Summer 2010

Improving resiliency using graph based evolutionary algorithms

Jayakanth Jayachandran

Department:

IMPROVING RESILIENCY USING GRAPH BASED EVOLUTIONARY ALGORITHMS

by

JAYAKANTH JAYACHANDRAN

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

2010

Approved by

Steve Corns, Advisor
Suzanna Long
Scott Grasman

ABSTRACT

Resiliency is an important characteristic of any system. It signifies the ability of a system to survive and recover from unprecedented disruptions. Various characteristics exist that indicate the level of resiliency in a system. One of these attributes is the adaptability of the system. This adaptability can be enhanced by redundancy present within the system. In the context of system design, redundancy can be achieved by having a diverse set of good designs for that particular system. Evolutionary algorithms are widely used in creating designs for engineering systems, as they perform well on discontinuous and/or high dimensional problems. One method to control the diversity of solutions within an evolutionary algorithm is the use of combinatorial graphs, or graph based evolutionary algorithms. This diversity of solutions is key factor to enhance the redundancy of a system design. In this work, the way how graph based evolutionary algorithms generate diverse solutions is investigated by examining the influence of representation and mutation. This allows for greater understanding of the exploratory nature of each representation and how they can control the number of solution generated within a trial. The results of this research are then applied to the Travelling Salesman Problem, a known NP hard problem often used as a surrogate for logistic or network design problems. When the redundancy in system design is improved, adaptability can be achieved by placing an agent to initiate a transfer to other good solutions in the event of a disruption in network connectivity, making it possible to improve the resiliency of the system.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Steve Corns who was instrumental during my entire course of my collegiate period. Also, I would like to thank my other members of the committee, Dr. Suzanna Long and Dr. Scott Grasman for their support.

TABLE OF CONTENTS

Page

ABSTRACT iii
ACKNOWLEDGMENTS iv
LIST OF ILLUSTRATIONS vii
LIST OF TABLES ix
SECTION
1. INTRODUCTION 1
2. EVOLUTIONARY ALGORITHMS 3
2.1. BACKGROUND 3
2.2. EVOLUTIONARY ALGORITHM PARAMETERS 5
2.2.1. Population Size 5
2.2.2. Models of Evolution 5
2.2.1. Crossover 6
2.2.2. Exploration and Exploitation 6
2.2.3. Mutation 7
2.2.4. Representation 7
2.3. GRAPH BASED EVOLUTIONARY ALGORITHMS 7
2.3.1. Motivation 7
2.3.2. Background 9
2.3.3. Taxonomy 10
3. COMPUTATIONAL EXPERIMENTS 11
3.1. BACKGROUND 11
3.2. EXPERIMENTAL DESIGN 12
3.3. EXPERIMENTAL RESULTS 15
3.3.1. Experiment-I 15
3.3.1.1 Ackley path function 15
3.3.1.2 Dropwave function 18
3.3.1.3 Rastrigin function (4 variables) 20

3.3.1.4 Rastrigin function (six variables) 22
3.3.1.5 Schwefel function 24
3.3.1.6 Shubert function 25
3.3.1.7 Cladogram 27
3.3.1.8 Summary 29
3.3.2. Experiment-II 30
3.3.2.1 Ackley path function 30
3.3.2.2 Dropwave function 32
3.3.2.3 Rastrigin function (4 variables) 34
3.3.2.4 Schwefel function 36
3.3.2.5 Shubert function 38
3.3.2.6 Summary 41
4. IMPROVING TRAVELLING SALESMAN PROBLEM SOLUTION DIVERSITY USING GRAPH BASED EVOLUTIONARY ALGORITHMS 42
4.1. BACKGROUND 42
4.2. EXPERIMENTAL DESIGN 43
4.3. EXPERIMENTAL RESULTS 46
4.3.1. KroA 47
4.3.2. KroC 48
4.3.3. Summary 49
5. CONCLUSIONS AND FUTURE WORK 50
APPENDIX 53
BIBILOGRAPHY 56
VITA 63

LIST OF ILLUSTRATIONS

Figure Page

2.1 Illustration of graphs used in the study 8
3.1 Ackley path function, gray encoding 16
3.2 Ackley path function, binary encoding 17
3.3 Ackley path function, real encoding 17
3.4 Dropwave function, gray encoding 18
3.5 Dropwave function, binary encoding 19
3.6 Dropwave function, real encoding 19
3.7 Rastrigin function (four dimensions), gray encoding 20
3.8 Rastrigin function (four dimensions), binary encoding 21
3.9 Rastrigin function (four dimensions), real encoding 21
3.10 Rastrigin function (six dimensions), gray encoding 22
3.11 Rastrigin function (six dimensions), binary encoding 23
3.12 Rastrigin function (six dimensions), real encoding 23
3.13 Schwefel function, gray encoding 24
3.14 Schwefel function, binary encoding 24
3.15 Schwefel function, real encoding 25
3.16 Shubert function, gray encoding 26
3.17 Shubert function, binary encoding 26
3.18 Shubert function, real encoding 27
3.19 Cladogram 28
3.20 Type-I Ackley path function, mutation 50 31
3.21 Type-II Ackley path function, mutation 200 31
3.22 Type-II Dropwave function, mutation 50 33
3.23 Type-II Dropwave function, mutation 100 33
3.24 Type-II Dropwave function, mutation 100 34
3.25 Type-I Rastrigin function (four dimensions), mutation 50 35
3.26 Type-I Rastrigin function (four dimensions), mutation 150 35

3.27 Type-I Schwefel function, mutation 50 37
3.28 Type-I Schwefel function, mutation 100 37
3.29 Type-I Schwefel function, mutation 100 38
3.30 Type-I Shubert function, mutation 50 39
3.31 Type-I Shubert function, mutation 100 39
3.32 Type-I Shubert function, mutation 150 40
3.33 Type-I Shubert function, mutation 200 40

LIST OF TABLES

Table Page

3.1 Graphs used in experiment I 14

3.2 Graphs used in experiment II 15

4.1 Graphs used in TSP experiment 44

4.2 Results of KroA 47

4.3 Results of KroC 48

1. INTRODUCTION

A system can fail when it faces unprecedented events that lead to its disruption of normal activities. Resiliency is a characteristic of a system which can provide the necessary elements to manage these disruptive events. Basic definition of resiliency is "the capacity of a system to tolerate disturbances while retaining its structure and function" [71]. Resiliency in a system is composed of three elements: accident avoidance, survival, and recovery. Certain elements are placed in a system that may avoid the occurrence of accidents. If it fails, the system can be designed to survive and may be able to recover from the disruption. Disruptions are of two types, type A or disruption of input and type B. Disruptions of input are disruptions which are caused by external random phenomenon. Disruptions due to change in environments are type A. When the disruption is systematically conceived is type B disruptions. Generally technical problems are categorized as type B. One of the important characteristic in achieving resiliency is adaptability. It is the capability of the system to adapt to unforeseen changes in the operating conditions. Apollo 13 is a good example of adaptability. Once the accident happened the crew of Apollo 13 adapted to manage on low power and survive. Redundancy is another attribute which helps in increasing adaptability. Redundancy is defined as multiple ways of performing a same function. A heuristic known as the functional redundancy heuristic states that there must be multiple ways to perform critical functions. In case of a disruption, the system can have the opportunity to perform those critical functions. In biological systems a high level of redundancy exists. Millions of cells perform identical functions. Also cells are generated and produced continually making loss of individual cells little difference [70]. These characteristics help to improve the resiliency of the system.

Evolutionary Algorithms are widely used optimization technique, also used in designing a system. EAs mimic the natural evolution from a solution population through computer simulation. It is highly used in the optimization problems from various fields like biology, art, mathematics, physics, and engineering design. The popularity of EAs is due to their flexibility, self adaptive and parallel search properties. Researchers like Goldberg [10], Maher [65] have used EAs as an exploratory tool in conceptual design.

Gero et al. [66], Parmee et al. [67] used EAs searches which progresses outside the initial design variable limits. Some of the applications of EAs include control system design [68], transportation problems and job scheduling and many more applications are discussed in [69].

An artificial geography can be imposed on the population of evolutionary algorithms using combinatorial graphs to control the rate at which information is shared during the process of the algorithms. This novel approach is referred as graph based evolutionary algorithms (GBEAs). This restriction on the mating behavior improves the diversity of solution in the population. The diversity in the population helps in avoiding local optima in deceptive problems and also generating diverse solutions. The diverse solutions created by GBEAs are a key to improve redundant solutions. Even though in some problems we can find an exact optimal solution, to improve redundancy multiple solutions are needed. The ability to provide a collection of good solutions for a particular system is invaluable when future system conditions are uncertain. The diverse solutions created by the GBEAs are a key to improving the redundancy.

The organization of the thesis is as follows. Section two contains information about evolutionary algorithms, their history and development. It also contains an overview of previous research completed in EA and its parameters along with the fundamentals of GBEAs, its characteristics and taxonomical properties are explained. Section three comprises of two computational experiments with their respective results. Experiment one studies the impact of representation on GBEAs and experiment two analyzes the dynamics of diversity. Section four contains the experiment conducted better understand the results of experiments one and two by applying it on traveling salesman problem. Section five summarizes conclusions and future work. In the appendix, an overview of graph theory and description of graphs are included.

2. EVOLUTIONARY ALGORITHMS

2.1. BACKGROUND

One of the ways to interpret the term 'evolutionary system' is to apply the Darwinian Theory of Evolution. Populations of individuals competing for limited resources, dynamically changing populations due to birth and death of individuals, an idea of fitness to each individual and variational inheritance are the main components which embraces Darwinian Theory of evolution. One of the earliest notions of evolutionary system can be seen in Friedman [1], where evolutionary mechanisms are suggested as a means of evolving control circuits for robots. In 1957, Box [2] uses a technique called evolutionary operation for improving industrial process. During the 1960s, the availability of digital computers for use as a modeling and simulation tool influenced the scientific community to use a simple idea such as evolutionary models for complex problem solving. Most of the current work in evolutionary algorithms can be traced to three strongly related but independently developed approaches: genetic algorithms, evolutionary strategies and evolutionary programming.

Holland used evolutionary processes in design and implementation of robust adaptive systems which deals with uncertain environment [3, 4]. These papers composed the fundamentals of "simple genetic algorithms" and subsequently studied by De Jong [5, 6, 7, 8], Goldberg [9, 13]. Rachenberg [14, 15] and Schwefel [16, 17] systematically approached on using evolutionary processes to solve difficult real-valued parameter optimization problems developed the basis of "evolutionary strategies". It is extended by Herdy [18], Kursawe [19]. Evolutionary programming, introduced by Fogel [20, 21] and broadened by Burgin [22, 23], Atmar [24] and Fogel [25, 26, 27] was initially attempted to create artificial intelligence. It was attempted to evolve finite state machines to predict events on the basis of earlier observations.

Evolutionary algorithms (EAs) are a powerful optimization technique following survival of the fittest. To understand the theory of evolution some basic definitions in biology are required. Chromosomes are thread like structures containing Deoxyribonucleic acid (DNA). DNA is a genetic material present in all living organisms and some viruses and a gene is a sequence of DNA. A gene may exist in alternate forms

that determine the expression of some particular characteristic (e.g. eye color, height). These forms are called alleles. Evolution is defined as the variations of allele frequencies in population over time. Here allele frequency means proportion of different alleles of a particular gene in a given population. So, if a creature is born or dies, the allele frequencies in the population change. For more details on molecular biology refer to [28].

Evolutionary algorithms are a stochastic search algorithm operating on a collection of randomly generated data structures (or *creatures*) referred to as the *population*. The population contains candidate solutions with explicitly computed fitness values. The *fitness* value is calculated using a fitness function, which is a measure of the quality of the solutions found by the heuristic. New solutions (*children*) are generated by blending (referred to as mating) existing individual data structures (*parents*), referred to as mating. The fitness values are used for replacement schemes, using newly found solutions to replace population members chosen randomly or with a bias based on fitness of that solution. A sample evolutionary algorithm is shown below.

Create an initial population.
Evaluate the fitness of the population.
Repeat
  Select pairs from the population to be parents, with a fitness bias.
  Copy the parents to make children.
  Perform crossover on the children (optional).
  Mutate the resulting children (probabilistic; optional).
  Place the children in the population.
  Evaluate the fitness of the children.
Until Done.

The parameters which influence the initialization of an EA are population size and representation. Representation is the data structure used and the crossover and mutation operators, called the variation operator when taken together. Crossover, mutation, and selection method mimic natural evolution, and are usually described as crossover rate and type, mutation rate and type, and the technique of how better offspring are passed on to next generation. These parameters within an EA influence the rate and

nature of convergence. The parameters are population size, representation of data structures, crossover, and mutation.

2.2. EVOLUTIONARY ALGORITHM PARAMETERS

2.2.1. Population Size. One of the earliest studies on population size by De Jong [5]. The results indicated the EAs achieved good performance when the population size is between 50 and 100. In the later part of 1980's and 1990's the size of the population selected was dependent on the perceived complexity of the problem [29] [12]. Smith proposed an algorithm which adjusts the population size with respect to the probability of selection error [30]. In early research, the population size for an algorithm is specified before running the algorithm until 1994, when the idea of variable population size was introduced [31]. This work introduced Genetic Algorithms with Varying Population Size (GAVaPS) which does not use any variation of selection. This algorithm applies the concept age of a chromosome, which is equivalent to the number of generations the chromosome stays alive. Thus, the age of chromosome replaces the concept of selection and it depends on the fitness of individual, influences the size of the population at every stage of the process. In GAVaPS lifetime of all individuals in the population is decreased at each generation. In 2000, Adaptive Population Size (APGA) was introduced by Back et al., where the life time of the fittest individual in each generation remains unchanged [62].

2.2.2. Models of Evolution. The technique followed in selecting parents and inserting children back into population is collectively called as models of evolution. Within a population, recombination between individuals is permitted. This information sharing is one of the reasons behind the creation of similar individuals causing a diversity loss or genetic drift. This diversity loss may result in convergence to a suboptimal solution in many types of problems. There are several techniques for selection of parent solutions, with the only requirement being that the method should be biased towards more fit individuals. Some of the popular selection methods include tournament selection where the population is shuffled randomly and divided into small groups where the fit individuals chosen to be parents and the resulting children replace the least fit individuals

In fitness proportional selection, parents are chosen in direct proportion to their fitness. In rank selection, individuals are ordered by ranks based on fitness and selected proportionately on the basis of that rank. A child insertion method is needed to insert children back into population. One method is to place the children in population at random (random replacement.) If the individuals are replaced with a probability inversely to their fitness is called fitness proportional replacement. In rank replacement, individuals are ranked opposite to rank selection and chosen to replace proportionately on the basis of ranks. Another method is to replace parents only if the children are more fit, referred to as elite replacement method.

2.2.1. Crossover. Sharing of material (or information) between data structures is referred to as crossover. One popular crossover technique is single point crossover [32], where a single point for crossover is selected uniformly at random along the length of the parent strings. An offspring is then generated by copying the first parent string until the crossover point is reached, then copying the second parent string from the crossover point to the end. Multiple point crossover can be performed by selecting two or more crossover points. Uniform crossover [32] works on each offspring gene independently, making a random choice as to which parent it should inherit information from. Crossover rate is the frequency with which the crossover operator is applied. Grefenstette [33] studied crossover types and rates as part of a study on evolutionary algorithm parameters. Kurusawe [34] showed the appropriate choice of crossover operator depends on the objective function, topology, and dimension of the objective function. Goldberg [29] showed that crossover rate highly influenced the diversity preservation in the population while studying the effects of altering evolutionary algorithm parameters.

2.2.2. Exploration and Exploitation. Exploration and exploitation are vital elements in problem solving by search. Exploration is provided by search operators (recombination and mutation) and exploitation achieved through selection. In graph based evolutionary algorithms by restricting the choices of co-parent exploration is emphasized more and selecting the best fit individual improves the exploitation. Improving resiliency in designing a system can be achieved by generating diverse designs for a same system. It is beneficial to have many novel designs to a system. As sometimes a selected design can be undesirable or unusable in an accident. Exploration helps in

attaining diverse solutions. Various studies about exploration and exploitation have been done over the years, producing many hypotheses about exploration and exploitation, but no general consensus has been reached on the fundamentals of exploration and exploitation [61]. One of the few characteristic which was proved is that exploration and is that exploration and exploitation is highly influenced by the representation of the problem [60].

2.2.3. Mutation. Mutation is the variation achieved by random changes in the data structures, facilitating local search and gradual introduction of diversity in the population. Mutation operators together with crossover operators are called variation operators. The frequency with which mutation is applied is the mutation rate. In binary strings mutation is performed by inverting bits. Some of the earliest work on mutation rate was done by De Jong [5]. Research conducted by Eiben has proven that mutation rates were helpful in improving convergence reliability [35]. There is evidence suggesting that different values of EA parameters might be optimal at different stages of algorithm [63]. One way to achieve this is to use self adaptive mechanisms to control mutation rate [36]. In these instances, varying mutation rates can improve the performance of EAs.

2.2.4. Representation. Representation in an evolutionary algorithm is the structure used together with the choice of variation operators. Classification of evolutionary algorithms is typically based on the choice of the representation. Genetic algorithms use bit strings, real-valued strings are used in evolutionary strategies and the application of expression trees resulted in genetic programming. There are more complex representations such as finite state machines, GP-automata, ISAc lists (if-skip-action). The choice of the data structure can greatly influence the effect of the variation operators.

2.3. GRAPH BASED EVOLUTIONARY ALGORITHMS

2.3.1. Motivation. There is no single measure of diversity in an evolutionary algorithm it can be thought of as a measure of number of different solutions present. Generally, the number of different values present is used as a measure although statistical measures like entropy are also used. In nature, there have been few problems with

diversity loss. There are few theories which explain why there is less diversity in nature. Irrespective of their fitness individuals are separated geographically in nature [37] genetic information. In some, mating provide the necessary diversity giving rise to robust life forms which are required to survive the environment.

In evolutionary algorithms, maintaining a useful diversity is important. In some problems no useful level of diversity is required while in some others a very rich set of diverse solutions may lead to converge in an undesirable local optimum or sometimes consume more computational time. In EAs, loss of diversity is managed through techniques such as high mutation rate, reducing the fitness of a population member in proportion to the other solutions that are essentially the same (niche specialization [32]), implementing memory structures and rejecting duplicate solutions (TABU search [38], [39]).

One way to handle diversity loss is to break the total population down into subpopulations of strings. Each one of these subpopulations could then execute as a normal genetic algorithm. At a predetermined number of generations, the subpopulations swap some solutions. This migration allows subpopulations to share genetic material. By introducing migration the island model is able to exploit differences in the various subpopulations this variation in fact represents a source of genetic diversity. Each subpopulation is an island and there is some designated way in which genetic material is moved from one island to another [64]. Some of the illustrations used in this study are shown in below (Fig. 2.1).

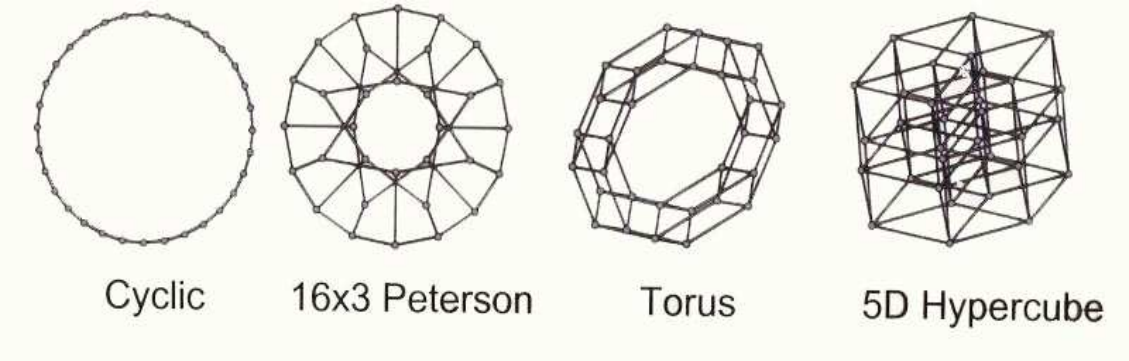

Figure 2.1 Illustration of graphs used in the study

2.3.2. Background. Imposing geography on the population also manages diversity in the population [40]. One method of imposing geography on a solution population is to use graph based evolutionary algorithms (GBEAs) [41].The degree of genetic information sharing within the population is controlled by the choice of combinatorial graphs, thus giving a balance between exploration and exploitation. A novel approach in preserving diversity is using combinatorial graphs which limit the spread of information within the population. A combinatorial graph or graph G is a collection V (G) vertices and E (G) edges where E (G) is a set of unordered pairs from V (G). Two distinct vertices of the graph are neighbors if they are members of the same edge. Degree of the vertex is the number of edges it contains. If all the vertices in the graph have the same degree, the graph is said to be regular. If the common degree of a regular graph is k, then the graph is called k-regular. A graph is connected if one can go from any vertex to any vertex by traversing in a sequence of vertices and edges. The diameter of a graph is the longest that a shortest path between any two of the vertices can be. The diameter is can be defined as the shortest path across the graph. A graph used to constrain mating in a population will be called the population structure. Choose a graph with vertex set V (G) and edges E (G), place an individual in each vertex of the graph G. For a mating, pick a vertex v from V (G) uniformly at random. A neighbor of v is chosen for mating with a fitness bias. Crossover and mutation are used to produce a single individual which may or may not be used to replace the individual with the old individual follows the local mating rule of the GBEA. The local mating rule will pick neighbors in individual with the old individual follows the local mating rule of the GBEA. The local mating rule will pick neighbors in direct proportion to their fitness (fitness proportional selection) and let the new individual replace the old if it is at least as fit as the individual it replaces. For mathematical background and types of graphs refer appendix. GBEAs have shown better performance that a standard evolutionary algorithm in some problems. An example of the use of GBEAs is the design wood-burning stove [42], where convergence time to an acceptable solution was decreased using an appropriate graph [43].

2.3.3. Taxonomy. Taxonomy is the science of classification based on measurable characters, with the resulting categoriation used to provide a conceptual framework of the parameters which conceptual framework of a priori knowledge of the parameters which the parameters which may have a high chances of yielding a better performance. A cladogram is a tree like diagram which shows the relationship between the problems used in the experiment. The data collected in the experiment used to create taxonomy of the problems used. Using the taxonomic characters, hierarchical clustering produces a cladogram that classifies the problems as more or less similar. The method used to construct the cladogram was a clustering technique called the "Unweighted Pair Group Method with Arithmetic mean" (UPGMA), which uses the performance of the graphs to construct a vector and then calculates the Euclidean distance between the problems to show similarity.

The selection of taxonomical character is very important. GBEAs provide taxonomical characters that are computable for any evolutionary computation problem that has a detectable solution. The time to solution varies for each problem and each graph in a complex manner. This complexity gives rise to the taxonomical character. These taxonomic characters are the normalized mean solution times for the problem on each graph. The taxonomy can be used to determine the importance of representation in algorithms, by analyzing the location and grouping of the problems.

3. COMPUTATIONAL EXPERIMENTS

3.1. BACKGROUND

Representation of a problem heavily influences the outcome of the algorithm, having an impact on the exploration and exploitation element of the search. Graph based evolutionary algorithms have been shown to provide taxonomical information on a variety of computational intelligence problems, but they have not been used to examine the differences due to multiple representations of the same problem. This experiment investigates the use of graph based evolutionary algorithms to provide information about the impact of representation on evolutionary algorithms. This information can enable a priori selection of a method that provides better performance when applied to a problem with a similar representation and/or solution search space. This experiment examines five optimization problems using three different representations: binary, gray coding, and real value encodings. The impact of these representations is explored using their performance on graphs as taxonomical characters. Also the study of representations will help in understanding of their explorative nature, as this phenomenon is very important in generating good redundant solutions.

For many evolutionary algorithms a key obstacle to finding the global optima is insufficient solution diversity, causing the algorithm to become mired in local optima. The diversity in solutions can be influenced by algorithm parameters including population size, mutation operator and diversity preservation techniques. A trade off can be seen between the initial diversity of the population size, introduction of new diversity from mutation, and the preservation of diversity from combinatorial graph. With an appropriate fusion of these three factors a level of diversity can be achieved to decrease the time to find the global optima and also to produce more diverse solutions. The trade off can be analyzed by using difference population size and different mutation values for a same problem.

3.2. EXPERIMENTAL DESIGN

The problems used in this study are common test problems from the literature [44]. Ackley path is a minimization problem with a continuous multimodal function obtained by modulating an exponential function with a cosine wave of moderate amplitude. Its topology has a flat outer region and a central hole or peak where the modulation by the cosine wave is more influential. The Dropwave function is a multimodal function with two variables. Its topology is like ripples on water at the outer 'edges' and a hole in the center. The Shubert is a multimodal function with the value equal to the product of summation of two cosine functions of the two variables. It has global optimum towards the center and local optima spreading out from center forming a shape similar to a plus sign. The version of the Schwefel function used here is a four variable, sinusoidal minimization problem. Rastrigin is based on the sum of squares De Jong function with the addition of a cosine function to introduce more local optima, making it more deceptive. Two versions of this problem were examined, one with four dimensions and one with six. All of these problems are highly multimodal and display some amount of deceptive behavior. The stopping criteria for each of these experiments was selected to be the actual optimum value for the Ackley, Dropwave, and Shubert functions, and a value within +/- 0.1% of the optimal solution for the Schwefel and the two Rastrigin functions.

The three representations used are also common methods from the literature; real-valued, binary, and gray coding representations. To generate initial population for all the problems except Ackley path, a string of fours integers is generated, where the first value being 0 or 1 and the rest of the string can contain values between 0 and 9. This string was used to produce an integer from 0 to 1023, with any values outside this range discarded and a new value determined. For Ackley path a string of 5 integers is generated where the first value ranging from 0 to 6 and the rest of the string contains values between 0 and 9. This string was used to produce an integer from 0 to 65535. The binary and gray coding representations used a bit string of length 10 that was evaluated to give an integer value from 0 to 1023, for Ackley path a bit string of 16 is used that was evaluated to give an integer value from 0 to 65535. In this way all of the representations used operated on the same range of integers with the only difference being how the string and its evaluation

represented the problem. The difference between the binary and the gray coding representations is the mapping used when the binary string is translated into an integer. A single point mutation in a binary representation has a much different effect when it occurs at different areas of the string. A single mutation at any location using gray coding has the same effect, making the mutation operator much less disruptive [45]. Two kinds of experiments were designed. For each of these representations single point crossover and single point mutation were used, with mutation flipping a bit for the binary and gray coding, while a value ranging from -200 to 200 selected uniformly at random was added to the real value representation. For the second experiment design only real encoding was used, but variations in mutation value are performed. Four different ranges of numbers are used in this study: -50 to 50, -100 to 100, -150 to 150, and –200 to 200 selected uniformly at random. All of the problems used a range of values from -5.12 to 5.11 for each of the variables of the search space, achieved by subtracting 512 from the integer value and dividing by 100.

For experiment-I, simulations were performed for 6 test problems using three different representations on each of the 15 graphs given in Table 3.1. All the graphs used are of population size 512 except one graph regular tree 510 which has 510 vertices. For each problem, 5000 independent simulations were made and the number of mating events required to find the correct solution was saved for each of these 1,350,000 simulations. If more than 10,000,000 mating events were required, the simulations were recorded as having failed to have found an answer. For each graph and problem, the mean and standard deviation of the number of mating events to solution were used to construct 95% confidence intervals for the mean time to solution. These results were first compared to evaluate graph performance (how many mating events were required to find the solution), and then they were used to determine similarities between the 18 different problem/representation combinations. For experiment-II, simulations were performed on real encoding using the graphs in Table 3.2. This table includes graphs with lower numbers of vertices (8 and 64) along with graphs containing 512 vertices. Simulations were performed on 6 test problems using the four different ranges of mutation values on each of the 20 graphs given in Table 3.2. For each problem 5000 independent simulations were conducted and the number of mating events required to find the correct solution was

recorded for each of these 2,400,000 simulations. If more than 10,000,000 mating events were required, the simulation was recorded as having failed. Again for each graph and problem, the mean and standard deviation of the number of mating events to solution were used to construct 95% confidence intervals for the mean time to solution. Based on these calculations, two types of figures were created; Type I figures contain only graphs with a number of failures below 250, with the number of failures (if any) encountered by the graphs indicated within the brackets. Mean, standard deviation and confidence interval are calculated only for the successful experiments. Type II figures are used to include the results of failed simulations of the graphs on these problems, with the number of mating events for the failed simulation recorded as 10,000,000.

Table 3.1 Graphs used in experiment I

Graph	Index	Regularity	Diameter	Mean Degree
Cycle	C512	2	256	2
Hypercube	H9	9	9	9
Complete	K512	511	1	511
Peterson-1	P256_1	3	129	3
Peterson-3	P256_3	3	46	3
Perterson-7	P256_7	3	22	3
Peterson-17	P256_17	3	18	3
Random Toroid	Rtor07_1	No	19	7.445
Toroid 16,32	T16_32	4	24	4
Toroid 4,128	T4_128	4	66	4
Toroid 8,64	T8_64	4	36	4
Simplexified	Z	4	19	4
RegularTree512,3	RT1n512de	3,1	16	1.996
RegularTree512,4	RT1n512d4	4,1	11	1.996
RegularTree510,5	RT1n512d5	5,1	9	1.996

Table 3.2 Graphs used in experiment II

Graph	Index Name	Vertices	Diameter	Degree
Hypercube3	H3	8	3	3
Cycle8	C8	8	4	2
Complete8	K8	8	1	7
Hypercube64	H6	64	6	6
Cycle64	C64	64	32	2
Complete64	K64	64	1	63
Toroid 8,8	T8_8	64	10	4
Peterson 32,7	P32_7	64	6	3
RegularTree64,3	RT1n64d3	64	10	1.969
Complete510	K510	510	1	509
Hypercube9	H9	512	9	9
Cycle512	C512	512	256	2
Complete512	K512	512	1	511
Toroid 4,128	T4_128	512	66	4
Toroid 16,32	T16_32	512	24	4
Peterson256,7	P256_7	512	22	3
Peterson256,23	P256_23	512	16	3
RegularTree510,5	RT1n510d5	512	9	1.996
RegularTree512,3	RT1n512d3	512	16	1.996
Simplexified	Z	512	19	4

3.3. EXPERIMENTAL RESULTS

3.3.1. Experiment-I. This section contains results for all five problems.

3.3.1.1 Ackley path function. The hypercube graph using gray encoding had the best performance (lowest mean time to solution) when compared to all the other graphs, including binary and gray representations. The gray encoding on the hypercube is followed by random toroid and the worst performance is shown from cycle graph (Fig

3.1). In the case of binary encoding the results are not statistically significant with the exception of the hypercube and complete graphs, which displayed poor performance (Fig. 3.2). All the other graphs have shown fairly similar performance. In real encoding hypercube graph shows the best performance followed by the complete graph (Fig. 3.3). These highly connected graphs are followed by intermediately connected graphs and then sparsely connected graphs.

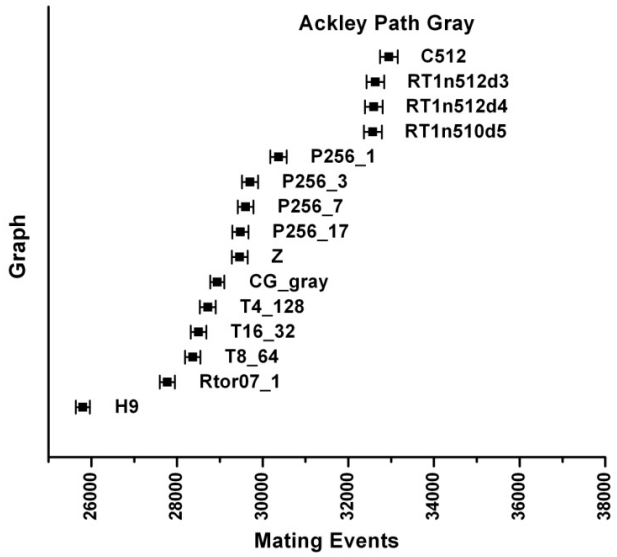

Figure 3.1 Ackley path function, gray encoding

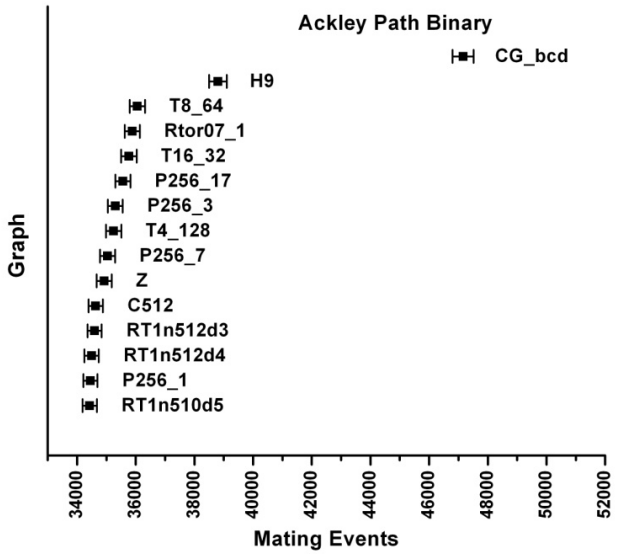

Figure 3.2 Ackley path function, binary encoding

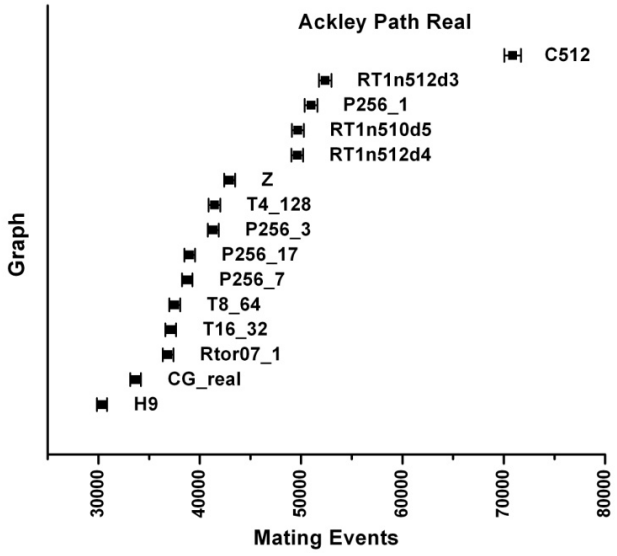

Figure 3.3 Ackley path function, real encoding

3.3.1.2 Dropwave function. In the context of mean time to solution gray encoding performed better than binary and real encodings. In gray encoding highly connected graph hypercube has performed best followed by complete (Fig. 3.5). The pattern followed in gray encoding is highly connected graphs performing well followed by intermediately connected graphs and the worst performing graphs being sparsely connected graphs. In binary encoding, the intermediately connected simplexified graph performed best followed by another intermediately connected graph, the random toroid (Fig. 3.5). Even though intermediately connected graphs performed best, the graphs did not show a clear pattern in performance on the basis of connectivity. Also, to be noted is that the highly connected graphs (hypercube, complete) performed poorly. Real encoding does not give statistically significant results due to very large confidence intervals (Fig. 3.6).

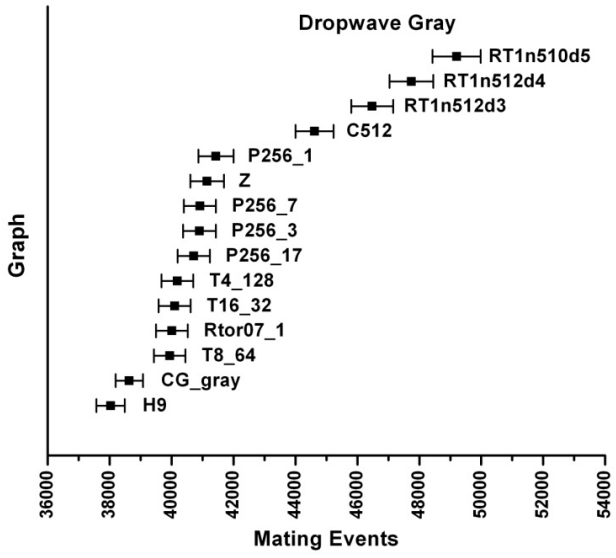

Figure 3.4 Dropwave function, gray encoding

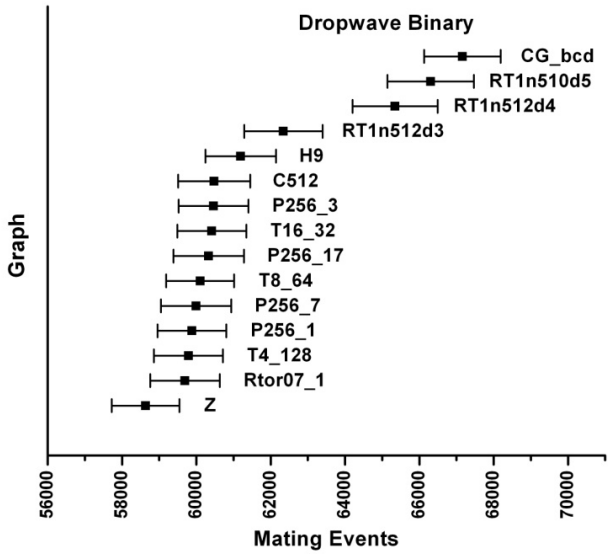

Figure 3.5 Dropwave function, binary encoding

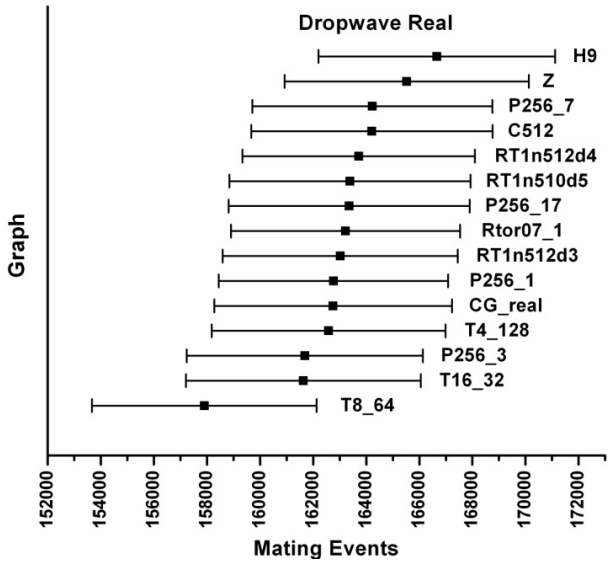

Figure 3.6 Dropwave function, real encoding

3.3.1.3 Rastrigin function (4 variables). In four dimensional Rastrigin function gray encoding has performed best, in which highly connected graphs have performed best. Highly connected graphs are followed by intermediately connected graphs and then sparsely connected graphs (Fig. 3.7). In binary encoding, highly connected graphs have performed badly. But rest of the graphs does not show a clear pattern in performance on the basis of connectivity (Fig. 3.8). Again real encoding provided statistically insignificant results (Fig. 3.9).

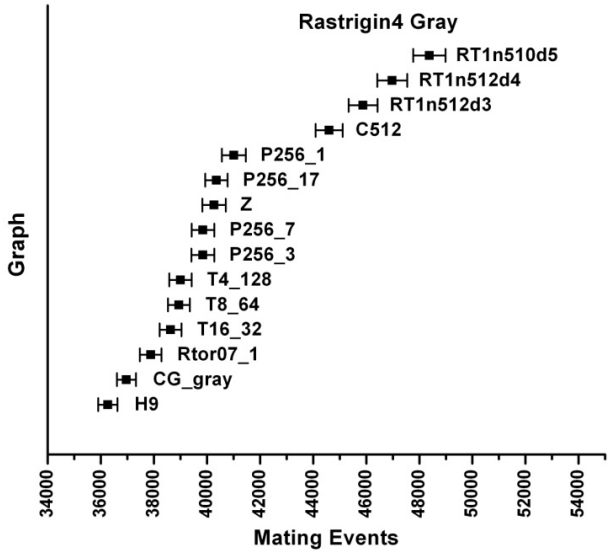

Figure 3.7 Rastrigin function (four dimensions), gray encoding

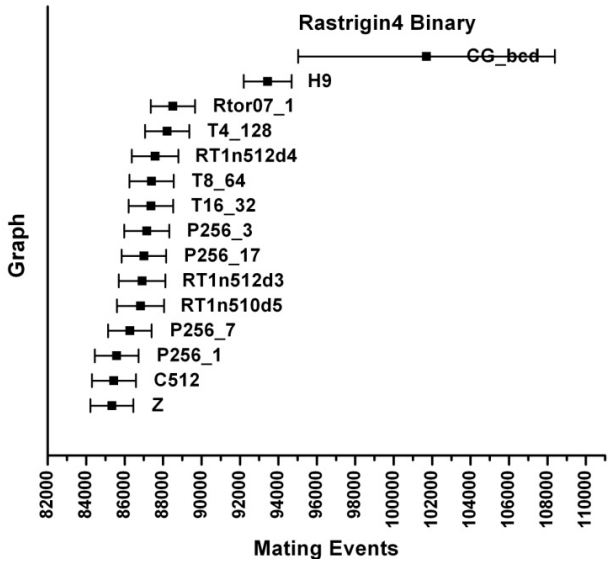

Figure 3.8 Rastrigin function (four dimensions), binary encoding

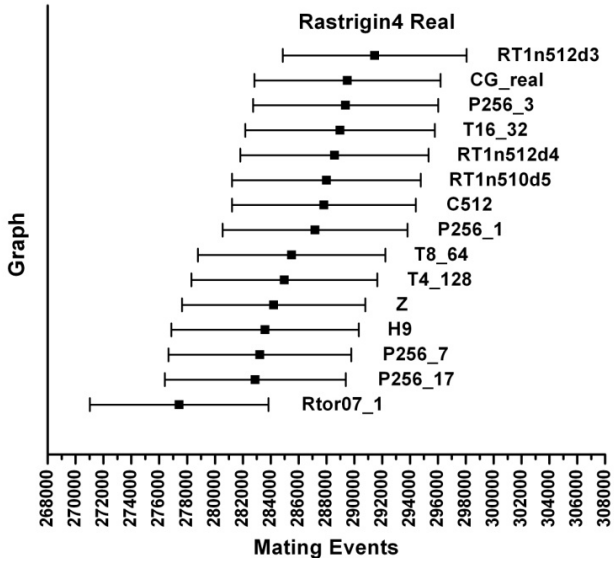

Figure 3.9 Rastrigin function (four dimensions), real encoding

3.3.1.4 Rastrigin function (six variables). In the six dimensional Rastrigin function, again gray encoding performed best and highly connected graphs (such as the hypercube) performed best (Fig. 3.10). One interesting fact here is that the other highly connected graph (complete) had much poorer performance; it is essentially ranked between intermediately connected graphs and sparsely connected graphs. In binary encoding, sparsely connected graphs performed best (Fig. 3.11). Also a clear pattern is visible, where sparsely connected graphs are followed by intermediately connected graphs and then highly connected graphs. In case of real encoding, there are no differences due to very large confidence intervals contributing to statistically insignificant result (Fig. 3.12).

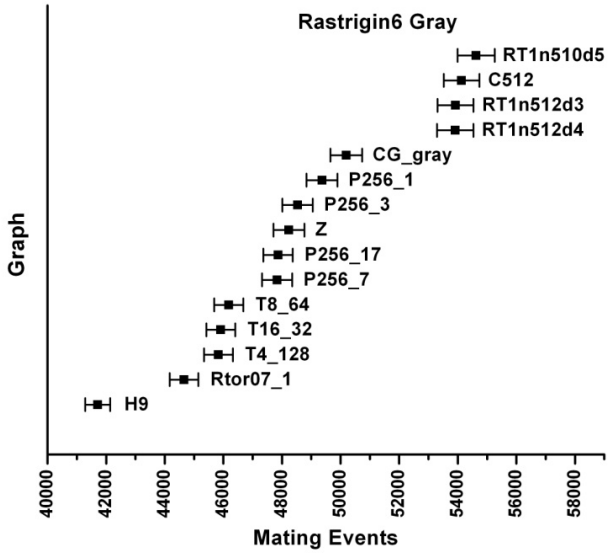

Figure 3.10 Rastrigin function (six dimensions), gray encoding

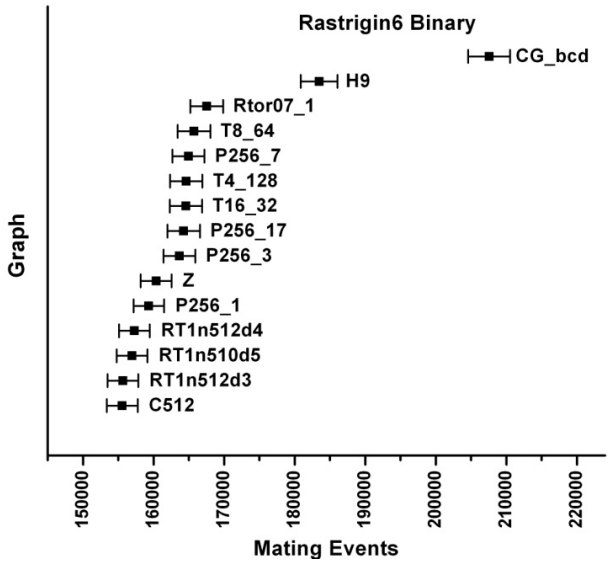

Figure 3.11 Rastrigin function (six dimensions), binary encoding

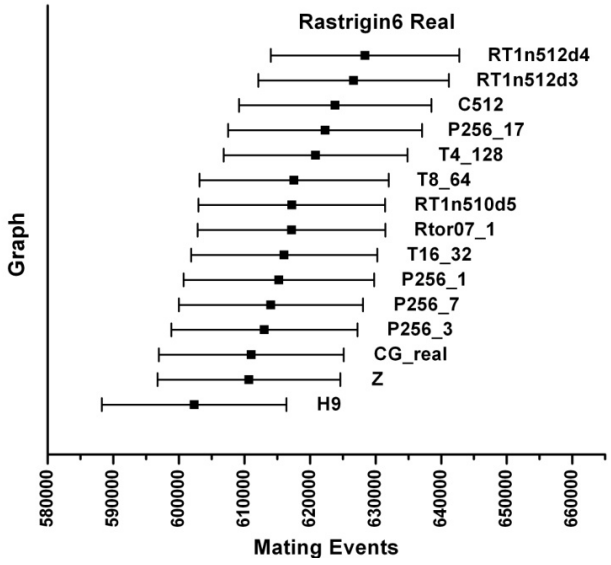

Figure 3.12 Rastrigin function (six dimensions), real encoding

3.3.1.5 Schwefel function. The hypercube graph in gray and real encoding and the random toroid graph in binary encoding are best performers for the Schwefel function. In gray encoding (Fig. 3.13) and real encoding (Fig. 3.14) highly connected graphs are followed by intermediately connected graphs and sparsely

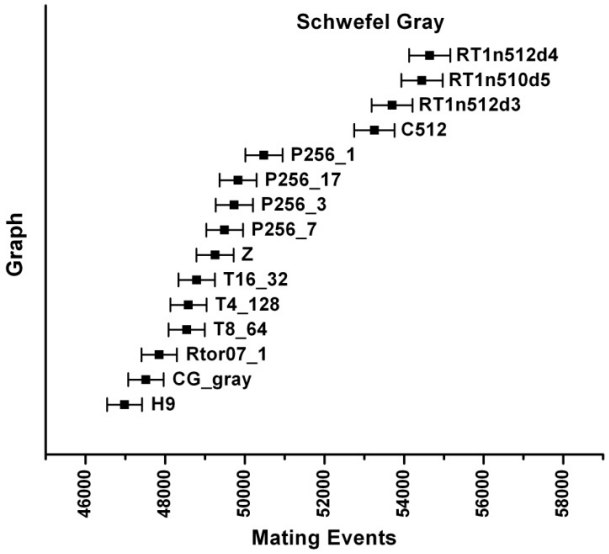

Figure 3.13 Schwefel function, gray encoding

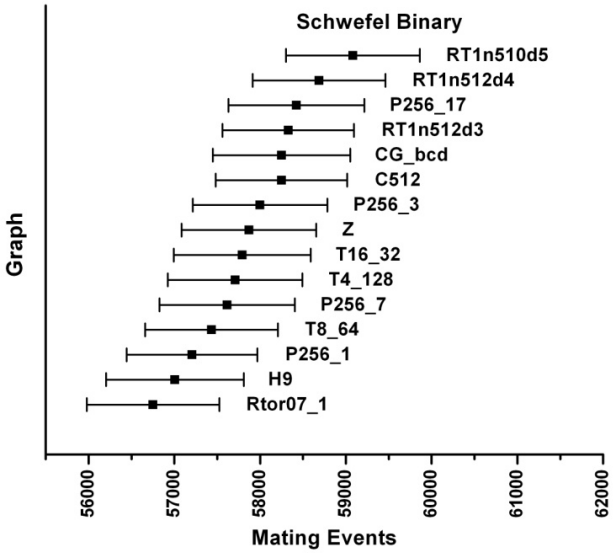

Figure 3.14 Schwefel function, binary encoding

connected graphs. Also sparsely connected graphs are well separated from other graphs. But in binary encoding there is no clear pattern in the graph performance (Fig. 3.15).

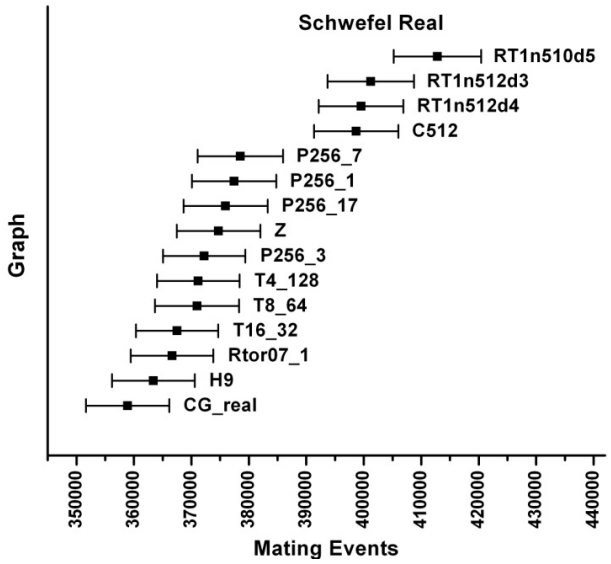

Figure 3.15 Schwefel function, real encoding

3.3.1.6 Shubert function. In the Shubert function the hypercube graph performed best in gray and binary encodings, while in real encoding the random toroid graph performed best. In gray encoding, it is interesting to find the other highly connected graph; complete graph has performed poorly (Fig. 3.16). Similarly in the binary encoding the complete graph performed poorly (Fig. 3.17). In real encoding, both intermediately connected graphs and highly connected graphs performed well and sparsely connected graphs are grouped separately from other graphs (Fig. 3.18).

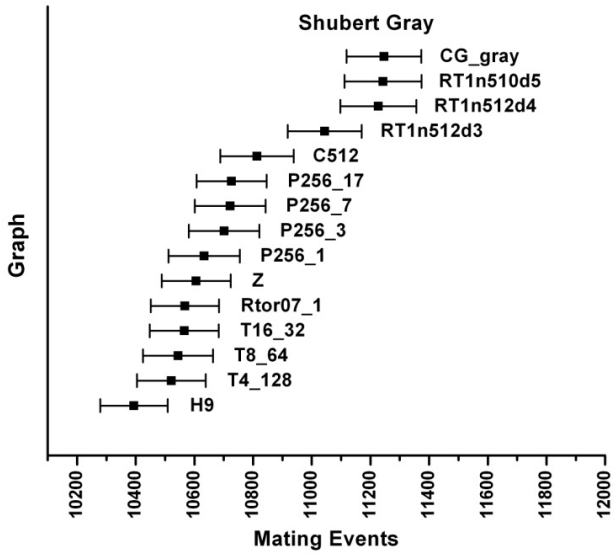

Figure 3.16 Shubert function, gray encoding

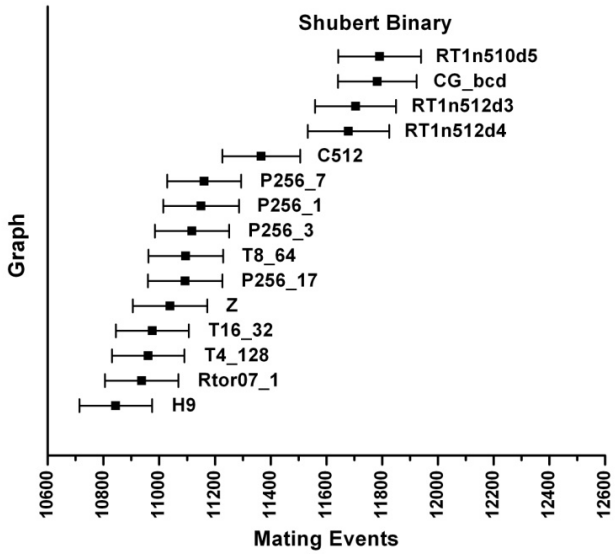

Figure 3.17 Shubert function, binary encoding

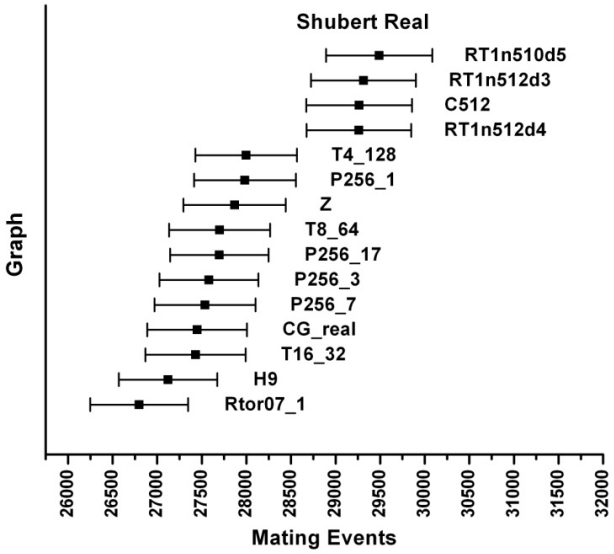

Figure 3.18 Shubert function, real encoding

3.3.1.7 Cladogram. The graph performance information from this work was used as input to the UPGMA algorithm to construct a cladogram (Fig. 3.19) to display similarity between problems. The results of the cladogram give no absolute groupings, as the problems and representations were mixed, but there were some observable trends. There was some clustering by problem type, such as the Shubert function performance in both the binary and gray coding representations. However, the largest clustering was by representation type, as five of the six gray coding representations were located at the far right of the figure. The real valued representations were found to the right of the cladogram, with a larger separation between the problems. Finally, the binary representation problems were mainly spread through the middle section of the cladogram.

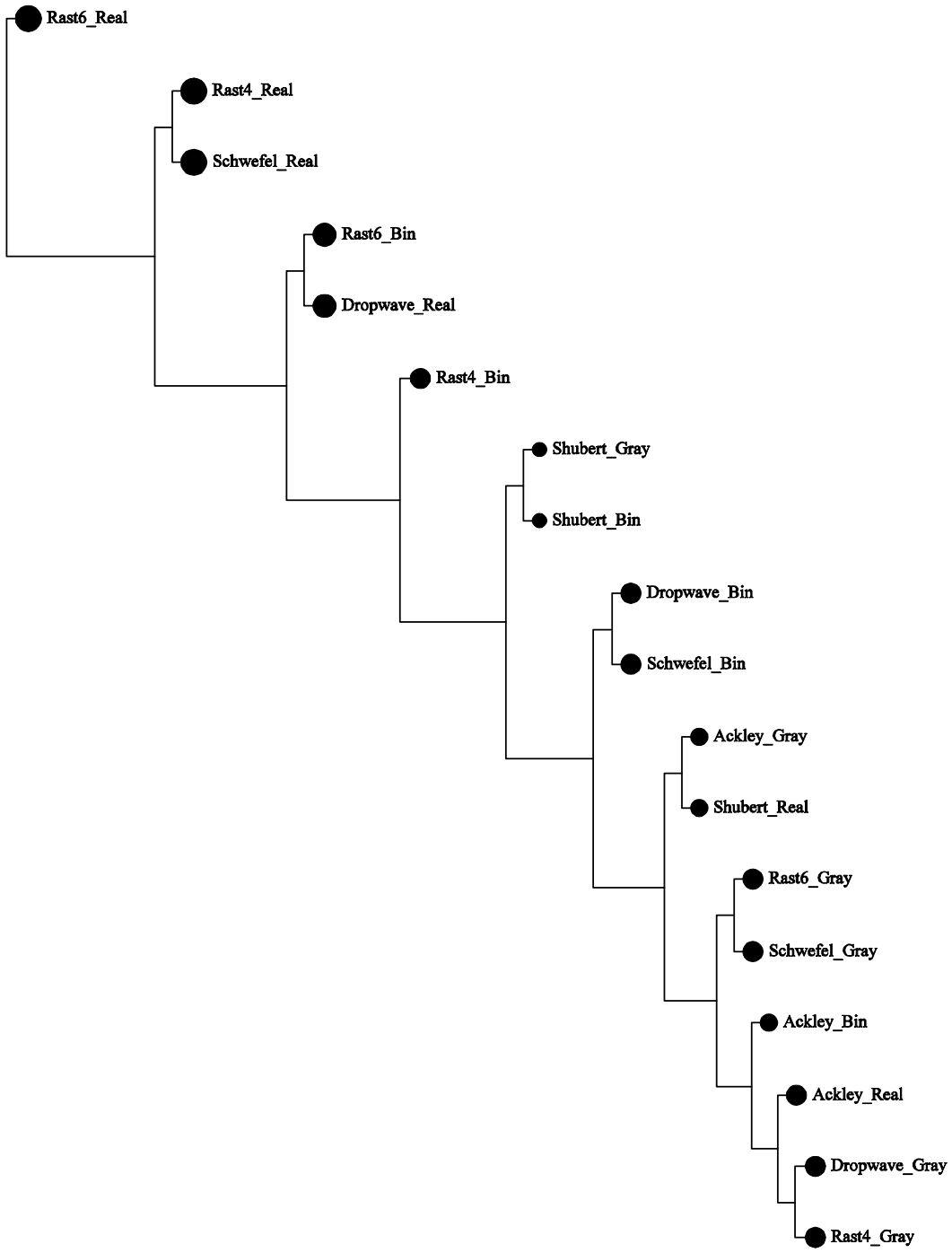

Figure 3.19 Cladogram

3.3.1.8 Summary. For all of the problems investigated here, the real value representations typically had no statistically significant differences in performance. The only exceptions were the Schwefel (Fig. 3.15) and Shubert functions, where the sparser graphs performed worse than the highly connected graphs. The gray coding representation had several statistically significant results, and of the three representations it performed best. For the Dropwave, Schwefel, and Rastrigin problem in four dimensions, the most highly connected graphs performing best, followed by the intermediate graphs, and the graphs performing the worst were the sparse graphs (Fig. 3.4). This remained true for the remaining three problems with the exception of the complete graph. For the gray coding Ackley path function, the complete graph had a performance similar to the intermediate graphs. For the gray coding six dimensional Rastrigin function, (Fig. 3.10) it performed worse than most of the intermediate graphs, although better than the sparse graphs. For gray coding Shubert function, the complete graph was one of the lowest performers, with a mean and confidence interval similar to the sparse graphs (Fig. 3.16).

The complete graph continued to perform poorly for the binary representation of the Shubert and Dropwave functions (Fig. 3.5), although the trend for these functions indicates that the intermediate graphs performed best. The remaining binary representation problems had results that were similar to each other, with the sparsest graphs performing best followed by the intermediate graphs and the most connected graphs performing the worst (Fig. 3.11). The real representation has produced statistically insignificant results characterized by large confidence intervals. The most likely cause of this is the disruptiveness of the mutation operator on real representation.. As the disruptiveness of the mutation operator increased, the amount of variation between the problems increased, as did the variation in time to solution. This disruptiveness allows the algorithms to explore the fitness space more than the other representations. Exploration of fitness space is critical in achieving redundant solutions. As this exploration can lead to diversity, a second experiment was conducted to study about diversity in real encoding. Even though diversity is a very useful element, unnecessary diversity can have negative impact on the performance of the algorithm. So, it is important to study about the dynamics of diversity, by understand it can help in

generating diverse solutions. Adapting to these diverse solutions in the event disruptions of its operations, these diverse solutions can help in continuing normal activities and improving the resiliency of the system.

3.3.2. Experiment-II. There are several trends prevalent in the problems examined here. The mutation value +/- 50 has shown better performance in the context of mean time to solution. In addition, for each problem and graph combination the mean time to solution increased as the mutation value increased when the number of failures is disregarded, although the number of failures decreased. For this experiment, if a graph fails to converge more than 250 times, the graph is considered as unsuitable for the problem with those particular parameters.

3.3.2.1 Ackley path function. For mutation value +/-50, all graphs of population size 8 and 64 had 250 or more failures. Graphs with a population size 510 or 512 had no failures, with highly connected graphs performing best (Fig. 3.20). When the mutation value is increased to +/- 100 and +/- 150, the number of failures for graphs with population sizes of 8 and 64 exceeded 250. The highly connected graphs (Hypercube, then complete) with population sizes of 510 and 512 performed best, although there was an increase in mean time to solution. For mutation value +/- 200, all graphs with population size 64 and 8 has failures except three dimensional Hypercube (population size 8, highly connected) and six dimensional Hypercube (population size 64, highly connected) (Fig 3.21). But the mean time to solution increases for all the graphs, as the mutation value is increased, the mean time to solution increases and the highly connected graphs perform best.

Except for the cases with a mutation value of +/- 200, population size 512 graphs were the best performers. This shows that this problem prefers an initial diversity through population size rather than mutation value and graphs. When the mutation value is increased, mean time to solution for population size 512 graphs increases. When the mutation value is increased to +/- 200, the six dimensional hypercube graph (population size 64, highly connected) becomes best performer. This is most likely due to the mutation operator now being able to range further across the search space making it easier for solutions to escape local optima. This allows the highly connected graphs to find solution pieces with the mutation operator and then benefit from quickly sharing

what they found. The larger population sizes are less likely to share this information due to the number of potential mating partners. This is evident from the increase in mean time to solution.

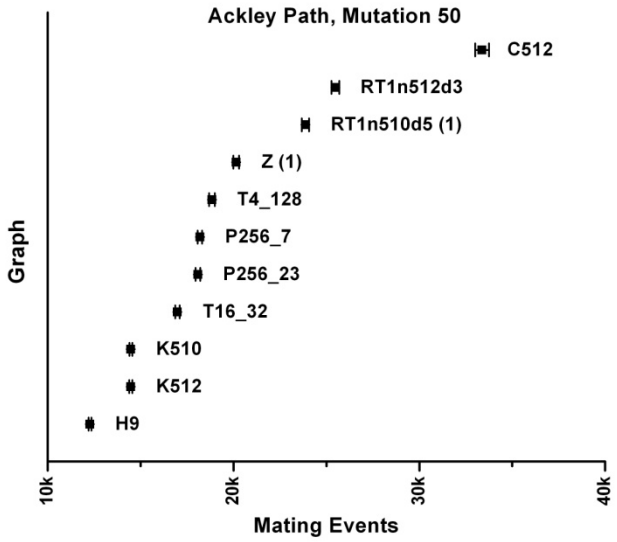

Figure 3.20 Type-I Ackley path function, mutation 50

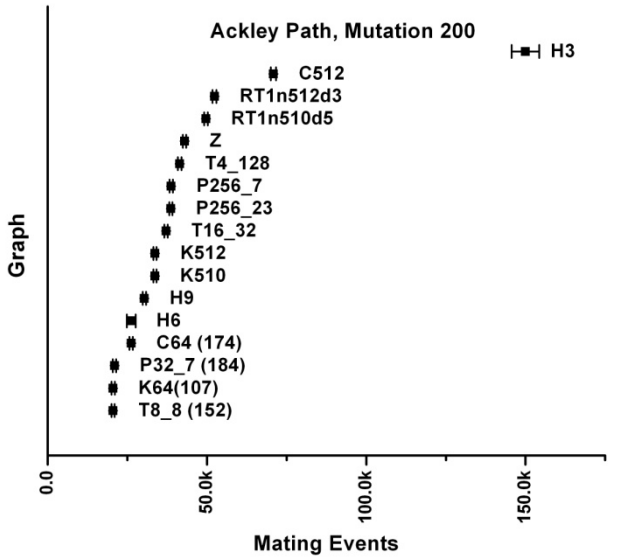

Figure 3.21 Type-II Ackley path function, mutation 200

3.3.2.2 Dropwave function. When solved with mutation value +/- 50, graphs with a population size of 8 performed best, followed by population size 64 and 512 (Fig. 3.22). The best performing graphs for this problem were the eight vertex graphs. The impact of population size is evident from the distinct grouping of graphs based on population size. When mutation value is increased to +/- 100, the separation by population appears to remain consistent, although both confidence interval and mean time to solution increases so that no statistically significant results are given between the population size 8 and population size 64 graphs. This indicates that the required diversity is already being met, so an increase in mutation value only makes the solution harder to find (Fig. 3.23).

These characteristics continue for mutation value +/- 150. When further increased to mutation value +/- 200, the 8 vertex complete graph fails more than 250 times but the only other statistically significant results is that the size 64 cycle graph performs better than the nine dimensional hypercube (Fig. 3.24). This problem is best solved with an initial population of size 8 and relatively low mutation value. Increasing the mutation value causes the mean time to solution and the confidence interval to increase. This is likely due to the larger mutation value creating too much disruption in the evolutionary mechanism as solutions jump from one fitness trough to another. The disruption at a mutation value of +/- 200 was sufficient to cause the population size 8 complete graph to fail more than 250 times, and likely would cause more failures as it approached re-sampling of the search space.

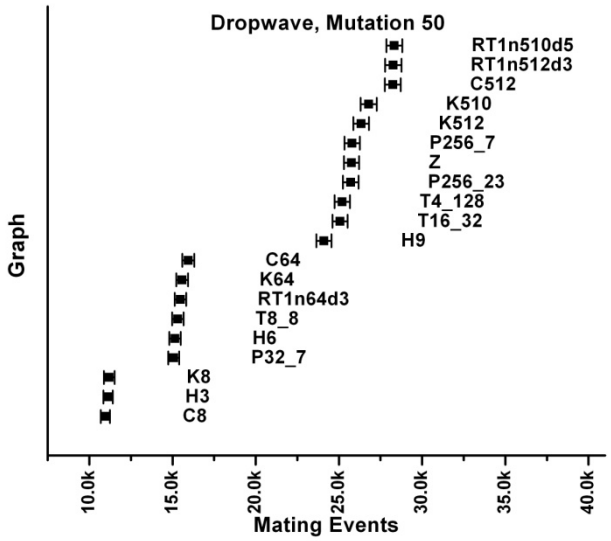

Figure 3.22 Type-II Dropwave function, mutation 50

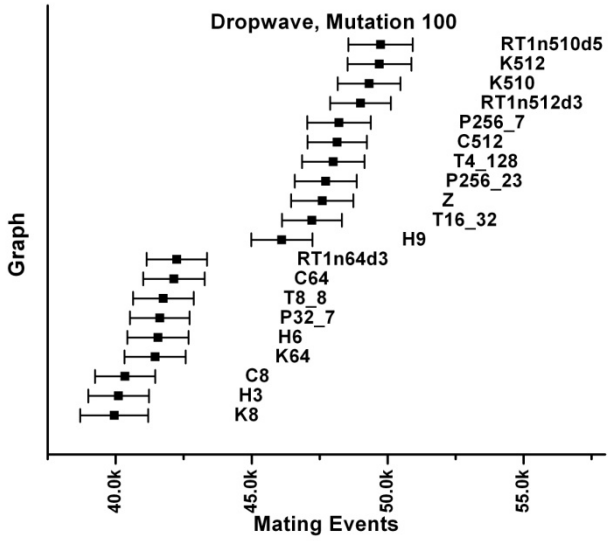

Figure 3.23 Type-II Dropwave function, mutation 100

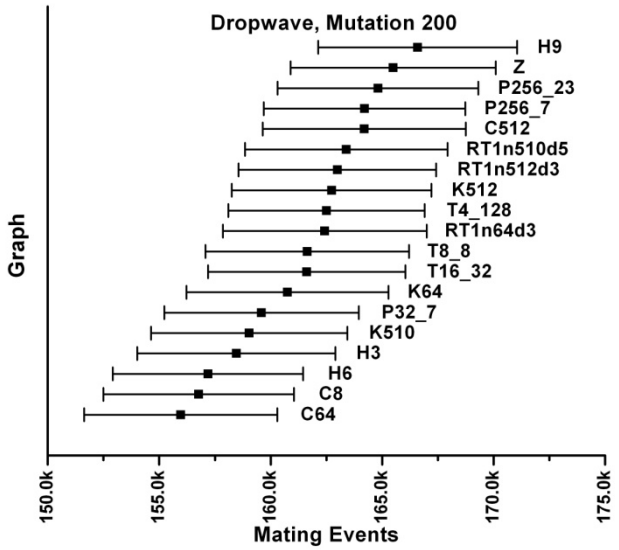

Figure 3.24 Type-II Dropwave function, mutation 100

3.3.2.3 Rastrigin function (4 variables). For mutation value +/- 50, highly connected graphs with population size 64 graphs performed best, followed by population size 512. The graphs with a population size of 8 all failed more than 250 of the runs. The graphs are grouped distinctly on the basis of population size (Fig. 3.25), and to a lesser degree by connectivity. As the highly connected graphs were preferred, the diversity needed for the problem was sufficient using a population size of 64. As the mutation value is increased, highly connected graphs of population size 8 no longer failed and displayed the best performance, followed by the graphs of population size 64 and the population size 512. The separation of graphs into population sizes also becomes much more distinct at this mutation value. When mutation value is increased further to +/- 150, graphs of population size 8 continues to perform better but with a significant increase in mean time to solution (Fig. 3.26). For mutation value +/- 200, mean time to solution and confidence intervals continued to increase for all the graphs and with the same distinct grouping of population sizes. This indicates that the problem is best solved in mutation value +/- 50, with the highly connected graphs of population size 64. When the mutation value is increased, the diversity introduction in population size 64 graphs becomes unnecessary increasing the mean time to solution. This increase in mutation value also

enables the population size 8 graphs to perform find solutions by allowing for a wider exploration of the search space. This indicates that the diversity created by mutation value is not necessary unless a small population size is used and diversity preserving graphs are not preferred for all mutation values.

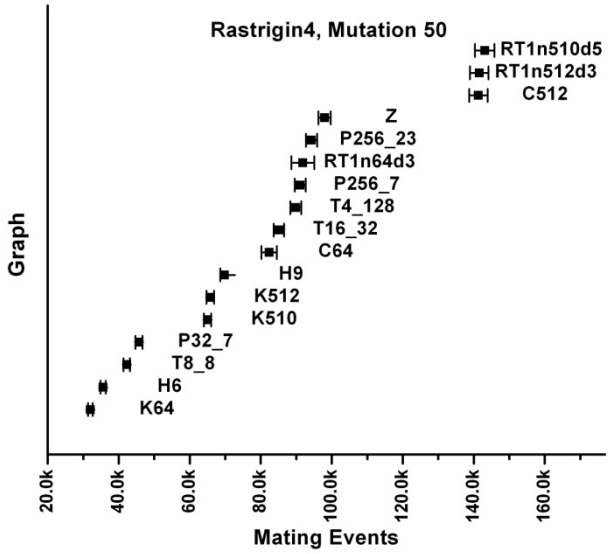

Figure 3.25 Type-I Rastrigin function (four dimensions), mutation 50

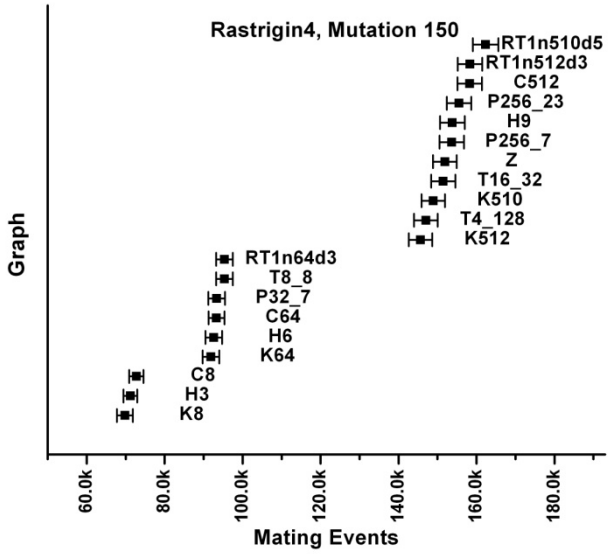

Figure 3.26 Type-I Rastrigin function (four dimensions), mutation 150

3.3.2.4 Schwefel function. For mutation value +/- 50, all graphs of population size 8 failed in more than 250 of their runs. Graphs with a population size of 64 performed better, but they all had failures of between 1 and 6 runs while the graphs with a population size of 512 had no failures but a larger mean time to solution (Fig. 3.27). The 64 vertex cycle graph yielded best performance when the failures were considered, preserving diversity in the algorithm as the diversity created by population size and mutation were insufficient. As mutation value is increased to +/- 100, the graphs with a population size of 64 had fewer failures and continued to outperform the population size 512 graphs, although with a significant increase in mean time to solution. Again, all graphs with a population size of 8 failed more than 250 times. The sparsely connected 64 vertex cycle graph was replaced by the highly connected six dimensional hypercube (Fig. 3.28). At mutation value +/- 150, graphs with population size 64 continues to perform best, but again with a high increase in mean time to solution.

When the mutation value was increased to +/- 200, none of the population size 8 graphs failed and they became the best performers, although mean time to solution again increased (Fig. 3.29). This shows that a population size of 8 has insufficient diversity to find the solution without a large mutation value (over +/- 150) to add diversity. It is interesting to note that when the number of failures and the mean time to solution are considered for the Schwefel function with a low mutation value, the sparse graphs outperformed the more connected graphs. This could indicate a trade off point between diversity types for these problems. As the mutation value is increased, the diversity from initial population and mutation value is sufficient for graphs with a population of size 64, and so they were preferred the larger population size graphs. When mutation is at +/- 200, all the population size 8 graphs start to converge with no failures, demonstrating another tradeoff between the types of diversity necessary for the problem to be solved. This shows that necessary diversity for this problem is a small population size of 8 and a higher mutation value. In addition, a sparsely connected graph may also perform well.

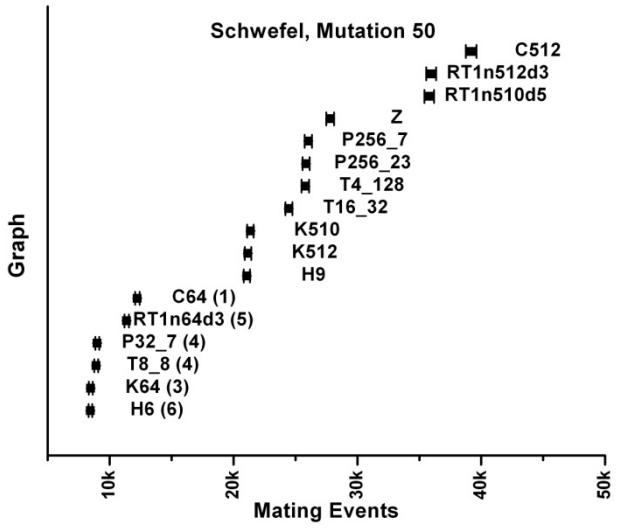

Figure 3.27 Type-I Schwefel function, mutation 50

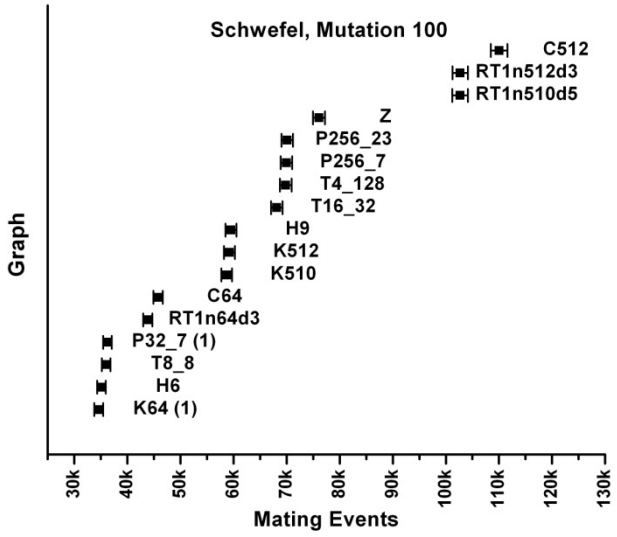

Figure 3.28 Type-I Schwefel function, mutation 100

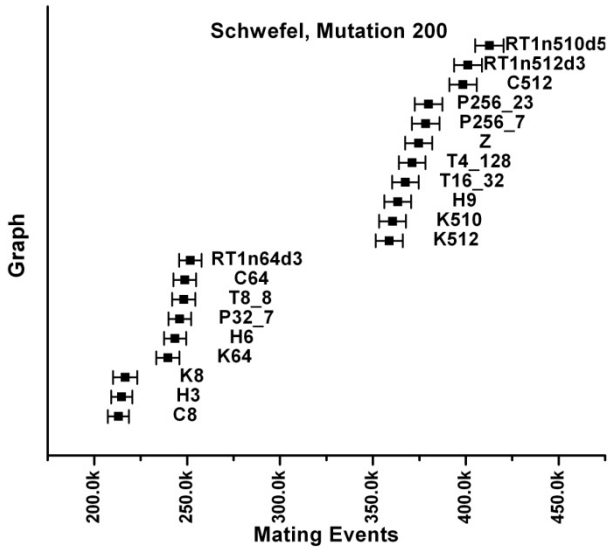

Figure 3.29 Type-I Schwefel function, mutation 100

3.3.2.5 Shubert function. As highly connected graphs are preferred the diversity needed for this problem is already contributed by the initial population and mutation value (Fig. 3.30). When the mutation value was increased to +/- 100, the mean time to solution of all population size 64 increased, although there were fewer failures in the graphs with a population size of 8 (Fig. 3.31). This trend continues when the mutation value is increased to +/- 150 (Fig. 3.32). When the mutation value was increased to +/- 200, the graphs with a population size of 8 had fewer failures, with the sparser graph having the fewest. This is likely due to the required diversity being augmented by the increase in mutation value (Fig. 3.33).

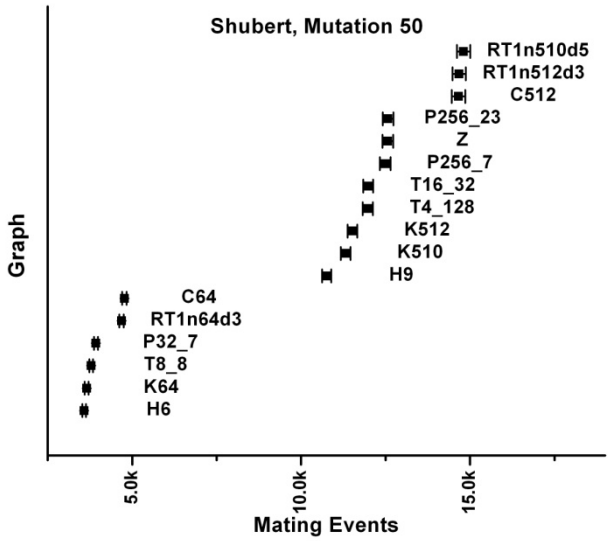

Figure 3.30 Type-I Shubert function, mutation 50

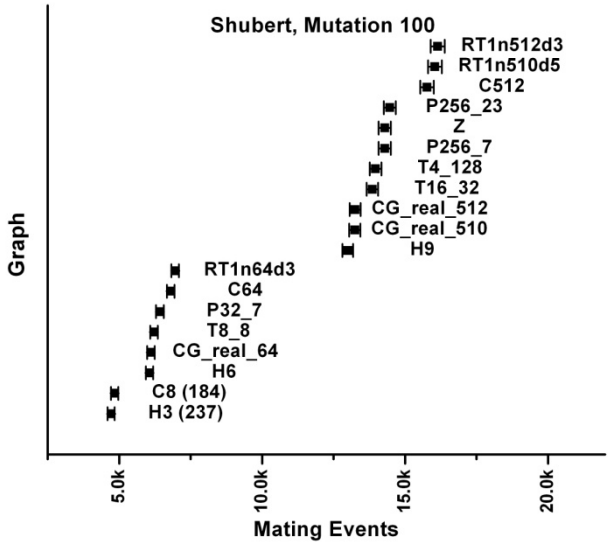

Figure 3.31 Type-I Shubert function, mutation 100

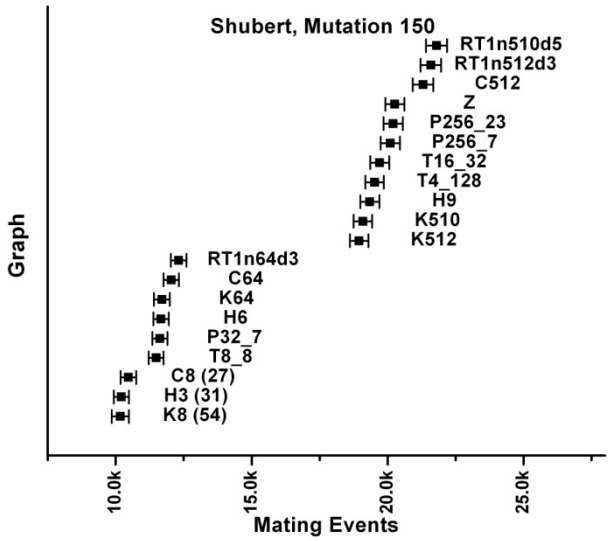

Figure 3.32 Type-I Shubert function, mutation 150

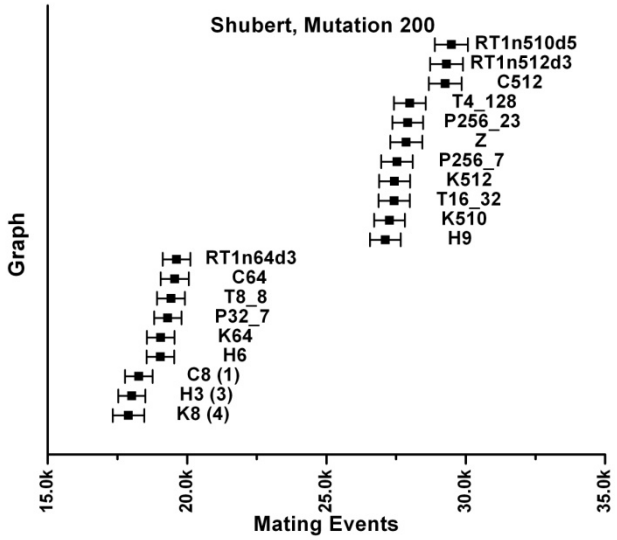

Figure 3.33 Type-I Shubert function, mutation 200

3.3.2.6 Summary. The problems in this experiment show that the type of diversity introduced by a diversity control mechanism has a strong influence on time to convergence to global optima. Using the same representation, this set of problems responded differently to different types of diversity. For example, the Ackley path problem requires a high initial diversity, while the Dropwave function needs low initial diversity and a relatively low mutation value. From the problems used in the study, only the Ackley path problem prefers a relatively high initial diversity through a population size of 512 while the others prefer population sizes of 8 or 64. When we consider the runs with a population size of 8, the solution for the Ackley path problem was found either quickly or not at all. This is likely due to a need for certain building block components which would need to be found by mutation if not in the initial population. If these pieces did not exist, the population could be taken over by solutions with local optima and impeding convergence. Once the mutation value was increased sufficiently to aid in developing these pieces, the small population graphs started to outperform the large population graphs, although the required number of mating events increased as the mutation value increases. In all of the problems, diversity preservation provided by graphs has a smaller effect, typical of this problem type [46]. From the results of experiment-I, it is clear that real encoding explores more than gray and binary encoding and is more likely to generate diverse solutions. This redundancy in solutions can be selected in place of the existing solution or design and help in continuing the operations of a system in the event of an accident. From experiment-II the importance of the extent of the diversity in the population is evident. It is critical to have moderate level of diversity as unnecessary diversity may lead in to undesirable performance of the algorithm.

4. IMPROVING TRAVELLING SALESMAN PROBLEM SOLUTION DIVERSITY USING GRAPH BASED EVOLUTIONARY ALGORITHMS

4.1. BACKGROUND

The traveling salesman problem (TSP) is one of the widely known non-deterministic polynomial time (NP) combinatorial optimization problem [47]. A salesman travels to a set of cities, visiting each only once. The solution is to find the shortest distance to visit each of these cities and then return to the starting city. Links of cities that are good in the short term do not necessarily lead to optimal complete routes. Many problems in science, engineering, bioinformatics, and scheduling can be formulated as traveling salesman problems. A simple explanation of a supply chain network is explained in this work. Supply chain models can be evaluated using traveling salesman problem, where each city can be considered as a warehouse or customer. The problem faced by the supply chain model is the vulnerabilities in the routes. The vulnerabilities can be caused by various problems like consequences from natural disasters, road conditions, etc. These can result in damages which can be immense and may cripple entire sections of the network, causing extensive financial damage. One of the methods to mitigate these risks is to build a resilient supply chain model capable of rerouting the transportation vehicles to circumvent these occurrences. One method to build a resilient supply chain network is to generate multiple good solutions and provide methods to transition between these solutions. This gives an opportunity to reroute the network if a hazard is encountered in the network. These multiple solutions do not add or delete cities but supply alternative solutions to the same problem. A similarity measure based on transposition of cities is used to determine the degree to which routes differ.

Evolutionary algorithms have been applied on TSP obtaining differing levels of success. Some of the earliest uses of evolutionary algorithms on TSP were by Goldberg et al. [48] and Grefenstette et al. [49]. Follow on research efforts using evolutionary algorithms have been applied to improve the performance and running time of TSP. various studies on the representation of TSP, crossover, mutation operators have been studied [29, 49, 51, 52, 53]. These have given several insights to the use of EAs for solving TSP. Improvements are continually made in EAs to solve TSP. Recently Wang et

al. proposed an improved greedy algorithm by combined with local search methods for TSP [76]. A memetic algorithm was used by Liu [75]. For some of the other recent research on approaching TSP using EAs refer to [77, 78, 79, 80]

Due to their flexibility and scalability EAs are used to improve resiliency in some domain specific systems. Hybrid genetic algorithm was used by He et al. [71] designed to generate back-up routing in telecommunications network based on shortest path problem. Abdullah et al. used hybrid genetic algorithms to design resilient high speed communication networks [72, 73]. Evolutionary algorithms were used in managing traffic in internet networks [74].

4.2. EXPERIMENTAL DESIGN

TSP is a problem in which there are N cities and a salesman must travel to all the cities, but only once and returns to the city where the salesman started. For each pair of cities the distance is known. Tour length, the order of the cities in which the salesman visits the cities, must be as small as possible. Let G = (V, E) be a complete, weighted graph. A Hamilton cycle of graph G is a cycle graph that connects each vertex of the graph only once. Each vertex can be considered as a city and the weights on the edges as distance or cost for traveling between the two cities. The traveling salesman problem is to find the Hamilton cycle with the minimum weight. TSP can be represented in various methods. The method used in this study is path representation, where i= (1, 2...N), a positive integer represents a city. It is perhaps one of the natural forms of representation. A tour 2-3-1-5-4-6 is represented as T= (2 3 1 5 4 6). Each city can be located in a two dimensional space using Euclidean co-ordinates. The distance between two cities can be found using Euclidean distance formula. Sum of all the Euclidean distances in a tour T gives the tour length.

Real value encoding is used as representation in evolutionary algorithm to match with the path representation of TSP. Creating initial population has two steps. First the population is generated using nearest neighbor algorithm. A greedy algorithm which selects the nearest unvisited city to the current city is used. In the next step a 2-optimal algorithm is applied to the population which was generated earlier using nearest

neighbor. Although the basic moves for 2-opt were first suggested by Flood [54], it was proposed by Croes [55] as an algorithm to be used on TSP. It is a local search algorithm where typically two edges are deleted from the tour and the nodes are reconnected in other possible positions that still yield a valid tour. This step is done only when the reconnected new tour is shorter. Continue removing and reconnecting the edges until no improvements can be made in the tour length. Now the tour is 2-optimal. Using 2-optimal algorithm will result in a tour length less than 5% above the Held-Karp bound [56]. The crossover operator used in partially mapped crossover (PMX). The crossover builds by swapping a subsequence of a tour between the two parents. The rest of the offspring are constructed from the original parents for which there is no conflict in the cities. For more details refer to [57]. Mutation rate is 100% and mutation operator is a simple form of mutation, swapping of two cities selected uniformly random. This simple form of mutation may not result in high disruptiveness to the algorithm. Graphs used in this study can be divided on the basis of number of vertices. Eight different types of graphs are used with a combination of different vertices creating 34 different graphs (Table 4.1). The number vertices used are 8, 64, 512, 1024.

Table 4.1 Graphs used in TSP experiment

Graph	Index	Vertices	Diameter	Degree
Hypercube3	H3	8	3	3
Cycle8	C8	8	4	2
Complete8	K8	8	1	7
Hypercube64	H6	64	6	6
Cycle64	C64	64	32	2
Complete64	K64	64	1	63
Toroid 8,8	T8_8	64	10	4
Peterson 32,7	P32_7	64	6	3
Regular Tree 64,3	RT1n64d3	64	10	1.969
Hypercube9	H9	512	9	9
Cycle512	C512	512	256	2

Table 4.1 Graphs used in TSP experiment continued.

Complete512	K512	512	1	511
Toroid 4,128	T4_128	512	66	4
Toroid 16,32	T16_32	512	24	4
Toroid 8,64	T8_64	512	36	4
Peterson 256,1	P256_1	512	129	3
Peterson 256,3	P256_3	512	46	3
Peterson 256,7	P256_7	512	22	3
Peterson 256,17	P256_17	512	18	3
Peterson 256,23	P256_23	512	16	3
Peterson 256,23	P256_23	512	16	3
RegularTree510,5	RT1n510d5	512	9	1.996
RegularTree510,4	RT1n510d4	512	11	1.996
RegularTree512,3	RT1n512d3	512	16	1.996
Simplexified	Z	512	19	4
Random Toroid	RTor07_1	512	19	7.445
Hypercube 10	H10	1024	10	10
Cycle1024	C1024	1024	512	2
Peterson 512,1	P512_1	1024	257	3
Peterson 512,3	P512_3	1024	88	3
Peterson 512,7	P512_7	1024	42	3
Peterson 512, 17	P512_17	1024	25	3
Toroid 16, 64	T16_64	1024	40	4
Toroid 4, 256	T4_256	1024	130	4
Toroid 8, 128	T8_128	1024	68	4

The two TSP problems used in this study each contain 100 cities. For each of the 34 graphs 100 independent simulations of the two problems were computed and the ending criterion for each run is 10000 mating events. Once the ending criteria were reached the loop was terminated and the highest fitness tour of that particular simulation

is recorded. After 100 simulations, the first tour recorded was selected as a reference and compared with the rest of the tours. The tours which were different from the reference tour were noted and the reversal distance [58] between the reference tour and the tour under comparison was calculated and recorded.

TSP solutions can be represented as permutations of tours. Consider tours T_a and T_b, where $T_a = (T_{a1}\ T_{a2}\ T_{a3}.....T_{an})$ and $T_b = (T_{b1}\ T_{b2}\ T_{b3}.....T_{bn})$. In this notation T_{ai} is denoted T_a (i). Typically, reversal of an interval [i, j] is the permutation Ta = (j j-1 ..... i). To calculate the reversal distance given permutations T_a and T_b, find a series of reversal R_1, R_2 ...R_n such that $T_a \cdot R_1 \cdot R_2 \cdot \cdots R_n = T_b$, and where n is minimum. In general, the reversal distance between T_a and T_b are equal to the reversal distance between $T_a^{-1} \cdot T_b$ and the identity permutation i, where T_a^{-1} denotes the inverse of T_a. Next, the input is taken as $\Pi = T_a^{-1} \cdot T_b$ and its reversal distance from the identity matrix i is calculated. After each reversal, the number of transpositions required for that particular reversal is noted and continued until the total reversal distance is computed. Here transposition is deletion and reinsertion of a edge from its original site. The number of transpositions between the permutations gives a dissimilarity measure between the two tours.

4.3. EXPERIMENTAL RESULTS

The two problems used in the study are kroA_100 and kroC_100 from TSPLIB [59] with optimal distance at 21282 and 20749 respectively. The columns in Tables 4.1 and 4.2 represent: A – Percentage of difference between the optimal distance and the distance of the reference tour, B – Number of dissimilar routes produced by that graph, C – Percentage of difference between the distance of the reference tour and the best tour distance produced by that graph, D - Percentage of difference between the distance of the optimal tour and the best tour distance produced by that graph, E – number of transpositions required to change the reference tour into the best tour found by the graph.

4.3.1. KroA. Of all 34 different graphs only 16 graphs gave dissimilar solutions indicating that all of the other graphs found the same best tour in every simulation. Only 2 graphs (C1024, T16_64) has generated diverse solutions in 1024 vertices group. In graphs with 512 vertices five graphs generated diverse solutions. They are complete 512, Hypercube 9, Peterson 256_7, Peterson 256_17 and Toroid 8_64. All the graphs with vertices 64 and 8 produced diverse solutions. The number of dissimilar solutions generated from graphs with vertices 1024 and 512 is 1. For graphs with 64 vertices the number of dissimilar solutions is one of 4, 5 and 6. All the graphs with eight vertices have generated 29 different solutions. The number of different solutions generated increases by the decrease of number of vertices of the graph.

Table 4.2 Results of KroA

Graphs	A	B	C	D	E
C1024	1.932	1	0.11	1.81	27090
T16_64	1.932	1	0.11	1.81	27090
K512	1.932	1	0.12	1.8	186
K64	1.932	5	0.06	2	22971
K8	3.198	29	1.22	1.93	24060
C64	1.932	6	0.06	2	22971
C8	2.471	29	0.65	1.8	11484
H3	2	29	0.06	1.93	23175
H6	2	4	0.06	1.93	23175
H9	1.932	1	0.12	1.8	186
P256_17	1.932	1	0.11	1.81	27090
P256_7	1.932	1	0.18	1.74	23385
P32_7	1.932	4	0.06	2	22971
RT1n64d3	2	5	0.06	1.93	23175
T8_64	1.932	1	0.11	1.82	27090
T8_8	1.932	4	0.11	1.82	22971

4.3.2. KroC. All the graphs with vertices 8 and 64 have produced different solutions and the rest of the graphs did not. For this instance, the number of dissimilar routes for graphs with 64 vertices was between 3 and 6. The complete graph with 8 vertices produced 26 dissimilar solutions and the three dimensional hypercube and cycle graph with eight vertices produced 30 dissimilar solutions. Again the number of dissimilar solution increases with the decrease in the number of vertices of the graphs.

Table 4.3 Results of KroC

Graphs	A	B	C	D	E
K64	1.429	5	0.21	1.64	23928
K8	4.806	26	3.22	1.42	44700
C64	1.429	3	0.21	1.64	23928
C8	1.429	30	0.21	1.64	23928
H3	5.674	30	4	1.42	34302
H6	1.647	4	0.21	1.42	22272
P32_7	1.429	3	0.21	1.64	23928
T8_8	1.429	6	0.2	1.22	12
RT1n64d3	1.429	3	0.21	1.64	23928

4.3.3. Summary. The use of GBEAs has produced diverse tours in both theproblems, kroA and kroC. In both the problems the number of dissimilar routes increases with the decrease in population size. This may due to the emphasis of the disruptivness created in the algorithm. In high polation size, this disruptivness is undermined, but in low population size it will be oppsite. A small disruptivess can be enhanced due to a very less choice of population members. The number of transpositions can be take as a metric which can be used to determine the difference between tours. The graphs with fewer vertices produced more tours that are dissimilar. In both these problems the graphs which produced the best tours among the graphs are the intermmediately connected graphs.

5. CONCLUSIONS AND FUTURE WORK

Resiliency is improved by redundancy and redundancy can be achieved in system design by using GBEAs to create diverse designs. The ability of GBEAs to control solution diversity allows the system design to be provided with several elements that helps to improve resiliency. One of the ways to achieve resiliency is to improve the adaptability of the system, and adaptability can be improved by redundancy. Redundancy in system design is the having multiple designs of the system. Adaptability can be implemented by placing an agent within the system to take advantage of the diverse solution set. It can be a software agent or a human agent capable of switching to the other design in an event of disruption. GBEAs are used to improve redundancy in system design by generating diverse solutions. To obtain the best results some important characteristics of GBEAs impacting diversity control, such as representation, must be considered.

Representation is the data structure used along with the choice of variation operators. This plays a significant part in the outcome of the results. In the experiment-I gray, binary and real encodings are used on the same problem. When compared with the gray and binary encoding, the real encoding has statistically insignificant results, although two characteristics are discernable in the real encoding results. The mean number of mating events varied widely, producing large confidence intervals and a high mean number of mating events. These two characteristics can be attributed to the element of exploration in the algorithm, where the algorithm is searching for building blocks to find the solution. The most likely cause of this is the disruptiveness of the mutation operator in the real valued representations compared to the gray coding and binary encoding. As the disruptiveness of the mutation operator increased, the amount of variation of number of mating events between the problems increased, as did the variation in time to solution. This phenomenon improves the level of diversity in the solutions. Diversity in solutions is resulted from the diversity in the population.

For some problems, too much diversity in the population can hinder the performance of the algorithm, so it is important to control the diversity in the population. By tuning the mutation value, the diversity in the population can be controlled to an

extent. As real encoding exhibits the element of exploration more than the other two representations it is further studied. Diversity is created initially by generating a random population. As the algorithm progresses diversity is induced through the mutation operator or preserved by a mechanism incorporated in the algorithm, in this case the use of GBEAs. High mutation value and rate may bring unnecessary diversity in the solutions and increase the time to solution very high. The requirement of diversity is mainly based on the fitness landscape of the problem. From the results of experiment-II it is evident that a trade off can be seen in each problem between diversity from population size, mutation value, and diversity preservation. For these problems, graphs come into play only when the diversity offered by the population size and mutation value is inadequate. In addition, different problems require different combinations of diversity, whether initial, injected, or maintenance, and so a single approach will not be adequate to provide the necessary diversity to all problems.

To better understand the results of experiment-I and II, consider the traveling salesman problem. It is a problem which is similar real encoding. The results from TSP experiment show that the diversity in the solution is mainly offered by the population size. As the number of vertices decreases the rate of information shared between the vertices increases. This allows for the evolving tours to quickly combine building blocks to form a high performance solution, generating diverse solutions in the graphs with lower vertices. As the number of number of vertices increases rate of information sharing decreases and it is more difficult to form the necessary building blocks. This can be seen in the column B in the Tables 4.2 and 4.3. Also to determine the diversity in the solutions the number of transpositions between two paths can be taken as a metric which can be used to determine the difference between tours. The graphs with fewer vertices produced more tours that are dissimilar. In both these problems the graphs which produced the best tours based on the distance are the intermediately connected graphs. This study shows that the diversity of the population is very important in generating diverse solutions. The diversity in solutions is a result from the appropriate combination of representation, mutation value and graphs. These diverse solutions increase the redundancy of the system design. In an event of accident, one of the diverse solutions can be used which allows the system to survive and recover from the accident. In order to use the other good solution,

the system must be adaptable. This adaptability can be realized by an agent based behavior, either by a human or software agent, sequentially improving the resiliency of the system.

Additional TSP problems with varying number of cities can be analyzed to better understand the working of this algorithm. This can result in changing of certain parameters as the number of cities varies. An improvement to the present algorithm can be made by using k-opt method, which will improve the quality of solutions. To understand more on generating diverse solutions using GBEAs, other parameters have to be studied. Mainly, the combined effect of representation and population size can be analyzed. Also the exploratory nature of other forms of representation has to be investigated; as all the real world problems cannot be represented using real valued encoding.

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.

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.

Definition 4 The n x m-torus, denoted $T_{n,m}$, has vertex set Z_n x Z_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.

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.

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 random graphs were 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. 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 $T_{4,8}$ 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 Rt (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 3. 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.

BIBILOGRAPHY

1. Friedman, G. Select feedback computers for engineering synthesis and nervous system analogy. Master's Thesis, 1956, UCLA.

2. Box, F. Evolutionary operation: A method for increasing industrial productivity. Applied Statistics, 1957 6(2), 81-101.

3. Holland, J. Outline for a logical theory of adaptive systems. JACM 1962, 9,297-314

4. Holland, J. Nonlinear environments permitting efficient adaptation. In computer and information sciences II, 1967. Academic Press.

5. K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. of Michigan, Ann Arbor, 1975, Dissertation. Int. 36(10), 5140B, University Microfilms no. 76-9381.

6. K. A. De Jong, "On using genetic algorithms to search program spaces," in Proc. 2nd Int. Conf. on Genetic Algorithms and Their Applications. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 210–216.

7. K. A. De Jong, "Are genetic algorithms function optimizers?" in Parallel Problem Solving from Nature 2. Amsterdam, The Netherlands: Elsevier, 1992, pp. 3–13.

8. K. A. De Jong, "Genetic algorithms are NOT function optimizers," in Foundations of genetic algorithms 2. San Mateo, CA: Morgan Kaufmann, 1993, pp. 5–17.

9. D. E. Goldberg, "Genetic algorithms and rule learning in dynamic system control," in Proc. 1st Int. Conf. on Genetic Algorithms and Their Applications. Hillsdale, NJ: Lawrence Erlbaum, 1985, pp. 8–15.

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

11. D. E. Goldberg, "The theory of virtual alphabets," in Parallel Problem Solving from Nature—Proc. 1st Workshop PPSN I. (Lecture Notes in Computer Science, vol. 496). Berlin, Germany: Springer, 1991, pp. 13–22.

12. D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations," Complex Syst., vol. 6, pp. 333–362, 1992.

13. D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik, “Rapid, accurate optimization of difficult problems using fast messy genetic algorithms,” in Proc. 5th Int. Conf. on Genetic Algorithms. San Mateo, CA: Morgan Kaufmann, 1993, pp. 56–64.

14. I.Rechenberg, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart, Germany: Frommann-Holzboog, 1973.

15. I.Rechenberg, Evolutionsstrategie ’94, in Werkstatt Bionik und Evolutionstechnik Stuttgart, Germany: Frommann-Holzboog, 1994, vol. 1.

16. H.-P. Schwefel, Evolutionsstrategie und numerische Optimierung Dissertation, Technische Universit¨at Berlin, Germany, May 1975.

17. H.-P. Schwefel, Evolution and Optimum Seeking. New York: Wiley, 1995 (Sixth-Generation Computer Technology Series).

18. M. Herdy, “Reproductive isolation as strategy parameter in hierarchically organized evolution strategies,” in Parallel Problem Solving from Nature 2. Amsterdam, The Netherlands: Elsevier, 1992, pp. 207–217.

19. F. Kursawe, “A variant of Evolution Strategies for vector optimization,” in Parallel Problem Solving from Nature—Proc. 1st Workshop PPSN I (Lecture Notes in Computer Science, vol. 496). Berlin, Germany: Springer, 1991, pp. 193–197.

20. L. J. Fogel, “Autonomous automata,” Ind. Res., vol. 4, pp. 14–19, 1962.

21. L. J. Fogel, “On the organization of intellect,” Ph.D. dissertation, University of California, Los Angeles, 1964.

22. G. H. Burgin, “On playing two-person zero-sum games against nonminimax players,” IEEE Trans. Syst. Sci. Cybern., vol. SSC-5, no. 4, pp. 369–370, Oct. 1969.

23. G. H. Burgin, “Systems identification by quasilinearization and evolutionary programming,” Journal of. Cybernetics. vol. 3, no. 2, pp. 56–75, 1973.

24. J. W. Atmar, “Speculation on the evolution of intelligence and its possible realization in machine form,” Ph.D. dissertation, New Mexico State Univ., Las Cruces, 1976.

25. L. J. Fogel, A. J. Owens, and M. J. Walsh, Artificial Intelligence Through Simulated Evolution. New York: Wiley, 1966.

26. D. B. Fogel, “An evolutionary approach to the traveling salesman problem,” Biological Cybern., vol. 60, pp. 139–144, 1988.

27. D. B. Fogel, “Evolving artificial intelligence,” Ph.D. dissertation, Univ. of California, San Diego, 1992.

28. Benjamin Lewin. Genes VII. Oxford University Press, New York, 2000.

29. D. E. Goldberg. Sizing populations for serial and parallel genetic algorithms. In Proceedings of the Third International Conference on Genetic Algorithms, pages 70–79. Morgan Kaufmann Publishers Inc., 1989.

30. Smith, R. E “Adaptively resizing population: An algorithm and analysis”, in proceedings of fifth international conference on genetic algorithms Morgan Kauffman Publishers 1993.

31. Jaroslaw Arabas, Zbigniew Michalewicz, and Jan Mulawka. GAVaPS-a genetic algorithm with varying population size. In Proceedings of the First IEEE Conference on Evolutionary Computation, pages 73–78, 1994.

32. Ashlock, D.(2006), Evolutionary Computation for Modeling and Optimization, Springer, ISBN 0-387-22196-4.

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

34. F. Kursawe, “Toward self-adapting evolution strategies,” in Proc. 2nd IEEE Conf. Evolutionary Computation, Perth, Australia. Piscataway, NJ: IEEE Press, 1995, pp. 283–288.

35. A. E. Eiben, E.H.L. Aarts, K.M. Van Hee. Global convergence of genetic algorithms: A Markov chain analysis. In Proceedings of the 1st conference on parallel problem solving from nature, No. 496 in Lecture notes in computer science, Springer, Berlin, Heidelberg, New York, 1991.

36. H.E. Aguirre, K. Tanaka, "Parallel varying mutation genetic algorithms," wcci, vol. 1, pp.795-800, Computational Intelligence, WCCI. Proceedings of the 2002 World on Congress on, 2002.

37. S. Wright, Evolution, W. B. Provine, Ed. Chicago, IL: Univ. of Chicago Press, 1986.

38. Glover, F. "Tabu Search — Part I," ORSA Journal on Computing 1989 1: 3, 190-206.

39. Glover, F. "Tabu Search — Part II," ORSA Journal on Computing 1990 2: 1, 4-32.

40. M. Kimura and J. Crow, “On the maximum avoidance of inbreeding,” Genet. Res., vol. 4, pp. 399–415, 1963.

41. D. Ashlock, J. Walker, and M. Smucker, “Graph based genetic algorithms,” In Proc. Congress on. Evolutionary Computation. San Francisco, CA, 1999, pp.1362–1368.

42. M. Bryden, Ashlock, McCorkle, “An Application of Graph Based Evolutionary Algorithms for Diversity Preservation”. In Proceedings of the 2004 congress on evolutionary computation. CEC 2004. Portland OR, USA.

43. Corns, Ashlock, McCorkle, Bryden, - Improving Design Diversity Using Graph based Evolutionary Algorithms, 2006 IEEE Congress on Evolutionary Computation.

44. 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

45. 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.

46. K. M. Bryden, D. A. Ashlock, S. Corns, and S. J. Wilson. Graph based evolutionary algorithms. IEEE Transaction on Evolutionary Computation, 10:550–567, 2006.

47. M. Garey and D. Johnson,” Computers and Intractability A Guide to the Theory of NP Completeness”, Freeman, San Francisco, 1979.

48. D. Goldberg and R. Lingle, “Alleles, Loci, and the Traveling Salesman Problem,” First International Conference on Genetic Algorithms and Their Applications, pp. 154-159, 1985.

49. J. Grefenstette, R. Gopal, B. Rosmaita, and D. Gucht, “Genetic Algorithms for the Traveling Salesman Problem,” First International Conference on Genetic Algorithms and Their Applications, pp. 160-168, 1985.

50. Grefenstette,J.J. Incorporating Problem Specific Knowledge into Genetic Algorithms. “Genetic Algorithms and Simulated Annealing” Morgan Kaufmann Publishers, San Mateo, CA, pp.42-60.1987.

51. Jog P., J.Y. Suh, and D.V. Gucht. “The Effects of Population Size, Heuristic Crossover, and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem”. Proceedings of 3rd International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA. pp.110-115. 1989.

52. Oliver, I.M., D.J. Smith, and J.R.C. Holland. “A Study of Permutation Crossover Operators on the Traveling Salesman Problem”. Proceedings of 2nd International Conference of Genetic Algorithms. Lawrence Erlbaum Associates. Hillsdale, NJ, pp 224-230, 1987.

53. Seniw, D. “A Genetic Algorithm for Traveling Salesman Problem”. Msc Thesis, University of North Carolina, Charlotte, NC, 1991.

54. M. M. Flood,“The traveling-salesman problem”, Operations Res. 4 (1956), 61-75.

55. G. A. Croes, “A method for solving traveling salesman problems,” Operations Res. 6 (1958), 791-812.

56. D.S. Johnson and L.A. McGeoch, “The Traveling Salesman Problem: A Case Study in Local Optimization,” November 20, 1995.

57. Michalewicz, Z., & Fogel, D. B. (2000). How to Solve It: Modern Heuristics. Berlin: Springer.

58. Kececioglu, J., and Sankoff, D. 1995. Exact and approximation algorithms for the inversion distance between two permutations. Algorithmica 13, 180-210.

59. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. Accessed February 10th, 2010.

60. D. L. Battle and M. D. Vose. Isomorphism of genetic algorithms. In Proceedings og 2nd International conference on genetic algorithms and their applications, Lawrence Erlbaum Associates, 1987.

61. Eiben, A. E. and Schippers, C. A. 1998. On evolutionary exploration and exploitation. Fund. Inf. 35, 1–16.

62. Thomas Back, Agoston Eiben, and Nikolai van der Vaart. An empirical study on GAs without parameters. In Proceedings of PPSN VI: the 6th International Conference on Parallel Problem Solving from Nature, pages 315–324, London, UK, 2000. Springer-Verlag.

63. A.E. Eiben, R. Kinterding, and Z. Michalewicz, “Parameter control in evolutionary algorithms”, IEEE transactions on evolutionary algorithms, Vol.3 no. 2, pp 124-141,1996.

64. D. Whitley, S. Rana, and R. Heckendorn, “Island model genetic algorithms and linearly separable problems,” in Proc. AISB Workshop Evolutionary Computation D. Corne and J. Shapiro, Eds., New York, 1997, pp. 109–125.

65. Maher. L. M., Poon, J. & Boulanger, S. (1995) Formalising Design Exploration as Co-Evolution: A combined gene approach. Preprints of the Second W P I F WGS.2 Workshop on Formal Design Methods for CAD

66. Gero, J and Kazakov, V (1998) Adapting Evolutionary computing in Creative Designing. 4th International Conference on Computational Models of Creative Design Heron Island, Australia, 6-10 December.

67. Parmee, I. C. and Bonham, C. (1998) Supporting Innovative and Creative Design using Interactive Designer/Evolutionary Strategies. 4th International Conference on Computational Models of Creative Design Heron Island, Australia, 6-10 December.

68. Claverie, J.M.; De Jong, K.; Sheta, A.F.; , "Robust nonlinear control design using competitive coevolution," Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, vol.1, no., pp.403-409 vol.1, 2000.

69. Mitsuo Gen, Runwei Cheng, Genetic Algorithms, John Wiley & Sons, Inc., New York, NY, 1999.

70. George, S., Evans, D., & Marchette, S. (2003). A Biological Programming Model fo Self-Healing. In *First ACM Workshop on Survivable and Self-Regenerative Systems (SSRS03)*. Available at www.cs.virginia.edu/~evans/pubs/ssrs.pdf.

71. He, L., & Mort, N. (2007). Hybrid genetic algorithms for telecommunications network back-up routing. BT Technology Journal, 18(4), 42–50.

72. A. Konak and A. E. Smith, \Designing resilient networks using a hybrid genetic algorithm approach," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, (New York, NY, USA), pp. 1279{1285, ACM Press, 2005.

73. Konak, A., & Bartolacci, M. R. (2007). Designing survivable resilient networks: a stochastic hybrid genetic algorithm approach. *Omega*, *35*(6), 645–658.

74. Banarjee, N., & Kumar, R. (2007). Multiobjective network design for realistic traffic models. In *Proceedings of generic and evolutionary computation conference (GECCO’07)*, (pp. 1904–1911), London, UK.

75. Y.-H. Liu. A Memetic Algorithm for the Probabilistic Traveling Salesman Problem. In: *Procs. 2008 IEEE Congress Evolutionary Computation (CEC 2008)*, Hong Kong. (2008).

76. Z Wang, H Duan, X Zhang. An Improved greedy genetic algorithm for solving travelling salesman problem. Proceedings of the 2009 Fifth International Conference on Natural Computation - Volume 05 374-378.2009.

77. Jin-Qiu Yang, Jian-Gang Yang, "A Novel Genetic Algorithm for Traveling Salesman Problem Based on Neighborhood Code," Intelligent Networks and Intelligent Systems, International Workshop on, pp. 429-432, 2009 Second International Conference on Intelligent Networks and Intelligent Systems, 2009.

78. Yanbing Liu, Jun Huang, "A Novel Genetic Algorithm and Its Application in TSP," Network and Parallel Computing Workshops, IFIP International Conference on, pp. 263-266, 2008 IFIP International Conference on Network and Parallel Computing, 2008.

79. S. Y. Yuen and C. K. Chow, "A non-revisiting genetic algorithm," in Proc. IEEE Congress on Evolutionary Computation, 2007, pp. 4583-4590.

80. P. C. Chang, W. H. Huang, J. Y. C. Liu, C. Chen, C. J. Ting, “Dynamic Diversity Control by Injecting Artificial Chromosomes for Solving TSP Problems,” Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence), pp. 542-549, 2008.

VITA

Jayakanth Jayachandran was born in Cuddalore, Tamilnadu, India, on April 20, 1986. Most part of his schooling was completed in Cuddalore. He pursued his Bachelors in Electrical and Electronics from Anna University, Chennai, India in 2007 and started Masters at Missouri University of Science and Technology in May 2008. His primary areas of interest are evolutionary computation, modeling and simulation and received M.S. in July of 2010.