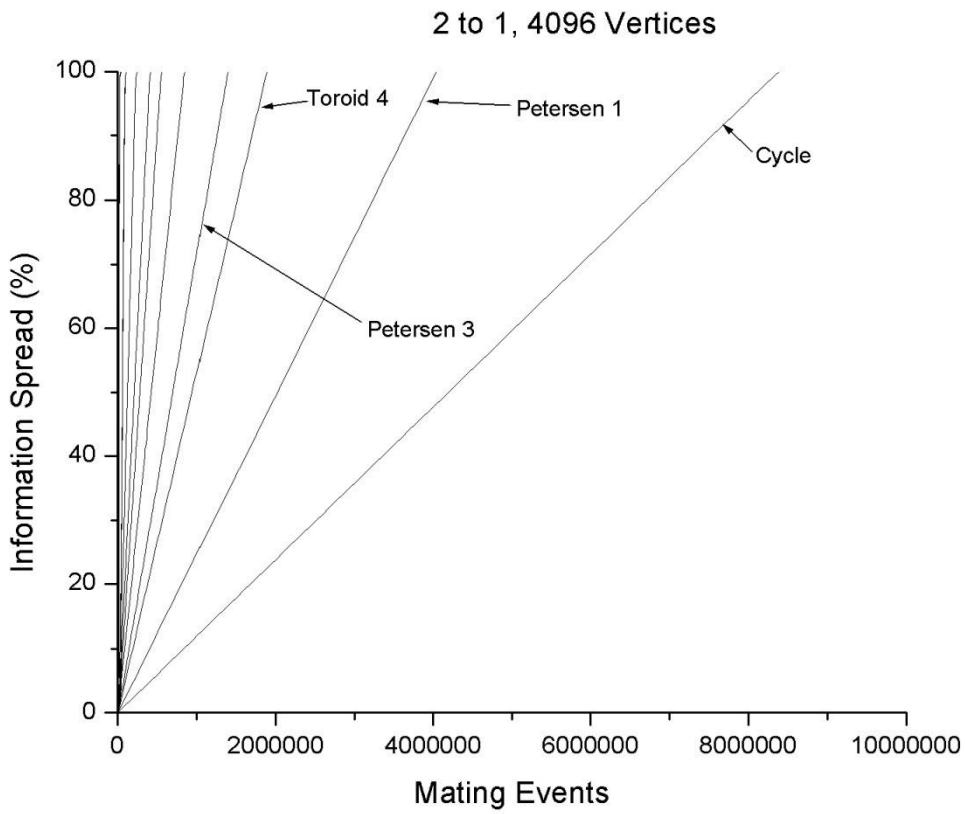

Figure 17, Information spread as a function of mating events and graph, 4096 vertices, r= 2.

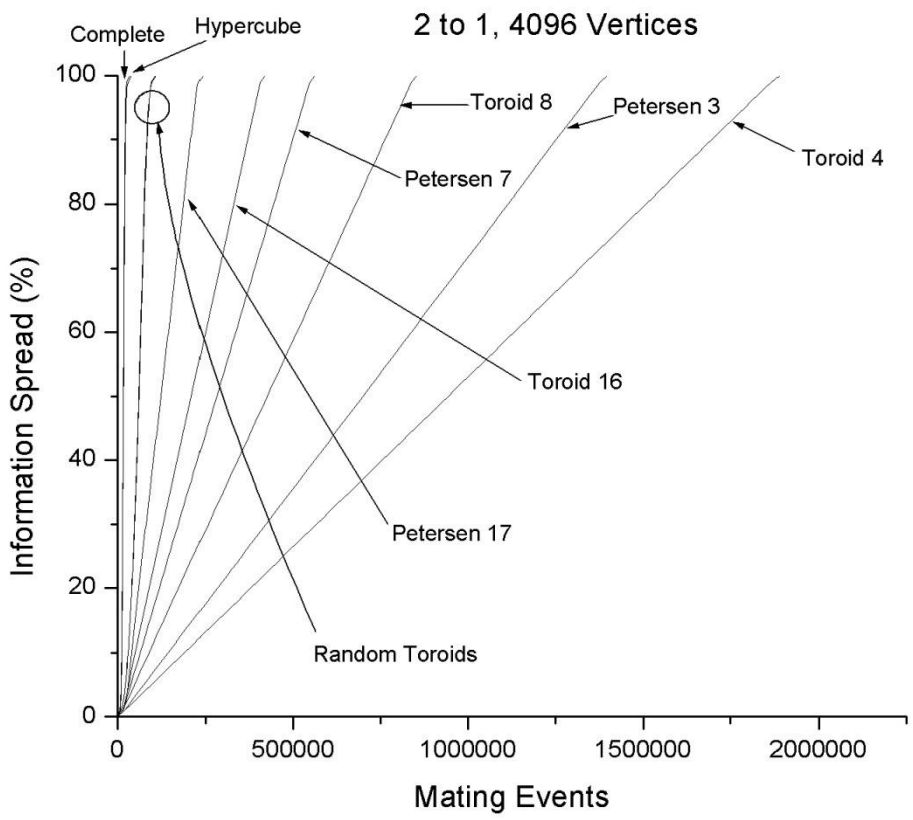

Figure 18, Information spread as a function of mating events and graph, 4096 vertices, r=2.

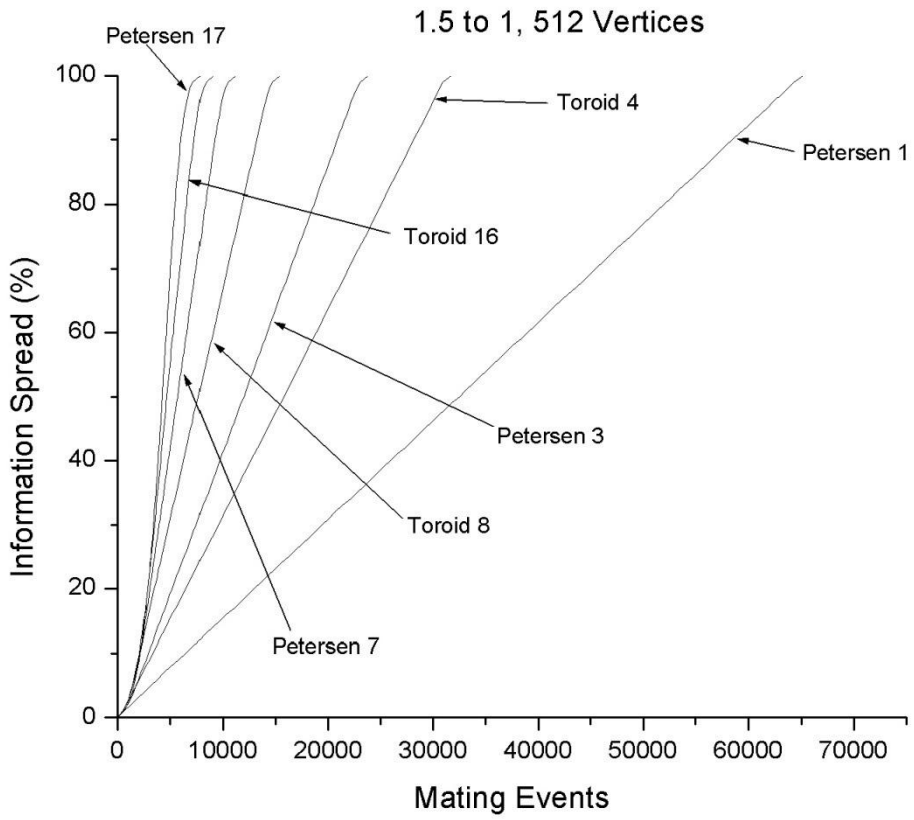

Figure 19, Information spread as a function of mating events and graph, 512 vertices, r= 1.5.

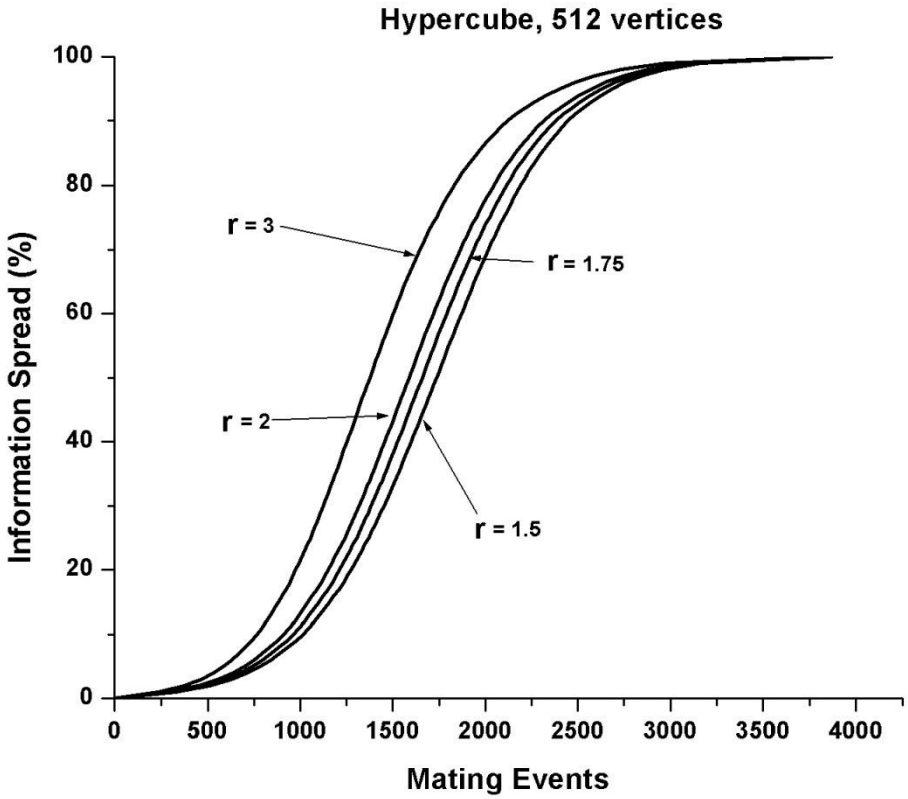

Figure 20, Information spread as a function of mating events and r for the hypercube graph, 512 vertices.

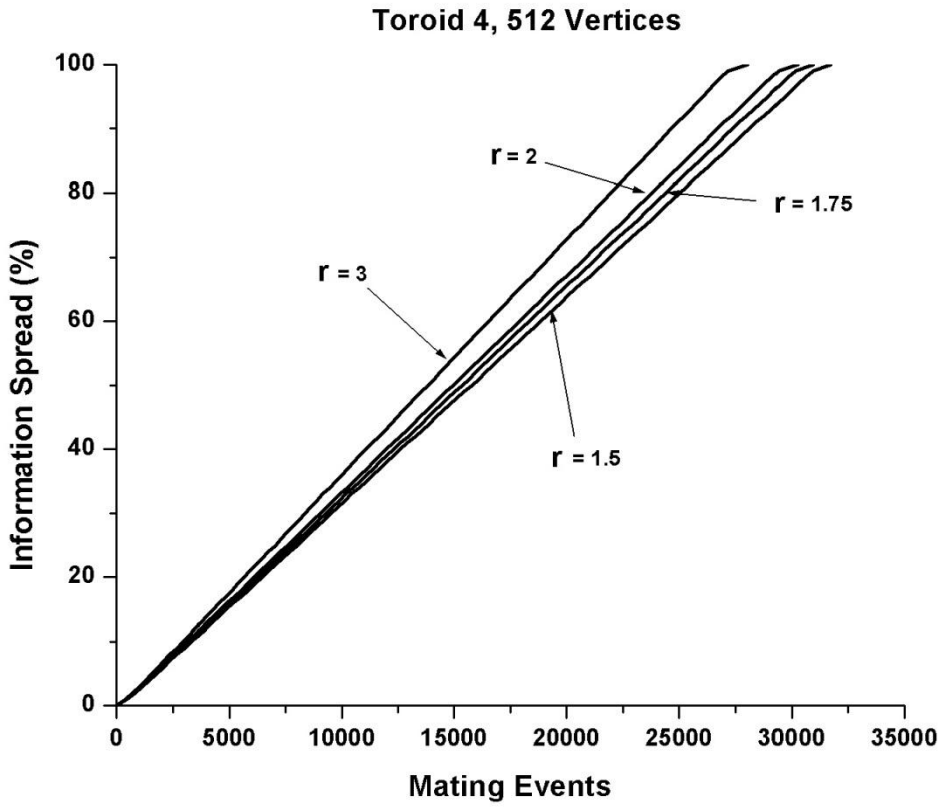

Figure 21, Information spread as a function of mating events and r for the toroid 4 graph, 512 vertices.

size (κ , Eq. 4-15), it can be seen that, for the hypercube graphs, the difference increases in fixed amounts as the population size is doubled (Fig. 22.) For the other types of

$$\kappa = \frac{Number\ of\ Mating\ Events}{Population\ Size} \tag{4-15}$$

graphs, this increase appears to be a doubling of the difference per increment (Figs. 23 and 24.) When κ at 100% information spread is compared to the graph diameters as shown in Tables 4 through 7, they are found to be roughly equal for all graphs but the complete and hypercube graphs (Figs. 25 and 26.)

To validate the results of the empirical experiments, the results of the complete and cycle graphs were compared to the analytical solutions. While the empirical experiments do have stochastic mating rules, the standard deviation of the results was insignificant. The analytical and empirical results for the cycle graph (Fig. 27) differed by 110 mating events compared to a mean value of 130,450 mating events for the experimental results. Both plot behaving linearly with identical slopes and a slight curvature at the tails of the plot. A comparison of the analytical and empirical results for the complete graph (Fig. 28) shows that the expected value plot leads the empirical plot at the beginning and the end of the curves, with a maximum difference of 146 from the experimental results mean value of 3237. Both of these results are well within a standard deviation of the empirical results, and so are considered valid results.

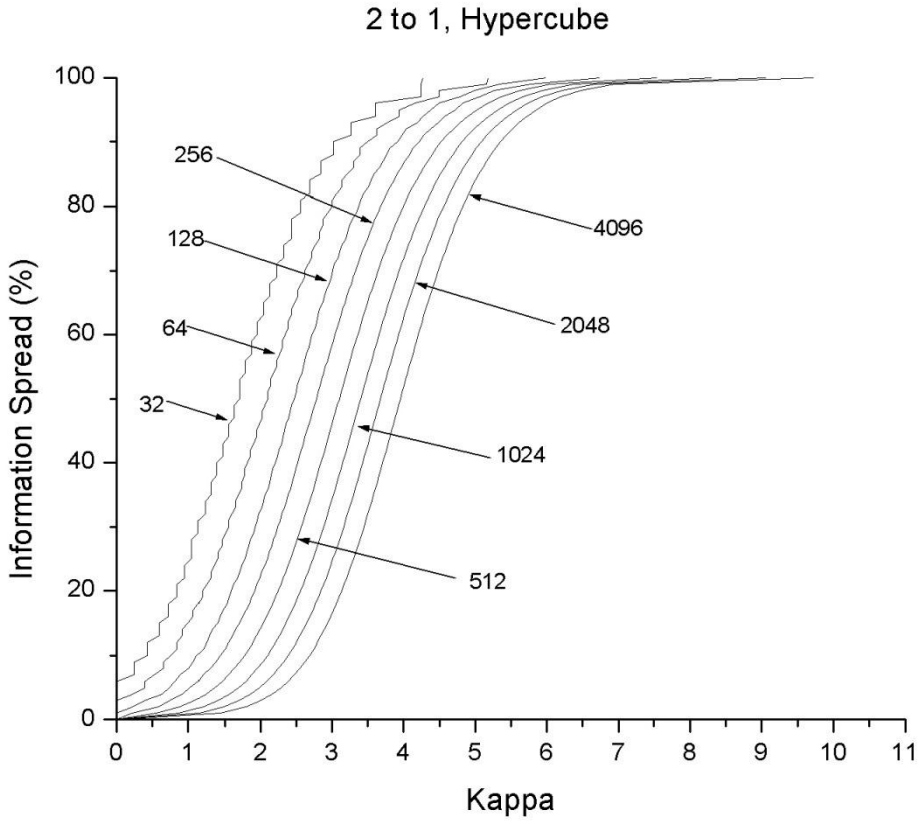

Figure 22, Information spread as a function of mating events/population size for the hypercube, r=2.

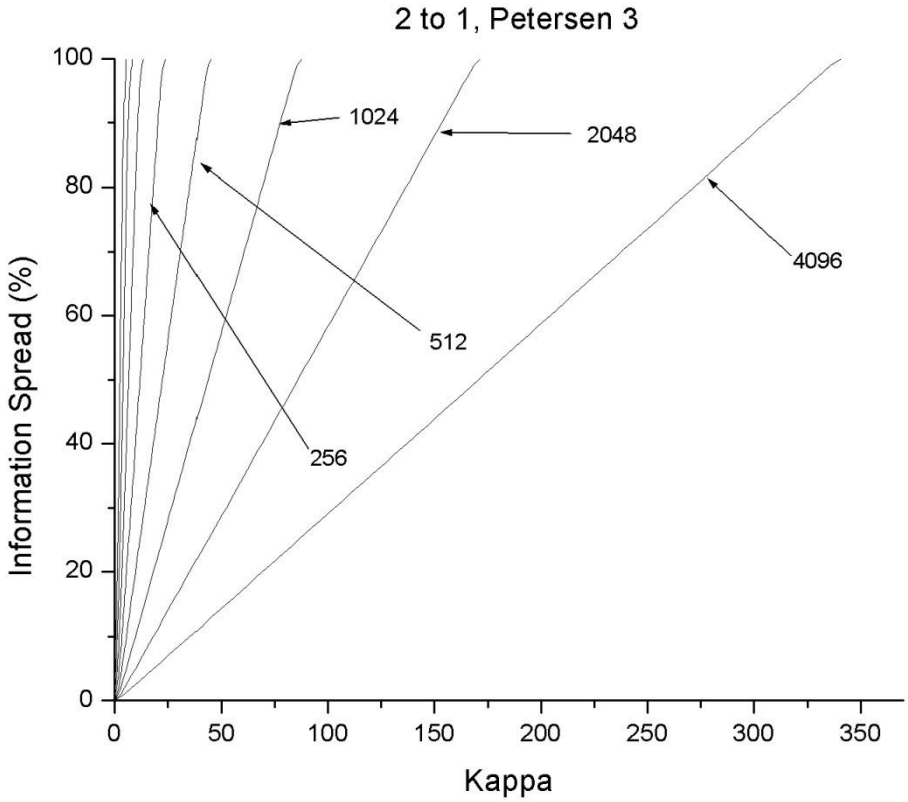

Figure 23, Information spread as a function of mating events/population size for the Petersen 3 graph, r=2.

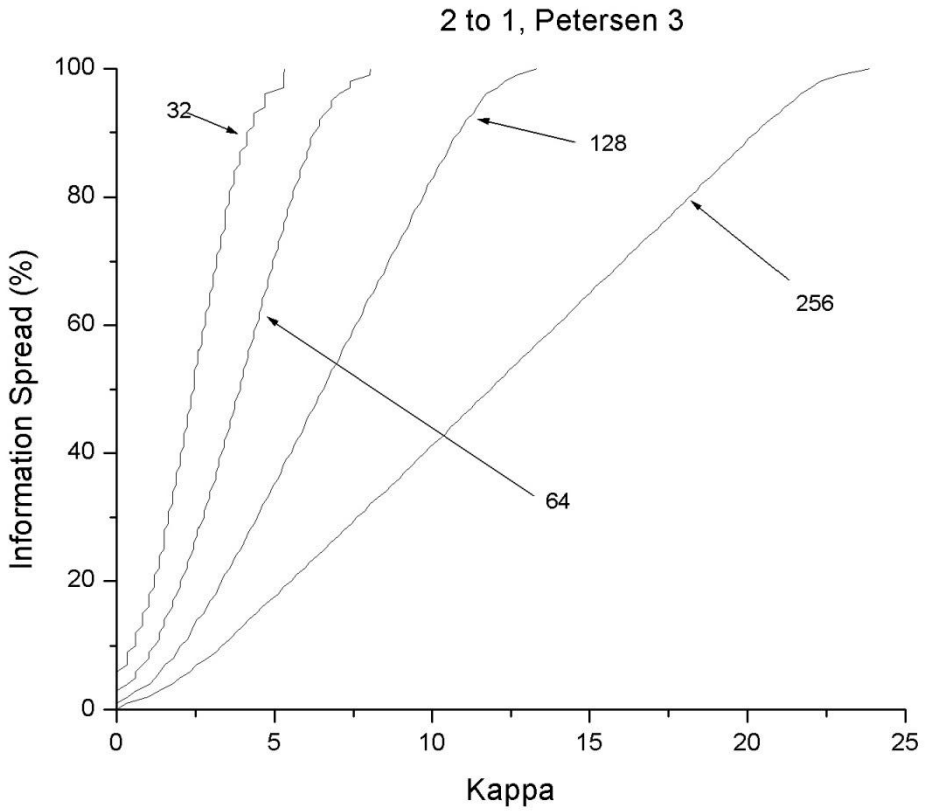

Figure 24, Information spread as a function of mating events/population size for the Petersen 3 graph, r=2.

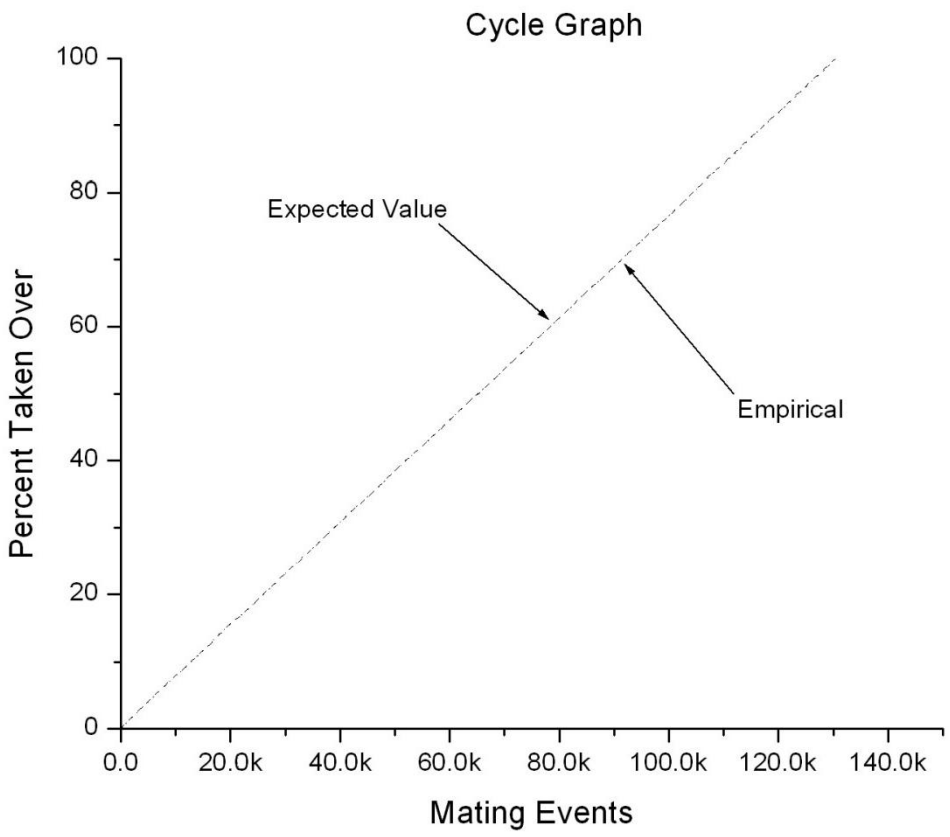

Figure 25, Takeover times for cycle graph by method, n=512.

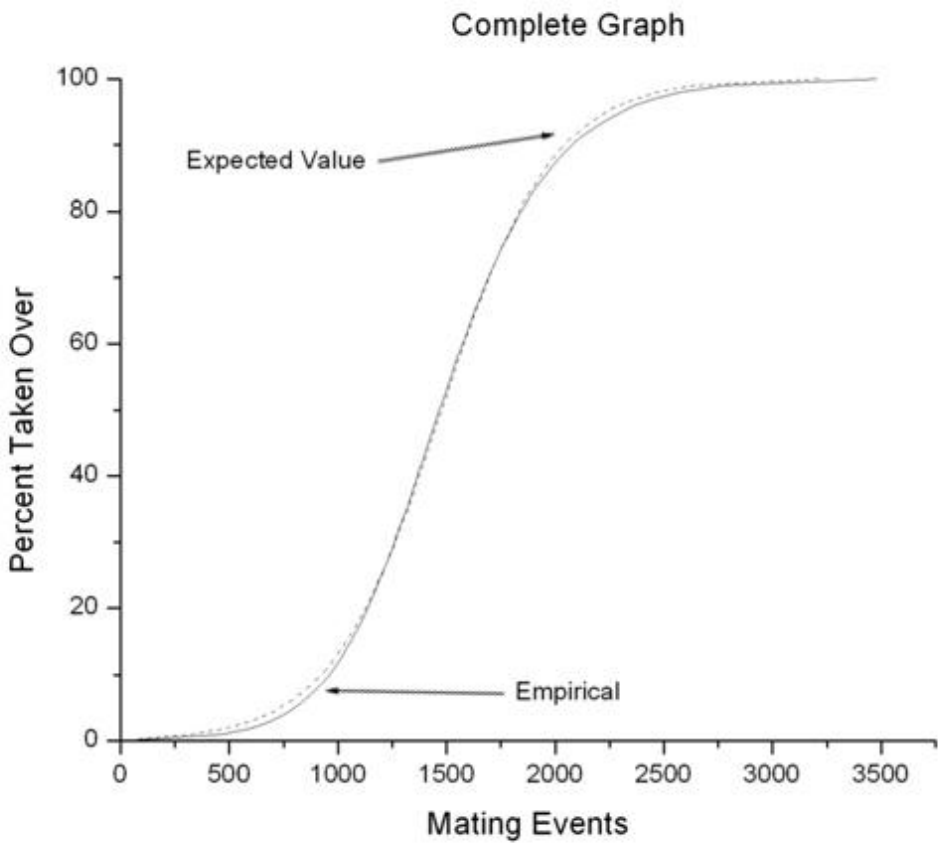

Figure 26, Takeover times for complete graph by method, n=512.

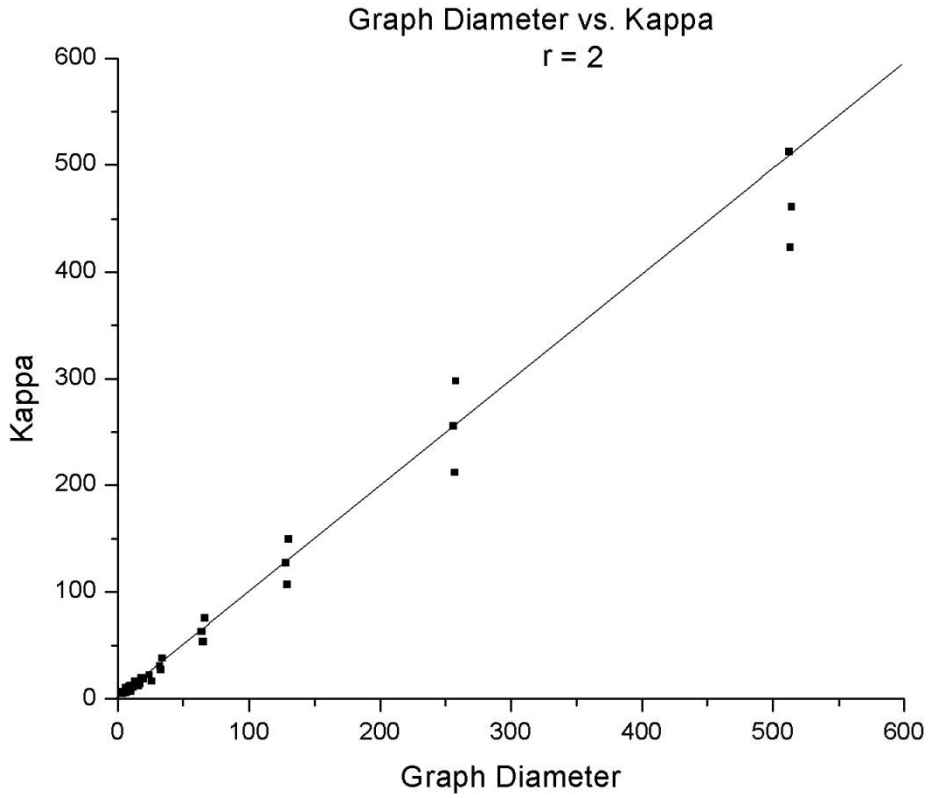

Figure 27, Kappa as a function of graph diameter for r=2.

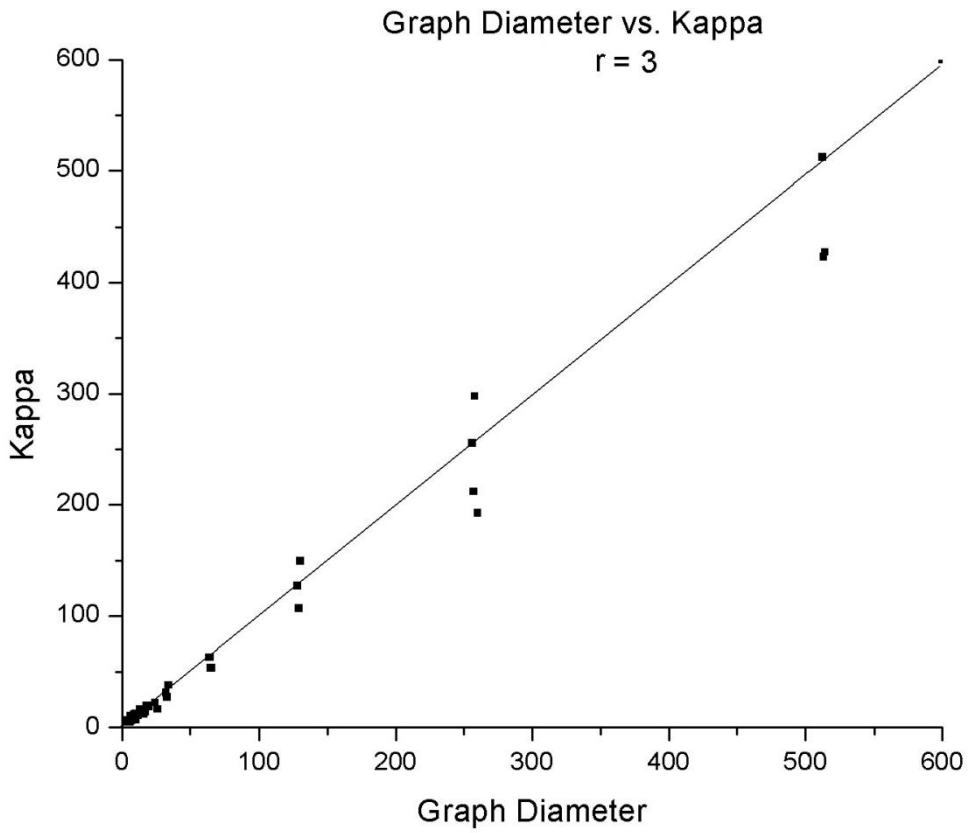

Figure 28, Kappa as a function of graph diameter for r=3.

This experiment was designed to investigate the effect that the choice of graph would have on the number of different designs found when using a GBEA. All of the experiments showed that there is a correlation between sparsity of graph used and how many different solutions are obtained.

For the sinusoid function, the complete graph yielded the smallest variety of solutions, finding about six per run regardless of the dimension of the problem. (Note that using the complete graph is equivalent to running a standard evolutionary algorithm.) Using the hypercube resulted in more solutions as the dimension of the problem increased, but only slightly. Using the remaining graphs resulted in a marked increase in the number of solutions found as the dimension of the problem increased. The toroid and Petersen graphs performed similarly, with the average number of solutions increasing as the diameter of the graphs increased. Performance using the Petersen-1 graph was similar to that using the cycle graph (Fig. 29.) The surprising result of this study is that the simplexified graph (designated Z) had the largest variety of solutions for higher dimension problems even though it has the same degree as the toroid graph (Fig. 30.)

The PORS problem exhibited similar trends with the exception of those for the simplexified graph (Fig. 31.) As the degree of the graph decreased, the number of different solutions present when the run completed increased. Since there was no single solution that dominated the others, the results were ranked by number of solutions found in each run. These rankings reflect generic solution types; whichever solution happens to be dominant for a particular run and not a particular solution. In this way, the statistical

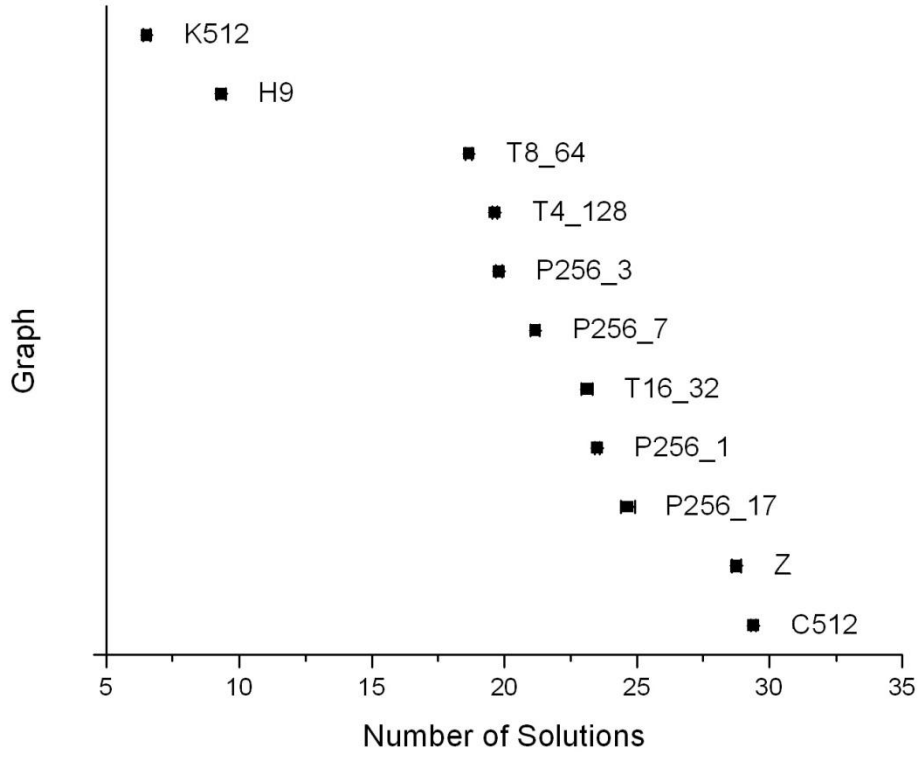

Figure 29, Number of solutions found by graph for the 3 dimensional sine problem.

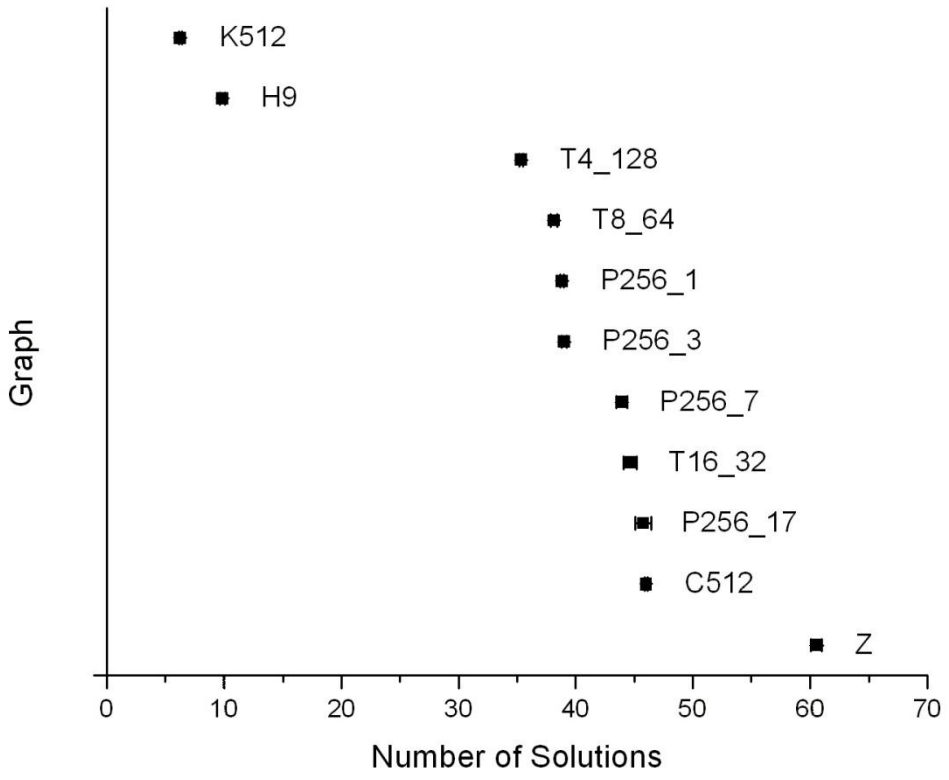

Figure 30, Number of solutions found by graph for the 9 dimensional sine problem.

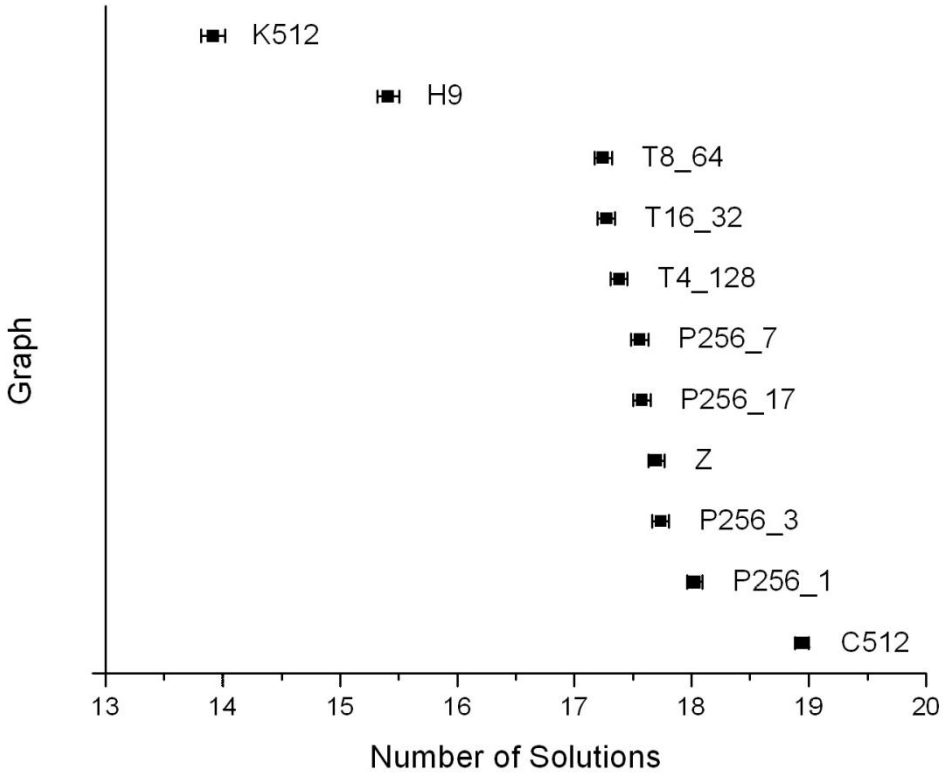

Figure 31, Number of solutions found by graph for the PORS16 problem.

analysis shows the expected number of solutions and the level of population dominance, but does not infer that any solution is overall superior to any other. The population on the complete graph was nearly 60% of the same type of solution, while there was a much more diverse population found on the cycle graph. When only distinct solutions were considered, these results were even more pronounced with nearly half of the population comprised of a single solution for the complete graph compared to about 25% for the cycle graph.

It can be seen that when the graph diameter and population size increase, the rate of information spread decreases, while increasing the fitness ratio has the opposite effect. The number of mating events for the entire graph to be exposed to a superior solution appears to be directly proportional to the population size and graph diameter, and inversely proportional to the fitness ratio. It is interesting to note that even for the complete graph the value for kappa increases as the population size increases, indicating that the takeover times do not scale directly with population size. It also appears that there is an inverse relationship between graph degree and takeover time, although this seems to be to a lesser degree than the other factors as seen by the intermixing of the Petersen graphs (degree 3) and the toroid graphs (degree 4.)

It has been a general practice (Golberg and Deb, 1991) to divide the number of mating events in a steady state algorithm by the half the population size to get an equivalent number of generations to compare these steady state algorithms to generational algorithms. While this serves well as an easy approximation to make comparisons, a

review of the kappa values found in this study indicates that error may be introduced by using this assumption. When fitness proportional selection is being used, the bias towards selecting a more fit mating partner skews the search towards exploiting the better solutions at the expense of the lower fitness solutions. This means that the less fit individuals are involved in fewer mating events as the solutions evolve, speeding convergence to a solution. As the graph connectivity decreases, this effect is decreases until its effect is nearly negligible in the cycle graph. This can be seen in tables 4 through 7, where the rate at which kappa increases for the hypercube graph decreases as population size increases, compared to the cycle graph where the rate is consistently half the population size.

Also shown in this study is that GBEAs enhance solution diversity. In the case of the real-valued problem in nine dimensions, the number of different solutions was increased by a factor of 9.6. When applied to a real world application of the sprayer nozzle (Engelbrecht, 2007), three times as many solutions were found despite using a small population size. These results support the theory that GBEAs enhance diversity in an evolving population. As shown by the number of different solutions found in the real-valued problem, the graph choice can have a strong impact on the diversity present in the population. The availability of multiple solutions could prove invaluable in design processes, not only by giving different alternatives to the decision maker in the early stages, but also by providing options further along in the process should the initial design be found infeasible.

Unfortunately, there is a tradeoff between the number of solutions found in the population and the number of mating events required to arrive at the desired number of solutions. Previous results have indicated that the PORS16 problem was best solved (fewest mating events required) by those graphs that had higher connectivity and smaller graph diameter. Given the current thinking on diversity preservation, this tradeoff is what one would expect. PORS16 is a simple optimization problem requiring little diversity to solve. The more difficult and deceptive problems are best approached using diversity preserving graphs, the same graphs that are shown here to find multiple solutions.

4.3. Conclusion

This study provides additional evidence that graphs can be used to tune the rate of information spread in a population to maintain and control diversity. Earlier studies have indicated that diversity is a key parameter in determining the performance of an evolutionary algorithm. In addition, this type of study provides a relatively quick and easy methodology for comparing graph performance. These results indicate that if there is an initial estimate of the deceptiveness of a landscape, a significant speed up can be realized by utilizing the proper graph and population size combination.

It was also found that the graph set used in the initial work (Bryden, et al., 2006) had several graphs that had identical performance on the problems examined. By removing these redundant graphs, a winnowed graph set can be developed. This winnowed set can range in size depending on the degree to which the researcher wishes to explore the effects of controlling the flow of information in the evolving population. Some guidelines

for graph selection using a population size of 512 are given in Table 8. Similar to the recommended set of test suite problems there are two recommended choices; a smaller graph set for use with problems that are computationally expensive and a list of additional graphs that may be used to further investigate the effects of controlling information flow. The smaller set of graphs comes from the observations of the population size study, where it was seen that the performance of the graphs separated into graph families. The smaller graph set is comprised of those graphs that showed performance indicative of their respectively families when the takeover times were calculated in Section 4.2. The list of additional graphs is comprised of the remaining variants of the Petersen and toroid graphs and the first random toroidal graph. The additional Petersen and toroid graphs were included because they allowed for smaller changes in diversity preservation compared to the smaller graph set. The random toroidal graphs displayed performance similar to the toroid and Petersen graphs, and so were not included in the smaller set, although the first random toroidal graph was included in the larger set as a representative of that family of graphs. These recommendations also hold for population sizes other than 512, although smaller population sizes will preclude some of the graphs that cannot be constructed or are redundant, as discussed in section 4.1.

While more information needs to be gathered to give definitive advice on which graph will give the best results, there is sufficient information to formulate some guidelines to the use of graph based evolutionary algorithms. To use these guidelines, some of the characteristics of the problem must be know a priori. These characteristics are the

Table 8, Winnowed graph sets for use in evaluation of evolutionary computation problems, population size of 512.

Smaller Graph Set:	Larger Graph Set Also Includes:
Complete	P256_1
Cyclic	P256_3
H9	P256_17
P256_7	T4_128
T8_64	T16_32
	RTor7_1

deceptiveness of the fitness landscape, the size of the character alphabet, and the number of variables in the chromosome of the problem being examined.

The first characteristic of the problem to consider is the deceptiveness of the fitness landscape for that problem. If this fitness landscape is thought to be deceptive, a lower amount of information flow is recommended. For many problems it may be impossible or impractical to determine if the problem is deceptive, in which case a moderate amount of diversity preservation would be recommended (for example, using one of the torus graphs.) The results of the PORS15 problem on various graphs (Section 4.1.1, Fig. 7) shows that for this deceptive problem, a graph with a high amount of diversity preservation performs best. However, for problems where it is uncertain whether the fitness landscape is deceptive, the better compromise solution would be those graphs with an intermediate amount of information flow. Depending on the computational resources that are available, two trials could also be run; one with a diversity preserving graph and one with a highly connected graph. This could be used to determine the deceptiveness of the fitness landscape and return an optimal solution with minimal wall time.

The second characteristic to consider is the size of the available character alphabet for discreet problems. For problems with a smaller available alphabet, the benefit of using a diversity preserving graph increases. This is largely due to the decreased difficulty in assembling building blocks to be assembled into the final solution. This can be seen by comparing the results of a small alphabet problem (PORS16; Section 4.1.1, Fig. 6) to the results of a problem with a larger alphabet (the north wall builder problem; Section 4.1.3,

Fig. 9.) The problem with a smaller alphabet shows twice the improvement at small population sizes compared to the large alphabet problem.

The third characteristic is the number of variables in the chromosome of the problem. This is related to the size of the available character alphabet, in that more information is contained in each population member. Because of this, a larger population size is required to provide sufficient diversity to construct the optimal solution. This also makes diversity preservation important to maintain this information, although there is also a benefit to sharing information if the problem is not deceptive.

The qualitative guidelines for graph selection can be described as follows:

- For a simple uni-modal problem, the complete graph is preferred (which is also equivalent to a standard evolutionary algorithm.)
- If the problem being examined is thought to be deceptive, a diversity preserving graph is recommended. The more deceptive the fitness landscape, the sparser the graph.
- For problems with a large alphabet and/or a large number of variables, a graph with an intermediate amount of diversity preservation is preferred, such as the torus or Petersen graphs.

It should be stressed that these guidelines for the use of GBEAs are a starting point for the analysis of the given problem. While these recommendations do not guarantee the

best time to solution for a particular problem, they should still deliver satisfactory performance.

5. APPLYING GRAPH BASED EVOLUTIONARY ALGORITHMS

An important goal of this research is to present graph based evolutionary algorithms as a tool in both the design process and for the modeling of systems. The preceding chapters have given some insight into the behavior of both evolutionary computation problems and the different graph structures used in GBEAs, now it remains to apply the knowledge gained to the solving of real world problems. In this way it can be shown that GBEAs are more than a classifying system and an interesting mathematical construct; they are a means to augment the engineering decision process. The benefits to engineering design can be seen in both the time necessary to find an optimized solution and in supplying a wider variety of acceptable solutions, promoting computational creativity.

Some of the problems described in Chapter Three fall under the category of applied problems. These problems explore issues relating to current scientific research as well as meeting the criteria of viable test problems. Examples would be the DNA barcode problem and the Steiner systems problems (Ashlock, Guo, and Qiu, 2002; Ashlock, Bryden, and Corns, 2005). The results of these experiments show that proper graph selection has a strong positive impact on time to solution. By researching applied problems such as these it is possible to develop both a taxonomy and a proposed test suite for evolutionary computation problems that are relevant to research topics of industry.

Many of the problems found in industry would not be acceptable as test problems. The most common reason for this is the amount of time required to complete a fitness evaluation. For example, Bryden, Ashlock, McCorkle and Urban (2002) used a GBEA to

improve the performance of third world cook stoves. The goal of this study was to determine the optimal placement of baffles in a Plancha stove. Plancha stoves are inexpensive and easily assembled stoves constructed of cast concrete with a metal cooking surface. These stoves are designed to replace open fires for cooking needs in rural Guatemala, decreasing the amount of fuel required and reducing health risks from open flames and the accumulation of smoke in dwellings. The design of these stoves is challenging due to limitations on construction methods that severely restrict the number of baffles that may be placed under the heating surface to direct the hot flue gases from combustion. The goal of this research was to optimize baffle placement under the heating surface, with the fitness being a measure of how uniform the stove top temperature was. Three strings representing baffles were used as chromosomes in the GBEA. These strings were passed to a commercial computational fluid dynamics (CFD) package to return the fitness of the solution. Each fitness evaluation required approximately 3 minutes to perform, making completion of one run very time intensive and a sufficient number of runs for statistical data unreasonable. However, from the data obtained, it was found that this optimization problem, when represented properly, was a fairly simple uni-modal problem and was solved best by the complete or hypercube graphs. This agrees with the results found in Chapter Three of this research for other uni-modal problems.

With the combined knowledge of how GBEAs act to control information flow and the affect this has on problems examined thus far, we are now ready to apply this tool to another applied problem. While several problems exist that would benefit from this method, the problem to be examined here relates to the use of antibiotics as performance

enhancers for the raising of swine for human consumption. This problem is interesting in that it brings together elements of mechanical engineering, biology, economics and public policy to address concerns in the environment, the health care industry, and agricultural businesses. The remainder of this chapter is dedicated to a discussion of this problem and how it can be approached using graph based evolutionary algorithms.

5.1. Bacteria and Swine Growth Model

Since the late 1940s antibiotics have been used as an additive to livestock feed (Cromwell, 2001.) Antibiotics have been shown to both increase the growth rate and decrease the mortality rate of animals when administered in sub-therapeutic doses (doses smaller than that required for disease treatment.) There is also a large body of evidence indicating that antibiotic use has led to improvements in feed to growth ratio, reproduction rates and overall animal health (Hays, 1977; Cromwell and Dawson, 1992; Zhi, Nightingale, and Quintiliani, 1988.) One perceived problem with the widespread use of antibiotics in the swine industry is that of antibiotic resistance development among foodborne bacteria, e.g. *Salmonella spp.* and *Campylobacter.spp*. It has been well established that when bacteria are exposed to antibiotics in vitro, resistance to that antibiotic can develop (Prescott, Baggot, and Walker, 2000.) Using this information it seems intuitive that the use of sub-therapeutic levels of antibiotics for performance enhancement could cause resistance in the bacteria present in the gastro-intestinal (GI) tract of swine. However, the actual risk of using sub-therapeutic levels on human health by way of the ratio of bacteria resistant to that antibiotic in either animal or human

reservoirs has not been quantified. This has led to debate as to whether performance enhancing substances should be restricted or banned (Cox, Copeland, and Vaughn, 2005; Singer, Cox, Dickson, Hurd, Phillips, and Miller, 2004; World Health Organization, 1997, 1998, and 2001.)

What is needed now is specific guidance as to what is prudent or optimal use. This might be defined as minimizing the fraction of antibiotic resistant bacteria present in the animal when it is sent to slaughter without comprising animal health and performance. By adjusting the number of times and the amount per treatment that the antibiotic is given to the animals it is possible to explore different options for swine production to minimize antimicrobial resistance.

5.2. The Need for Bacteria Models

To understand the need for modeling bacteria in the swine GI tract it is necessary to define the risk that the bacteria pose to human health. A stepwise risk assessment-based model (Fig. 32) for determining this risk was introduced by Hurd (2006.) While this diagram was made for use in the poultry industry, it is a general model that can be applied to any food animal production system.

The release assessment portion starts by the administration of an antibiotic (in this case a macrolide) to the animal. This can lead to antibiotic resistance (RzD selected) in the

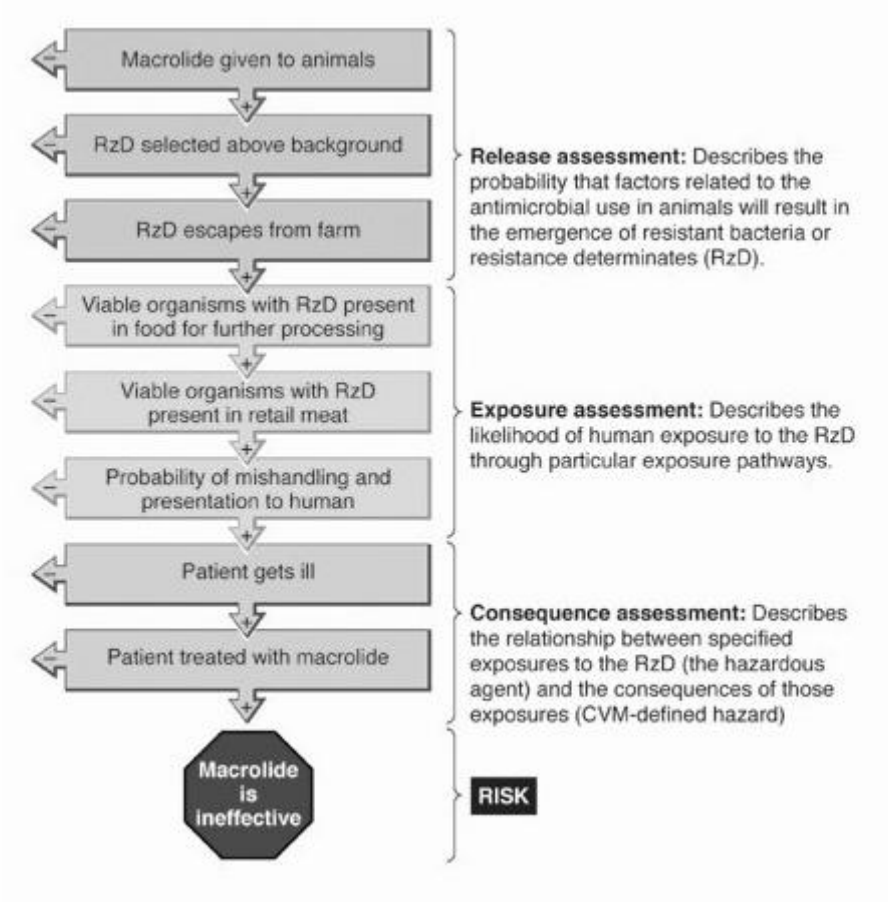

Figure 32, Stepwise Risk Assessment-based approach for estimating the impact on human health from macrolide resistance that develops on poultry farms (Hurd, 2006).

bacteria present in the GI tract. If the animal appears healthy, it will most likely be sent to the abattoir, taking the resistant bacteria with it. The exposure assessment is a measure of factors that may cause the bacteria from the GI tract to be transported to the meat intended for retail sale. This could be from a number of sources, including mishandling of the product, undetected ruptures of the GI tract components or cross-contamination from other carcasses. It also takes into account the probability that this meat will be mishandled and eaten by a human. The final assessment is the consequences, which entails the probability that the patient will get ill from the bacteria and be treated with the antibiotic in question, resulting in treatment failure due to resistance in the bacteria. In previous studies, exploration of these assessments (Hurd, Enoe, Sorensen, Wachmann, Hald, and Greiner, 2004) has shown that a relationship between resistant bacteria arriving in animals at the abattoir and human does exist. It is then apparent that an investigation of steps that could be taken on-farm to lower this risk would be beneficial.

While constructing the model, it is necessary to keep in mind the objectives of this research: to minimize the risk to human health and the cost to raise the animals. The cost of raising the animals is related to how quickly the animals can be brought to market weight. The benefits in weight gain achieved by using antibiotics decreases the amount of time that the animals need to be fed and housed on the farm, decreasing the costs associated with raising the animals. While these economic benefits of sending the animals to the abattoir earlier are easily understood, the affect on human health from antibiotic use as performance enhancers requires additional discussion. To consider the impact on human health, we can apply this objective function:

$$ID = DS * HC_s + DR * HC_R \tag{5-7}$$

Where: ID is number of human illness days

DS is human illness days per case due to susceptible bacteria

DR is human illness days per case due to resistant bacteria

HC_S is the number of cases due to susceptible bacteria

HC_R is the number of cases due to resistant bacteria

The number of human cases can be simulated using an attribution factor that will be determined using historical data that correlates the bacteria level in animals to the number of reported human cases that can be attributed to pork. Data similar to this has already been applied to an investigation of the effectiveness of Denmark's salmonella control program, although this dealt with the number of animals in each herd that tested positive for the bacteria (Hurd, et al., 2004.) There is no evidence that human illnesses are more severe or longer lasting from bacteria that are resistant to the antibiotics used in pork production. However, if it were possible to reduce any possible risk of antibiotic resistant bacteria entering the food chain at no additional cost, it would be a desirable course of action.

5.3. Bacteria, Antimicrobials, and Swine

Antimicrobials are substances that inhibit the growth of or kill microorganisms in low concentration (Prescott, Baggot, and Walker, 2000.) Antibiotics are antimicrobials that

are produced by a microorganism to kill other microorganisms, although the two words are often interchanged. For humans, antibiotics are most commonly used to treat illness caused by bacterial infections. These substances inhibit the bacterium's ability to reproduce effectively or damage the cell function directly. The end effect is that, when used therapeutically, the bacteria die at a faster rate than they reproduce until the concentration is negligible or eradicated.

Antibiotics have varying levels of effectiveness against different bacteria. This is further complicated by antibiotic resistance that can developed either by mutation of the bacteria or the sharing of genetic information in the form of plasmids from other resistant bacteria. This resistance may or may not be permanent, depending on the bacteria and the antibiotic. Resistance to one antibiotic can also translate into resistance to other antibiotics in the same class, although this varies mainly by the class of the antibiotic. These complications make it clear that care must be exercised when administering antibiotics to humans or animals, as the acquisition of antibiotic resistance is a complex relationship.

Unlike human usage, antibiotics are given to poultry and swine for three different reasons; performance enhancement (often incorrectly termed "growth promotion"), disease prevention, and disease treatment. The regimens (dose, route, and length of time) for administering antibiotics to swine vary depending on the animals' age and physical condition, and are based on manufacturer's recommendations and advice from veterinarians. Typically, a low level of antibiotic is added to the feed of the animal to

enhance performance, resulting in an increase in feed efficiency (decrease the amount of feed the animals eat for the same amount of weight gain.) This effect is generally thought to be from a change in the microbiological ecosystem of the GI tract. When there is cause for taking preventative measures, such as when the animals are weaned or other times when it is suspected the animal may easily contract a disease, a larger dose is administered. This can be done by either increasing the amount of antibiotic added to the feed or by adding a water soluble antibiotic to the water given to the animals. When using antibiotics for disease treatment they are normally administered by either addition to drinking water or by injection. Given that there are thousands of different variants of bacterium existent in the GI tract of an animal and hundreds of different antimicrobial treatments available, the task of finding a superior regimen is challenging.

5.4. Previous Bacteria Models

The modeling of a living system is different from most mechanical models in that we understand the underlying relationships more fully in most of the mechanical models in use today. Great strides have been made in gaining understanding of biological systems, but there is still much to be learned. In our case, this includes the interactions between the host animal and the bacteria and between different bacteria in an animal's gastro-intestinal ecologies. The following is an overview of the methods used to model these systems.

5.4.1. Early Mathematical Models

The Malthus Equation is generally considered the first mathematical model for population growth (Schmidt, 1992.) Introduced in 1798, this is a simple formula stating that the rate at which a population increases is equal to the population size multiplied by a growth factor. While this was not introduced as a mathematical model, it can be written as:

$$\frac{dN}{dt} = \mu_{\max} N \tag{5-1}$$

which on integration yields:

$$N = N_O e^{\mu_{Max} t} \tag{5-2}$$

Where: N = the population density at time t

N_O = the initial population density

M_{Max} = the maximum specific growth rate

The first model to be applied to population growth of microorganisms was the logistic equation (Schmidt, 1992.) This model incorporated the availability of resources into the equation by adding the environment's carrying capacity (K) into the equation:

$$\mu = \mu_{\max} \frac{1 - N}{K} \tag{5-3}$$

Where μ is the specific growth rate and is used instead of the maximum specific growth rate in equations 5-1 and 5-2. These equations can give a good idea of how populations grow, but since there is no input into what is controlling the rate, it is difficult to apply any factors that would allow for control of the maximum population size.

The next step in growth modeling was developed by the French scientist Monod in 1949. The Monod equation added a factor to account for limits on population growth rate imposed by limitations in the system (Schmidt, 1992):

$$\mu = \mu_{\max} \frac{S}{K_S + S} \tag{5-4}$$

Where: S = the concentration of limiting substrate

K_S = the half-saturation constant

The model introduced by Monod dealt with a finite amount of substrate to sustain the population (Eq. 5-4), but other variants exist. This model is also more flexible in that it can be used to fit growth curves that are not symmetrical.

Most of the modern models used now are based on the Monod equation, with modifications incorporated to address special circumstances. Some of these circumstances include problems modeling energy expended by bacteria to sustain life

(maintenance energy) and the production of toxic waste by the bacteria. Variations of this equation are also used in simulation packages available on the internet (Food Safety First, 2006; Food Safety and Inspection Service, USDA, 2006.) Past modifications to this model also included the effects of antibiotics on the growth rates (Hochhaus and Derendorf, 1995; Nolting and Derendorf, 1995; Zhi, Nightingale, and Quintiliani, 1988.)

The Monod model and those models based on it are flexible and can be used to match many observed growth curves of bacteria in-vitro. In general practice, they are used on systems with a finite amount of substrate for the bacteria to grow from. They can take into account the presence of antibiotic, if the proper modifications are applied, but it is left to the user to determine what these modifications are.

5.4.2. Lipsitch and Levin

Lipsitch and Levin (1977) did work that focused on the development of antimicrobial resistance in bacteria. This research started with the logistic equation, but added in a loss term to account for interaction with antibiotics. This was expanded to investigate multidrug treatments, different dosing regimens and non-adherence scenarios. Statistical models were used to predict when and how many of the bacteria present gain resistance based on treatment dose and timing.

The model can take into account multiple antibiotics at once with varying dosings, but only models one bacteria type. It is limited in that it only considers the acquisition of

antibiotic resistance through mutation. It also uses discrete values for the levels of antibiotic present. In a non-laboratory biological environment it may not be possible for drug levels to change quickly, depending on the biological half-life of the antibiotic in question.

5.4.3. Nikolaou and Tam

Nikolaou and Tam (2005) introduced a model that stressed that the level of antibiotic resistance in bacteria varies across the population. They argued that to proper model these resistance levels the growth rate should not be a single term, but based on the cumulants of the distribution representing the resistance of the bacterial population. This allows for extrapolation of growth trends past the 24-hour in-vitro period that was used in the study.

The strong points of this model were that it modeled both bacteria and antibiotics at the same time, while accounting for varying levels of antibiotic resistance in the bacterial population. However, it requires prior knowledge of bacteria growth data to find the growth rate distribution. It also only examines one type of bacteria and one antibiotic at a time, making it ill suited for use in a model that requires multiple strains of bacteria to be accounted for.

5.4.4. The Chemostat

The Chemostat model is also attributed to Monod (Panikov, 1995; Schmidt, 1992) circa 1950. This model was originally developed to investigate continuous cultivation of

bacteria so that useful by-products of the bacteria (such as yeast or ethanol) could be collected for human use. This model can be thought of as a bio-reactor in which substrate is continuously added to the bacteria population to either grow additional bacteria for cultivation or provide energy and resources to the bacteria so the desired by-product will be produced. In the previous models the amount of substrate is normally decreased as the model progresses. In contrast, the amount of substrate in this model is controlled to represent systems in which there is a continuous introduction of materials for growth and possibly the removal of toxins.

This model is the basis for most steady state bacteria models. It uses the Monod equation, but is modified to account for the addition of substrate, and possible other bacteria or compounds (Abrosov and Kovroc, 1977). To date, there has been no work that combines a chemostat model with the acquisition or development of antibiotic resistance in bacteria, although this would be a natural extension of this model.

5.4.5. BacSim

The BacSim model (Kreft, Booth, and Wimpenny, 1998; Kreft, 2006) is an agent based bacterial growth model that models the bacterium individually. Each bacterium is represented by an agent. This agent interacts with the other agents (bacteria) in this virtual world. As it is an individual-based model, it investigates concerns at a microscopic level, such as substrate diffusion and variations in cell size. The cells can then be viewed as a group, representing a colony of the bacteria type being modeled. The

model is built using various other models, such as the cell division model proposed by Donachie and Robinson (1996) and the maintenance cost model by Herbert (1958). Java code has been posted on the internet that allows for experimentation with the model (Kreft, 2006). The URL for this application is http://www.theobio.uni-bonn.de/people/jan_kreft/bacsim.html.

This model is very detailed, dealing with the growth, reproduction, and death of every cell present. The model is versatile enough to be used for many different types of organisms, and it appears that it could be adapted to multiple types of bacteria at once, although this has not yet been attempted. The detail of the model is also one of its weaknesses. When dealing with bacteria levels that can easily reach millions, this model can rapidly become cumbersome. To date there have been no attempts to implement interaction with antibiotics in either their effect on the growth rate or on the bacteria gaining resistance at this level of detail.

5.4.6. Summary of Bacteria Growth Models

There are several methods for modeling the growth of bacteria, with this being just a short listing. The models currently available are either general out of a necessity to be widely applicable or they are specialized to a point where they are not suitable for an investigation of bacteria and antibiotic interactions in a host animal. It is be possible, however, to use parts of these models as a basis for a new model that includes bacteria

and antibiotic interactions in a system that has a continuous inflow of both substrate and antimicrobial agents.

5.5. Modeling Swine

To accurately model the ecology in the gastro-intestinal tract of swine, it is an accurate model of the animal's growth and GI tract is necessary. One of the major concerns of pork producers is the rate of weight gain in the animal, which is why sub-clinical levels of antibiotics are used. Feed is eaten by the animal and the animal gains weight. The more the animal weighs, the more feed it eats. This volume of feed and water is also important because it is the method by which performance enhancing antibiotics are administered. The rate at which the animals grow and the amount of feed and water they consume is well documented (Lewis and Southern, 2001), and so an accurate model can be developed.

Growing-finishing animals usually start with a weight ranging from 17-24 kilograms, or about 40-50 pounds (Cromwell, 2001.) As an animal gains weight, the rate at which it gains weight increases until it reaches market weight (around 230 pounds.) The fidelity of this weight gain model can be low, such as in Cromwell's work (2001) given in Table 9, or higher fidelity models can be constructed (Schnickel and Craig, 2001) such as:

$$WTGAIN_t = C\left(1-\exp\left(-mt^a\right)\right)+e_{it} \qquad (5\text{-}5)$$

Table 9, Animal weight gain in pounds per hour without antibiotics and with antibiotics (Cromwell, 2001).

Weight Range	Weight Gain (Pounds), No Antibiotic Used	Weight Gain (Pounds), Antibiotic Used
Weight < 55 Pounds	0.035750	0.04125
55 Pounds – 110 Pounds	0.054083	0.06050
Weight > 110 Pounds	0.063250	0.06600

where WTGAIN is the expected weight gain from birth to time t, C is the mature weight, m is an exponential growth decay constant, a is a kinetic order constant, and e_{it} is the residual weight gain for the i^{th} pig at time t, with all constants calculated using live weight data from weaner pigs to market weight animals. A method should be chosen depending on the desired level of fidelity of the model. It should be kept in mind that with a higher fidelity model, there is an increase in the computation resources necessary to find that information.

In most studies, there is also a correlation between antibiotic usage and weight gain, with animals fed antimicrobials in the feed gaining weight at a higher rate (Hays, 1977; Zhi, Nightingale, and Quintiliani, 1988.) Although it should be noted that there does exist some evidence that there is only a benefit in the early stage of animal growth (Dritz, Tokach, Goodband, and Nelssen, 2002.) Another model necessary in modeling the gastro-intestinal tract of swine is the size of the animals GI-tract, and how much feed and water the animal consumes in a day.

The amount of antibiotic ingested by the animal is directly related to the amount of food and water consumed by the animals on a given day. To determine these inputs, it is first necessary to identify the amount of energy intake. This can be calculated based on the animal's weight:

$$DEI = 1250 + 188BW - 1.4BW^2 + 0.0044BW^3 \quad (5\text{-}6)$$

Where DEI is the digestible energy intake in kcal/day and BW is the body weight of the animal in kilograms. Assuming an average energy content for feed of 3400 kilocalories per kilogram (National Research Council, 1998), the mass of feed consumed per day can be calculated. To determine the amount of water ingested by an average animal each day, a comparison between feed intake and water intake can be used. Research by Braude, Clarke, Mitchell, Cray, Franke, and Sedgwick (1957) shows that when swine are allowed to eat as much feed as they desire, they consume approximately 2.5 kilogram of water per kilogram of feed.

To calculate the concentration of antibiotic or bacteria in the GI tract of an animal, it is first necessary to determine the volume of the GI tract. For a fully grown animal, the GI tract has a volume of about 27.5 liters (Patience, Thacker, and deLange, 2005), including the stomach, cecum, small intestine, and colon. To determine the size of the GI tract of the animal from weaner pig to market weight, it is assumed that the volume of the GI tract increases at the same rate as the animal gains weight.

These equations and relationships make it possible to construct a model of the rate of weight gain in swine, also taking into consideration the amount of feed and water the animals consume. This modeling of the GI tract of swine allows for calculating the amount of antibiotic that is ingested by the animals, and by using the volume of the GI tract it is possible to make reasonable comparisons on the antibiotic concentrations present. It also makes it possible to determine the bacteria concentrations present. By

using the proper combination of models and making some modifications, it is possible to make a reasonable computation model of the ecology of an animal's GI tract.

5.6. New Bacteria/Swine Model

To investigate the interactions between swine, antibiotics, and bacteria it is necessary to bring together several different models that span different size and time scales. When the GI tract of a typical animal is considered, it is necessary to couple the macro-scale model of the animal's growth and feed intake with the micro-scale models representing the growth rate of the bacteria and the biological decay of the antibiotic. In addition to size, the differences in time scales must also be taken into account. The time necessary for a weaner pig to market weight is almost six months (National Research Council, 1998.) The amount of time necessary for any antibiotic ingested to be processed is measured in hours (Nolting and Derendorf, 1995), while the time necessary for a bacteria population to double is typically about 2 hours (Doyle and Roman, 1981.) In the case of bacteria modeling, it is also necessary to construct a new model borrowing from those models discussed in Section 5.4. By discretizing these models, it is possible to couple them across the necessary time scales. By using the relationships outline in Section 5.5, the coupling between size scales can be accomplished. It is then possible to perform iterations on these models to encompass the life cycle of a pig and determine the affects of the antibiotics and bacteria on its growth.

5.6.1. Equations and Differencing

The use of GBEAs for analysis of this risk mitigation problem requires that we first develop a model of the interactions of bacteria and antibiotic within an average animal's GI tract. This model takes into account not only the concentration of bacteria and antibiotic, but also the efficacy of the antibiotic and the resistance of the bacterial population. For the initial model one bacteria and one antibiotic will be taken into account, although a discussion of multiple antibiotics will be included for future consideration (Chapter 7), as later models would require multiple antibiotics to be evaluated on different bacteria types. In constructing this model, the following assumptions are made:

1. The pH and water activity of the environment remain constant throughout the process.
2. The amount of resistant bacteria on carcasses in the chiller is proportional to the amount present when the animal arrives at the abattoir.
3. For animals in which the bacteria are present, the bacterial concentration is initially at a steady state level.
4. Animal growth is proportional to animal health.
5. When multiple antibiotics are used, they have the same decay rate and they combine according to Loewe additivity (Lipstich and Levin, 1997) (models without this assumption would be beyond the scope of this work.)
6. As the concentration of bacteria increases above a threshold level, the probability of human illness being caused by that bacteria increases.

With these assumptions, it is possible to construct a differential equation that describes the growth rate for the bacteria of interest that would be applicable to evaluating human health risks. A bacterial concentration for both susceptible and resistant bacteria needs to be established for use in determining the affect on human health. Using the assumed initial bacteria concentration, it is possible to determine the growth dynamics of susceptible and resistant bacteria when exposed to an antibiotic:

$$\frac{dN_s}{dt} = K_{Gs} N_s(0)\left[1 - \frac{\sum_{i=1}^{n} N_i(t)}{N_{\max}}\right] - K_{ks} N_s(0)\frac{C(t)}{C(t) + C_{50}} \tag{5-8}$$

$$\frac{dN_r}{dt} = K_{Gr} N_r(0)\left[1 - \frac{\sum_{i=1}^{n} N_i(t)}{N_{\max}}\right] - K_{kr} N_r(0)\frac{C(t)}{C(t) + C_{50}} \tag{5-9}$$

$$\frac{dC}{dt} = -K_{Abx} C(0) \tag{5-10}$$

Where: N is the concentration of bacteria

s is susceptible bacteria

r is resistant bacteria

n is the total number of bacteria types

i is the designation of which bacteria is being examined

N_{max} is the maximum supportable amount of bacteria

K_G is the growth rate of bacteria

K_k is the kill rate of bacteria for the antibiotic

K_{abx} is the decay rate for the antibiotic

C_S is the antibiotic concentration

C_{50} is the concentration for half the maximum killing of bacteria

These equations can be discretized to give:

$$N_{s,j+1} = e^{K_{Gs}} N_{s,j}\left[1 - \frac{\sum_{i=1}^{n} N_{i,j}}{N_{\max}}\right] - e^{K_{ks}} N_{s,j} \frac{C_j}{C_j + C_{50}} \tag{5-11}$$

$$N_{r,j+1} = e^{K_{Gr}} N_{s,j}\left[1 - \frac{\sum_{i=1}^{n} N_{i,j}}{N_{\max}}\right] - e^{K_{kr}} N_{r,j} \frac{C_j}{C_j + C_{50}} \tag{5-12}$$

$$C_{j+1} = e^{-K_{Abx}} C_j \tag{5-13}$$

where j designates the current time step. Discretization of these equations allows for the solving of the final concentrations of bacteria. It also provides a means for which the antibiotic concentration can be increased, simulating either the use of antibiotic in feed or

the administration of therapeutic treatments. This model can be used for any combination of bacteria and one antibiotic by inserting an additional equation with the appropriate growth rate for each bacteria type being modeled and then using the sum of all bacteria concentrations to determine the amount present.

5.6.2. Programming

To investigate the antibiotic regimen problem the model must first be written as a computer program to be used as a fitness function for the GBEA. This was done by iterating through the discretized models for a representative number of time steps while incrementing the weight of the animal dependent on the concentration of antibiotic present in the GI tract. The details and methodology of coding the model are as follows.

To begin the evaluation of the regimen the controlling variables are first initialized. The weight of the animal is initialized to thirty pounds to represent an average weight of a weaner pig arriving at the finishing facility. The concentration of antibiotic is started at a value that corresponds to the amount of antibiotic it would receive while nursing and during weaning (Cromwell, 2001; Dritz, et al., 2002.) The concentration of bacteria is initialized to an amount found to exist in animals at delivery (Corns, Hurd, Ashlock, and Bryden, 2006.) The minimum inhibitory concentration (MIC) of the antibiotic for each bacteria type is initialized to values found in literature (Singer, et al., 2004). These bacteria concentrations and the MIC values for the antibiotic are then allowed to vary for each bacteria group, using a triangular distribution. These distributions are centered on

initial values with endpoints at +/-10% of the corresponding value in the case of the bacteria concentrations and +/-2.5% of the value for the MIC values. Using these triangular distributions makes it possible to account for the inherent variability of the bacteria concentrations found in an animal and their susceptibility to antibiotics. With these values set the model gives a representation of both the animal and its GI tract as it enters the production facility.

The model is now ready to be used to simulate the growth of the animal to market weight by iterating through the equations given in Section 5.6.1. A time step of one hour was selected to capture the changes in both the antibiotic and bacteria concentrations. For each one hour interval new values are calculated based on the previous time step values. First, the average amount of feed and water ingested by the animal is calculated based on the animal's current weight using Eq. 5-6. These values are then used to find the amount of antibiotic taken in by the animal. The antibiotic content of the feed and the water are determined for the antibiotic regimen being evaluated at the current time step and the total antibiotic intake is calculated. The volume of the animal's GI tract is then found, also based on the animal's weight, and used to determine the antibiotic concentration for the current time step. Using Eq. 5-13 the decrease in antibiotic concentration is found and subtracted from the concentration of the previous time step. Adding the antibiotic intake concentration to this gives the total concentration for the current time stage of the iteration.

The next step is to find the new bacteria concentrations. The total amount of bacteria present in the GI tract is calculated and compared to the carrying capacity of the GI tract to find the limitations for the growth calculations. The change in bacteria concentration is then calculated using Eqs. 5-11 and 5-12 for all of the bacteria types being considered. Any changes between resistant and susceptible bacteria are then determined based on the current antibiotic concentration. This accounts for the offspring of susceptible bacteria expressing resistance to the antibiotic in question.

The final step of the iteration is to determine the weight gain of the animal. The values from Table 9 are used to determine the amount of weight gained in that hour. If the current antibiotic concentration is above a threshold value, the "antibiotics present" column value is added to the weight. Otherwise, the "no antibiotics present" column value is added to the current weight.

When the iterative process is complete, the fitness value for the regimen is found. This is done by first calculating the total amount of antibiotic added to the feed and water. A penalty function is applied if the amount of antibiotic administered is significantly higher than the standard use. For the model used in this study, this limit was set to correspond to label usage for growth promotion with fourteen days of use to prevent disease. The fitness is then calculated using the final weight of the animal, modifying this by the percent of bacteria that are resistant to the antibiotic and then applying the penalty function, if necessary.

To take into account the uncertainty associated with the initial bacteria levels and the minimum inhibitory concentration (MIC) of the antibiotic, a Monte Carlo simulation is used for each fitness evaluation. Each Monte Carlo simulation consists of 500 trials of the model. For each trial, different samplings from the triangular distributions for the initial amount of each bacteria strain and the MIC values for the antibiotic are used. Mean values for the fitness are calculated using the mean animal weight and percentage of bacteria that are resistant to the bacteria. The standard deviation of the fitness value is also calculated to make it possible to calculate a 95% confidence interval on the results.

5.6.3. Optimization

Using the previously described model, eight different graphs have been used to find a near optimal antibiotic regimen to minimize both bacterial resistance to antibiotics and the weight gain of the animal. For this set of experiments, tylosin phosphate was examined as the performance enhancer, with two strings hold information on the amount of antibiotic to be added to either the feed or the water. The first string contains information on the amount of antibiotic added to the feed ranging from 0 to 100 grams per ton in 10 gram increments. Each value represents one week of feed, for a total of 24 values (168 days). The second string determines how much antibiotic is added to the water for disease prevention, ranging from 0, 125, or 250 milligrams per gallon. This string is of length 84, representing two-day periods that the antibiotic may be added to the water. When the GBEA is initialized, the amount of antibiotic added to the feed is allowed to vary within allowable label usage, while the amount added to the water is

initialized to zero. These two strings are the chromosomes being manipulated by the GBEA, and are also the input for the overall model.

A population size of 128 solutions was used with single point crossover and one mutation on each string per mating event. The mutation operator selects a string value at random and changes it to a new value selected at random from the range of possible values. The fitness of the individual solutions is determined based on the final weight of the animal, the percentage of bacteria that are resistant to the antibiotic and to a lesser extent the concentration of both susceptible and resistant bacteria at the end of the iterative process. Eight different graphs were used in this study, with five replicates for each graph type conducted. For each of these experiments, 500 mating events were conducted and the best result from each was recorded.

5.6.4. Results

This goal of this research is not only to explore the utility of GBEAs, but also to find an antibiotic regimen that will minimize human health risk by managing the concentration and resistance level of the bacteria present in the animal as it is delivered to the abattoir. It is also necessary to maintain acceptable animal health and growth to make the optimized regimen financially attractive to the producers. To evaluate the results of this experiment, it is first necessary to compare the two indicators of these objectives.

The results of the experiment indicate that the two objectives are in competition with each other. The use of antibiotics promotes growth, but also increases bacterial resistance to that antibiotic in the model as it is currently implemented. The decrease in human risk is represented by the decrease in the percentage of bacteria that are resistant to the antibiotic. The economic benefit to the pork industry is represented by the animal's weight at the end of the simulation. A lower animal weight does not mean that the animal will be delivered weighing less, but instead that the amount of time necessary for the pig to reach market weight is increased, with an associated increase in the cost to feed and house the animal.

Initially, the graphs selected were to determine which would have the best performance when applied to this problem. The eight graphs used in this experiment are a winnowed set derived from those found earlier in this study (Section 4.3, Table 8.) However, instead of one graph performing optimally on this problem, a variety of solutions were found that were different compromises between higher animal weight and a lower percentage of resistant bacteria. The general trend was that diversity preserving graphs yielded higher animal weights while graphs with a high rate of information flow improved the percentage of antibiotic resistant bacteria. The range of solutions found by the graphs made it possible to construct a Pareto-optimal frontier, allowing for a direct comparison of the tradeoffs between the two objectives. The results of the forty trials were compared to determine if any of the solutions were dominated in both weight and percent of bacteria resistant. Because the values of the objective functions were real valued, the results were rounded to a tenth of a pound for the weight and a tenth of a percent for the

percent of resistant bacteria. If the solution found had a lower final animal weight that other solutions, the percentage of bacteria resistant to the antibiotic for that solution was compared to the higher weight solutions. If any of those higher weight solutions had a lower percentage of bacteria resistant to the antibiotic, they were considered a dominated solution and removed from consideration. After removing the dominated solutions from the population, seven of the solutions remained from those found using GBEAs.

To allow for a comparison to existing methods of raising these animals, two baseline experiments were conducted. The first baseline experiment was to determine the results when the conventional use of antibiotics was applied. In this model, 40 grams of the antibiotic were added to every ton of feed. This gave a final animal weight of 276.1 pounds with 80.9% of the *Campylobacter.spp* present in the GI tract resistant to tylosin phosphate. For the second baseline experiment, no antibiotic was used (antibiotic free, or ABF), yielding a final animal weight of 256.5 pounds with only 40.2% of the *Campylobacter.spp* present in the GI tract resistant to tylosin phosphate. These results closely match published antibiotic resistance levels and animal weights for these regimens (Thakur and Gebreyes, 2005.) Using the results of the work by Thakur and Gebreyes provides validation of the model was validated against existing conditions found at animal production sites.

The results of these baseline experiments and the non-dominated regimens found by the GBEAs are given in Table 10, along with the results from the best-of-five standard

Table 10, Delivered Weight and Percent of Resistant Bacteria by Graph or Method.

Graph or Method	Delivered Animal Weight (lbs.)	Percent of Bacteria Resistant
ABF Method	256.5	40.2%
P64_7 Graph	273.4	43.7%
P64_7 Graph	273.8	46.3%
T4_32 Graph	273.9	46.9%
P64_17 Graph	274.6	49.8%
H7 Graph	274. 7	57.8%
C512 Graph	274.8	59.7%
K128 Graph	274.7	67.8%
P64_7 Graph	275.4	72.5%
Conventional Method	276.1	80.9%

evolutionary algorithm. These results were used to construct the Pareto optimal frontier (Fig. 33) making it possible to compare the benefits of the regimens found.

The results that make up the Pareto front in Figure 33 give a wide selection of solutions to the use of antibiotics in swine production. A closer examination of the solutions shows that the regimen found by the Petersen graph (k=7, Fig. 34) is a distinctly different solution to the problem than that found by a standard evolutionary algorithm (complete graph, Fig. 35.) While the standard evolutionary algorithm made changes to the conventional method of antibiotic use, the GBEA found a regimen that more strongly resembles pulse administration of medication. This is noteworthy because this is an emergent behavior in the algorithm; no information was provided a priori to promote this solution. This pulse administration of medication is similar to that found in chemotherapy treatments used for cancer patients. While this type of therapy has been used in the pork industry before (Dewey, Cox, Straw, Bush, and Hurd, 1999), it is typically reserved for the treatment of illness. While careful consideration would need to be taken to fully assess the impact on micro flora in the animal's GI tract, this could prove to be a novel technique to minimize the antibiotic resistance in bacteria that could lead to foodborne illness.

5.7. Conclusions

The use of graph based evolutionary algorithms for the antibiotic regimen problem has been shown to give superior results to those found using a standard evolutionary algorithm. More importantly, the use of diversity preserving graphs yields novel solutions

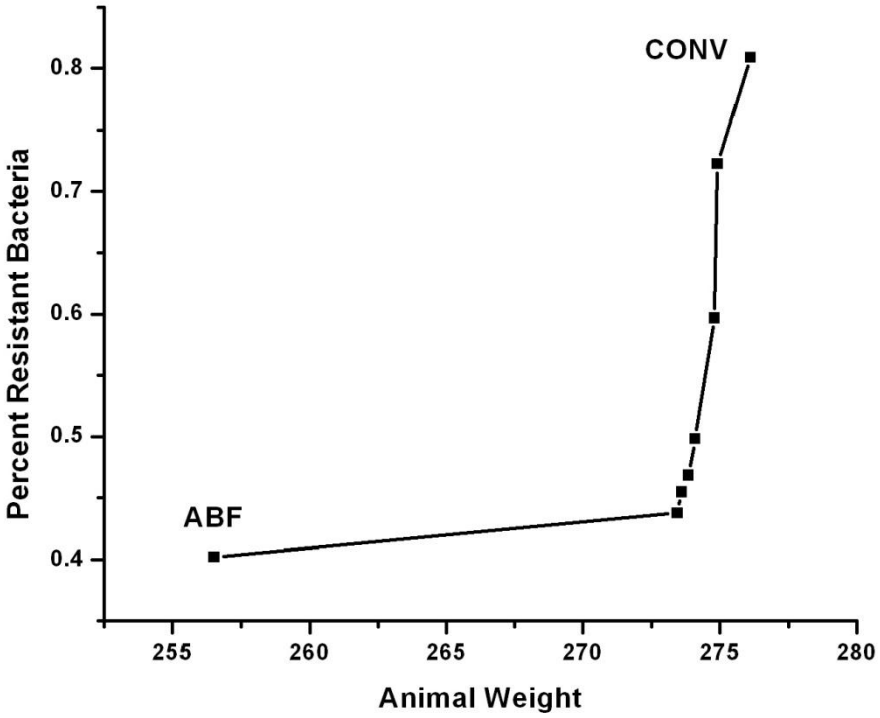

Figure 33, Percent of Campylobacter.spp resistant to antibiotic vs. Animal Weight at Abattoir.

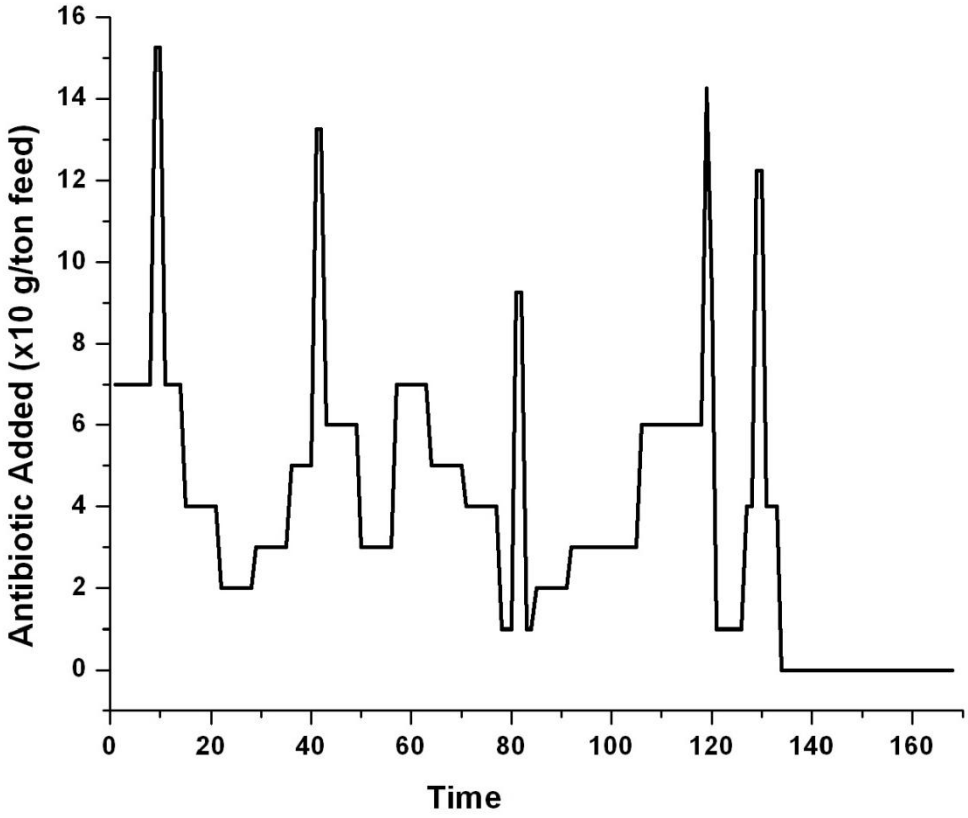

Figure 34, Antibiotic regimen found by Petersen graph (k=7) for antibiotic regimen problem.

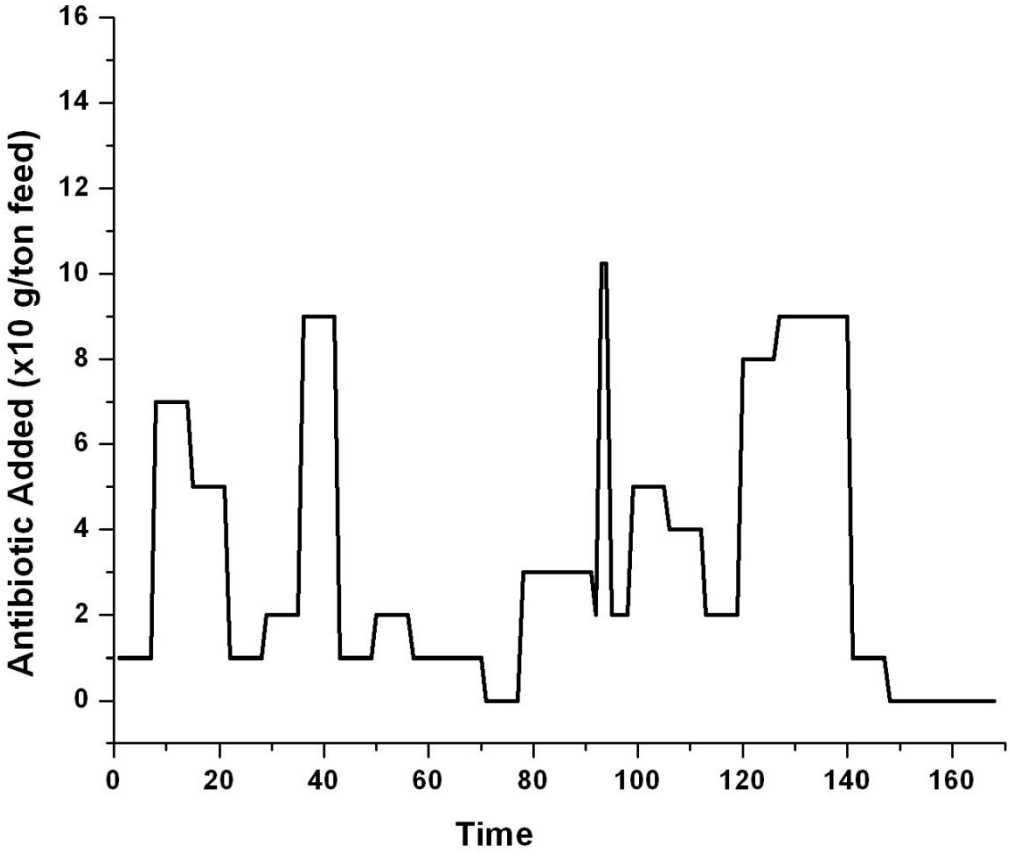

Figure 35, Antibiotic regimen found by standard evolutionary algorithm (complete graph) for antibiotic regimen problem use.

to the antibiotic regimen problem. When the results of this experiment are compared to the findings of Sections 3 and 4, it is possible to theorize what is occurring in the evolving population in the various graphs. The highly connected graphs quickly find a method to decrease the amount of resistance displayed by the bacteria, but in doing so increase the amount of antibiotic used to a level where any additional antibiotics could quickly incur a fitness penalty. This is due to the rapid sharing of information increasing the average fitness, removing solutions that have very low antibiotic use and corresponding low animal weight. For the graphs with a modest amount of diversity preservation, the slower information flow allows the regimens with lower antibiotic use to remain in the population longer, giving sufficient time for the algorithm to develop solutions taking advantage of the addition of antibiotic to the water supply. Normally reserved for disease treatment or prevention, the addition of antibiotic to the water supply is the most efficient method for administering pulse doses of antibiotic. The addition of antibiotics through the water supply is modeled using a 2 day period rather than the 7 day period for antibiotic addition to feed, allowing for a more precise addition of antibiotic. This behavior is also displayed by the sparse graphs, although with the slow transfer of information the population members are less able to take advantage of the information related to pulses found elsewhere in the population. Further examination of the problem would make it possible to test this theory, although the computational resources necessary would be significant.

The use of graph based evolutionary algorithms has provided novel solutions to be address potential health risks from antibiotic use while largely maintaining the economic

benefits gained. These results provide nearly the same benefits realized by the conventional use of antibiotics as performance enhancers while increasing the percentage of resistant bacteria only slightly compared to antibiotic free facilities. This compromise would make it possible to lower the potential risk to human health at little to no cost to the producer. With proper validation, this would provide a "win-win" situation for the pork industry, environmental groups and consumers.

6. CONCLUSIONS

The use of population structures to control the flow of information in an evolving population of solutions has been demonstrated. While this work has focused on the use of GBEAs to control the rate of diversity loss, it should be noted that GBEAs are just one tool for controlling information flow and diversity loss. By placing restrictions on which population members may be selected during mating events, barriers are placed against the flow of information. This makes it possible to control the rate at which diversity is lost in a population of evolving solutions, allowing building blocks and complete solutions time to mature. Depending on the amount of diversity necessary, this can lead to a faster convergence time or to the discovery of multiple satisfactory solutions. The amount of diversity preserved is indirectly related to the flow of information in the evolving population. Applying these various graphs against a test suite of problems has made it possible to evaluate the graphs and determine which are best suited for the evaluation of evolutionary computation problems. The evaluation of the graph sets was done in tandem with the evaluation of evolutionary computation problems. In this way, representative graph sets and test suites were developed, increasing the utility of GBEAs and making it possible to construct an unbiased test suite of problems.

Controlling the flow of information in evolutionary algorithms can improve the time to convergence to an optimal solution. While the benefit varies by problem, all but the one-max problem had some amount of speed up when compared to a standard evolutionary algorithm. The PORS15 problem, a problem with a deceptive fitness function, had nearly a 24x decrease in the number of mating events to completion with a population size of

256, the largest improvement in time to convergence for any of the problems examined. This increase in performance is a result of several barriers to the flow of information preventing the destruction of necessary building blocks by superior yet sub-optimal solutions.

The results of Figure 14 and the shift of preferred graph in the population size study highlight a difference between the amount of diversity initially present in an evolutionary algorithm and the preservation of diversity. The use of a diversity preserving tool shifts the required population size to the left of Figure 14, decreasing the population size necessary for best algorithm performance. This is due to a tradeoff between initial diversity in the population of solutions and the preservation of the diversity present as evolution proceeds. After a sufficient population size is reached to provide the necessary information for an optimal solution, preserving diversity can be used as a surrogate to a larger population size. The mechanism by which diversity preservation does this is by maintaining copies of the necessary information to keep it available rather than providing a large number of copies. This allows for a smaller population size to be used when investigating problems, decreasing the number of fitness calls and thereby decreasing the amount of time necessary to find a satisfactory solution. This is especially beneficial when applied to problems with expensive fitness calls, such as implementations using computational fluid dynamics solvers (Bryden, et al., 2002).

The need for and preferred type of diversity can be used as a means to classify evolutionary computation problems. For problems with simple, non-deceptive fitness

landscapes there is little need for diversity. As problem landscaped become more multi-modal and deceptive, diversity and diversity preservation become more important. In addition, as the number of variables and the size of the available alphabet increases the use of diversity preserving tools becomes more important. By developing a taxonomy of these problems, it is possible to apply a priori knowledge of a new problem to determine a population size and diversity preservation scheme.

This taxonomy of evolutionary computation problems also serves as a tool for the development of an unbiased test suite of problems. Just as graph preference can be used as a characteristic to classify problems, using a collection of problems that show a wide range of preferred graph would allow for an unbiased evaluation of any new graphs. This test suite would also provide a diverse sample of evolutionary computation problems that could be used to evaluate other proposed algorithms, giving researchers the ability to weigh the utility of different methods to make an informed decision as to which to include in their work.

Controlling the flow of information makes it possible to find several acceptable solutions to an evolutionary computation problem. Diversity preservation allows superior solutions enough time to develop a foothold in a local graph neighborhood. After these solutions have had an opportunity to mature, they begin to interact at the active edges in the population structure. If they are similar, the solutions will gradually merge together, leaving only those solutions that have distinct differences. A larger number of barriers to the flow of information gives more opportunities for the development of these local

solutions, although there is an added computational cost for these extra solutions and so more run time is necessary when compared to achieving a single solution.

By providing a variety of solutions for the design process these methods provide a benefit to engineering design by adding a computationally inexpensive method to increase the flexibility of the design process. This benefit is even more pronounced when using a system of systems approach (Fig. 36.) When bringing together multiple components to construct an assembly or system, an optimal component design may not lead to an optimal system design. Careful design of the models used to investigate the system can overcome these issues, but as more and more components are added to the system this quickly becomes unmanageable. Optimizing both components and the entire system using evolutionary computation methods and proper diversity management can automatically negotiate the necessary tradeoffs for this optimization. Compare this problem to a deceptive problem, where optimal component level design is not the optimal component design for the system. In the same manner in which necessary building blocks are preserved in a deceptive problem, these component designs necessary to the overall system optimization are preserved until they can be utilized in the global optimization process.

A result of the antibiotic regimen experiment was the emergence of a novel solution to the problem that resembles one currently implemented in medical science. While the

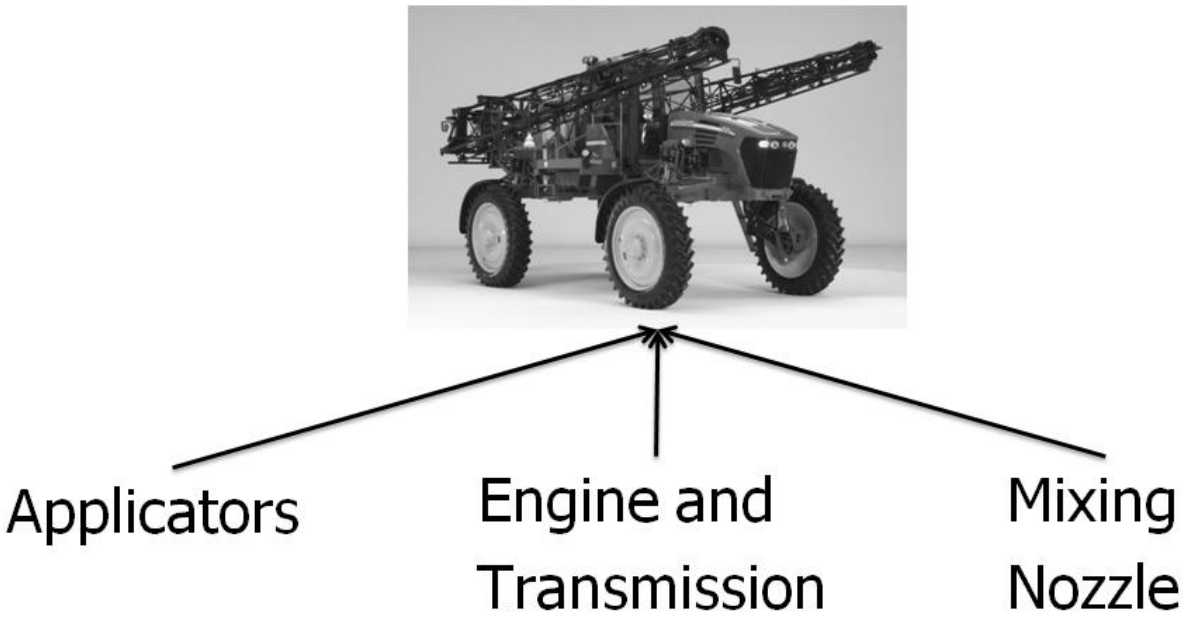

Figure 36, Sytem of systems concept for engineering optimization (John Deere Company, 2008)

graph set used was the larger recommended set of graphs, the guidelines for graph selection recommended the use of a graph that has a modest amount of diversity preservation. This is due mainly to the large number of variables used in the representation of the problem. When the results were analyzed this recommendation was confirmed, with the Petersen graph giving the best results, and also finding a novel pulsed method for providing the antibiotics to the animals.

This pulse administration of antibiotics found by the GBEA is an example of how these techniques can be used by engineers as a tool to inspire creativity, similar to the concept of computational creativity (Saunders, 2002.) By controlling the flow of information within the evolving population of solutions, several novel solutions can be developed. When used in the design process, this tool can suggest a variety of novel solutions for the user to consider for the problem at hand, allowing human input to include design requirements that cannot be included in computer code.

7. FUTURE RESEARCH

There are four distinct areas where this work can be extended. First, the classification of evolutionary computation problems and the continuation of the taxonomy and test suite work, which will continue to provide information on the similarities and differences of differently classes of problems. This information would be applied to two other areas of future work, the development of novel designs that encourage creativity and the optimization of multilevel models of systems. Finally, there are several research avenues that the antibiotic regimen problem could be used to pursue, both in improvements to the current model and expansion to other areas of investigation.

The development of a taxonomy of evolutionary computation problems and the use of this information for test suite development will be a work in progress for several years. Sufficient information for preliminary recommendations exists, but more problems need to be examined to develop a more useful taxonomy of problems. While a complete taxonomy of problems is unlikely to be achieved, continued contributions to this work would increase the database of problems and allow researchers to find problems more similar to the one they would like to examine. Tied to this taxonomy work is the development of a test suite of evolutionary computation problems. As more information is gathered on the nature of the problems being approached with these techniques, a more representative and less biased test suite can be produced, giving better insight into new techniques.

The use of diversity preserving tools to find multiple solutions could also be used to promote creativity, such as computational creativity in art (Romero and Machado, 2008.) One example would the use of GBEAs to investigate an image segregation problem (Karthikeyan, Bryden and Ashlock, 2005). In this study, a gene was constructed that held generator points and numerical weights. These were used to construct panes of the image using Voronoi tessellation with the color of the pane determined by the color of the pixel specified by the generator points. While the initial intent of this research was to find a novel method for image compression, it was found that this method created a stained glass effect by segmenting different colored tiles. By changing the numerical weights and generator points, different effects could be generated to form an artistic expression. Similar applications could be applied in both visual and audio artistic endeavors.

The use of these tools to control the flow of information in engineering optimization would be a benefit to design on a systems level. Utilizing more flexible frameworks for engineering design allows professionals to bring together larger sets of data and analyze more and more components and systems. To optimize these larger simulations, it is necessary to provide a tool that allows for a larger degree of design flexibility. This can be achieved by managing the amount of diversity present in and evolutionary algorithms. By combining simulations using different graph based evolutionary algorithms, computational resources can be assigned to components which are key to the design or are known to have several optimal configurations. By selecting an appropriate solution presented by the graph based evolutionary algorithm, novel designs can be developed that

are superior to those that already exist or that can provide solutions to the problems of the future.

There are also several opportunities to improve the modeling of the interactions between bacteria, antibiotics and swine. The model proposed in Chapter 5 is adequate for the evaluation of graph based evolutionary algorithms, but there are several areas where it can be expanded to encompass the issues of antibiotic use in pork production. A variety of solutions to the problem of finding a superior antibiotic regimen have been found that satisfy the limits placed on the model, but there are several more factors that should be considered. More bacteria, such as *Salmonella.spp* should be added to the model to better represent the risk to human health. This could be done by adding another bacteria term to model, with the associated growth and mortality coefficients and MIC values for the antibiotic being evaluated.

It would also be beneficial to consider more that on type of antibiotic, as it is not uncommon for one type of antibiotic to be used as a performance enhancer while a different antibiotic would be used for disease control or prevention. A slight adjustment to the second term of equations 5-8 and 5-9 and to equation 5-10 allows for modeling more than one antibiotic treatment:

$$\frac{dN_s}{dt} = K_{Gs} N_s(0)\left[1 - \frac{N_s(t) + N_{ra}(t) + N_{rb}(t)}{N_{\max}}\right] - K_{ksa}\frac{C_a(t)/C_{50a} + C_b(t)/C_{50b}}{1 + C_a(t)/C_{50a} + C_b(t)/C_{50b}} \tag{7-1}$$

$$\frac{dN_{ra}}{dt} = K_{Gra} N_{ra}(0)\left[1 - \frac{N_s(t) + N_{ra}(t) + N_{rb}(t)}{N_{\max}}\right] - K_{krb}\frac{C_b(t)}{C_b(t) + C_{50b}} \tag{7-2}$$

$$\frac{dN_{rb}}{dt} = K_{Grb} N_{rb}(0)\left[1 - \frac{N_s(t) + N_{ra}(t) + N_{rb}(t)}{N_{\max}}\right] - K_{kra}\frac{C_a(t)}{C_a(t) + C_{50a}} \tag{7-3}$$

$$\frac{dC}{dt} = -K_{Abx} C_a(0) \tag{7-4}$$

$$\frac{dC}{dt} = -K_{Abx} C_b(0) \tag{7-5}$$

Where the ‘a’ and ‘b’ subscripts represent different antibiotics. This can be repeated multiple times for the administration of additional antibiotics to the regimen. It should be noted that there is a significant chance of drug interactions that should be taken into consideration when using multiple antibiotics. This can also be combined with expansions to the model to account for multiple types of bacteria that are a potential human health risk. It should be noted, however, that many of the drug interactions that would be encountered in this work are unknown and could require years of medical research to discover, and several more years to adequately model.

While this model addresses the concentrations of bacteria and antibiotic in the animal as delivered to the abattoir, there is also a potential risk that the bacteria and antibiotic may be released to the environment while the animal is being grown (Qiang, Macauley, Mormile, Surampalli, and Adams, 2006.) This could be included in the model by the introduction of terms that track these concentrations during the growth period of the animal. The addition of these considerations would increase the complexity of the model as well as necessitating a good deal of research and evaluation to determine how best to incorporate the impact of the antibiotic and resistant bacteria on the surrounding environment. Some of these considerations would be the size of hog lagoons and the effectiveness of oxidation methods to break down the antibiotics found in the effluent streams.

With these additions in place, this model could also be implemented as a tool for use in the finishing facilities. If the amount of antibiotic administered to the animals were varied either by a necessary intervention or human error, this tool would allow the operator to determine a new regimen that would meet the same goals while taking into account the new antibiotic levels.

APPENDIX: GRAPH THEORY OVERVIEW

A *combinatorial graph* or *graph* (G), is a collection of vertices (V(G)) and edges (E(G)) where E(G) is a set of unordered pairs from V(G). Two vertices of the graph are *neighbors* if they are members of the same edge. The *degree* of the vertex is the number of edges containing that vertex. If all vertices in a graph have the same degree, the graph is said to be *regular*, and if the common degree of a regular graph is k, then the graph is said to be *k-regular*. If you can go from any vertex to any other vertex traveling along vertices and edges of the graph, the graph is *connected*. The *diameter* of a graph is the longest that the most direct path between any two of the vertices can be, or in other words, the shortest path across the graph. A graph used to constrain mating in a population can be called the *population structure*. The general strategy for graph based evolutionary algorithms is to use the graph to specify the geography on which a population lives, permitting mating only between neighbors, and finding graphs that preserve diversity without hindering progress due to heterogeneous crossover. Additional information on combinatorial graphs can be found in (West, 1996.)

List of Graphs

In this section, the graphs used in this study are defined, as well as those necessary to properly describe those used.

Definition 1 *The* complete graph *on n vertices, denoted* K_n*, has n vertices and all possible edges.*

Definition 2 *The n-*cycle*, denoted C_n, has vertex set Z_n. Edges are pairs of vertices that differ by 1 (mod n) so that the vertices form a ring with each vertex having two neighbors A C_{32} graph is shown in Fig 1(a).*

Definition 3 *The n-*hypercube*, denoted H_n, has the set of all n character binary strings as its set of vertices. Edges consist of pairs of strings that differ in exactly one position. A 5-hypercube is shown in Fig 1(d).*

Definition 4 *The n* x *m-*torus*, denoted $T_{n,m}$, has vertex set Z_nxZ_m. Edges are pairs of vertices that differ either by 1 (mod n) in their first coordinate or by 1 (mod m) in their second coordinate, but not both. These graphs are n* x *m grids that wrap (as tori) at the edges. An 8* x *4-torus is shown in Fig 1(c).*

Definition 5 *The generalized* Petersen graph *with parameters n, k, denoted* $P_{n,k}$*, has vertex set 0,1,…,2n-1. The two sets of vertices are both considered to be copies of Z_n. The first n vertices are connected in a standard n-cycle. The second n vertices are connected in a cycle-like fashion, but the connections jump in steps of size k (mod n). The graph also has edges joining corresponding members of the two copies of Z_n The graph $P_{32,5}$ is shown in Fig 1(b).*

Four classes of random graphs were added to the graph set in hopes that more insight into the usefulness of the technique. The first three graphs are generated using edge moves

(Ashlock, Walker, and Smucker, 1999) in a randomized algorithm that corresponds to a type of random graph (a probability distribution on some set of graphs).

Definition 6*An*edge move *is performed as follows. Two edges {a,b} and {c,d} are found that have the property that none of {a,c}, {a,d}, {b,c}, or {c,d} are themselves edges. The edges {a,b} and {c,d} are deleted from the graph, and the edges {a,c} and {b,d} are added. Notice that edge moves preserve the regularity of a graph if it is regular.*

Random Graphs

The last of the random graphs was generated by randomly placing vertices on a unit torus (a unit square that is wrapped at the edges.) In order to place a control on the degree of the graph, this distance was varied with the population size (Table 11.) Three different instances of each graph class were produced for use in this research.

The first three random regular graphs were generated using the following algorithm. Starting with a regular graph, 3000 edge moves are performed on vertices selected uniformly at random from those that are valid edge moves. Initially, the random graphs were labeled according to the degree of the graph, but since the degree of the graphs may change when the number of vertices is changed, these numbers are now merely labels, only necessarily showing the degree of the graphs for population size of 512. For 3-regular graphs, the Petersen size one graph was the starting point. For 4-regular graphs, the starting point was $T_{n,m}$ graph with the largest radius for that population size (ie $T_{4,8}$

Table 11, Separation distances for random tori generation.

Population size	Separation
32	0.35
64	0.30
128	0.20
256	0.15
512	0.07
1024	0.05
2048	0.03
4096	0.02

for 32 vertices, $T_{8,m}$ for 64 and 128 vertices, and $T_{16,m}$ for 256 vertices and above), and the 9-regular graph was started with a hypercube graph. These graphs are denoted R (n,k,i) in this study, with n being the number of vertices, k being the degree for population size 512 (as described above), and i is the instance of the graph.

For the final set of three random graphs, a number of points equal to the population size were placed on a unit torus. Edges were created with these points if they were within a certain distance from each other, varying for each population size, as outlined in Table 11. These values were selected to try to maintain a roughly equal degree of graph for each population size. After generation, the graph was checked to see if it was connected, and rejected if the test failed. These graphs are denoted RT(r,i), where r is the maximum separation from another point where an edge would still be created, and i is the instance of the graph.

REFERENCES

Abrosov, N.S. and Kovroc, B.G. (1977) *Analysis of the Structure of Community of Unicelluar Organisms*, Nauka Publisher, Novosibirsk.

Ackley, D. L. and Littman, M. L. (1992) "A Case for Distributed Lamarckian Evolution," Working Paper, Cognitive Science Research Group. Bellcore, New Jersey.

Altekruse, S.F., Cohen, M.L. and Swerdlow, D.L. (1997) "Emerging Foodborne Diseases," *Emerging Infectious Diseases*, Vol. 3, No. 3.

Altenberg, L. (1995) "The Schema Theorem and Price's Theorem," *Foundations of Genetic Algorithms 3*, edited by Whitley, L. D. and Vose, M. D., Morgan Kaufmann Publishers, San Francisco, CA, pp. 23-49.

Arabas, J., Michalewicz, Z., and Mulawka, J. (1994) "GAVaPS – a Genetic Algorithm with Varying Population Size," *Proceedings of the First IEEE Conference on Evolutionary Computation*, 73-78. Kauffman.

Ashlock, D. A. (2006) *Evolutionary Computation for Modeling and Optimization*, Springer Science+Business Meida, Inc., New York, NY.

Ashlock, D.A.,Bryden, K.M. and Corns, S.M. (2005) "Graph Based Evolutionary Algorithms Enhance the Location of Steiner Systems." *Proceedings of the 2005 Congress on Evolutionary Computation*, pp. 1861-1866. IEEE Press.

Ashlock, D.A.,Bryden, K.M., Corns, S.M. and Schonfeld J. (2006), "An Updated Taxonomy of Evolutionary Computation Problems Using Graph-Based Evolutionary Algorithms," *Proceedings of the 2006 IEEE World Congress on Computational Intelligence,* pp. 403-410. IEEE Press.

Ashlock, D., Guo, L. and Qiu, F., (2002) "Greedy Closure Genetic Algorithms," *Proceedings of the 2002 Congress on Evolutionary Computation*, pp. 1296-1301, Piscataway, NJ. IEEE Press.

Ashlock, D. A. and Lathrop, J. I. (1998a) An Arithmetic Test Suite for Genetic Programming, ISU Applied Mathematics Report AM98-1.

Ashlock, D. A. and Lathrop, J. I. (1998b) "A fully Characterized Test Suite for Genetic Programming," *Evolutionary Programming VII*, pp. 537-546.Springer, New York.

Ashlock, D. A., Walker, J., and Smucker, M. (1999) "Graph Based Genetic Algorithms," *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1362-1368. Morgan Kaufmann, San Francisco.

Banzhaf, W. G., Nordin, P., Keller, R. E., and Francone, F. D. (1998) *Genetic Programming: An Introduction*. Morgan Kaufmann, San Francisco.

Bartholomew, M.J.,Vose, D.J., Tollefson, L.R., and Travis, C.C. (2005) "A Linear Model for Managing the Risk of Antimicrobial Resistance Originating in Food Animals," *Risk Analysis*, Vol. 25, No. 1, pp. 99-108.

Belew, R. (1992) "Paradigmatic over-fitting," *GA-Digest*, 6(18).

Blaize, M., Knight, D. and Rasheed, K. (1998) "Automated Optimal Design of Two-Dimensional Supersonic Missile Inlets," *Journal of Propulsion Power* 14, pp. 890-898.

Braude, R., Clarke, P.M., Mithcell, K.G., Cray, A.S., Franke, A. and Sedgwick, P.H. (1957) "Unrestricted Whey for Fattening Pigs," Journal of Agricultural Science, Vol. 49: pp. 347

Bryden, K. M., Ashlock, D. A., Corns, S. M., and Willson, S.J. (2006) "Graph Based Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computations*, Vol. 10:5, pp. 550-567.

Bryden, K. M., Ashlock, D. A., McCorkle, D. S., and Urban, G. L. (2002) "Optimization of Heat Transfer Utilizing Graph Based Evolutionary Algorithms," *International Journal of Heat and Fluid flow*, 24(2), pp. 267-277.

Canibe, N. and Jensen, B.B. (2003) "Fermented and Nonfermented Liquid Feed to Growing Pigs: Effect on Aspects of Gastrointestinal Ecology and Growth Performance," *Journal of Animal Science*, Vol. 81, pp. 2019-2031.

Corns, S. M., Hurd, H. S., Ashlock, D. A., and Bryden, K. M. (2006) "Evolutionary Optimization of an Antibiotic Feed Regimen Applied to Multiple Bacteria," in *Intelligent Engineering Systems through Artificial Neural Networks*, edited by C. H. Dagli et al., ASME Press , vol. 16: pp. 255-260.

Cox, L.A. (2004) "Potential Human Health Benefits of Antibiotics used in Food Animals: A Case Study of Virginiamycin," *Environment International*, Vol. 31, pp.549-563.

Cox, L.A., Copeland, D., and Vaughn, M. (2005) "Ciprofloxacin Resistance Does Not Affect Duration of Domestically Acquired Campylobacteriosis," *Journal of Infectious Diseases*, 191, pp. 1565-1566.

Cromwell, G.L. (2001) "Antimicrobial and Promicrobial Agents," in *Swine Nutrition, 2nd Ed.* Edited by Lewis, A.J. and Southern, L.L., CRC Press LLC.

Cromwell, G. L. and Dawson, K. A. (1992) "Antibiotic Growth Promotants," In *Emerging Agricultural Technology: Issues for the 1990's*, Office of Technology Assessment, U.S. Congress, Washington, D.C.

Davidor, Y., Yamada, T., and Nakano, R. (1993) "The ECOlogical framework II: Improving GA Performance at Virtually Zero Cost," *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 171-175. Morgan Kaufman.

John Deere Company (2008) Image from www.leetractor.com/newequip.htm, last accessed April 14, 2008.

DeJong, K. A. (1975) *An analysis of the behavior of a class of genetic adaptive systems*, (Doctoral dissertation, University of Michigan, 1975). Dissertation Abstracts International, 36(10), 5140B.

Dejong, K. A. and Spears, W. M. (1990) "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms," *Parallel Problem Solving from Nature: 1st Workshop (PPSN 1)*, v496, edited by H-P. Schwefel and R. Manner, pp. 38-47. Springer-Verlag.

Dewey, C. E., Cox, B. D., Straw, B. E., Bush, E. J., and Hurd, H. S. (1999) "Use of Antimicrobials in Swine Feeds in the United States," *Swine Health and Production*, Vol. 7, No. 1, pp. 19-25.

Donachie, W. D. and Robinson, A. C. (1996) "Cell Division: Parameter Values and the Process," In *Excherichia coli and Salmonella typhimurium: Cellular and Molecular Biology*, 2nd ed., Edited by F. C Neidhardt and others, pp. 1578-1593, American Society for Microbiology, Washington, D.C.

Doyle, M.P. and Roman, D.J. (1981) Growth and survival of *Campylobacter fetus*susp.*jejuni* as a function of temperature and pH. J. Food Protect. 44(8), pp. 596-601.

Dritz, S.S.,Tokach, M.D., Goodband, R.D., and Nelssen, J.L. (2002) "Effects of Administration of Antimicrobials in Feed on Growth Rate and Feed Efficiency of Pigs in Multisite Production Systems," *JAVMA*, Vol. 220, No. 11, pp. 1690-1695.

Engelbrecht, J. J. (2007) "Optimization of a hydraulic mixing nozzle," (Master's thesis, Iowa State University, 2007).

Fabbri, G. (1997) "A Genetic Algorithm for Fin Profile Optimization," *International Journal of Heat and Mass Transfer*, Vol.40, No.9, 2165-2172.

Fernandes, C. and Rosa, A. (2001) "A Study on Non-random Mating and Varying Population Size in Genetic Algorithms Using a Royal Road Function," *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol.1, pp. 60-66.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966) *Artificial Intelligence through Simulated Evolution*. Wiley, New York.
Giacobini, M., Tomassini, M., Tettamanzi, A. G. B., and Alba, E.(2005) "Selection Intensity in Cellular Evolutionary Algorithms for Regular Lattices," *IEEE Transactions on Evolutionary Computation*, Vol.9, No.5, pp. 489-505.

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

Goldberg, D. E. and Deb, K., (1991) "A Comparative Analysis of Selection Schemes Using in Genetic Algorithms," *Foundations of Genetic* Algorithms, G. J. E. Rawlins Ed., pp. 69-93. Morgan Daufmann, San Mateo, CA.

Goldberg, D. E., Deb, K., and Clark, J. H. (1992) "Genetic Algorithms, Noise and the Sizing of Populations," *Complex Systems,* Vol.6, No.4, pp. 333-362.

Goldberg, D. E., Deb, K., and Thierens, D. (1993) "Towards a Better Understanding of Mixing in Genetic Algorithms," *Journal of the Society of Instruments and Control Engineers* 32(1), pp. 10-16.

Goldberg, D. E., Sastry, K. and Latoza, T. (2002) "On the Supply of Building Blocks," *Proceedings of GECCO-2002*, pp. 328-335.

Grefenstette, J. J. (1986) "Optimization for Control Parameters for Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol.SMC-16, No. 1.

Gusfield, D. (1997) *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, MA.

Hacker, K. A., Eddy, J. and Lewis, K. E. (1992) "Efficient Global Optimization Using Hybrid Genetic Algorithms," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*.

Harik, G., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L. (1997) "The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations," *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*. IEEE press.

Haupt, R. L. and Haupt, S. E. (1998) *Practical Genetic Algorithms*. New York, John Wiley & Sons.

Haupt, R. L. and Haupt, S. E. (2000) "Optimum Populations Size and Mutation Rate for a Simple Real Genetic Algorithm that Optimizes Array Factors," *Applied Computational Electromagnetics Society Journal*, v. 15, 2, pp. 94-102.

Hays, V. W. (1977) *Effectiveness of Feed Additive Usage of Antibacterial Agents in Swine and Poultry Production*, Office of Technology Assessment, U.S. Congress, Washington, D.C. (Edited version: Hays, V.W. (1981). *The Hays Report*, Rachelle Laboratories, Inc., Long Beach, CA.)

Herbert, D. (1958) "Some Principles of Continuous Culture," *Resumes de Travaux Presentees a Sessions de Rapports: 7th International Congress International de Microbiologie, Stockholm*, pp. 381-396.

Hochhaus, G. and Derendorf, G. (1995) "Dose Optimization Based on Pharmacokinetic-pharmacodynamic Modeling,"In *Handbook of Pharmacokinetic/pharmacodynamic correlation*, Ed. H. Derendorf and G. Hochhaus, CRC Press, Inc., Boca Raton, Fl.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Hurd, H.S.,Enoe, C., Sorensen, L. Wachmann, H., Hald, T., and Greiner, M. (2004) "Risk-based Optimization of the Danish Pork *Salmonella* Program.", Project Report, International EpiLab, Copenhagen, Denmark.

Hurd, H.S. (2006) "Assessing Risks to Human Health from Antibiotic Use in Food Animals," *Microbe*, Vol.1, No. 3, pp. 115-119.

Jang, M. and Lee, J. (2000) "Genetic Algorithm Based Design of Transonic Airfoils Using Euler Equations," in *Collection of Technical Papers-AIAA/ASME/ASCE/ASC Structures, Structural Dynamics and Materials Conference*, 1(2), pp. 1396-1404. Atlanta, GA.

Karthikeyan, B., Bryden, K.M. and Ashlock, D.A. (2005) "Low-Impact Image Segmentation by Balance Weighted Voronoi Tessellations," *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 15, pp. 533-541. ASME Press, New York.

Keane, A.J. (1994) "Experiences with Optimizers in Structural Design," *Proceedings of the 1st Conference on Adaptive Computing in Engineering Design and Control*, pp. 14-27. Published by the University of Plymouth, UK.

Kim, E-Y. (2005) "Analysis of Game Playing Agents with Fingerprints," (Doctorial Dissertation, Iowa State University, 2005).

Kimura, M. and Crow, J. (1963) "On the Maximum Avoidance of Inbreeding," *Genetic Research*, 4, pp. 399-415.

Kinnear, K. (1994) *Advances in Genetic Programming*. The MIT Press, Cambridge, MA.

Konkel, M. E., Monteville, M. R., Riveral-Amill, V. and Joens, L. A.(2001) "The Pathogenesis of *Campylobacter Jejuni*-Mediated Enteritis," *Current Issues in Intestinal Microbiology*, 2(2), pp. 55-71, 2001.

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

Kreft, J.U., Booth, G. and Wimpenny, J.W.T. (1998) "BacSim, A Simulator for Individual-based Modelling of Bacterial Colony Growth," *Microbiology*, Vol. 144, pp. 3275-3287.

Kreft, J.U. (2006) Personal web-site, http://www.theobio.uni-bonn.de/people/jan_kreft/bacsim.html, last accessed March 23, 2008.

Lee, M.K., Billington, J.B. and Joens, L.A. (2004) "Potential Virulence and Antimicrobial Susceptibility of *Campylobacter jejuni* Isolates from Food and Companion Animals," *Foodborne Pathogens and Disease*, Vol. 1, No. 4, pp. 223-230.

Lewis, A.J. and Southern, L.L. (2001) Editors, *Swine Nutrition*, 2nd Ed., CRC Press, Boca Raton, FL.

Lipsitch, M and Levin, B.R. (1997) "The Population Dynamics of Antimicrobial Chemotherapy," *Antimicrobial Agents and Chemotherapy*, Vol.41, No.2, pp. 363-373.

Loewe, S. (1953) "The Problem of Synergism and Antagonism of Combined Drugs," *Arzneimittel Forschumg*, Vol. 3, pp. 285-290.

Martin, T. J. and Dulikravich, G. S. (1992) "Implicit and Explicit Sensitivities for Optimization of Cooled Turbine Blades," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1992.

Mathias, K. and Whitley, D. (1994) "Transforming the Search Space with Gray Coding," *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, v /1, pp. 513-518.

Mayr, E and Ashlock, P. D. (1991) *Principles of Systematic Zoology*. McGraw-Hill, New York.

McCorkle, D. S., Bryden, K. M., and Carmichael, C. G. (2003) "A New Methodology for Evolutionary Optimization of Energy Systems," *Computer Methods in Applied Mechanics and Engineering*, 192(44-46), pp. 5021-5036.

McEliece, R. (1977) *The Theory of Information and Coding*. Addison-Wesley, Reading, MA.

Miller, J. F. and Thomson, P. (2000) "Cartesian genetic programming," *Proceedings of the European Conference on Genetic Programming*, pp. 121–132, London, UK. Springer-Verlag.

Food Safety First (2006) "Microbial Growth Simulation Program," http://www.foodsafetyfirst.org/pdfs/micro%20growth.pdf, last accessed April 15, 2008.

Food Safety and Inspection Service, United States Department of Agriculture (2006) "Microbial Pathogen Computer Modeling," http://www.fsis.usda.gov/regulations_&_policies/Notice_25-05/index.asp, last accessed April 13, 2008.

Mueller, M., Pena, A. and Derendorf, H. (2004) "Issues in Pharmacokinetics and Pharmacodynamics of Anti-Infective Agents: Kill Curves versus MIC," *Antimicrobial Agents and Chemotherapy*, Vol. 48, No. 2, pp. 369-377.

Mühlenbein, H. (1991) "Darwin's Continent Cycle Theory and Its Simulation by the Prisoner's dilemma," *Complex Systems*, 5, pp. 459-478.

Mühlenbein, H., Schomisch, M., and Born, J. (1991) "The Parallel Genetic Algorithm as Function Optimizer," *Proceeding of the Fourth International Conference on Genetic Algorithms*, pp. 271-278. Morgan Kaufmann Publishers.

MDO Test Suite. (2002). NASA Langley Research Center, Multidisciplinary Optimization (MDO) Branch, http://mdob.larc.nasa.gov/mdo.test, last accessed October 17, 2003.

National Research Council (1998) *Nutritional Requirements of Swine*, 10th ed., Washington, D.C., National Academy Press.

Nikolaou, M. and Tam, V.H. (2005) "A New Modeling Approach to the Effect of Antimicrobial Agents on Heterogeneous Microbial Populations," *Journal of Mathematical Biology*, vol 52, No. 2, pp. 154-182.

Nimwegen, E. V. and Crutchfield, J. P. (2001) "Optimizing Epochal Evolutionary Search: Population-Size Dependent Theory," *Machine Learning*, v. 45, n 1, pp. 77-114.

Nolting, A. and Derendorf, H. (1995) "Pharmacokinetic/Pharmacodynamic modeling of antibiotics," In *Handbook of Pharmacokinetic/pharmacodynamic correlation*, Ed. H. Derendorf and G. Hochhaus, CRC Press, Inc., Boca Raton, Fl.

Panikov, N.S. (1995) *Microbial Growth Kinetics*, Chapman & Hall, London.

Parmee, I. C. (2001) *Evolutionary and Adaptive Computing in Engineering Design*. Springer-Verlag London Limited.

Patience, J. F., Thacker, P. A., and deLange, C. F. M. (2005) *Swine Nutrition Guide, 2nd edition,* Prairie Swine Centre, Inc.

Prescott, J.F.,Baggot, J. D., and Walker, R.D. (2000) *Antimicrobial Therapy in Veterinary Medicine, 3rd ed.*, Iowa State University Press, Ames, IA.

Qiu, F., Guo, L., Wen, T.J., Ashlock, D.A.,and Schnable, P.S. (2003) "DNA Sequence-Based Barcodes for Tracking the Origins of Ests from a Maize CDNA Library Constructed Using Multiple MRNA Sources," *Plant Physiology*, 133, pp. 475-481.

Qiang, Z., Macauley, J., Mormile, M., Surampalli, R., and Adams, C. (2006) "Treatment of Antibiotics and Antibiotic Resistant Bacteria in Swine Wastewater with Free Chlorine," *J. Agriculture and Food Chemistry*, 54, pp. 8144-8154.

Rechenburg, I. (1984) "The Evolution Strategy: A Mathematical Model of Darwinian Evolution," in *Synergetics: From Microscopic to Macroscopic Order*, Frehlend, E. Ed., *Springer Series in Synergetics*, Vol. 22; pp. 122-132.

Reynolds, C. (1992) An Evolved, Vision-based Behavioral Model of Coordinated Group Motion. In Jean-Arcady Meyer, Herbert L. Roiblat, and Stewart Wilson, editors, *From Animals to Animats 2*, pp. 384-392. MIT Press.

Romero, J. and Machado, P. (2008) *The Art of Artificial Evolution*, Springer-Verlag London Limited.

Rosenquist, H., Nielsen, N.L.,Sommer, H.M., Norrung, B., and Christensen, B.B. (2003) "Quantitative Risk Assessment of Human Campylobacteriosis associated with Thermophilic Campylobacter Species in Chickens," *International Journal of Food Microbiology*, Vol. 83, pp. 87-103.

Rudolph, G. (2000a), "Takeover Times and Probabilities of Non-Generational Selection Rules", in *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA, pp. 903-910.

Rudolph, G. (2000b), "On Takeover Times in Spatially Structured Populations: Array and Ring", in *Proceedings of the Second Asia-Pacific Conference on Genetic Algorithms and Aplications*, Hong Kong, pp. 144-151.

Salipante, S.J. and Hall, B.G. (2003) "Determining the Limits of the Evolutionary Potential of an Antibiotic Resistance Gene," *Journal of Molecular Biology and Evolution*, 20(4), pp. 653-659.

Sarma, J. and De Jong, K. (1996)"An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms." *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV)*, pp 236–244.

Saunders, R. (2002) *Curious Design Agents and Artificial Creativity*, (Doctoral dissertation, University of Sydney, 2002).

Schaffer, J. D., Caruana, R. A., Eshelman, L. J., and Das, R. (1989) "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization," *Proceeding of the Third International Conference on Genetic Algorithms*, pp. 51-60. Morgan Kauffman.

Schnickel, A. P. and Craig, B.A. (2001) "Nonlinear Mixed Effects Model for Swine Growth," Purdue University 2001 Swine Research Report, Purdue University, Lafayette, IN.

Schmidt, S. K. (1992) "Models for Studying the Population Ecology of Microorganisms in Natural Systems," in *Modeling the Metabolic and Physiologic Activities of Microorganisms*, ed. by C. J. Hurst, pp. 1-59, John Wiley and Sons, NY.

Schwefel, H. P. (1975) "Evolutions strategie und Numerische Optimierung," (Doctoral dissertation, Technical University of Berlin, 1975).

Singer, R.S., Cox, L.A., Dickson, J.S., Hurd, H.S., Phillips, I., and Miller, G.Y. (2004) "Potential Risks and Benefits of Tylosin Use in Poultry," Interscience Conference on Antimicrobial Agents and Chemotherapy. Paper C2-1986, Poster #3011, Washington, DC.

Smith, J. and Fogarty, T. C. (1996) "Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm," *Proceedings of the IEEE International Conference on Evolutionary Computing*, pp. 318-323.

Spears W. M. (1993) "Crossover or Mutation?" *Foundations of Genetic Algorithms 2*, pp. 221 – 237. Morgan Kaufmann.

Spears, W.M. (2006) "Genetic Algorithms (Evolutionary Algorithms): Repository of Test Functions," http://www.cs.uwyo.edu/~wspears/functs.html, last accessed April 3, 2008.

Syswerda, G. (1991) A Study of Reproduction in Generational and Steady State Genetic Algorithms. *Foundations of Genetic Algorithms*, pp. 94-101. Morgan Kaufmann.

Thakur, S. and Gebreyes, W.A. (2005) "Prevalence and Antimicrobial Resistance of *Campylobacter* in Antimicrobial-Free and Conventional Pig Production Systems," *Journal of Food Protection*, Vol.68, No.11, pp. 2402-2410.

Urban, G. L., Bryden, K. M., and Ashlock, D. (2002) "Engineering Optimization of an Improved Plancha Stove," *Energy for Sustainable Development*, 6(2), pp. 5-15.

Van Boekel, M. A. J. S. (1996) "Statistical Aspects of Kinetic Modeling for Food Science Problems," *Journal of Food Science*, Vol.61, No. 3, pp. 477-485.

Vavak, F., Jukes, K., and Fogarty, T. C. (1997) "Adaptive Combustion Balancing in Multiple Burner Boiler Using a Genetic Algorithm with Variable Range of local Search," *The proceedings of the 7th International Conference on Genetic Algorithms*, pp. 719-726. Morgan Kaufmann.

West, D. B. (1996) *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ 07458.

Whitley, D. (1989) "The Genitor algorithm and selection pressure: Why Rank Based Allocation of Reproductive Trials is Best," *Proceedings of the 3rd ICGA*, pp. 116-121. Morgan Kaufmann.

Whitley, D. and Starkweather, T. (1990) "GENITOR II: A Distributed Genetic Algorithm," *Journal of Experimental and Theoretical Artificial* Intelligence, vol. 2, pp. 189-214.

Whitley, D., Mathias K., Rana S., and Dzubera, J. (1996) "Evaluating Evolutionary Algorithms," *Artificial Intelligence*, vol 85, num 1-2, pp. 245-276.

World Health Organization (1997) "The Medical Impact of the Use of Anti-Microbials in Food Animals: Report and Proceedings of a WHO Meeting, Berlin, Germany.

World Health Organization (1998) "Use of Quinolones in Food Animals and Potential Impact on Human Health: Report and Proceedings of a WHO Meeting, Geneva, Switzerland.

World Health Organization (2001) "Monitoring Anti-Microbial usage in Food Animals for the Protection of Human Health: Report of a WHO Consultation," Oslo, Norway, September 2001.

Wolpert, D. H. and Macready, W. G., (1995) "No Free Lunch Theorems for Search," Santa Fe Institute, Santa Fe, NM. Tech. Rep. SFI-TR-05-010.

Wolpert, D. H. and Macready, W. G., (1997) "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computations*, Vol.1, No.1, pp. 67-82.

Wright, S. (1986) *Evolution*. University of Chicago Press. Edited and with introductory Materials by W. B. Provine.

Wu, A. S., Lindsay, R.K., and Riolo R. (1997) "Empirical Observations on the Roles of Crossover and Mutation," *Proceedings of the Seventh International Conference on Genetic Algorithms*, pp. 362-369. Morgan Kaufmann, San Francisco.

Wu, A. S. and DeJong, K. A. (1999) "An Examination of Building Block Dynamics in Different Representations," *Proceedings of the Congress on Evolutionary computation*, vol. 1, pp. 715-721. IEEE Press.

Yu, T. and Miller, J. F. (2002) "Finding needles in haystacks is not hard with neutrality," *EuroGP '02: Proceedings of the 5th European Conference on Genetic Programming*, pp. 13–25, London, UK. Springer-Verlag.

Zhi, J., Nightingale, C.H., and Quintiliani, R. (1988) "Microbial Pharmacodynamics of piperacillin in Neutropenic Mice of Systematic Infection Due due to *Pseudomonas aeruginosa*," *Journal of Pharmacokinetics and Biopharmacy*, Vol. 16, pp. 355-375.

Zimmerman, D.R. (1986) "Role of Subtherapeutic Antimicrobials in Animal Production," *Journal of Animal Science*, 62 (Suppl. 2):6.