2012

An investigation of exploitation versus exploration in GBEA optimization of PORS 15 and 16 Problems

Kaelynn Adele Koch
Iowa State University

An investigation of exploitation versus exploration in GBEA optimization of PORS 15 and 16 Problems

by

Kaelynn Adele Koch

A thesis submitted to the graduate faculty in partial fulfillment of the requirements for the degree of Master of Science

Major: Mechanical Engineering

Program of Study Committee:

Kenneth Mark Bryden, Major Professor

Xinwei Wang

Halil Ceylan

Iowa State University

Ames, Iowa

2012

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help and support of those around me. I would like to thank my advisor, Mark Bryden, for his guidance and patience in completing this thesis. Also, I would like to thank my colleagues for listening to my ideas and giving me suggestions. Finally, I would like to thank my friends and family, especially my mother Julie and David, for keeping me grounded and having an endless supply of understanding and encouragement.

This work was performed at the Ames Laboratory under contract number DE-AC02-07CH11358 with the U.S. Department of Energy. The document number assigned to this thesis/dissertation is IS-T 3056.

CHAPTER I. INTRODUCTION

Optimization routines have long been used in engineering to find the optimal parameters for both theoretical and real world problems. The motivation behind researching optimization routines is to cut down on methods that are costly in time, energy, and materials. Gradient-based optimization algorithms are relatively simple to implement. However, there are many optimization problems that cannot readily be solved with gradient-based optimization algorithms. In searching for alternative methods to optimize complicated problems, researchers have looked to optimization processes in nature. Evolutionary algorithms (EAs) are a broad class of optimization methods that incorporate biological processes and phenomena, including natural selection and survival of the fittest to optimize a problem. EAs have enjoyed particular success in problems that are deceptive, discontinuous, and multidimensional. However, EAs are slow and require numerous function calls. In the case of high fidelity models of complex systems (e.g., computational fluid dynamics) this can result in prohibitive demands on computational time.

To address these issues, researchers have looked at other variables in nature that impact the natural selection and evolutionary process. In nature, barriers such as geography and mating rituals restrict mating and preserve diversity. Earlier research has identified combinatorial graphs as a means to mimic the geographical restrictions to mating in evolutionary algorithms (Bryden et al 2006). Combinatorial graphs are a collection of vertices and edges. Each member of the evolving solution set is placed at a

vertex of the combinatorial graph. Vertices that are connected together by edges and solutions are only permitted to mate with those solutions on adjacent edges.

A variety of test problems were used for initial studies including: one-max, the De Jong functions, the Griewangk function, the self avoiding random walk problems (SAW), and the plus one recall store problems (PORS). These problems are resistant to optimization with gradient based optimizers, such as hill climbing. Depending on the problem, the choice of combinatorial graph was found to significantly impact the time to solution. It was hypothesized that the variations in time to solution are driven by the competing mechanisms of exploration and exploitation (Corns 2008). Deceptive problems need to maintain diversity and prefer greater exploration. This permits population members to be isolated from one another, allowing for the exploration of the fitness landscape as the population evolves. Simpler problems prefer exploitation to quickly eliminate ineffective options and drive the rest of the population to evolve toward the current best solution. However, this hypothesis remains largely untested.

This thesis explores this hypothesis by examining two contrasting problems that embody the hypothesized tradeoff between exploration and exploitation, the PORS 15 and 16 problems. The PORS 15 problem is a difficult, deceptive problem. In contrast PORS 16 is a relatively simple problem. When solved using a graph based evolutionary algorithm (GBEA) PORS 15 is solved more than ten times faster using a cycle graph than with a complete graph. PORS 16 is solved 1.25 times faster using complete graph than a cycle graph. This thesis uses this observation as the starting point to examine exploration and exploitation in GBEAs.

The organization of this thesis is as follows. Chapter 2 explores the development of evolutionary algorithms, evolutionary algorithms that use topology, and graph based evolutionary algorithms. Chapter 3 outlines the computational experiments used to examine how graphs impact the tradeoff between exploration and exploitation in solving PORS 15 and 16. Chapter 3 also explores the development of the PORS problems and their characteristics that make them a good means of testing the aforementioned thesis. Chapter 4 outlines the results of the experiments. Chapter 5 discusses these results for the use of GBEAs. Chapter 5 also discusses how these results relate to the proposed hypothesis and provide guidance. Chapter 6 explores the implications of this experiment, the conclusions we have arrived to, and future directions of this research.

CHAPTER II. BACKGROUND

As briefly discussed in Chapter 1, evolutionary algorithms are designed to mimic the natural selection process of survival of the fittest (Alba and Cotta 2006). Biological principles such as inheritance, natural selection, and genetic operators are programmed in an iterative format to evolve a population in search of the optimum solution to a problem. Genetic operators include crossover, recombination, and mutation. Fitness can be encoded as binary strings of 1s and 0s to represent chromosomes or other complex structures that are problem specific. These chromosomes interact and are manipulated through genetic operators. Evolutionary algorithms refer to a large class of algorithms that use any of the aforementioned mechanisms for optimization. These include genetic algorithms (GAs), genetic programming, evolutionary programming, and evolution strategies (Alba and Cotta 2006). Evolutionary algorithms have enjoyed a great deal of success in solving problems that cannot be readily solved using gradient-based optimization. For example, deceptive problems and problems with discontinuous search spaces are not easily solved with gradient-based optimization.

Evolutionary algorithms are not immune from falling into the traps of local optima or local minima that occur gradient-based optimizers. An evolving population in EAs must retain sufficient diversity to avoid driving the population toward a local optima or minima. However the mechanisms for managing overall population diversity are poorly understood and underdeveloped. The population can be injected with random members that replace those whose fitness values and parameters are overrepresented, but determining when and how to perform this operation is difficult. Also, there is no

guarantee that these newly generated members could drive the older members out of the traps.

In nature, evolution is a process that can take millions of years. As EAs are directly based off of this process, they can be computationally expensive. This is particularly a problem with engineering design problems. For example, computational fluid dynamics problems are computationally expensive. In the design of fluids systems, changes cannot be described with low fidelity or engineering models in a satisfactory manner. Computational fluid dynamics are often used to model thermal and fluid system performance within the design, but the compute time required for CFD limits its usefulness as a design and optimization tool (McCorkle et al 2003).

Many methods for reducing the computational time of using CFD in optimization have been developed. These include using EAs with reduced versions of the Navier-Stokes equations that are solved implicitly or by marching to reduce the computational expense (Tannehill et al. 1997). Examples of these reduced equations are the Euler equations or the viscous shock-layer equations. In other cases, low detail models are used to evolve designs and to construct a high detail model that can be verified and refined further with an EA. Neural networks can also manipulate design structures during evolution to make sure that only designs that meet the design criteria are considered (Bryden et al 2002). The approach taken in Bryden et al 2002 was to limit the number of function calls by imposing a combinatorial graph on the evolving population to mimic geographic obstacles to the spread of genetic information in an evolving population.

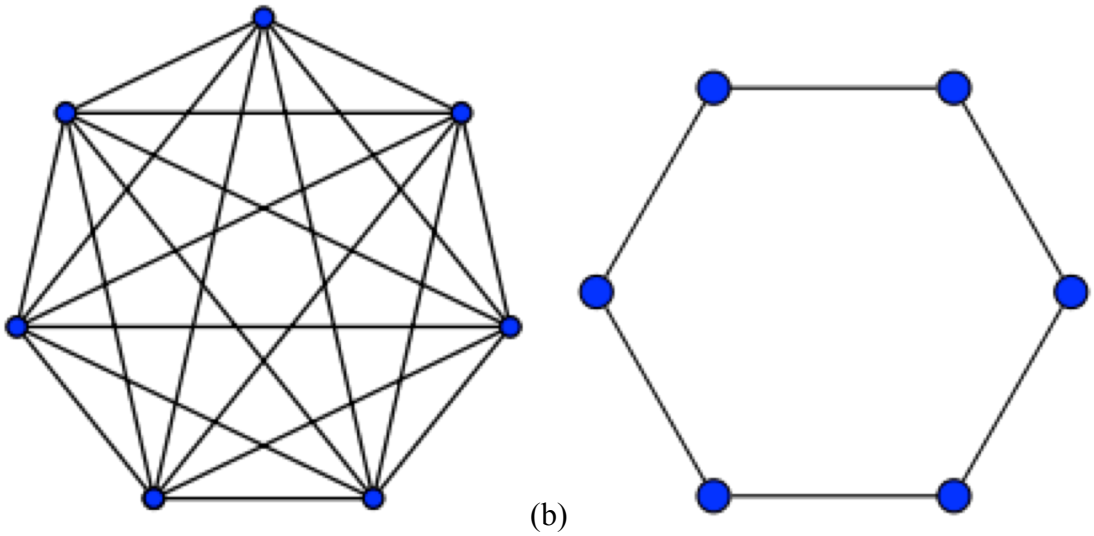
Spreading genetic information in nature is not easy. Physical and social obstacles exist that prevent the sharing of genetic information. Examples of these obstacles include

sex, mating rituals, cultural customs, and geography. Incorporating these obstacles into EAs could overcome the shortcomings EAs have. For example, most EAs consider each population member to be both male and female; therefore population members are free to mate with all other members. Some GAs have incorporated female and male differentiation with the goal of promoting either global search or local searchers (Garcia-Martinez 2005). Other means of including natural constraints have been more widely adopted. One of the most popular is the imposition of an artificial geography onto an evolving population and it takes many forms. Island genetic algorithms have subpopulations that only interact with each other through migration (Parmee 2001).

Geographical constraints in nature have managed to produce varied ecosystems. A dramatic case of this is the island of Madagascar. Having broken off of the African subcontinent before the evolution of large mammals, they are noticeably absent from the island whereas they can be found almost everywhere else in the world. In addition to this, many of the species are specific to the island and the ecosystem does not resemble African ecosystem despite its geographically close proximity. Common methods that use geography could speed up computational time are island model GAs and some kinds of parallel GAs. The intent of these methods is to either speed up convergence or preserve diversity among the population to avoid premature convergence. Single population and multiple population parallel GAs, also referred to as island parallel GAs, use topology in different ways to impact mating interactions between individuals (Cantu Paz 1998). Island GAs fall into serial and parallel categories. Serial island GA process all islands on the same processor whereas island parallel GAs have distinct groups of populations that evolve on separate computer nodes in order to improve computation time. Island model

GAs consist of separate populations that mostly evolve independently of one another. Interaction between populations occurs at preset intervals through “migration”. Genetic information from the islands is exchanged infrequently through migration between algorithms according to the kind of topology used to connect the islands (i.e. hypercube, toroidal, etc.). It has been shown that island model GAs can outperform standard GAs for certain types of problems. However, according to Cantu Paz the mechanisms responsible for this increase are not fully understood. Single population parallel GAs consist of one spatially distributed population. Ideally, there is one individual assigned to every processing element available, so that the algorithm can be parallelized and run more efficiently. The form of topography they utilize is “neighborhoods”. The selection and mating of population members are restricted to neighborhoods that overlap, permitting some kind of interaction among all individuals of the population. Because the information exchange between processors is not as computationally expensive as the genetic algorithm itself, this process does make the algorithm run more efficiently than a serial genetic algorithm. Some researchers have attempted to combine the above types of genetic algorithms to formulate “hierarchical” GAs in order to improve performance. The theory behind combining these two types is that smaller partial solutions could be constructed and shared to create larger partial solutions. (Cantu Paz 1998)

As noted earlier, combinatorial graphs have been used to impose “geography” on an evolving set of solutions. Combinatorial graphs can be understood as artificial geographical structures. There are many ways that one could use combinatorial graphs in combination with evolutionary algorithms. Figure 2.1 shows a few of the combinatorial graphs that have been used in GBEA optimization.



(a) (b)

Figure 2.1. Examples of Combinatorial Graphs: (a) a complete graph and (b) a cycle graph with seven vertices.

Combinatorial graphs are a collection of vertices connected to one another through edges. An edge consists of two vertices connected to one another. These two vertices are referred to as neighbors. Collections of edges can be thought of as a path. Graph diameter is defined as the largest number of edges along the shortest path between two vertices on the graph (Bryden, Ashlock, Corns, and Wilson 2006). Diameter can also be thought of as the shortest path to the furthest member across the graph.

In GBEAs, a single population member is placed on a vertex of the graph and permitted to only interact with other vertices it is connected to by a single edge (neighbors). This manner of connecting population members is very similar to the island

genetic algorithm, except that in the case of the GBEA there is a single member evolving at each vertex of the graph rather than a subpopulation. There are also no migration operators. One mating event occurs at a time in GBEAs, making them steady state evolutionary algorithms. A population member on the graph is selected at random and its co-parent is chosen through roulette selection of its neighbors as determined by the graph selected. Children are created through an exchange of random genetic information, known as crossover.

It has been theorized that the choice of graph can manage the rate of information transfer and modify the diversity of the evolving population. The rate of information transfer is related to the diameter of the graph used (Corns 2008). However, to date there has been no direct evidence that the average diversity of the population is either preserved or destroyed with certain graphs.

GBEAs have been found to improve performance on some problems, specifically difficult or deceptive problems. Each problem solved prefers a particular graph or family of graphs (Bryden and Ashlock 2006). However, not all problems preferred high levels of diversity and subsequently performed better with a complete graph. These problems were typically unimodal or highly polynomial. Performance of particular graphs has been found to be heavily dependent on population size, as seen in Figs. 2.2 and 2.3.

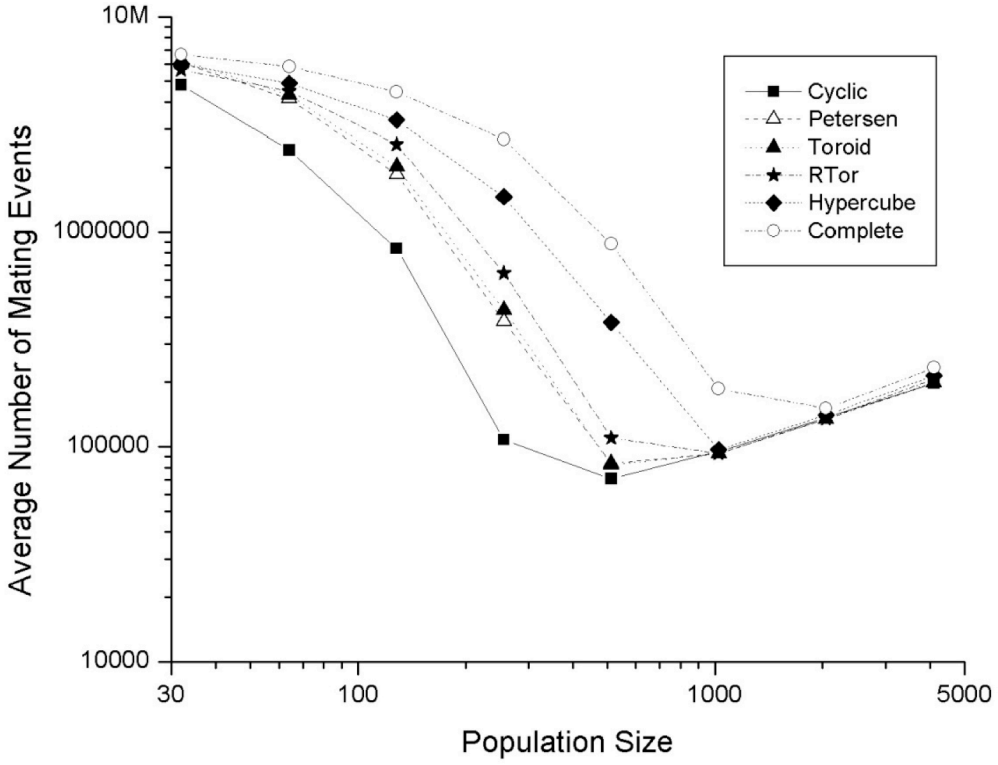


Figure 2.2. Performance of GBEAs for PORS 15 as a function of population size and graph.

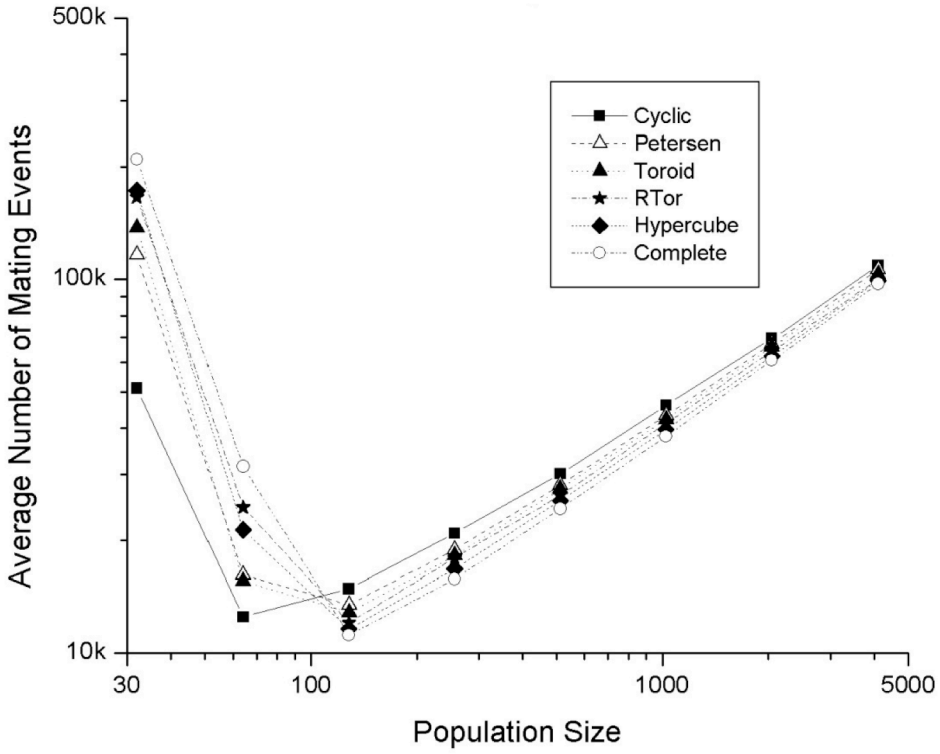


Figure 2.3. Performance of GBEAs for PORS 16 as a function of population size and graph.

Figures 2.2 and 2.3 display the results Corns obtained for the PORS problems when solved with GBEAs. PORS problems are solved with the fewest number of mating events, on average, with a population size of 512. However, each problem preferred a different graph. The cycle graph consistently solved the PORS 15 with the fewest number of mating events. PORS 16 was solved with the fewest number of mating events with the complete graph with a moderate population size. Corns attributed these diametric performances to the graph diameter and problem type. As discussed, the rate of

information transfer is related to the diameter of a graph. PORS 16 is a simple problem that does not require diversity to find a solution. Therefore, speeding up the rate of information transfer would not have an impact on finding a solution efficiently. However, PORS 15 is a difficult problem that requires diversity to find the optimal solution. Slowing the rate of information transfer would preserve diversity, therefore the graph with the largest diameter, the cycle graph, performs best overall.

Corns also theorized that GBEAs can provide a structured way to mediate the competing processes of exploitation and exploration. Graph selection modifies the rate and path of information spread, and performance can be geared either in favor of exploitation or exploitation. Exploitation manipulates the strongest population members to find a solution. The parameters of these population members are manipulated and weaker population members are eliminated quickly. This is referred to as burning out diversity in EAs. However, these weak population members may have components that can improve fitness and removing them from the population may hinder or prevent convergence on the true optima. Exploration is used as a means of preserving diversity by allowing competing solutions within the solution set time to evolve.

Some qualitative guidelines exist for selecting a graph for GBEA optimization that balances time-to-solution and diversity preservation. These guidelines are outlined in Corns 2008 and primarily refer to simple fitness landscapes. Simple fitness landscapes benefit best from exploitation that comes from using graphs with small diameter (i.e., complete graph or hypercube graph) and the exploration time being mediated with the population size. Theoretical time to solution can be estimated by the following equation:

$$\text{Mean takeover time } = C(dn) \qquad \text{(Eq. 1)}$$

In Eq. 1, d is the diameter of the graph, n is the population size, and C is a proportionality constant related to the graph type and in some cases a weak function of population size. Takeover times are the time required will take for a population to be taken over by a single solution. For more information on calculating takeover times in GBEAs, see Corns 2008.

When designing an EA for a particular problem, mediating the tradeoff between exploration and exploitation is important. Is it better to stick with the current best solution and exploit it to find the local optimum and risk discarding potentially valuable solutions? Exploitation can expedite the process and provide a serviceable solution, regardless of whether it is a true optimum or not. Or is it wiser to continue to explore for other solutions that appear to be weaker, but that may result in finding even better parameters to manipulate? Exploration can be time consuming, but can yield better quality results than focusing solely on exploitation would. All optimization problems are different and determining what kinds of fitness landscapes are best suited to these mechanisms is often a process of trial and error. This thesis examines how exploitation and exploration are brokered using particular graphs for two different types of problems.

CHAPTER III. COMPUTATIONAL EXPERIMENT

The goal of this work is to examine the tradeoff between exploitation and exploration in GBEA optimization. To do this, computational experiments are used to examine how solutions evolve in PORS 15 and 16 problems when solved using GBEAs. The experiment is comprised of three components; the graphs and the population, the evolutionary algorithm rule set, and the example problems. The complete, hypercube, and cycle graphs were used for this experiment. A fixed population size of 512 was used as per the guidelines from Corns thesis (Corns 2008). The example problems used were the PORS 15 and 16 problems. PORS 15 represents the subset of difficult PORS problems and PORS 16 represents the subset of PORS problems that are easiest to optimize.

Plus one recall store (PORS) is an optimization problem based on the idea of a simple calculator with four buttons: plus, one, store, and recall. Integer addition and store are classified as operations, and one and memory recall are classified as terminals. The goal is to arrange a fixed number of keystrokes in a way that maximizes the numerical result. The number of keystrokes is generally identified in the description of the problem. For example, PORS 3 would have 3 keystrokes and PORS 12 would have 12 keystrokes. There are a number of ways to express PORS. It is best to start algebraically to understand the basic operations involved. Consider the PORS 6 problem with the following keystrokes. Figure 3.1 shows the six strokes, their order algebraically, and the numerical result.

1+1 Sto + Rcl=4

1+1=2, Sto=2, Sto + Rcl= 2+2=4

Figure 3.1. Algebraic representation of the solution to PORS 6.

PORS 15 and 16 represent deceptive and non-deceptive problems and have contrasting results at a population size of 512 when solved with GBEAs. The PORS 15 problem is solved quickest with the graph with the largest diameter, the cycle graph, and slowest with the graph with the shortest diameter, the complete graph. The PORS 16 problem is solved with the smallest number of mating events by the graph with the complete graph and the largest number of mating events with the cycle graph. Corns identified that the rate of information transfer within the graphs is directly related to graph diameter and this rate of information transfer impacts how solutions are formulated for a particular problem. Because the problems are in the same family and have contrasting results, they are ideal for making comparisons between exploration and exploitation in GBEAs. As noted, this thesis examines the hypothesis put forth by Corns 2008, the choice of graph can be used to manage the exploration of the solution space versus the exploitation of the best population members.

The order of keystrokes can also be expressed in a parse tree format to resemble chromosomes. Parsing is a process by which a compiler takes parts of the source code and produces a structure in memory that represents the parts of a program that can be executed (Ashlock 2005). Parse trees can be used to represent complex mathematical expressions by creating recursive structures. These structures when broken down into their simplest form are binary trees with one root at the top and a leaf to either side of the root. For example + is the node and both leaves are 1, shown in Fig. 3.2 below.

Figure 3.2. Parse tree representation of 1 + 1.

These trees are evaluated left to right, bottom to top, as shown below in Fig. 3.3. The value of each node is after the semicolon.

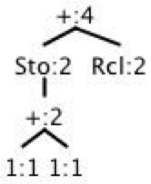


Figure 3.3. Parse tree representation of the solution to PORS 6.

Each solution to the PORS efficient node use problem, with greater than two nodes, must be constructed from one or more of the six subtrees in Fig. 3.4. (McCorkle, Ashlock, Bryden 2011)

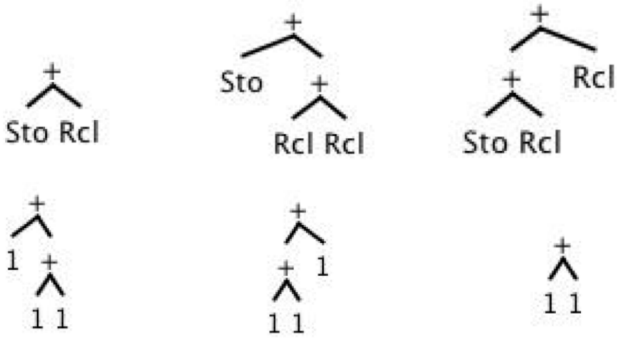


Figure 3.4. The six subtrees found in all optimal PORS structures.

PORS 16 has 24 possible solutions. Each of these consists of a combination of the six building blocks and result in the numerical result of 36. PORS 16 does not have any

local optima, like PORS 15, and is classified as a non-deceptive problem because these structures cannot trap the algorithm. This is a problem that seems to have no need for diversity and there are no local optima for the algorithm to get hung up on as a result (Corns 2008). One of the 24 optimum solutions is arrived at with little difficulty when using the complete graph. The possible solutions are shown in Fig. 3.5

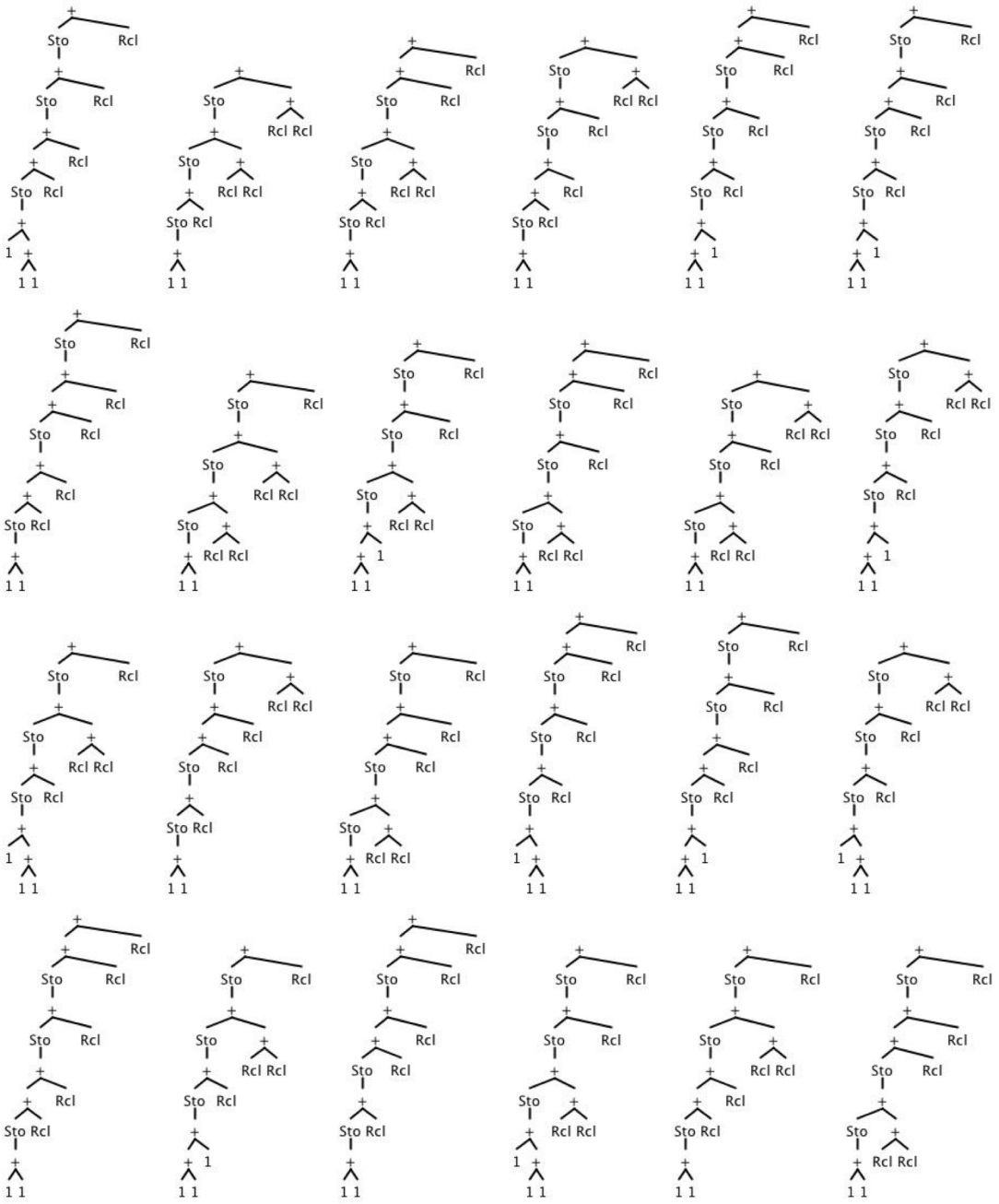


Figure 3.5. Solutions to PORS 16.

In contrast, the solution to PORS 15 is unique and evaluates to 32. The parse tree contains four (+ Sto Rcl) subtrees and one (+ 1 1) subtree, as shown in Fig. 3.5.

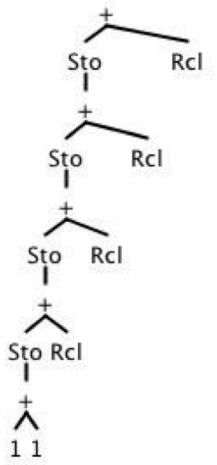


Figure 3.6. The optimal structure for PORS 15.

PORS 15 is a deceptive problem, as there are many ways to achieve a high fitness value that is not necessarily the maximum and cannot be built upon to reach the true maximum. Even more challenging is that solutions resulting gin a score of 27 do not contain the two critical building blocks to required to reach 32. As a result, if the solution converges to 27 it is very difficult to reach 32 since mutations invariably result in less fit structures eliminated by mating. The highest fitness achievable below 32 is 27. The four

trees that evaluate to 27 are shown below in Fig 3.7.

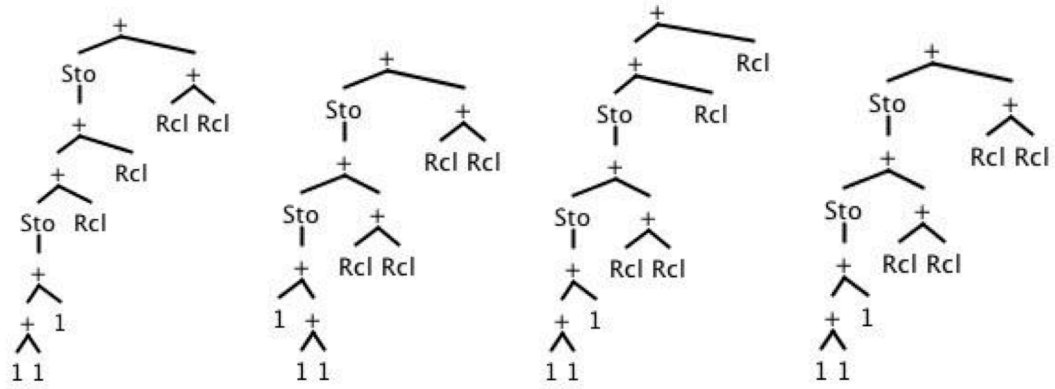


Figure 3.7. The four parse trees with 15 nodes that evaluate to a fitness of 27.

Upon breaking down everything to the six structures present in optimum solutions, all but (+ 1 1) and (+ Sto Rcl) are present in these structures. Therefore the essential blocks for finding the optimum of 32 do not exist, and the structures that evaluate to a fitness of 27 for PORS 15 have been declared to have no value towards the optimum solution of PORS 15 (Ashlock and Lanthrop 1998). Herein is the deceptive nature of the problem.

The next highest fitness below 27 is 24. The substructures essential to optimum fitness are present, as can be seen in Fig 3.8.

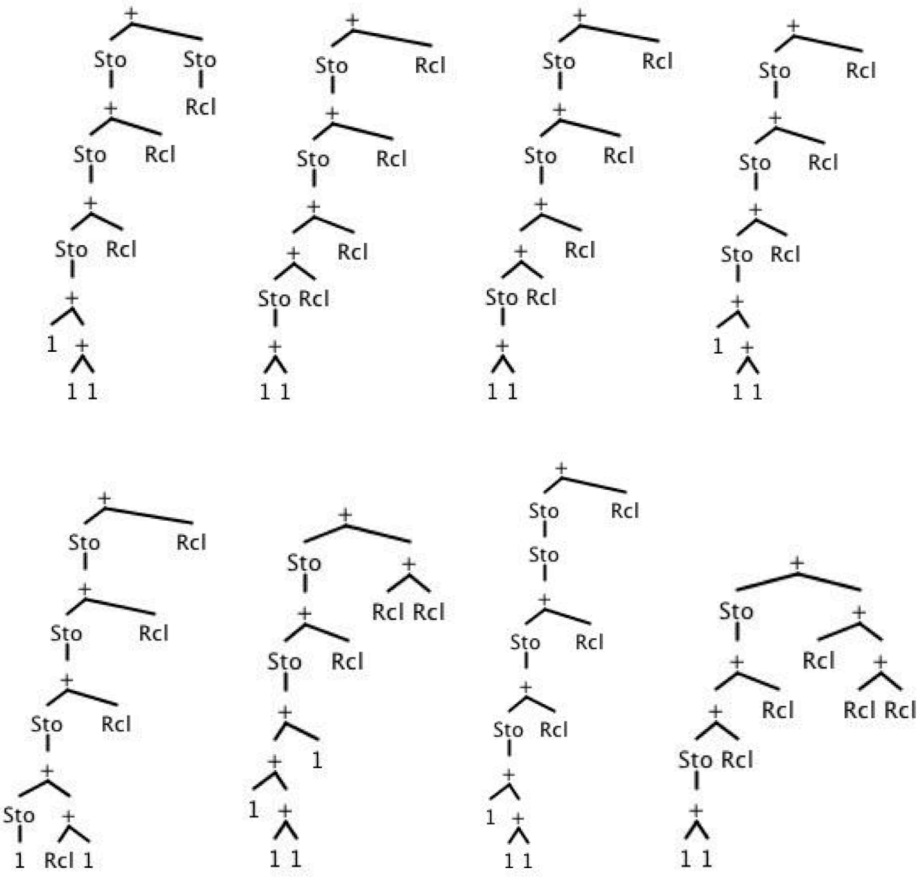


Figure 3.8. The eight parse trees with 15 nodes that evaluate to a fitness of 24.

It is possible to achieve the optimum of 32 from a tree with a fitness of 24, especially where there are (+ Sto Rcl) chains. There is large amount of diversity left in these structures that can be exchanged to formulate the optimum solution of 32.

The graphs studied in this paper are the complete graph, hypercube graph, and cycle graph. Selection of these graphs was based on their contrasting performance, as discussed in Chapter 2, on the PORS 15 and 16 problems for population size 512. The complete graph has 512 vertices that are all connected to each other. This is the traditional, unaltered genetic algorithm. There are no geographical limits being imposed on mating. The complete graph has the smallest diameter. The hypercube graph consists

of a set of all 512 character binary strings as its set of vertices. The edges consist of pairs of strings that differ in exactly one position. This graph has the second smallest diameter, being 9 for population size 512. The cycle graph links all population members into a ring of 512 vertices. Each vertex of the graph has two neighbors with which it can interact. The cycle graph has the largest diameter, being of size 256 for population size 512.

The general parameters for the GBEA used in this experiment for each of the graphs are as follows:

- The program converges when a structure that contains the known optimum fitness for a preset number of nodes is found.
- The EA is terminated if it fails to meet the convergence criteria before 10 million mating events. Failures are recorded and included in the average.
- Population size of 512 is set and does not fluctuate. Initial parse tree values are randomly assigned.
- Crossover causes parents to exchange random subtrees to create new children.
- If crossover results in a higher number of nodes than specified, resulting children have either the left or right subtrees promoted and then the topmost subtree may be removed to reduce the total number of nodes to be equal to or less than that specified by the problem.
- If the study includes mutation, mutation is then performed at a ten percent chance. Some cases were run without mutation to determine the role of crossover.
- If the resulting children have higher fitness than their parents, the parents are replaced. This is defined as elite replacement.

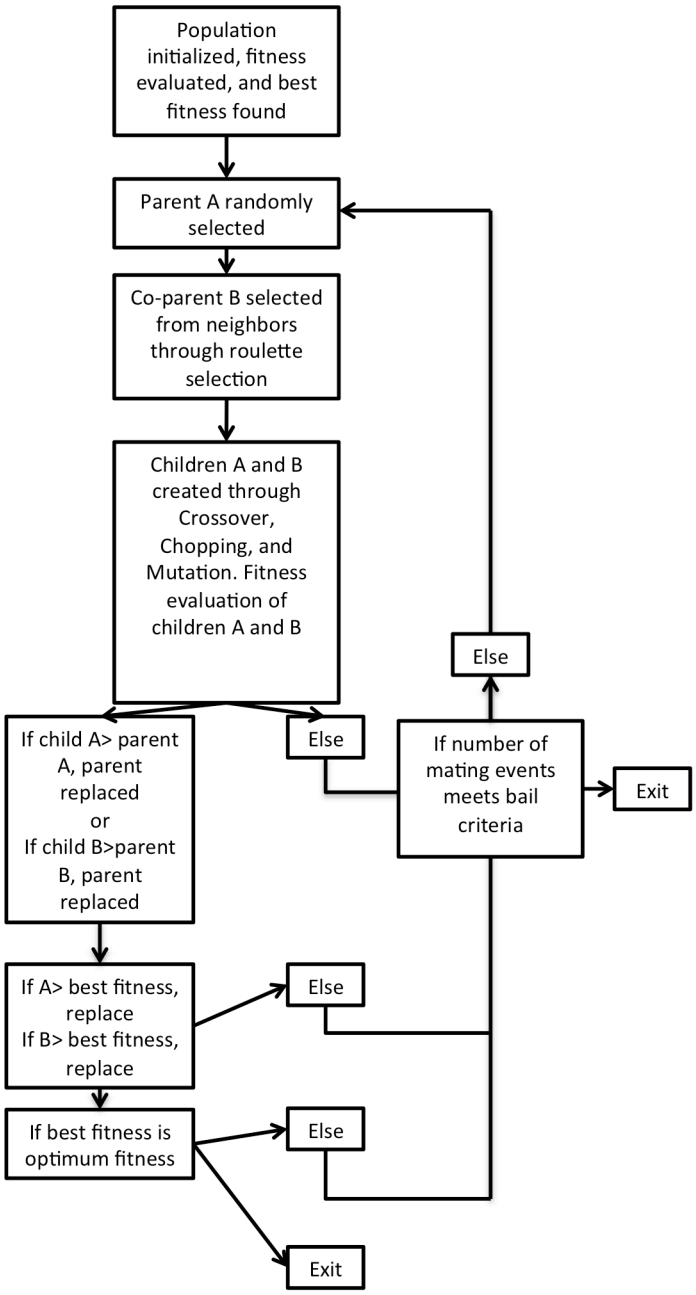


Figure 3.9. Flowchart of GBEA.

Parents are chosen through roulette selection of the available neighbors on the graph. Random subtrees are chosen from the parent and co-parent and are exchanged. Next, the resulting trees are chopped, if necessary, such that the number of nodes is less than or equal to N. Chopping first promotes either the main left or right subtree, and then removes the top subtrees if necessary to bring n<=N. Mutation is then performed by selecting a random subtree and replacing it with a randomly generated subtree of the same size. Elite replacement is then used, meaning that if the child is better than its corresponding parent, it replaces that parent. The results of crossover, chopping, and mutation for each mating event are recorded for analysis. The majority of the mating events do not result in replacement; therefore they are non-constructive. Constructive mating events are those that result in the replacement of one or both of the parents by a child with better fitness.

Analysis of the PORS problems when solved with GBEAS has been done before, as discussed in Chapter 2. While it has been theorized that using a graphical structure preserves diversity, there has been no direct evidence that this is the case. To determine how solutions are formulated in PORS 15 and 16, it is necessary to determine the average population fitness at a given mating event, the population diversity, and the average number of constructive mating events necessary to find a solution. Studies were done without mutation and with a 10% mutation rate to remove a degree of randomness from evaluating what is typical in evolving a solution in GBEAs.

Guidelines for calculating the average number of constructive mating events followed all other parameters for GBEAs as mentioned above in addition to some extra

calculations. The constructive mating events need to be related to the total mating events to determine what typical behavior for the problem and graph selected is. Every time a parent is replaced, it is counted as a constructive mating event. The number of constructive mating events for a given run is summed. 10,000 runs were performed and the constructive mating events required to reach a solution were averaged and the standard deviation was calculated. Some averages do include failures and the number of failures is recorded.

The guidelines for the average population fitness experiments are as follows. Ten populations with different randomly generated initial structures evolved for a specified graph with GBEA optimization. The parameters for convergence as stated above held true. Failures were recorded as well as successes if they arose. Average population fitness is calculated for each mating event, regardless of whether it is constructive or not. This value is stored along with its corresponding mating event. Depending on the problem, average fitness was stored for every 100 or 1000 mating events. Average population fitness for one run is averaged with the other nine for every 100 or 1000 mating events for each graph studied for the PORS 15 and 16 problems.

CHAPTER IV. RESULTS

As noted earlier, the PORS 15 and PORS 16 problems were run on the compete, hypercube, and cycle graphs with 10% and 0% mutation providing twelve test cases. For each test case the average number of mating events, the average fitness as a function of mating event, the average number of constructive mating events, and the population diversity were determined. The average number of mating events was determined by summing the mating events until convergence was achieved for each run, for 10,000 runs and then averaging them. Failures were included in the average as the number of mating events at which the algorithm terminated if convergence was not met. The average number of constructive mating events was determined by summing all of the events that lead to the replacement of a parent for each run over 10,000 runs and then averaged. Runs that resulted in failures had their constructive mating events included as well. Summing the fitness values for all population members at a given mating event and averaging them determined the average fitness as a function of mating event for the population. This calculation was performed until the program converged or failed. Data was collected for ten runs. For every 100 or 1000 mating events, the average fitness of the populations was averaged and plotted as a function of mating events. Phylogenetic trees were constructed for a few runs to demonstrate how modifications occur in GBEA optimization and see how the solution evolved over time.

Figure 4.1 displays both the phylogenetic tree of population member 44 and the structural changes made to 44 through mating events when the child replaced the parent tree. The phylogenetic tree identifies the co-parent that played a role in the replacement of 44. The progression of population member 44's initial structure to the final structure for the PORS 15 efficient node use problem using the complete graph is shown on the right hand side of the figure. Population member 44 begins with an initial parse tree that evaluates to 8, which is expressed as 44-8 in the diagram. Many mating events do not result in replacement of either parent parse tree. The initial parse tree for population member 44 was randomly generated when the population was seeded at the beginning of the run. As discussed in Chapter 3, parent A is selected randomly and parent B is selected among the parent A's neighbors on the graph through roulette selection. In this case for the complete graph, all 512 members are adjacent and can interact with each other.

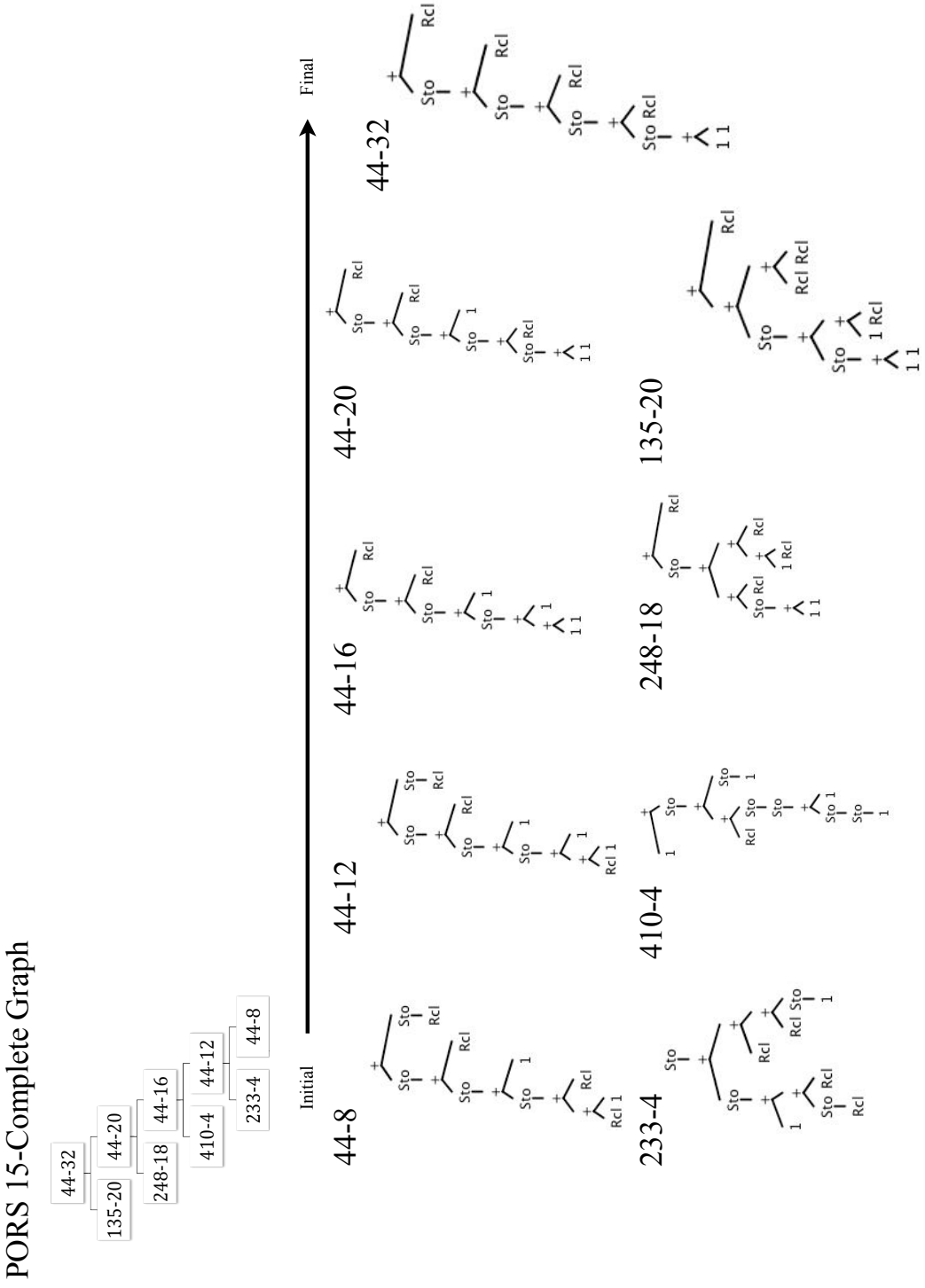


Figure 4.1. The phylogenetic tree of the population member with the best fitness for PORS 15 when solved with a complete graph.

Population members 44-8 and 233-4 mate and this results in the child corresponding to 44-8 replacing its parent with a new parse tree with a fitness of 44-12. Exchange of genetic information between parents in crossover did not change the structure of 44-8. Mutation of the second lowest subtree from Rcl to 1 resulted in a change of structure and a higher fitness of 12. 44-12 replaced 44-8.

The next mating event with population member 44 that resulted in the replacement of 44-12 was with 410-4. The top right subtree of (Sto Rcl) of 44 was replaced through crossover with (Rcl) from 410-4. The lowest left leaf on 44 was changed from Rcl to 1 through mutation. This resulted in 44-12 being replaced by the new tree that evaluated to 44-16 that was generated through crossover and mutation.

Population member 44-16 was next replaced in a mating event with 248-18. Genetic information from 44-16 was exchanged through crossover and resulted in the removal of the building block (+ (+ 1 1) 1) in exchange for one of essential building blocks (+ (Sto (+ 1 1) 1). Mutation resulted in (+(Sto (+ 1 1) 1) changing to (+ (Sto (+ 1 1) Rcl). This change resulted in two of the essential building blocks for PORS 15 aligning and 44-16 being replaced by the newly generated tree, 44-20.

The final mating event that resulted in the replacement of 44 was with 135-20. From 135-20, 44-20 obtained a single (Rcl) that replaced the right leaf node of (1) in the middle. No mutation occurred. This change in structure that resulted from crossover with 135-20 produced a fitness of 32. The algorithm terminated at this point because it had found a tree that matched the preset fitness required for convergence. Note that the algorithm does not know what the optimum subtree looks like, just the fitness it evaluates to.

Mutation plays a significant role in creating children with higher fitness values than their parent. Three of the four mating events involved a mutation that improved the fitness of the parse tree. Crossover impacted structure three out of four times as well and involved both small and large sections of genetic information being exchanged. Crossover is also primarily responsible for passing on larger structures of value. Mutation plays a minor role in tweaking structures exchanged through crossover to improve their value to the structure and to discover the blocks essential to the formulation of the optimal tree.

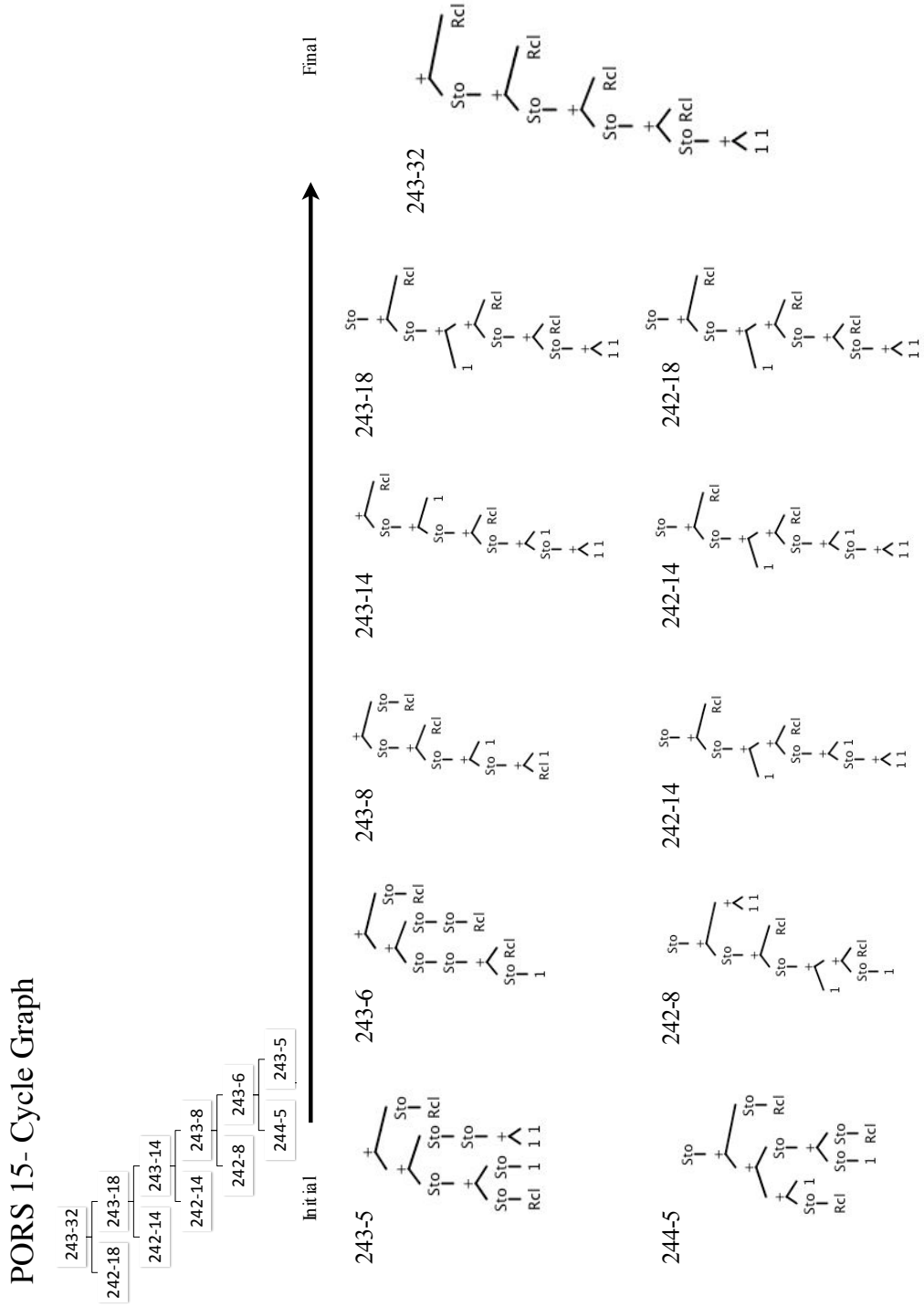


Figure 4.2. The phylogenetic tree of the member with the best fitness for PORS 15 when solved with a cycle graph.

Figure 4.2 provides another example of solution formulation for the PORS 15 efficient node use problem using a cycle graph. In a cycle graph, each population member is adjacent to the one that precedes it and the one that follows numerically. This means that each population member only has two options for co-parents chosen through roulette selection. Population member 243 begins with a structure that evaluates to a fitness of 5. This initial structure was generated randomly when the initial population was seeded at the beginning of the run, same as with the complete graph discussed above.

The first mating event that results in the replacement of member 243-5 is with 244-5. Crossover results in 243 receiving a (Sto Rcl) chain in exchange for (Sto (+ 1 1)). Mutation resulted in the left subtree being replaced. This creates a structure for 243 that evaluates to 6 and replaces 243-5. Population member 243-6 experiences modifications again when mating with 242-8. Crossover results in changing the left subtree. Mutation replaces the subtree exchanged below the (+ Sto Rcl) subtree. The combination of these two events results in the structure of 243-8.

Population member 243-8 experiences a large change in structure in the next mating event it has with 242-14. A large portion of genetic information from 242-14 is exchanged a small portion of the preexisting left subtree of 243-8. The structure at this point in time has more than 15 nodes and needs to be chopped. This chopping removes the top most tree bringing the essential building block of (+ Sto Rcl) to the top of the parse tree of 243. There are no modifications made to this structure through mutation. The resulting structure of 243-14 replaces 243-8.

The structure of 243 has a significant change resulting from a mating event 242-14, the same structure that it produced a fit child with in the mating event before. Again

information is exchanged through crossover that results in a structure of 243 that has too many nodes. These nodes are then chopped with removes the essential building block from the top of the structure of 243. However, mutation changes a small right hand leaf from a 1 to a Rcl. The structure now has a fitness of 18 and replaces 243-14. The bottom half of the structure resembles the bottom half of the optimum PORS tree.

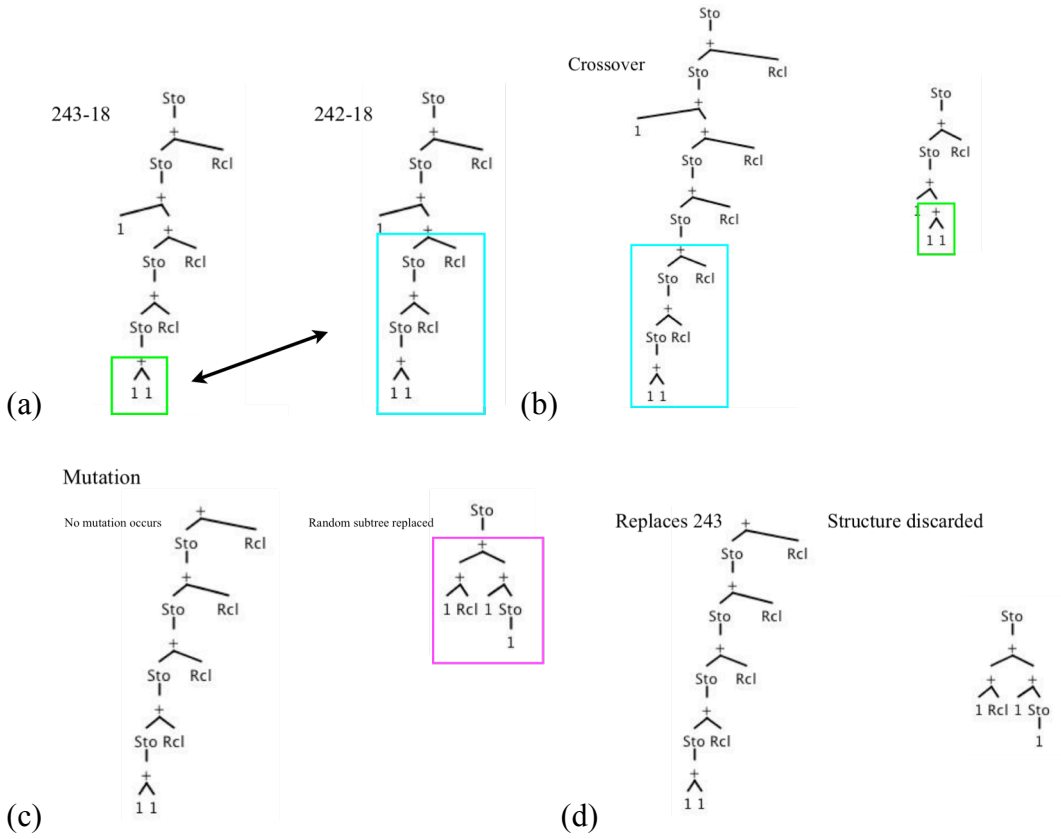


Figure 4.3. Mating event in a cycle graph solving the PORS 15 problem.

The last mating event between 243-18 and 243-18 is depicted in detail in Fig 4.3. As shown in (a) and (b) the lower portion of 242-18 was exchanged and built the optimum PORS tree at the bottom of the structure of 243. Again, there are too many nodes and the top subtrees are removed as per the chopping rules after crossover

occurred. This results in the optimum structure formulation in (c). Mutation does not occur in (d) for 243 and the structure is left unaltered leading to the convergence of the algorithm because a structure with the optimum fitness of 32 has been found.

Population member 243 experiences significant changes as it evolves and all mating events had at least two of the main methods of structure alteration used. Chopping is particularly prevalent in this particular example. Five constructive mating events involving the replacement of 243 were necessary to find the optimum structure.

PORS Nodes	Mutation rate in %	Graph	Mean mating events to solution	Mean number of constructive mating events	Non-constructive per constructive mating events	Failures
15	10%	Complete	912,614 (+/-5.88)	2,749 (+/-0.38%)	332.0	785
		Hypercube	360,751 (+/-9.1%)	2,651(+/-.37%)	136.06	276
		Cycle	71,509 (+/- 1.21%)	2,522 (+/-0.35%)	28.36	0
	0%	Complete	1,378,770 (+/-9.7%)	2,749 (+/-0.97%)	501.6	1,359
		Hypercube	596,338 (+/-15%)	2,130 (+/-1.0%)	279.97	574
		Cycle	243,453 (+/-22%)	2,101 (+/-1.1%)	115.87	197
16	10%	Complete	24,155.4 (+/-0.75)	1,937 (+/-0.36)	12.47	0
		Hypercube	25,530.4 (+/-0.76%)	1,970 (+/-0.35%)	12.96	0
		Cycle	30,396.8 (+/-0.86%)	2,060 (+/-0.34)	14.76	0
	0%	Complete	12,743.5 (+/-1.4%)	1,592 (+/-0.83%)	8.01	0
		Hypercube	14,055.8 (+/-0.71%)	1,625 (+/-0.40%)	8.65	0
		Cycle	22,459 (+/-1.02%)	1,749 (+/-0.86)	12.84	0

Table 4.1. Constructive mating events for PORS 15 and 16 averaged over 10,000 runs for 0% and 10% mutation rates.

As shown in table 4.1 the mean mating events to solution for the PORS 15 problems with 10% mutation rate match the results from Corns thesis. The mean mating events to solution with a 0% mutation rate for the PORS 15 problems increased for every graph. The complete graph almost doubled incidents of failure and took 1.5 times as many mating events to converge with a 0% mutation rate. The hypercube graph almost doubled its failures as well and took 1.65 times more mating events to converge with 0%

mutation. In the case of the cycle graph, the amount of failures went from 0 to 197 and took 3.4 times more mating events to converge without mutation. However, there is a significant amount of error in the results from the cycle graph. This is due to the increase in failures in 0 failures with 10% mutation to 179 at 0% mutation. Mutation rate has a significant impact on convergence time for PORS 15. Without mutation, solving PORS 15 requires more mating events and results in more cases of failure.

The mean number of mating events to solution in Table 4.1 for the PORS 16 problems with 10% mutation rate also match the results from Corns thesis. Mean mating events to solution with a 0% mutation rate significantly dropped for all graphs studied. No failures to converge were recorded in any cases of the runs for PORS 16. Convergence time in the complete graph with 0% mutation rate took 52% of the time in comparison to runs with a 10% mutation rate. Similarly, the hypercube graph took 55% of the time to converge as runs with 10% mutation. The cycle graph also experienced a speedup in convergence, taking 36% less of time to converge with 0% as compared to with 10% mutation. Again, changing the mutation rate significantly impacts convergence time in PORS 16. However, in this case the lack of mutation speeds up convergence time.

Many mating events occur in the process of an evolutionary algorithm, however not all of them lead to the formulation of new structures. Constructive mating events are the backbone of evolutionary algorithms. Without them, the population could not evolve. In the case of the PORS 15 problems with 10% mutation, there is a significant difference in the ratio of constructive mating events to non constructive mating events for all graphs. In the case of the complete graph with 10% mutation rate 1:332 mating events on average was constructive. The hypercube graph had a constructive mating event every 136 mating

events on average. On average, every 28th mating event in a cycle graph resulted in a new structure.

Without mutation, the occurrence of constructive mating events was rarer in the PORS 15 problems. The likelihood of a new structure being introduced into the population is related to the increase in mating events discussed above for PORS 15. On average, it took 1.5 more mating events to obtain a new structure in the compete graph without mutation. The hypercube graph took twice as long to locate new structures without mutation and the cycle graph took four times longer than it did with a 10% mutation rate.

In the case of PORS 16, different trends are observed with constructive mating events. With a 10% mutation rate, the complete graph was most likely to find new structures every 1:12.47 mating events. The hypercube graph followed closely behind with every 1:12.96 mating events resulting in the replacement of an older member. The cycle graph had the worst chance of finding new structures, a 1 in 14.76 chance.

Without mutation, these trends drastically changed when solving the PORS 16 problems. As all graphs took less time to converge with a 0% mutation rate, the number of mating events required for a constructive event decreased as well. New structures were found 38% faster in the case of the complete graph, 33% faster for the hypercube graph, and 13% faster in the cycle graph. Mutation rate has a significant impact on the overall time to convergence and the frequency of constructive mating events. In the case of the PORS 15 problems, not having any mutation retarded both of these metrics of performance. The PORS 16 problems, however, benefited from the lack of mutation.

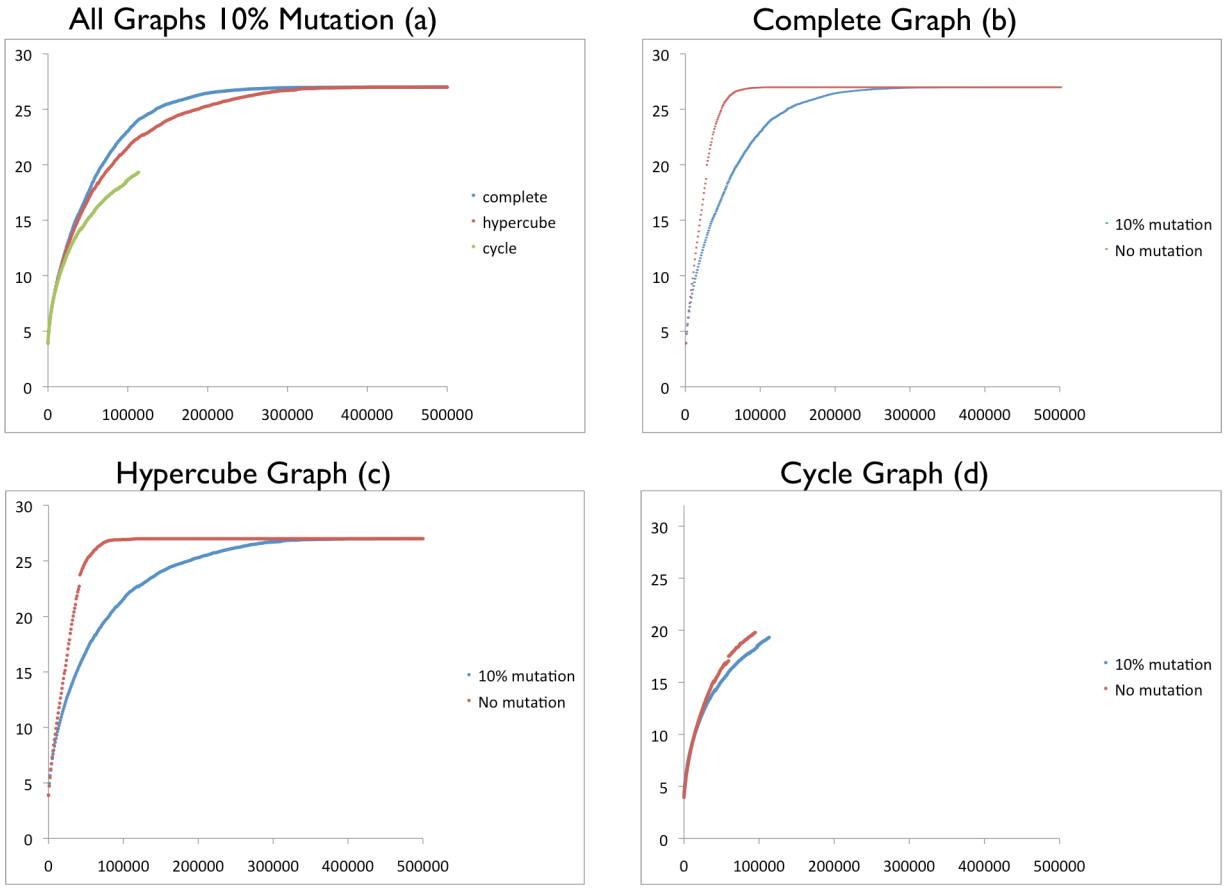


Figure 4.4. Average population fitness for PORS 15 as a function of mating events.

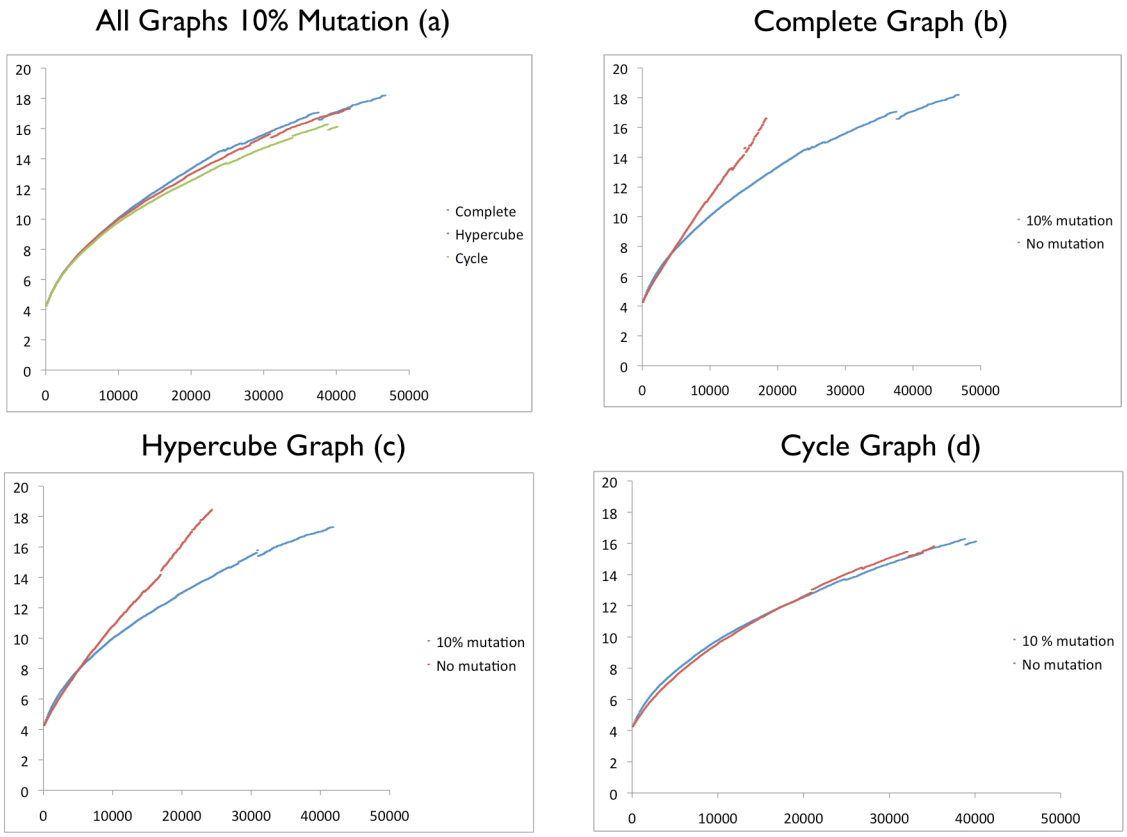


Figure 4.5. Average population fitness for PORS 16 as a function of mating events.

Figure 4.5 contains all of the results regarding the average fitness of the population as a function of mating events for PORS 15. In total, there were sixty runs made to collect data for PORS 15. Each trend line represents ten runs that were averaged to sample how the average population fitness changed as the number of mating events increased within the algorithm, as outlined in chapter 3. For each run, a new population was randomly seeded.

As shown in Fig. 4.5 (a) the graph contains the results of the runs made for the complete, hypercube, and cycle graph with a 10% mutation rate for PORS 15. All graphs begin at zero mating events with roughly the same average fitness. Average population fitness for the complete graph increases at a steady rate until approximately 150,000 mating events have occurred. At this point, the average fitness begins to plateau at 27. All runs for the complete graph with an average fitness of 27 failed. They were included to determine what happens in the population when the algorithm fails to converge. The algorithm fails to converge when the population solely consists of structures that evaluate to 27. As discussed in Chapter 3, these structures cannot contribute to formulating the optimum structure. Failures are defined as they were in the above section, if a parse tree is not found with the optimum fitness value for that number of nodes within 10 million mating events then the run is considered a failure. The hypercube graph climbs in a similar manner to the complete graph initially. However, the average fitness increases at a slower rate just before 50,000 mating events than the complete graph. If the algorithm runs far enough along, the population will plateau at 27. Again, as with the complete graph, if the average population fitness reached 27 the run always failed to converge. The behavior of the average fitness in the cycle graph is significantly different than the

previous two. Average fitness climbs at a much slower rate and never exceeds twenty for the runs that were performed. As seen in Table 4.1, the cycle graph does not have any failures for PORS 15 when run with a 10% mutation rate. The longest run in this case was just over 150,000 mating events, far below the 10 million maximums that the complete graph and hypercube graph had when they failed to converge. The cycle graph with 10% mutation is able to locate structures that will further evolve the population and lead to convergence to the optimal solution.

When PORS 15 was run with a mutation rate of 0%, significant changes in the mean number of mating events till solution for all graphs were observed, as seen in Table 4.1. The average population fitness for the complete graph, Fig 4.5 (b), with a 0% mutation rate climbed significantly steeper than with a 10% mutation rate. The average fitness plateaus at 27 before 100,000 mating events had occurred. Again, when the average fitness reached 27, those runs always failed. The hypercube graph, Fig 4.5 (c), also displays a similar trend to the complete graph with a 0% mutation rate. Average fitness increased much more quickly and a plateau was reached around 75,000 mating events, considerably faster than those runs with the 10% mutation rate. The cycle graph, Fig 4.5 (d), also experienced higher overall average fitness values as the number of runs progressed with a 0% mutation rate. However, when compared to runs made with a 10% mutation rate the shift is not as dramatic as those observed with the complete graph and hypercube graph. Failures are experienced in with these settings for PORS 15, as noted in Table 4.1. Average fitness still does not exceed 20, but no failures were included in the data set graphed, so one could expect a fitness plateau if one of those runs were observed during sampling.

Figure 4.6 contains all of the results regarding the average fitness of the population as a function of mating events for PORS 16. The parameters mentioned above for PORS 15 are the same for PORS 16. Figure 4.6 (a) for PORS 16 contains the results of the runs made for the complete, hypercube, and cycle graph with a 10% mutation rate. All graphs begin at zero mating events with roughly the same average fitness.

Average fitness in the complete graph climbs the quickest of out of the three graphs. It is closely followed by the hypercube graph and the cycle graph comes in last. This is also the order of their average performance for PORS 16, as noted in Table 4.1. The peak average fitness values for the complete graph reach around 28. The peak observed for the hypercube graph is roughly 27 and the cycle graph does not exceed 26. There are no plateaus in average fitness. As noted in the discussion of Table 4.1, no failures were recorded for PORS 16 with either 10% or 0% mutation. When the complete graph, Fig. 4.6 (b), is run with a 0% mutation rate, the average fitness of the population climbs much more steeply than with a 10% mutation rate. It climbs linearly at a mutation rate of 0%, where as at 10% it seems to follow a logarithmic pattern. Average fitness is more than double that of the runs at 10% at the fitness peak of 26 for runs at 0% mutation rate. A similar pattern can be observed with the hypercube graph, Fig. 4.6 (c). However, there is not such a disparity of fitness when the mutation rate of 0% peaks. The hypercube graph with 0% mutation rate also appears to climb linearly and at 10% it seems to be logarithmic. In the case of the cycle graph, Fig. 4.6 (d), there does not seem to be a significant variability for the rate at which average fitness increases as a function of mating event. At 10% and 0% both appear to climb logarithmically and reach a similar peak around 26 at roughly the same time.

Again we observe that the mutation rate has a significant impact in what goes on in the GBEA optimization of PORS 15 and 16 with the complete, hypercube, and cycle graph. Though it did not have an impact on which graph solved the problem most efficiently, it did impact the performance of each graph. A mutation rate of 0% always increased the rate at which the average fitness climbed, with the exception being the cycle graph for PORS 16. At this time there is not a distinguishable difference with the data set sampled for this experiment.

CHAPTER V. DISCUSSION OF RESULTS

As stated in chapters 1 and 2, the goal of this work is to explore the tradeoff between exploration and exploitation in GBEA optimization using the PORS 15 and 16 problems. In Corns thesis, he hypothesized that these mechanisms resulted either in the success of failure of a graph for a particular problem. In this context, exploration refers to preference of preserving population diversity as the solution set evolves. Exploration would allow competing solutions. However, this could increase the time to convergence but ensure that no stone was left unturned. The mechanism of exploitation would seek to burn out the diversity in the solution set quickly. The advantage of exploitation is that it will efficiently drive the solution set towards a solution. However, there is no guarantee that the solution is the right one because other solutions are not given the time to develop.

The population does not evolve without constructive mating events. Yet many mating events do not yield new structures. The results for replacement ratios showed that the graph that arrived to the solution quickest had the highest ratio of constructive mating events. The small standard deviation in the number of constructive mating events for the 10% mutation rate in the cases of PORS 15 and 16 is unexpected. There is a very strong correlation with the number of constructive mating events required to find the solution for a given problem, for a specific graph. The evolution of the population is not a completely random process. Each graph for PORS 15 and 16 has its own rules in regards to the frequency of constructive mating events. For example, the complete graph in PORS 15 has the least chance of any given mating event to result in a new parse tree that further evolves the population. However, this is turned around in PORS 16, where it has

the greatest likelihood of discovering new structures. Mutation rate impacts the likelihood of a mating events occurrence, but does not alter overall performance trends.

The role of mutation in the PORS 15 and 16 problems was unexpected. In PORS 15, the removal of mutation from mating events increased the average number of runs required to solution, failures, and decreased the likelihood of discovering a new structure for any given mating event. In PORS 16, something completely different was observed when mutation was removed from mating. Every graph converged quicker and had a greater likelihood of discovering new structures than they did with the 10% mating rate. Mutation rate serves as a means to prevent structures from being lost as the algorithm progresses and injects new structures into the population. PORS 15 performs best with mutation, as it needs the diversity that it provides to explore the solution space and find the optimum solution. If key building blocks for the only solution are lost, there is no chance for them to be recovered if mutation does not play a role in mating. The higher rates of failure for PORS 15, the increase in time to solution, and the decrease in frequency of constructive mating events points to PORS 15 depending heavily on diversity preservation techniques and preferring the exploration of the solution space. PORS 16 performs best without it because it doesn't require diversity to find an optimum solution, after all, there are 24 of them to find. There is no danger to convergence if there isn't an injection of diversity through mutation. Mutation retards the frequency of constructive mating events and time to solution for the PORS 16 problems.

The average fitness of the population as a function of mating event yields another metric for comparing graph performance for a particular problem. The complete graph in the case of PORS 15 and PORS 16 climbs quickest. This suggests that this is the

preferred behavior of the complete graph. It drives the population towards structures of high average fitness quickly. However, as observed in PORS 15, this is a detriment. Every time the population reached an average fitness of 27, the run failed to converge. As discussed in chapter 3, there are four structures that evaluate to 27 in PORS 15. They are all local optima and they do not have the components necessary to formulate the optimum solution of 32. When the population reaches the average fitness of 27, it consists only of these four structures and therefore can never converge. This explains the high failure rates in the complete graph and the hypercube graphs for PORS 15. However, this behavior is rewarded in PORS 16. There are no local optima for the population to get stuck in, and the faster a graph climbs, the quicker it will locate one of the 24 possible solutions for PORS 16. The absence of mutation increases the rate that the fitness climbs in all cases, except for the cycle graph in PORS 16 where there was no notable difference for this experiment.

Exploration is characterized by permitting competing solutions to evolve and preserving diversity overall so that this goal can be achieved. From this experiment, PORS 15 is best solved with methods that prefer exploration. It requires diversity because there is only one solution and local optima it can get held up on. Based off of this experiment, mutation is a mechanism that preserves diversity. The absence of mutation in PORS 15 can result the loss of critical pieces of the optimum solution, leading to failure. The rate at which average population fitness changes also impacts PORS 15. The cycle graph exhibits a slower change in population fitness in both PORS 15 and 16, pointing to this behavior being due to the graph itself, not the problem. The gentle change in fitness can be seen as the preservation of diversity throughout the population as the run

progresses, and therefore the cycle graph is graph that prefers to explore the solution space. This is also confirmed by the greater frequency of constructive mating events for the cycle graph, despite its lower overall average population fitness.

Exploitation is characterized by the burning out of diversity. It drives the population to the current best solution and does not permit competing solutions the time to evolve easily, as exploration would. In this experiment, PORS 16 is best solved with methods that lead to exploitation of the solution set. Graphs that behaved poorly for PORS 15 excelled when solving PORS 16. The complete graph and the hypercube graph increased in average fitness very quickly as a function of mating event, as they did in PORS 15. However, this behavior led to competing solutions being eliminated and the population driving towards the stronger members. If the average fitness of the population climbs quickly for a particular graph, that graph is a mechanism of exploitation. There was no need to preserve diversity because there were 24 solutions for the algorithm to find and preserving it only retarded the rate of convergence for PORS 16. The complete graph also had a higher frequency of constructive mating events, again proving that the solution was best found through exploitation. When mutation was removed, PORS 16 was solved almost twice as fast. The rate of constructive mating events increased. Mutation is a mechanism of exploration and the fact that PORS 16 performs better without it further supports that PORS 16 is best solved through exploitative methods.

At the end of Corns thesis, a few guidelines for graph selection for a particular problem were provided. These guidelines focused on simple problems. In addition, many of Corns' observations are confirmed through this experiment. Deceptive problems, such as the PORS 15 problem prefer exploration of the solution space. Graphs with large

diameters, such as the cycle graph, promote exploration.. Mutation also plays a key role in exploration as well, as without it essential structures can be lost. This experiment points to the recommendation of graphs with large diameter, slow changes in overall fitness, and mutation to be best suited towards deceptive problems. Non-deceptive problems, such as PORS 16 prefer exploitation. Burning out diversity allows for the quick assembly of solution. Graphs with small diameter, such as the complete graph and hypercube graph, have large changes in average fitness as a function of mating events. These observations confirm Corns' initial guidelines from Chapter 2.

The results of this thesis demonstrate that exploration and exploitation play a vital role in the solution of PORS 15 and PORS 16 problems with GBEA optimization. Exploration and exploitation can be mediated by choice of graph and by the mutation rate. The above guidelines regarding deceptive problems add additional support to Corns thesis regarding the selection of a particular graph and mutation rate for a desired problem.

CHAPTER VI. CONCLUSIONS AND FUTURE WORK

The goal of this work was to identify the roles of exploration and exploitation in the solving of the PORS 15 and 16 efficient node use problems with GBEA optimization. This work was built off Steve Corns thesis about the rate of information transfer in GBEAs in relation to the graph diameter used. This research used various metrics to evaluate constructive mating events and the average population fitness as a function of mating event, and how mutation rate impacts these trends. The frequency of constructive mating events was directly tied to the success of a particular graph for a given problem solved with GBEA optimization. Graphs whose average fitness climbed quickest are mechanisms of exploitation. The graphs shown in this paper to exhibit this behavior, the complete and hypercube graph, found solutions quickly for simple problems but their exploitive behavior burned diversity too quickly from the population and lead to higher instances of failure and lower instances of constructive mating events for problems that are deceptive. Graphs with a slower climb in average fitness were exploratative in nature. This proved to be a boon when solving deceptive problems, as it was insured that structures essential to the solution were not lost. This exploratative nature retarded convergence for simple problems, as they prefer to burn diversity rather than preserve. Mutation rate seems to be directly related to exploration. The higher the mutation rate, the more exploration will occur in the solution set as diversity can be maintained and injected as the algorithm progresses. The complete absence of mutation promotes exploitation.

While this work yields proves that exploration and exploitation can be manipulated by graph choice in GBEA optimization, it remains to be seen if these guidelines can be applied to problems that have not been solved by GBEAs before and predict what kind of graph is best suited towards a particular problem. Future areas of work would be analyzing another set of deceptive and non-deceptive problems in the same family to see if the same trends observed for the PORS problems hold true. In addition, determining the extent at which mutation operates in the tradeoff between exploration and exploitation would be valuable for determining additional parameters that can be tweaked. This work also focused on graphs that had either the smallest diameter or the largest diameter. More graphs need to be classified, including those that have diameters in between those used for this experiment, to further this research.

REFERENCES

Alba, E. & Cotta, C. (2006). “Evolutionary Algorithms.” In Olariu, Stephan and Zomaya, Albert Y. (Eds.), *Handbook of Bioinspired Algorithms and Applications (pp. 1.3-1.19).* Boca Raton, FL*:* Chapman & Hall/CRC.

Ashlock, D.A. & Lathrop, J.I. (1998). “A fully Characterized Test Suite for Genetic Programming,” *Evolutionary programming VII*, 537-546. Springer, New York.

Ashlock, D.A. (2005). Evolutionary Computation for Modeling and Optimization. Springer-Verlag New York, Inc., Secaucus, NJ.

Bryden, K.M, Ashlock, D.A., McCorkle, D.S., & Urban, G.L. (2002). “Optimization of Heat transfer using Graph Based Evolutionary Algorithms,” International Journal of Heat and Fluid Flow, 24(2), 267-277.

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

Cantú-Paz, E (1998). “A survey of parallel genetic algorithms,” *Calculateurs Parallèles, Réseaux et Systèmes Répartis*, vol. 10, no. 2, pp. 141–171, 1998.

Corns, S.M (2008). *The Role of Information Flow in Engineering Optimization.* PhD. Iowa State University, Ames, IA

Garcia-Martinez, C & Lozano, M. (2005). “Hybrid real-coded genetic algorithms with female and male differentiation”, IEEE Congress on Evolutionary Computation, vol 1 pp 896-903, 2005.

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

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

McCorkle, D.S., Bryden, K.M., Charmichael, C.G. (2003). “A New Methodology for Evoluiotnary Optimization of Energy Systems,” Computer Methods in Applied Mechanics and Engineering, 192(44-46):5021-5036 (2003)

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

Tannehil, J.C., Anderson, D.A. & Pletcher R.H. (1997*).* Computational Fluid Mechanics and Heat Transfer 2nd Edition. 2nd Edition, US: Taylor and Francis.