

KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, KUMASI  
COLLEGE OF SCIENCE  
DEPARTMENT OF MATHEMATICS

A GENETIC ALGORITHM MODEL FOR TRAVELLING SALESMAN PROBLEM  
(TSP)

A thesis submitted to the School of Graduate Studies  
Kwame Nkrumah University of Science and Technology in partial fulfilment of the  
requirements for the degree of Master of Philosophy (Applied Mathematics)

By  
DANIEL GYAMFI (BSc. MATHEMATICS)

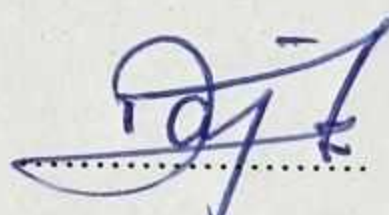
AUGUST 2013

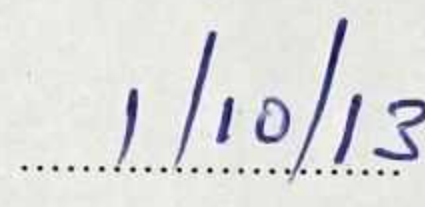


# DECLARATION

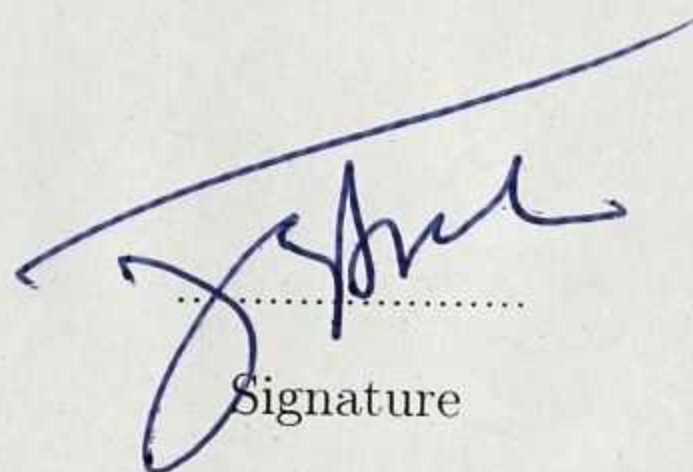
This thesis is a true work of the undersigned candidate and it has not been submitted in any form to any organization, institution or body for the award of any degree. All inclusions as well as references from works of previous authors have been duly acknowledged.

Daniel Gyamfi  
(PG6201511)

  
Signature

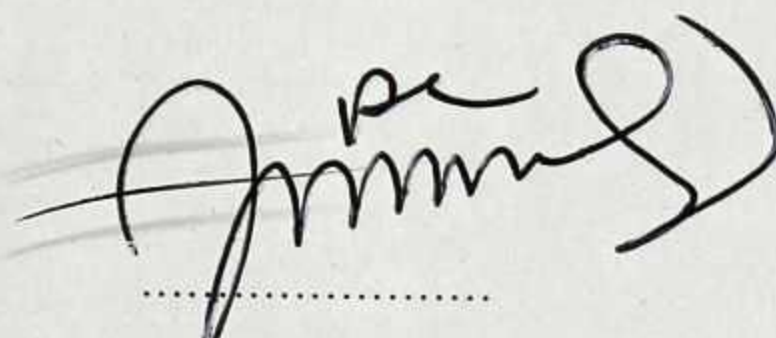
  
Date

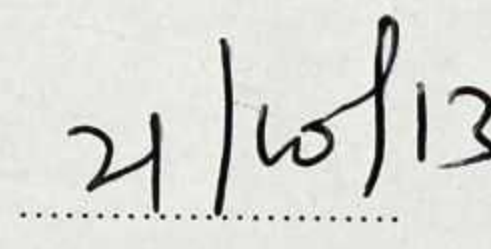
Certified By  
Dr. J. Ackora-Prah

  
Signature

  
Date

Prof. S. K. Amponsah  
Head of Department

  
Signature

  
Date

# DEDICATION

This work is dedicated to my beloved wife Dorcas and children Darius, Ecclezhiah and Esther and to all those who will benefit from this work.

LIBRARY  
KWAME NKRUMAH  
UNIVERSITY OF SCIENCE & TECHNOLOGY  
KUMASI



# ACKNOWLEDGEMENT

I will praise you Oh Lord with all my heart. I am always thankful to the Almighty God for perseverance and achievements.

I wish to thank Dr. J. Ackora-Prah, my supervisor, for his whole-hearted cooperation and great encouragement given in this successful endeavour. May his heart desires come to fruition.

Many thanks to Prof. S. K. Amponsah, Mr. K. Darkwah Dr. R. K. Avuglah, Dr. K. Baah Gyamfi and Mr. E. Harris for their immense support both physically and spiritually towards the completion of this work. God bless them all.

I am also indebted to the entire staff of the Department of Mathematics for their support towards the completion of this work. God richly bless you all.

Finally, I cannot hold back my appreciation to my fellow students, friends and loved ones especially Justice and Richard who in divers ways contributed to this work. Thank you all.



# ABSTRACT

Genetic algorithm is a search heuristic that mimics the process of evolution. This work discusses the concept and design procedure of Genetic Algorithm and explores a well established methodology of the literature to realise the workability and application of genetic algorithms.

Genetic algorithms are used to model the Travelling Salesman Problem. MATLAB simulations was carried out to find the optimal route of Zoomlion Ghana Limited as 7 6 9 4 1 2 3 5 8. Due to the unique nature of the operations of Zoomlion Ghana Limited, Kumasi, the optimal route was divided into two sub-routes. The two sub-routes were found to be 9 6 7 9 and 9 4 1 2 3 5 8 9. The corresponding distances (fitness) of the two routes were calculated to be 18.72 km and 66.22 km respectively, the total distance being 84.94 km. This shows that the total travelling distance of the company (i.e. 98.42 km), has been reduced by 13.48 km.



# Contents

DECLARATION	ii
DEDICATION	iii
ACKNOWLEDGEMENT	iv
ABSTRACT	v
1 INTRODUCTION	1
1.1 BACKGROUND . . . . .	1
1.2 STATEMENT OF PROBLEM . . . . .	2
1.3 OBJECTIVES . . . . .	3
1.4 METHODOLOGY . . . . .	3
1.5 PROBLEM JUSTIFICATION . . . . .	4
1.6 SCOPE/ORGANIZATION OF STUDY . . . . .	5
2 LITERATURE REVIEW	6
2.1 EVOLUTIONARY ALGORITHMS . . . . .	6
2.1.1 Evolutionary Strategies . . . . .	8
2.1.2 Evolutionary Programming . . . . .	9
2.1.3 Genetic Programming . . . . .	10
2.1.4 Genetic Algorithm . . . . .	12



2.2	A REVIEW OF WORKABILITY AND APPLICATION OF GENETIC ALGORITHMS (GAs) . . . . .	14
2.3	SIMULATED ANNEALING . . . . .	25
2.4	STOCHASTIC HILL CLIMBING . . . . .	26
2.4.1	Stochastic Hill Climbing Algorithm . . . . .	27
<b>3</b>	<b>METHODOLOGY</b>	<b>29</b>
3.1	WORKING PRINCIPLES OF GENETIC ALGORITHMS . . . . .	29
3.2	ENCODING . . . . .	32
3.2.1	Binary Encoding . . . . .	33
3.2.2	Permutation Encoding . . . . .	33
3.2.3	Value Encoding . . . . .	34
3.2.4	Tree Encoding . . . . .	34
3.2.5	Fitness Function . . . . .	34
3.2.6	Reproduction . . . . .	35
3.3	GENETIC ALGORITHM (GA) OPERATORS . . . . .	35
3.3.1	Selection Techniques in Genetic Algorithm (GA) . . . . .	35
3.3.2	Crossover . . . . .	42
3.3.3	Mutation . . . . .	54
3.4	REPLACEMENT . . . . .	59
3.4.1	Random Replacement . . . . .	60
3.4.2	Weak Parent Replacement . . . . .	60
3.4.3	Both Parents . . . . .	60
3.5	SEARCH TERMINATION (Convergence Criteria) . . . . .	61
3.6	HOW GENETIC ALGORITHMS WORK . . . . .	61
3.7	WHEN TO USE A GENETIC ALGORITHM . . . . .	62
3.8	BUILDING BLOCK HYPOTHESIS . . . . .	63
3.9	THE SCHEMA THEOREM . . . . .	64



3.10 NO-FREE-LUNCH THEOREM . . . . .	66
3.11 DISTINCTION BETWEEN GENETIC ALGORITHMS WITH OTHER OPTIMIZATION TECHNIQUES . . . . .	67
3.12 THE FLOYD-WARSHALL ALGORITHM . . . . .	68
<b>4 GENETIC ALGORITHM MODEL FOR TRAVEL SALESMAN PROBLEM (TSP)</b>	<b>69</b>
4.1 MODEL . . . . .	70
4.2 DATA . . . . .	71
4.3 ENCODING . . . . .	73
4.4 INITIAL POPULATION . . . . .	77
4.5 CROSSOVER AND MUTATION . . . . .	78
4.6 FITNESS FUNCTION . . . . .	81
4.7 OPTIMAL ROUTE WITH DISPOSABLE SITES . . . . .	83
4.8 PERFORMANCE COMPARISON . . . . .	84
<b>5 CONCLUSION AND RECOMMENDATION</b>	<b>86</b>
5.1 CONCLUSION . . . . .	86
5.2 RECOMMENDATION . . . . .	87
<b>REFERENCE</b>	<b>88</b>
<b>APPENDIX</b>	<b>93</b>
<b>A FLOW CHART SHOWING GENETIC ALGORITHM MODEL FOR TSP</b>	<b>94</b>



# Chapter 1

## INTRODUCTION

Evolutionary algorithms (EAs) are population-based meta heuristic optimization algorithms that use biology-inspired mechanisms and survival of the fittest idea in order to refine a set of solution iteratively. The last decade has witnessed many researches in solving combinatorial optimization problems (COPs). Numerous algorithms were proposed for applying to this problem. One of these algorithms is named Genetic Algorithm.

Genetic algorithms (GAs) are computing algorithms constructed in analogy with the process of natural evolution where the elements of search space are binary strings or arrays of elementary types. They are algorithms for optimization of hard combinatorial optimization problems in a quick way and its solution is reliable and accurate. Once again, the advantage of applying GAs to hard combinatorial optimization problems such as Travel Salesman problem (TSP) lies in the ability to search the solution space in a broader way than heuristic methods based upon neighbourhood search.

### 1.1 BACKGROUND

Charles Darwin stated the theory of natural evolution in the origin of species. Over many generations, natural populations evolve according to the principle of natural selec-



tion (survival of the fittest) to reach certain remarkable tasks. Natural evolution works so well in nature, as a result it should be interesting to simulate natural evolution and develop a method, which solves concrete, and search optimization problems.

In nature, an individual in population competes with each other for resources such as food, shelter and so on. Also members of the same species often compete to attract mate for reproduction.

Those individuals which are most successful in surviving and attracting mates will have relatively large number of offspring. Poorly performing individuals will produce few to even no offspring at all. This means that the genes from the highly adapted or " fit " individuals will spread to an increasing number of individuals in each successive generation. The recombination of good characteristics from different ancestors can produce " best fit " offspring whose fitness is better than the parent. In this way, species evolve to become more and well suited to their environment.

In 1975, Holland developed this idea to describe how to apply the principles of natural evolution to optimization problems and built the first Genetic Algorithms. Holland's theory has been further developed and now Genetic Algorithms (GAs) stand up as a powerful tool for solving search and optimization problems from a variety of sources by simulating evolution. Today GAs are used to solve complicated optimization problems like, job shop scheduling, games playing and Travel Salesman Problem (TSP).

## **1.2 STATEMENT OF PROBLEM**

There are several issues that makes waste collection a problem. Apart from the fact that some road networks are bad, most settlement are badly planned and this poses a great challenge in transporting solid waste. These problems also results in too much



time being used in the collection and transporting process.

In an attempt to carry out their duties, the Zoomlion Limited end up spending more on their process due to the problem of routing (Travelling Salesman Problem) leading to a higher drop of income (drift to their income). Some attempt has been made to optimize their routing problem which resulted in a reasonable cost benefit analysis as per Samson et al (2012)

However in their proceeds the application of Ant Colony Algorithm was implemented but in an ideal case as reviewed by literature, Genetic Algorithm is known to have produce a better result than Ant Colony. To this assertion, this thesis sort to approach the routing problem faced by Zoomlion Limited, Kumasi using Genetic Algorithm (GA). However, Genetic Algorithm has not yet been applied to waste management in our country to conclude its effectiveness as a better approach as far as waste disposal is concerned.

### 1.3 OBJECTIVES

For the above stated problem to be resolved, it is the objective of this work;

1. to apply Genetic Algorithm to generate the optimal route for Zoomlion Ghana Limited
2. minimize the total travelling distance of the vehicles

### 1.4 METHODOLOGY

In order for these objectives to be accomplished, the following methods are required;

- acquire data from the company in a form of a secondary data. For consistency it is the same data that Samson's et al used.



- preprocessing of data using the Floyd-Washall's algorithm.
- application of genetic algorithm to the resulting output of Floyd-Washall's algorithm.
- performance comparison to some existing methods.

## 1.5 PROBLEM JUSTIFICATION

Through the years the travelling salesman problem (TSP) has attracted a lot of attention from academic researchers and industrial practitioners. There are several reasons for this. Firstly, the TSP is very easy to describe, yet very difficult to solve. No polynomial time algorithm is known with which it can be solved. This lack of any polynomial time algorithm is a characteristic of a class of NP complete problems, of which the TSP is a classic example. Secondly a lot of information is already known about the TSP, it has become a kind of "test" problem.

Moreover the TSP is broadly applicable to a variety of routing and scheduling problems, new combinatorial optimization methods are often applied to the TSP so that an idea can be formed of their usefulness. Finally, a great number of problems actually treated with heuristic techniques in artificial intelligence are related with the search of the best permutation of  $n$  elements.

Although, the main motivation behind this work is to find the optimal route (tour) to minimize the total distance travelled as Zoomlion Ghana Limited carry out their duties which will again minimize their operational cost. This work would also help to achieve the following.

- it would help manage waste in Kumasi and Ghana as a whole efficiently at a reduced cost.



- it would help in restructuring our road networks thereby facilitating the duties carried out by the employee.
- it would promote government and private sector investment since it will be seen as a profitable venture.
- huge investment into solid waste management would seek to promote competition in waste management companies thereby ensuring regular collection of solid waste and hence having cleaner and healthier communities.

## 1.6 SCOPE/ORGANIZATION OF STUDY

The first chapter introduces the research, the background of this research, the problem statement, objective, a brief methodology of the genetic algorithm as well as problem justification. The second chapter reviews Evolutionary Algorithms, some heuristic methods and other earlier research carried out in Genetic Algorithms. A detailed methodology of the genetic algorithm is discussed in the chapter three and the application of GA to TSP is presented in chapter four. The fifth chapter of this work presents the conclusion and recommendation based on this research conducted.



## Chapter 2

# LITERATURE REVIEW

### 2.1 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EAs) are computer programs that attempt to solve complex problems by mimicking the process of Darwinian evolution. In an EA, a number of artificial creatures search over the space of the problem. They compete continually with each other to discover optimal areas of the search space. It is hoped that over time the most successful of these creatures will evolve to discover the optimal solution.

The artificial creatures in EAs, known as individuals, are typically represented by fixed length strings or vectors. Each individual encodes a single possible solution to the problem under considerations. For instance, in order to construct an EA to search the conformation space of a molecule, each angle of rotation around a flexible bond could be encoded as a real number. Concatenating these numbers gives a string which can be used within an EA. Thus, each individual would encode a specific set of torsion angles. EAs manipulate pools or populations of individuals. The EA is started with an initial population of size  $\mu$  comprising random individuals (that is, each value in every string is set using a random number generator). Every individual is then assigned a fitness value. To generate a fitness score the individual is encoded to produce a possible solution to



the problem. The value of this solution is then calculated using the fitness function. Population members with high fitness scores therefore represent better solutions to the problem than individuals with lower fitness scores. Following this initial phase the main iterative cycle of the algorithm begins. Using mutation (perturbation) and recombination operators, the  $\mu$  individuals is then formed from the  $\mu$  individuals in the current population and the  $\lambda$  children. the  $\lambda$  children are assigned fitness scores. A new population of  $\mu$  individuals is then formed from the  $\mu$  individuals in the current population and the  $\lambda$  children. This new population becomes the current population and the iterative cycle is repeated. At some point in the cycle evolutionary pressure is applied. That is, the Darwinian strategy of the survival of the fittest is employed and individuals compete against each other. This is achieved by selection based on fitness scores, with fitter individuals more likely to be selected. The selection is applied either when choosing individuals to parent children or when choosing individuals to form a new population.

Traditionally four main variants of the evolutionary algorithms are defined: L. Sekanina, (2005). Evolution strategies (ESs), developed in Germany by I. Rechenberg, (1973) and H. P. Schwefel, (1981); evolutionary programming (EP) originally developed by L. J. Fogel et al. (1966) and subsequently refined by D. B. Fogel, (1995); genetic programming developed by John Koza, (1992) and genetic algorithms (GAs) developed by J. H Holland, (1992) and thoroughly revealed by D. E. Goldberg, (1989). Each of these three algorithms has been proved capable of yielding approximately optimal solutions given complex, multi-modal, non-differential, and discontinuous search spaces. Success has also been achieved for noisy and time-dependent landscapes. A simple description of each technique is given here and more formal descriptions are given by T. Bäck and Schwefel, (1993) T. Bäck, (1996) and D. B Fogel, (1995).



### 2.1.1 Evolutionary Strategies

Bienert, Rechenberg and Schwefel developed *evolutionary strategies* (ES) in the 1960s, T. Bäck, (1996), for optimization purposes in industrial applications. Like genetic programming, ESs made no distinction between genotype and phenotype. Each individual is represented as a real-valued vector. In the simplest form, two-individuals ES employs a mutation operator which mutates each vector element. Mutation is regarded as the primary operator, and aggregates a normal-distribution random variable with zero mean and a pre-selected standard deviation value. If a child gets better fitness than the parent, the child becomes the new parent. Otherwise, the parent is mutated to create a new child. This selection schema is known as (1+1)-ES. It is interesting that this approach traditionally belongs to evolutionary algorithms, even if only a one-member population exists.

A pivotal feature of evolutionary strategies - which distinguishes the approach from others - is that an individual consists not only of an element of search space but also of a set of control (strategy) parameters. Evolutionary strategies operate with these *strategy parameters* (for instance, with the standard deviation and rotation angle) and these parameters are evolved together with variables within the individual. This is known as *self - adaptation*.

Current state-of-the-art evolutionary strategies provide various genetic operators and two major selection scenarios:  $(\mu + \lambda)$ -ES and  $(\mu, \lambda)$ -ES. Evolutionary strategy selects its parent solutions deterministically. The  $(\mu + \lambda)$ -ES picks the best  $\mu$  individuals from both child and parent populations. The  $(\mu, \lambda)$ -ES picks the best  $\mu$  individuals from just the child population.



### 2.1.2 Evolutionary Programming

Evolutionary programming (EP) was originally developed by L. J. Fogel et al. (1966) for the evolution of finite state machines using a limited symbolic alphabet encoding. Subsequently D. B. Fogel extended the EP to encode real numbers, thus providing a tool for variable optimization, D. B. Fogel, (1995). Individuals in the EP comprise a string of real numbers, as in ESs. EP differs from GAs and ESs in that there is no recombination operator. Evolution is wholly dependent on the mutation operation, which uses a Gaussian probability distribution to perturb each variable. The standard deviations correspond to the square root of a linear transform of the parents' fitness score (the user is required to parametrize this transform). To overcome parametrization problems associated with the linear transform Fogel developed meta-evolutionary programming (meta-EP), D. B. Fogel (1992). In meta-EP individuals encode both object variables and variances (one variance for each object variable). As in ESs the variances are self-adapted and used to control the Gaussian mutation operator.

Selection pressure is applied in the EP when forming a new population from parents and offspring of the mutation operator, using a mechanism called tournament selection. Stochastic  $q$ -tournament selection is employed, where  $q$  is a parameter of the algorithm. Let  $U$  be the union of all parents and offspring. For each member  $m$  of  $U$ ,  $q$  opponents are selected from  $U$  at random. A count is then made of the number of opponents that have worse fitness scores than  $m$ . The  $\mu$  individuals with the highest tournament counts go on to form the new population. Note that as  $q$  increases the selection pressure in the algorithm increases and the selection process becomes increasingly deterministic. One side-effect of this selection process is that the best individual is always present in the new population.

Like ESs, applications using EP are rare in computational chemistry, though some successes have been achieved by B. T. Luke (1994) and D.K. Gehlhaar et al. (1995). Like ESs, EP is a technique best suited to parameter optimization.



### 2.1.3 Genetic Programming

*Genetic programming* [Banzhaf, W., Nordin, P., Keller, R. E., Francone, F. D.; (1998)] was developed by John Koza in 1992, to allow automatic programming and program induction. It may be viewed as a specialized form of genetic algorithm, which manipulates with variable length chromosomes (i.e. with a specialized representations) using modified genetic operators. Unlike genetic algorithms, genetic programming does not distinguish between the search space and the representation space. However, it is not difficult to introduce genotype-phenotype mapping for any evolutionary algorithm formally (it could be one-to-one mapping in its simplest form).

The genetic programming search space includes not only the problem space but also the space of representation of the problem, i.e. genetic programming is also able to evolve representations. The theoretical search space of genetic programming is the search space of all possible (recursive) compositions over a primitive set of symbols (e.g. constants, functional symbols of different arities, etc.) which the programs are constructed over.

The progress are represented either as trees or in a linear form (e.g. machine-language instructions). Crossover is considered as a major operator for genetic programming (however, it has been criticized heavily, as can be seen from the discussion in Banzhaf, W., Nordin, P., Keller, R. E., Francone, F. D.; (1998)). It interchanges randomly chosen subtrees of the parent's trees without the syntax of the programs being disrupted. Mutation picks a random subtrees and replaces it by a randomly generated one. Genetic programming traditionally evolves symbolic expressions in a functional language like LISO. However, any useful structure may be utilized nowadays.

An evolved program can contain code segments which when removed from the program would not alter the result produced by the program (e.g. the instruction  $a = a + 0$ ), i.e. semantically redundant code segments. Such segments are referred to as *introns*. The



size of the evolved program can also grow uncontrollably until it reaches the maximum tree depth allowed while the fitness remains unchanged. This effect is known as *bloat*. The bloat is a serious problem in genetic programming, since it usually leads to time-consuming fitness evaluation of the effect of search operators. Once it occurs, the fitness almost always stagnates. The problem and the benefits of introns and bloat and their relation are discussed, for example, in Banzhaf, W., Nordin, P., Keller, R. E., Francone, F. D.; (1998).

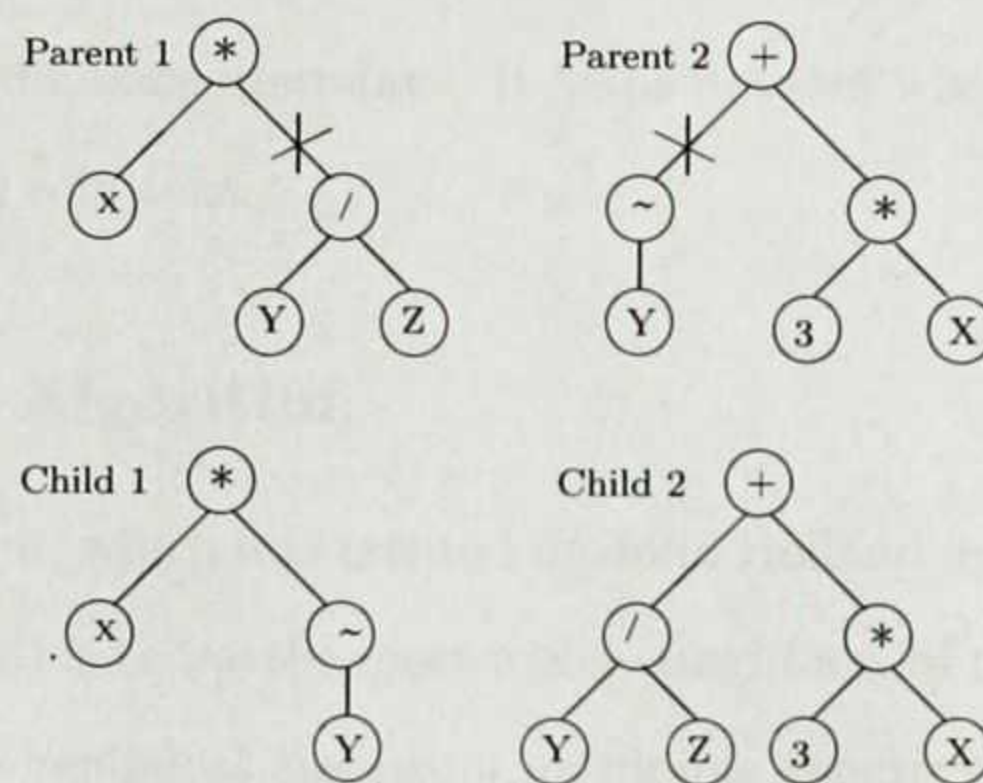


Fig. 2.1. A simple crossover in genetic programming

Genetic programming can also employ a number of advanced genetic operators. For instance, *automatically defined functions* (ADF) allow the definition of sub-programs that can be called from the rest of the program. Then the evolution is to find the solution as well as its decomposition into ADFs together.

The fitness function is either application specific for a given environment or it takes the form of symbolic regression. In all cases, the evolved program must be executed in order to find out what it does. The outputs of the program are usually compared with the desired outputs for given inputs. Generally, we cannot wait for the end of a program execution, since it is not certain if it ever stops (Bäck, T., 1996). Therefore, a terminating mechanism of the fitness evaluation process must be introduced.



Some papers compare genetic programming to machine-learning techniques. W. Banzhaf et al. (1998) claim that "the genetic programming representation is a superset of all other machine learning representations. Therefore, it is theoretically possible for properly designed genetic programming systems to evolve any solution that any other machine learning system can produce." J. Koza has also included similar statements in his book. As R. Poli mentioned, genetic programming is not quite ready as a replacement for standard software development. It is unclear whether it could ever be, but it is more than competitive in some domains nowadays. It helps in cases when humans do not have any idea of how to create solutions.

#### 2.1.4 Genetic Algorithm

The *genetic algorithm*, which was created by John Holland in 1973 and made famous by David Goldberg (1989) is today the most widely used form of the evolutionary algorithm. Its simplex form, the canonical algorithm, or simple genetic algorithm, work as follows, Bentley, P. (1999):

##### Simple Genetic Algorithm.

initialize population with random alleles

REPEAT

    evaluate individuals

    select individuals into 'mating pool'

    REPEAT

        take two parents from 'mating pool'

        randomly crossover to generate two offspring

        randomly mutate offspring

        place offspring into population

    UNTIL population is filled with new offspring

UNTIL termination condition is satisfied



This algorithm traditionally operates with a binary, integer, character or a real-valued vector stored in the chromosome of fixed length. One-point *crossover* is the simplest genetic recombination operator and, for example, it creates offspring **ABcdef** and **abCDEF** from two parents **ABCDEF** and **abcdef** if a random crossover point is 2, as seen in Fig. 2.2. Crossover is often used

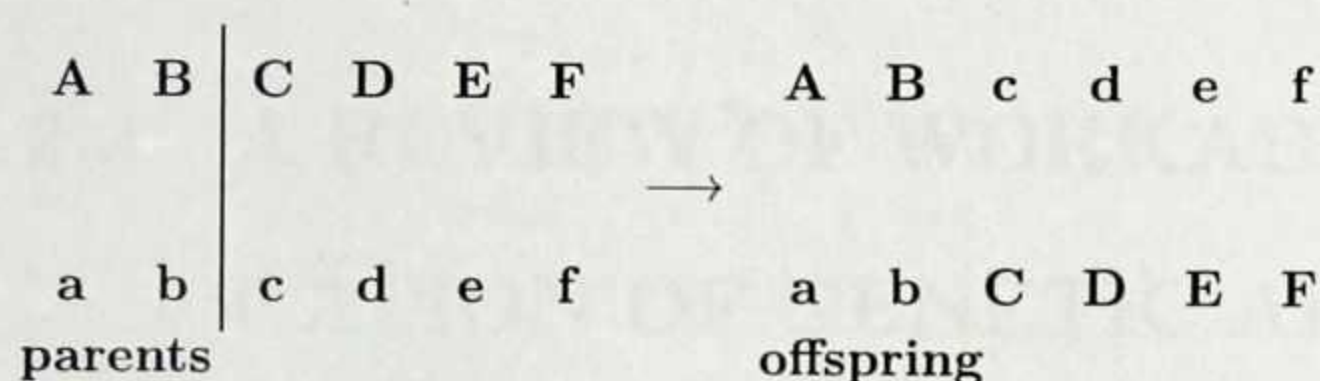


Fig. 2.2 A simple one-point crossover in a genetic algorithm

about 70% of the time to generate offspring, for the remaining 30% offspring are simply clones of their parents. *Mutations* occur rarely and usually swap a bit of the individual. *Selection* is typically implemented as a probabilistic operator, using the relative fitness  $p(a_i) = \phi / \sum_{j=1}^{\mu} \phi(a_j)$ , which determines the selection probability of an individual  $a_i$  ( $\mu$  denotes the population size). This method is also known as *roulette wheel* selection. The algorithm terminates when a sufficient solution is found or the time (generation) limit is exhausted.

For practical problems, the simple genetic algorithm is often considered as a basis for many enhancements, including: heuristic generation of the initial population, multi-point or more complicated crossover, elitism preserving the best individual for the next generation, more realistic selection, etc.

The selection mechanism is based solely on the fitness value. We would mention other variants to the roulette wheel. *Tournament selection* works by taking a random uniform



sample of a certain size  $q > 1$  from the population, selecting the best of these  $q$  individuals to survive for the next generation, and repeating the process until the new population is filled. In the case of *deterministic selection*,  $\mu$  parents create  $\lambda (\lambda > \mu)$  offspring and the best  $\mu$  offspring individuals are deterministically selected to replace the parents. An analysis that shows how different selection mechanisms provide selective pressures has been performed. [Back, T., 1996; Goldberg, D., 1989]

## 2.2 A REVIEW OF WORKABILITY AND APPLICATION OF GENETIC ALGORITHMS (GAs)

Chiong and Ooi Koon (2007), in their paper, a comparison between genetic Algorithms and Evolutionary Programming based on cutting stock problems stated that both GAs and Evolutionary (EP) are the well known optimization methods that belong to the class of Evolutionary Algorithms (EA) which have generally been recognised to have successfully solved many problems in recent years, especially with respect to engineering and individual problems. In their paper they looked at how these two methods tackle the one-dimensional cutting stock problem (CSP) and drew comparison to the effectiveness of GA and EP in solving CSP. Chiong and Ooi Koon (2007) proposed an improved algorithm based on the combination of GA and EP, using the steady-state replacement strategy and an order-based SR mutation. They believed that their proposed algorithm can improve the optimisation results of CSP significantly. In their concluding remarks, it was stated that WA are heuristic in nature and finding good design for the parameter settings is a major task and requires substantial experience and suggested that an extensive experiments need to be done to verify their proposed algorithm and make it more effective.

Wei (2009) described the application of Genetic Algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures. In this paper



Wei et al (2009) proposed a self-organised genetic algorithm for text clustering based on ontology method. They stated that the common problem in the fields of text clustering is that the document is represented as a bag of words, while the conceptual similarity is ignored. They took advantage of the thesaurus-based and corpus-based ontology to overcome this problem. However, they argued that the traditional corpus-based method is rather difficult to tackle. A transformed latent semantic indexing (LSI) model which can appropriately capture the associated semantic similarity was proposed and demonstrated as corpus-based ontology in this archive.

To investigate how ontology methods could be used effectively in text clustering. Wei et al (2009) implemented two hybrid strategies using various similarity measures. Their experimental result shown that Wei et al (2009) method of genetic algorithm in conjunction with the ontology strategy, the combination of the transformed LSI-based measure with the thesaurus-based measure apparently outperforms that of ... with traditional similarity measures. In conclusions Wei et al (2009) clustering algorithm also efficiently enhances the performance in comparison with standard GA and K-means in the same similarity environments.

A. Venables and G. Tan (2007) presented A 'Hand on' strategy for teaching Genetic Algorithms to the undergraduates. This paper describes a 'hands on' strategy to introduce and teach genetic algorithms to undergraduate computing students. By borrowing an analogical model from senior biology classes.

Venable et al (2007) described several introductory exercises that transport students from an illustration of natural selection in *Biston betula* moths, onto the representation and solution of differing mathematical and computing problems. In their paper their discussion cover terms such as population, generation, chromosome, gene, mutation and crossover in both their biological and computing contexts most importantly, the tasks underline the two key design issues of genetic algorithms: the choice of an appropriate algorithm representation and a suitable fitness function for each specific instances. Finally,



students were introduced to the notion of schema upon which genetic algorithms operate.

Kannan, Sasikumar and Devika (2010) recast a Genetic Algorithm approach for solving a closed loop supply chain model. A case study of battery recycling.

In this paper the authors developed a closed loop mixed integer linear programming model to determine the raw material level, production level, distribution and inventory level, disposal level and recycling level at different facilities with the objective of minimizing the total supply chain costs. The model is solved by the proposed heuristics based genetic algorithm (GA) and for smaller size problem the computational results obtained through GA are compared with the solutions obtained by (GAMS) optimization software.

Kannan et al (2010) paper it was revealed that for smaller size problems the GAMS software provides better results but with worst computational time but some larger-size real-world problems which cannot be solved by GAMS or other commercial software are only solved by the proposed heuristics based GA as a solution methodology for the larger problem sizes. Moreover, it was revealed that the proposed methodology performs very well in terms of both quality of solutions obtained and computational time. Based on the above validation, the authors proposed model in this research was tested with some real data extracted from the battery industry sources and achieved a cost reduction of 32.4% for the battery manufacturing industry by integrating the forward supply chain with the reverse supply chain.

Kannan et al (2010) concluded that the mathematical model is generalized enough to be relevant to most types of OEMs or hazardous material involved manufacturers.

Since the model was considered for single objective optimization, the authors proposed that in the future the multi objective model should be considered.

The problem of drawing graphs nicely contains several computationally intractable sub-problems It is on this note that Eloranta and Makinen (2001) presented a paper Tim



GA: A genetic algorithm for drawing undirected graphs. The authors indicated that it is natural to apply genetic algorithms to graph drawing. Their paper introduces a genetic (Tim GA) which nicely draws undirected graphs of moderate size. The aesthetic criteria used are the number of edge crossings, even distribution of nodes, and edge length deviation. Eloranta et al (2001) indicated that although Tim GA usually works well, there are some unsolved problems related to the genetic crossover operation of graphs and concluded that Tim GA's search is mainly guided by the mutation operations.

The weight constrained shortest path problem (WCSPP) is one of the most several known basic problems in combinatorial optimization. Because of its importance in many areas of applications such as computer science, engineering and operations research. Many researchers have extensively studied the WCSPP and Khaled et al (2005) presented a paper 'Reduction of search space by applying controlled Genetic operators for weight constrained shortest path problem(WCSPP).

In this paper, the authors have proposed a computationally fast method to find WCSPP in a graph. The Floyd-Warshall's algorithm was used to approximate the chromosome length which revealed that if the dimension of the chromosome is just equal to the dimension of the optimal shortest path, some controlled schemes of genetic operators on list chromosome representation were adopted. This approach gave a near optimum solution with smaller elapsed generation than classical GA technique. From further analysis on the matter a new generalised schema theorem is also developed from the philosophy of Holland's theorem.

Various applications of Genetic Algorithms to the problem of image segmentation are explored by Keri Woods (2007) on his paper Genetic Algorithms: Colour image segmentation to discuss the feasibility of using genetic algorithms to segment general colour images and also discuss the issues involved in designing such algorithms.

Keri Woods (2007) indicated that Genetic Algorithms are commonly used approach to



optimising the parameters of existing image segmentation algorithms and stated that the major decisions are choosing a method of segmentation to which genetic algorithms will be applied, finding a fitness function that is a good measure of the quality of image segmentation and finding a meaningful way to represent the chromosomes. Keri Woods (2007) used modified GAs and Hybrid GAs to solve this problem.

A new stereo matching approach using a genetic algorithm has been presented to improve the conventional stereo matching method by Han et al. (2001) on their paper entitled stereo matching using genetic algorithm with adaptive chromosomes. Han et al. (2001) adapted GAs as an efficient search technique to obtain corresponding points between stereo images.

Accordingly the authors also adopted genetic operators for the circumstances of stereo matching such that a  $2D$  disparity set was used as an individual and a fitness function was composed of intensity similarity and disparity smoothness constraints which are commonly used in stereo matching.

In order to acquire a consistent disparity map relative to the image appearance, Han et al. (2001) extracted a region of the input image, divided by zero-crossing points which was used in the determination of the chromosome shape. As a result, a disparity output that coincides with the input was obtained without any modification to the matching algorithm by Han et al. (2001) and consequently resulted in the improvements in output quality as well as in convergence speed.

A recast of genetic algorithms and the Evolution of Neural Networks for language processing by Jaine T. (2009) present ways in which he have used GAs to find which Neural Networks (NN) parameter values produce natural language task. In addition to this, the system has been modified and studied in order to evaluate ways in which coding methods in the GA and the NN can affect performance. In the case of GA coding, an evaluation method based on schema theory is presented. This methodology can help



determine optimal balances between different evolutionary operators such as crossover and mutation, based on the effect of different ways of resending words and sentences at the output layer is examined with binary and floating points schemes.

A computational technique based on a real coded genetic algorithm for microwave imaging purposes was discussed by Carsi. S. (2000). This study solves a non-linear inverse scattering problem for short range microwave imaging. Comparative analysis of the results obtained by approximate formulations and binary coded genetic algorithms (GAs) is made. Further, a hybrid version is presented and preliminary tested.

A dynamic routing control based on genetic algorithm can provide flexible real time management of the dynamic traffic changes in broadband networks. It was demonstrated through computer simulations using genetic algorithms by Shimamoto N. (2000). The proposed technique can generate the exact solution of path arrangement that keeps the traffic loss rate below the target value, even after changes in traffic.

A genetic algorithm for shortest path routing problem and the sizing of population was applied by Chang Wook et. al (2002). Variable length chromosomes and their genes have been used for encoding the problem. The proposed algorithm can cure all feasible chromosomes with a simple repair function. A population sizing equation is emphasized using computer simulations.

The multiple destination routing algorithm was formulated for finding a minimal cost tree which contains designated source and multiple destination nodes to satisfy certain constraints in a given communication network. The simulation studies for sparse and dense network demonstrate the robustness and efficiency of the proposed algorithm in terms of yielding high quality solutions.



A novel approach to solve very large scale integration (VLSI) channel and switch box routing problems was discussed by Lienig et. al (1997). This approach is based upon genetic algorithms that run on a distributed network of workstations. An extensive investigation shows the qualitatively better results and significantly reduction in occurrence of cross talk.

Usefulness of heuristic algorithms as the search method for diverse optimization problem is examined by Jang Sung Chun et. al (1998). Immune algorithms, genetic algorithms, evolutionary algorithms were compared on diverse optimization problems and the results reveal the out-performance of genetic algorithms. Based on genetic algorithms, surface permanent magnet synchronous motor is designed.

The problem of premature convergence in genetic algorithms optimization was discussed by Mori. N. et. al(1996). A novel thermodynamical genetic algorithm approach was suggested, which adopts the concepts of temperature and entropy in the selection rule. Simulated annealing was used to maintain diversity of the population. A comparison of the thermodynamical genetic algorithm approach with the simple genetic algorithm is carried out taking a knapsack problem.

Takagi-Sugeno-Kang (TSK) type recurrent fuzzy network. is proposed by Chia Feng Juang (2002), which develops from a series of recurrent fuzzy if-then rules with Takagi-Sugeno-Kang (TSK) type consequent parts. Takagi-Sugeno-Kang (TSK) type recurrent fuzzy network with supervised learning is suggested for the problems having on-line training data. To demonstrate the superiority of Takagi-Sugeno-Kang (TSK) type recurrent network, it is applied to dynamic system. By comparing the results the efficiency of Takagi-Sugeno-Kang (TSK) type recurrent fuzzy network is verified.

Optimization of antenna and scattering patterns using genetic algorithms was demon-



strated by Haupt et. al (1995). The proposed algorithm encodes each parameter into binary sequences called a gene and a set of gene is called chromosome. Several examples have been taken and implemented in MATLAB. For optimal solution of antenna patterns and back scattering radar cross-section pattern.

An orthogonal genetic algorithm approach for multimedia multicast routing was suggested by Qing Fu Zhang et al (1999). It can be investigated that the search space is statistically sound and is well suited for parallel implementation and execution. The implementation results reveal that the orthogonal genetic algorithms can find near optimal solution within moderate number of generations for practical problem sizes.

Dimeo R and Lee (1995) suggested a novel approach for boiler turbine control system using genetic algorithms. Proportional integral controller and a state feedback controller for non-linear multi-input-multi-output (MIMO) plant model is developed. Experimental results show the optimal control of boiler using genetic algorithms.

Channel assignment problem in hexagonal cellular network with two band buffering was discussed by Ghosh S. C. et al (2003). An algorithm is presented for solving channel assignment problem using elitist model of genetic algorithm which shows the optimal results within a reasonable computational time.

A vehicular wire antenna was designed using genetic algorithms used for both global positioning system (GPS) and indium systems. The antenna was simulated using numerical electromagnetic codes and then fabricated and tested. The voltage standing wave ratio (VSWR) and circular polarization radiation patterns were compiled and measured. Altshuler et. al (2000) suggested a new approach, which uses genetic algorithms in conjunction with electromagnetic code, produces configurations that are unique and seem to outperform more conventional design.



Design of direct form of a finite word length, finite impulse response (FIR) low pass filter was proposed by Xu and Daley (1995). The results of the proposed design techniques are compared with an integer programming technique and it is inferred from the results that genetic algorithm based technique outperforms the traditional approach.

Design of optimal disturbance rejection using genetic algorithms was suggested by Krohling and Rey (2001). The method was proposed to design an optimal disturbance rejection proportional integral derivative (PID) controller. A condition for disturbance rejection of control system is described which is further formulated as a constrained optimization problem. A constraint optimization problem to optimize integral of time and absolute error (ITAE) was tested by proportional integral derivative (PID) controller as applied to servo motor system. A double genetic algorithm was applied for solving constraint optimization problem. Simulation results demonstrate the performance and validity of the methods.

Length of Yagi-Uda antenna was optimized by Jones and Joines (1997) using genetic algorithms. To illustrate the capabilities of the method, the length and spacing of several Yagi-uda antennas are optimized for various performance characteristics.

Genetic algorithms were applied to pattern recognition problem by Raymer M. L. et. al (2000). A new approach is suggested to feature extraction in which feature selection and feature extraction was simultaneously done using genetic algorithms. The genetic algorithm optimizes a feature weight vector used to scale the individual features in the original pattern vectors.

Wei-Yen Wang and Li (2003) proposed a novel approach to adjust both the control points of B-spline membership functions and the weights of fuzzy neural network using



reduced form genetic algorithms (RGA). Simulation results show the faster convergence of the evolution process and effectiveness of reduced form genetic algorithms.

Problem of finding robust for flexible solutions for scheduling problems for real world application was suggested by Jensen M. T, (2003). Experimentally, it is shown that using a genetic algorithm, it is possible to find robust and flexible schedules with a low makespan.

Scheduling of hydraulically coupled plants can be approximated by genetic algorithms. An effective approach was suggested by Po-Hung Chen and Chang (1996) to 24 hrs ahead generation scheduling of hydraulically coupled plants. Experimental results show that the genetic algorithm approach obtains a more highly optimal solution than the conventional dynamic programming method.

Design of finite impulse response filter using genetic algorithms was suggested by Suckley D. et. al (1991). Genetic algorithm was used for automatic rapid and minimal computational complexity to design a filter.

An early paper in terms of genetic algorithms and its applications was presented by Tang K. S et. al (1996) which elaborates the genetic algorithm technology and its comparison with other optimization techniques. The genetic algorithm procedures were discussed to implement signal processing applications for infinite impulse response (IIR) adaptive filtering, time delay estimation, active noise control and speech processing, which are being implemented and described.

Hong Y. (2002) applied genetic algorithms on economic dispatch for congregation units considering multi-plant multi-buyer wheeling, which transmits microwaves to design load buses via wheeling. Varying the weights coefficient for penalty functions and determina-



tion of gene variables using genetic algorithms were discussed. The IEEE 30 and IEEE 188 bus system were used as test systems to illustrate the applicability of the proposed method.

Engineering applications of genetic algorithms were introduced by Man and Tang (1996). This study reveals how genetic algorithms can be integrated to form the framework of design tool for industrial engineers. An attempt has also been made to explain why, when and how to use genetic algorithms as an optimization tool for process controllers.

To find the shortest path, genetic algorithms can be used to encode a path in graph into a chromosome. The proposed approach have been tested by Gen M. et. al (1997) with different size from 6 nodes to 70 nodes and from 10 edges to 211 edges. The encouraging results using genetic algorithms can find the optimum very rapidly and with very high probability.

Genetic algorithm can be applied for system identification of both continuous and discrete time systems. They are effective in both domains for finding poles and zeroes. Simulations for minimum and non-minimum phase systems and a system with unmodelled dynamics were presented by Kristinsson and Dumont (1992).

Harmonic optimization of multilevel converter using genetic algorithms was proposed by Ozpineci et. al (2004). The optimization technique is applied to multilevel inverter to determine optimum switching angles for cascaded multilevel inverters for eliminating some higher order harmonics while maintaining the required fundamental voltage.

Simulations tuning of power system damping controller using genetic algorithms was done by Do Bomfim et. al (2000). Damping controller structures are assumed to be fixed consisting lead lag filter during the study. The study reveals the efficacy of genetic algorithms for the proposed system.



## 2.3 SIMULATED ANNEALING

In metallurgy and material science, annealing is a heat treatment of material with the goal of altering its properties such as hardness. Simulated annealing was originally inspired by formation of crystal in solids during cooling i.e., the physical cooling phenomenon. It is a method that simulates the thermodynamic process in which a metal is heated to its melting temperatures and then is allowed to cool slowly so that its structure is frozen at the crystal configuration of lowest energy. The slower the cooling, the more perfect is the crystal formed. By cooling, complex physical systems naturally converge towards a state of minimal energy. For an infinitely slow cooling, this method is certain to find the global optimum. The only point is that infinitely slow consists in finding the appropriate temperature decrease rate to obtain a good behaviour of its algorithm.

The system moves randomly, but the probability to stay in a particular configuration depends directly on the energy of the system and on its temperature as in Gibbs law.

Gibbs law gives this probability as:

$$p = e^{\frac{E}{kT}}$$

Where  $E$  stands for the energy,  $k$  is the Boltzmann constant and  $T$  is the temperature.

Research has revealed that Simulated Annealing algorithms with appropriate cooling strategies will asymptotically converge to the global optimum. In describing Simulated Annealing as used to solve a minimizing objective function of an optimization problem, the algorithm that follows is used.

### Algorithm for Simulated Annealing

Algorithm begins  $p_{new.g} \leftarrow initial\ guess$

$p_{cur} \leftarrow p_{new}$

$p^* \leftarrow p_{new}$



$t \leftarrow 0$  while termination Criterion is not satisfied do

$\delta E \leftarrow f(p_{new}.x) - f(p_{cur}.x)$

if  $\delta E \leq 0$  then

$p_{cur} \leftarrow p_{new}$

if  $f(p_{new}.x) < f(p^*.x)$  then  $P^* \leftarrow p_{cur}$

else

$T \leftarrow \text{get Temperature}(t)$

if random (generate)  $< e^{\frac{\delta E}{T_k}}$  then  $p_{cur} \leftarrow p_{new}$

update temperature

$t \leftarrow t + 1$

return  $P^*.x$

end

Simulated Annealing is a serious computer to Genetic Algorithms. Both Genetic Algorithms and Simulated Annealing are derived from analogy with natural system evolution and both deal with the same kind of optimization problem.

However, it is less efficient compared to the Genetic Algorithm since it only deals with one individual at each iteration. In light of this, Simulated Annealing is faster and simple or easier to implement. The Simulated Annealing can be used to determine the optimal layout of printed circuit board or the travelling salesman problem.

## 2.4 STOCHASTIC HILL CLIMBING

Hill climbing is a very old and simple search and optimization algorithm for continuous uni-modal functions. It uses a kind of gradient to guide the direction of the search. In principle, hill climbing algorithms perform a loop in which the currently known best solution is used to search for a new one. Stochastic hill climbing (also called stochastic



gradient descent) which is one of such methods consists of choosing randomly a solution in the neighbourhood of the current solution and retains this new solution only if it improves the objective function.

On multi-modal functions, the algorithm is likely to stop on the first peak it finds even if it is only a local optimum. This is a problem of hill climbing. To avoid this problem, it is advisable to repeat several hill climbs each time starting from a different randomly chosen point after the first local optimum. This method is sometimes known as iterated hill climbing. Once different local optimal points have been obtained, the global optimum can easily be observed. However, if the function of interest is very noisy with many small peaks then definitely stochastic hill climbing is not the best method. Nevertheless the advantage of this method is that it is easy to implement to achieve a fairly good solution faster.

Stochastic hill climbing usually starts from a randomly selected point. In describing the algorithm, below is a well stated outline.

### 2.4.1 Stochastic Hill Climbing Algorithm

Input:  $f$ : the objective function subject to minimization

Data:  $p_{new}$ : the new element created

Data:  $p^*$ : the (currently) best individual

Output:  $x^*$ : the best element found

1.  $p^* \leftarrow create$  (Implicitly:  $p^*.x \leftarrow gpm(p^*.g)$ )
2. while terminating Criterion is not satisfied do
3.  $p_{new}.x \leftarrow gpm(p_{new}.g)$



4. if  $f(p_{new}.x) < f(p^*.x)$  then  $p^* \leftarrow p_{new}$

5. return  $p^*.x$

6. end



# Chapter 3

## METHODOLOGY

### 3.1 WORKING PRINCIPLES OF GENETIC ALGORITHMS

The workability of genetic algorithm (GA) is based on Darwinian's theory of survival of the fittest. Genetic algorithms (GAs) may contain a chromosome, a gene, set of population, fitness, fitness function, breeding, mutation, and selection. Genetic algorithms (GAs) begin with a set of solutions represented by chromosomes, called population. Solutions from one population are taken and used to form a new population, which is motivated by the possibility that the new population will be better than the old one. Further, solutions are selected according to their fitness to form new solutions, that is, offspring. The above process is repeated until some condition is satisfied. Algorithmically, the basic genetic algorithm (GAs) is outlined as below;

1. Start with a randomly generated population of  $n$   $l$ -bit chromosomes (candidate solutions to a problem).
2. Calculate the fitness  $f(x)$  of each chromosome  $x$  in the population
3. Repeat the following steps until  $n$  offspring have been created:



- (a) Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done with replacement.
  - (b) With probability  $p_c$  (the "crossover probability" or "crossover rate"), cross over the pair at a randomly chosen point (chosen with uniform probability) to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents. (Note that here the crossover rate is defined to be the probability that two parents will cross over in a single point. There are also "multi-point crossover" versions of the GA in which the crossover rate for a pair of parents is the number of points at which a crossover takes place.)
  - (c) Mutate the two offspring at each locus with probability  $p_m$  (the mutation probability or mutation rate), and place the resulting chromosomes in the new population. If  $n$  is odd, one new population can be discarded at random.
4. Replace the current population with the new population.
  5. Go to step 2.

Each iteration of this process is called a *generation*. A GA is typically iterated for anywhere from 50 to 500 or more generations. The entire set of generations is called a *run*. At the end of a run there are often one or more highly fit chromosomes in the population. Since randomness plays a large role in each run, two runs with different random-number seeds will generally produce different detailed behaviours. GA researchers often report statistics (such as the best fitness found in a run and the generation at which the individual with that best fitness was discovered) average over many different runs of the GA on the same problem. The simple procedure just described is the basis for most applications of GAs. There are a number of details to fill in, such as the size of the population and the probabilities of crossover and mutation, and the success of the algorithm often depends greatly on these details. —



As a more detailed example of a simple GA, suppose that  $l$  (string length) is 8, that  $f(x)$  is equal to the number of ones in bit string  $x$  (an extremely simple fitness function, used here only for illustrative purposes), that  $n$  (the population size) is 4, that  $p_c = 0.7$  and that  $p_m = 0.001$ . (Like the fitness function, these values of  $l$  and  $n$  were chosen for simplicity. More typical values of  $l$  and  $n$  are in the range 50-1000. The values for  $p_c$  and  $p_m$  are fairly typical.)

The initial (randomly generated) population might look like this:

Chromosomes Label	Chromosomes String	Fitness
A	00000110	2
B	11101110	6
C	00100000	1
D	00110100	3

Table 3.01:

A common selection method in GAs *fitness-proportionate selection*, in which the number of times an individual is expected to reproduce is equal to its fitness divided by the average of fitness in the population. (This is equivalent to what biologists call “viability selection”.)

Genetic Algorithm is a random search optimization technique that has it roots in the principle of genetics. Before a GA can be run, a suitable coding (or representation) for the problem must be advised. We also require a fitness function, which assigns a figure of merit to each coded solution. During the run, parents must be selected for the reproduction , and recombined to generate offspring.

Genetic diversity or variation is a necessity for the process of evolution. The genetic operators used in GAs maintain genetic diversity. Genetic operators are analogues to



those which occur in real world. These operators are:

- Selection
- Crossover
- Mutation

As already mentioned, GAs evolve a population of individuals according to the process of natural selection. During this process, genetic operators create new individuals from highly fit old individuals. These operators are used after the coding process and the genetic algorithm enters the reproduction stage. Holland's introduction of a population based algorithm with selection, crossover, inversion and mutation was a major innovation. These operators are used at different stages of the GA.

In addition to these operators, there are some parameters of GAs. One important parameter is Population Size. The population size says how many chromosomes are in population, if there are only few chromosomes, then GA would have a few possibilities to perform crossover and only a small part of search space is explored, if there are many chromosomes, then GA slows down. Research shows that after some limit, it is not useful to increase population size, because it does not help in solving the problem faster. The population size depends on the type of encoding and the problem.

## 3.2 ENCODING

It is assumed that a potential solution to a problem may be represented as a set of parameters (for example, the dimensions of the beams is a bridge design). These parameters, known as *genes* are joined together to form a string of values referred to as a *chromosome*. Holland (Hol 75) first showed, and many still believe, that the ideal way for encoding is to use a binary alphabet for the string. For instance, if our problem is to maximize a function of three variables,  $F(x, y, z)$ , we might represent each variable



by a 10 – *bit* binary number (suitably scaled). Our chromosome would therefore contain three genes, and consist of 30 binary digits. In genetic terms, the set of parameters represented by a particular chromosome is referred to as *genotype*. The genotype contains the information required to construct an organism which is referred to as the *phenotype*. The same terms are used in GAs. For example, in a bridge design task, the set of parameters specifying a particular design is the genotype, while the finished construction is the phenotype. The fitness of an individual depends on the performance of the phenotype. This can be inferred from the genotype, i.e. it can be compared from the chromosome, using the fitness function. Various encoding techniques used in genetic algorithms (GAs) are binary encoding, permutation encoding, value encoding, and tree encoding.

### 3.2.1 Binary Encoding

It is the most common form of encoding in which the data value is converted into binary strings. Binary encoding gives many possible chromosomes with a small number of alleles. A chromosome is represented in binary encoding as shown in Fig. 3.01

Chromosome 1	<table><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	1	0	1	0	0	1
1	0	1	0	1	0	0	1		
Chromosome 2	<table><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr></table>	1	1	0	0	1	0	1	1
1	1	0	0	1	0	1	1		

Fig. 3.01. Binary Encoding

### 3.2.2 Permutation Encoding

Permutation encoding is best suited for ordering or queuing problems. Travelling salesman problem is a challenging problem in optimization, where permutation encoding is used. In permutation encoding, every chromosome is a string of numbers in a sequence as shown in Fig. 3.02



Chromosome 1	<table><tr><td>3</td><td>4</td><td>2</td><td>7</td><td>1</td><td>5</td><td>6</td><td>8</td></tr></table>	3	4	2	7	1	5	6	8
3	4	2	7	1	5	6	8		
Chromosome 2	<table><tr><td>8</td><td>3</td><td>6</td><td>1</td><td>2</td><td>7</td><td>4</td><td>5</td></tr></table>	8	3	6	1	2	7	4	5
8	3	6	1	2	7	4	5		

Fig. 3.02. Permutation Encoding

### 3.2.3 Value Encoding

Value encoding can be problems where some complicated values such as real numbers are used. Value encoding is a technique in which every chromosome is a string of some values and is used where some more complicated values are required. It can be expressed as shown in Fig. 3.03

Chromosome 1	2.4351	3.8609	4.110	6.783
Chromosome 2	north	south	East	West

Fig. 3.03 Value Encoding

### 3.2.4 Tree Encoding

Evolving expressions or programs such as genetic programming is the best suited technique for programming languages. In tree encoding, every chromosome is a tree of some objects, functions or commands in programming languages.

### 3.2.5 Fitness Function

A fitness function must be devised for each problem to be solved. Given a particular chromosome, the fitness function returns a single numerical "fitness", or "figure of merit", which is supposed to be proportional to the utility" or ability" of the individual which that chromosome represents. For many problems, particularly function optimization, it is obvious what the fitness function should measure; it should just be the value of the function. However, this is not always the case, for example with combinatorial optimization.



tion. In a realistic bridge design task, there are many performance measures we may want to optimize: strength/weight ratio, span, width, maximum load, cost, construction time or, more likely, some combination of all these. The fitness function quantifies the optimality of a solution (chromosome) so that a particular solution may be ranked against all the other solutions. The function depicts the closeness of a given solution to the desired result.

### **3.2.6 Reproduction**

At the reproduction phase of the GA, individuals are selected from the population and recombined, producing offspring which will comprise the next generation. Parents are selected randomly from the population using a scheme which favours the more fit individuals. Good individuals will probably be selected several times in a generation, poor ones may not be at all. Having selected two parents, their chromosomes are recombined, typically using the operators of GAs.

## **3.3 GENETIC ALGORITHM (GA) OPERATORS**

The simplest form of genetic algorithm involves three types of operators: selection, crossover and mutation.

### **3.3.1 Selection Techniques in Genetic Algorithm (GA)**

After deciding on an encoding, the second decision to make in using a genetic algorithm is how to perform selection, that is, how to choose the individuals in the population that will create offspring for the next generation and how many offspring each will create. The purpose of selection is, of course, to emphasize the fitter individuals in the population in hopes that their offspring will in turn have even higher fitness. According



to Darwin's evolution theory "survival of the fittest", the best ones should survive and create new offspring. Selection has to be balanced with variation from crossover and mutation (the "exploitation/exploration balance"). Because, too strong selection means that suboptimal highly fit individuals will take over the population, reducing the diversity needed for further change and progress while, too weak selection will result in too slow evolution. Selection means extract a subset of genes from an existing population, according to any definition of quality. Every gene has a meaning, so one can derive from the gene a kind of quality measurement called fitness function. Following this quality (fitness value), selection can be performed. As was the case for encodings, numerous selection schemes have been proposed in the GA literature. Some of the most common methods will be described. These descriptions do not provide rigorous guidelines for which method should be used for which problem; this is still an open question for GAs. All the various selection operators essentially do same thing. They pick from current population the strings of above average and insert their multiple copies in the mating pool in a probabilistic manner. The Selection operators are also called Reproduction operators. The most commonly used methods of selection that will be described are:

- Roulette wheel selection
- Boltzmann Selection
- Rank Selection
- Steady state selection
- Tournament selection

### **Roulette Wheel Selection (Fitness-Proportionate Selection)**

Roulette wheel selection, also known as Fitness Proportionate selection, is a genetic operator, used for selecting potentially useful solutions for recombination. In fitness-proportionate selection, the chance of an individual being selected is proportional to its



fitness, greater or less than its competitor's fitness. Conceptually, this can be thought as a game of Roulette.

The Roulette wheel simulates 8 individuals with fitness values  $F_i$ , marked at its circumference; e.g., the 5<sup>th</sup> individual has a higher Fitness than others, so the wheel would choose the 5<sup>th</sup> more than other individuals. The fitness of the individuals is calculated as the wheel spun  $n = 8$  times, each time selecting an instance, of the string, chosen by the wheel pointer.

The probability of  $i^{th}$  string is  $P_i = F_i / (\sum_{j=1}^n F_j)$ , where  $n = \text{number of individuals}$ , called population size;  $P_i = \text{probability of } i^{th} \text{ string being selected}$ ;  $F_i = \text{fitness for } i^{th} \text{ string in the population}$ . Because the circumference of the wheel is marked according to a string's fitness, the Roulette-wheel mechanism is expected to make  $F/\bar{F}$  copies of the  $i^{th}$  string.

Average fitness  $= \bar{F} F_j / n$ ; Expected count  $= (n + 8) * P_i$

Cumulative Probability  $= \sum_{i=1}^{n=5} P_i$

Typically, early in the search the fitness variances in the population is high and a small number of individuals are much fitter than the others. Under fitness proportionate selection, individuals and their descendants will multiply quickly in the population, in effect preventing the GA from doing any further exploration. This is known as "premature convergence." In other words, fitness proportionate selection early on often puts too much emphasis on exploration of highly fit strings at the expense of exploration of other regions of the search space. Later in the search, when all individuals in the population are very similar (the fitness variance is low), there are no real fitness differences for selection to exploit, and evolution grinds to a near halt. Thus, the rate of evolution depends on the variance of fitness in the population.



## Boltzmann Selection

Sigma scaling keeps the selection pressure more constant over a run. But often different amounts of selection pressure are needed at different times in a run for example, early on it might be good to be liberal, allowing less fit individuals to reproduce at close to the rate of fitter individuals, and having selection occur slowly while maintaining a lot of variation in the population. Later it might be good to have selection be stronger in order to strongly emphasize highly fit individuals, assuming that the early diversity with slow selection has allowed the population to find the right part of the search space. One approach to this is "Boltzmann Selection" (an approach similar to simulated annealing), in which a continuously varying temperature" controls the rate of selection according to a pre-set schedule. The temperature starts out high, which means that selection pressure is low (i.e., every individual has some reasonable probability of reproducing). The temperature is gradually lowered, which gradually increases the selection pressure, thereby allowing the GA to narrow in ever more closely to the best part of the search space while maintaining the appropriate" degree of diversity.

Simulated annealing is a method used to minimize or maximize a function. The system in thermal equilibrium at a temperature  $T$  has its energy distribution based on the probability defined by  $P(E) = \exp(-E/kT)$  where  $k$  is the Boltzmann constant.

The above expression suggests that a system at a higher temperature has almost uniform probability at any energy state, but at lower temperature it has a small probability of being at a higher energy state. Thus by controlling the temperature  $T$  and assuming that the selection process follows Boltzmann probability distribution, the convergence of the algorithm is controlled.



## Rank Selection

Rank selection is an alternative method whose purpose is also to prevent too-quick convergence. In the version proposed by Baker (1985), the individuals in the population are ranked according to fitness, and the expected value of each individual depends on its rank rather than on its absolute fitness. There is no need to scale fitness in this case, since absolute differences in fitness are obscured. This discarding of absolute fitness information can have advantages (using absolute fitness can lead to convergence problems) and disadvantages (in some cases it might be important to know that one individual is far fitter than its nearest competitor). Ranking avoids giving the far largest share of offspring to a small group of highly fit individuals, and thus reduces the selection pressure when the fitness variance is high. It also keeps up selection pressure when the fitness variance is low: the ratio of expected values if individuals ranked  $i$  and  $i + 1$  will be the same whether their absolute fitness difference are high or low.

The linear ranking method proposed by Baker is as follows: Each individual in the population is ranked in increasing order of fitness, from 1 to  $N$ . The user chooses the expected value  $Max$  of the individual with rank  $N$ , with  $Max > 0$ . The expected value of each individual  $I$  in the population at time  $t$  is given by

$$ExpVal(i, t) = Min + (Max - Min) \frac{rank(i, t) - 1}{N - 1}$$

where  $Min$  is the expected value of the individual with rank 1. Given the constraints  $Max > 0$  and  $i^{ExpVal(i, t) = N}$  (since population size stays constant from generation to generation), it is required that  $1 < Max < 2$  and  $Min = 2 - Max$ . At each generation the individual in the population are ranked and assigned expected values according to the above equation. Baker recommended  $Max = 1.1$  and showed that this scheme compared favourably to fitness proportionate selection on some selected test problems. Rank selection has a possible disadvantage: slowing down selection pressure means that the GA will in some cases be slower in finding highly fit individuals. However, in many cases



the increased preservation of diversity that results from ranking leads to more successful search than the quick convergence that can result from fitness proportionate selection. a variety of other ranking schemes (such as exponential rather than linear ranking) have also been tried.

### **Steady State Selection**

The GAs described so far has been generational. At each generation the new population consist entirely of offspring formed by parents in the previous generation (though some of these offspring may be identical to their parents). In some schemes, successive generations overlap to some degree, some portion of the previous generation is retained in the new population. The fraction individuals at each generation has been called the "generation gaps" (De Jong, 1975). In steady state selection, only a few individuals are replaced in each generation, usually a small number of the least fit individuals are replaced by offspring resulting from crossover and mutation of the fittest individuals. Steady states GAs are often used in evolving rule-based systems (e.g., classifier systems) in which incremental learning (and remembering what has already been learned) is important and in which members of the population collectively (rather than individually) solve the problem at hand.

### **Tournament Selection**

The fitness proportionate methods described above require two passes through the population at each generation: one ~~pass~~ to compute the mean fitness (and, for sigma scaling, the standard deviation) and one pass to compute the expected value of each individual. Rank scaling requires sorting the entire population by rank a potentially time consuming procedure. Tournament selection is similar to rank selection in terms of selection pressure, but it is computationally more efficient and more amenable to parallel im-



plementation. Two individuals are chosen at random from the population. A random number  $r$  is then chosen between 0 and 1. If  $r < k$  (where  $k$  is a parameter, for example 0.75), the fitter of the two individuals is selected to be a parent; otherwise the less fit individual is selected. The two are then returned to the original population and can be elected again.

### Example of Selection

Evolutionary Algorithm is to maximize the function  $f(x) = x^2$  with  $x$  in the integer interval  $[0,31]$ , i.e.  $x = 0, 1, 2, \dots, 30, 31$ .

1. The first step is encoding of chromosomes; 5-bits are used to represent integers up to 31
2. Assume that the population size is 4
3. Generate initial population at random. They are chromosomes or genotypes; e.g. 01101, 11000, 01000, 10011
4. Calculate fitness value of each individual.

(a) Decode the individual into an integer (phenotypes),

01101  $\rightarrow$  13; 11000  $\rightarrow$  24; 01000  $\rightarrow$  8; 10011  $\rightarrow$  19;

(b) Evaluate the fitness according to  $f(x) = x^2$ ,

13  $\rightarrow$  169; 24  $\rightarrow$  576; 8  $\rightarrow$  64; 19  $\rightarrow$  361.

5. Select parents for crossover based on their fitness in  $P_i$  (Probability of  $i_{th}$  string being selected). Out of many methods for selecting the best chromosomes, if roulette-wheel selection is used, then the probability of the  $i^{th}$  string in the population is

$$P_i = F_i / \left( \sum_{j=1}^n f_j \right), \text{ where}$$



$F_i$  is the fitness for the string  $i$  in the population, expressed as  $f(x)$

$P_i$  is probability of the string  $i$  being selected,

$n$  is number of individuals in the population, is population size,  $n = 4$

$n * p_i$  is expected count

String Number	Initial Population	X Value	Fitness $F_i$ $f(x) = x^2$	$P_i$	Expected Count $n * p_i$
1	01101	13	169	0.14	0.58
2	11000	24	576	0.49	1.97
3	01000	8	64	0.06	0.22
4	10011	19	361	0.31	1.23
Sum			1170	1.00	4.00
Average			293	0.25	1.00
Max			576	0.49	1.97

Table 3.02: Example of Selection

The string number 2 has maximum chance of selection

### 3.3.2 Crossover

Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user definable crossover probability. Crossover selects genes from parent chromosomes and create a new offspring. It could be said that the main distinguishing feature of a GA is the use of crossover. Crossover takes two individuals, and cuts their chromosomes strings at some randomly chosen position, to produce two head segments and two tail segments. The tail



segments are then swapped over to produce two new full length chromosomes Crossover is not usually applied to all pairs of individuals selected for mating. A random choice is made where the likelihood of crossover being applied is typically between 0.6 and 1.0. If crossover is not applied, offspring are produced simply by duplicating the parents. The crossover operators to be discussed are;

- One-Point Crossover
- Two-Point Crossover
- Uniform Crossover
- Arithmetic Crossover
- Heuristic Crossover
- Multi-Point Crossover (N-Point Crossover)
- Three Point Crossover
- Crossover with Reduced Surrogate
- Shuffle Crossover
- Precedence Preservative Crossover (PPX)
- Ordered Crossover
- Partially Matched Crossover (PMX)
- Cyclic Crossover

The operators are selected based on the way chromosomes are encoded



One-Point Crossover (Single Point Crossover)

One-point crossover is the simplest form. The idea here is, to recombine building blocks on different strings. Single point crossover has some shortcomings, though. For one thing, it cannot combine all possible schemas. For example, it cannot in general combine instances of 11\*\*\*\*\*1 and \*\*\*\*\*11\*\* to form an instance of 11\*\*11\*1. Likewise, schemas with long defining lengths are likely to be destroyed under single point crossover. Eshelman, Caruana, and Schaffer (1989) call this "positional bias": the schemas that can be created or destroyed by a crossover depend strongly on the location of the bits in the chromosome. Single point crossover assumes that short, low order schemas are the functional building blocks of strings, but one generally related bits together on a string, since particular bits might be crucial in more than one schema. The tendency of single point crossover to keep short schemas intact can lead to the preservation of hitch-hikers; bits that are not part of a desired schema but which, by being close on the string, hitch-hike along with the beneficial schema as it reproduces. Many People have also noted that single point crossover treats some loci preferentially: the segments exchanged between the two parents always contain the endpoints of the strings.

One point crossover operator randomly selects one crossover point and then copy everything before this point from the first parent and then everything after the crossover point copy from the second parent. The crossover would then look as shown below.

Consider the two parents selected for crossover;

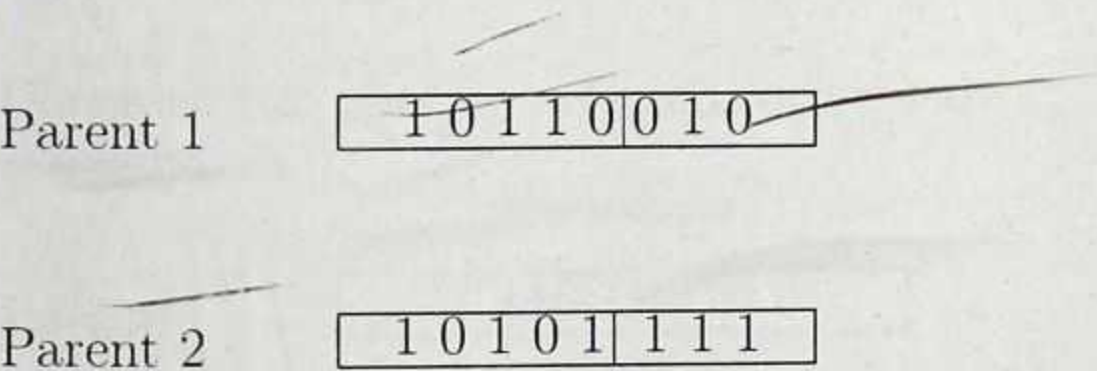


Fig. 3.04: Single Point Crossover



Interchanging the parents chromosomes after the crossover points -

The offspring produced are:

Offspring 1	<table><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	1	0	1	1	0	<table><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1
1	0	1	1	0						
1	1	1								
Offspring 2	<table><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	1	0	1	0	1	<table><tr><td>0</td><td>1</td><td>0</td></tr></table>	0	1	0
1	0	1	0	1						
0	1	0								

Fig. 3.05: Single Point Crossover

The symbol |, a vertical line, is the chosen crossover point.

Two Point Crossover

To reduce bias and this endpoint effect, many GA practitioners use two-point crossover, in which two positions are chosen at random and the segments between them are exchanged. Two-point crossover is less likely to disrupt schemas with large defining lengths and can combine more schemas than single-point crossover. In addition, the segments that are exchanged do not necessarily contain the endpoints of the strings. Again, there are schemas that two-point crossover cannot combine. GA practitioners have experimented with different numbers of crossover points. As mentioned, the two-point crossover operator randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these two points to produce two new offspring (children) for mating in the next generation.

Consider two parents selected for crossover:

Parent 1	110 110 10
Parent 2	011 011 00

Fig. 3.06: Two Point Crossover



Interchanging the parents chromosomes between the crossover points -

The offspring produced are:

Offspring 1	<table border="1"><tr><td>1 1 0</td><td>∴ 0 1 1</td><td>∴ 1 0</td></tr></table>	1 1 0	∴ 0 1 1	∴ 1 0
1 1 0	∴ 0 1 1	∴ 1 0		
Offspring 2	<table border="1"><tr><td>0 1 1</td><td>∴ 1 1 0</td><td>∴ 0 0</td></tr></table>	0 1 1	∴ 1 1 0	∴ 0 0
0 1 1	∴ 1 1 0	∴ 0 0		

Fig. 3.07: Two Point Crossover

In the figure above, the dotted lines indicate the crossover point.

### Uniform Crossover

Uniform crossover operator decides with some probability (known as the mixing ratio) which parent will contribute how the gene values in the offspring chromosomes. The crossover operator allows the parent chromosomes to be mixed at the gene level rather than the segment level (as with one and two point crossover). Uniform crossover works as follows; for each bit position 1 to L, randomly pick each bit from either of the two parent strings. This means that each bit is inherited independently from any other bit and that there is in fact no linkage between bits. It also means that uniform crossover is unbiased with respect to defining length. Some practitioners believe strongly in the superiority of "parameterized uniform crossover," in which an exchange happens at each bit position with probability  $p$  (typically  $0.5 \leq p \leq 0.8$ ). Parameterized uniform crossover has no positional bias; any schemas contained at different positions in the parents can potentially be recombined in the offspring. However, this lack of positional bias can prevent co-adapted alleles from ever forming in the population, since parameterized uniform crossover can be highly disrupted of any schema.

Consider the two parents selected for crossover;



Parent 1      

1	1	0	1	1	0	0	1	0	0	1	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Parent 2      

1	1	0	1	1	1	1	0	0	0	0	1	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Fig. 3.08: Uniform Point Crossover

If the mixing ratio is 0.5 approximately, then half of the genes in the offspring will come from parent 1 and other half will come from parent 2.

The possible set of offspring after uniform crossover would be:

Offspring 1      

1	1	0	1	1	0	0	1	0	0	0	1	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Offspring 2      

1	1	0	1	1	1	1	0	0	0	1	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Fig. 3.09: Uniform Point Crossover

**Arithmetic Crossover**

Arithmetic crossover operator linearly combines two parent chromosomes vectors to produce two new offspring according to the equations:

Offspring 1 =  $a * Parent1 + (1 - a) * Parent2$

Offspring 2 =  $(1 - a) * Parent1 + a * Parent2$

Where  $a$  is a random weighing factor chosen before each crossover operation.

Consider two parents (each of 4 float genes) selected for crossover:

Parent 1      (0.3)   (1.4)   (0.2)   (7.4)

Parent 2      (0.5)   (4.5)   (0.1)   (5.6)



Applying the above two equations and assuming the weighing factor  $a = 0.7$ , applying above equations we get two resulting offspring.

The possible set of offspring after arithmetic crossover would be:

Offspring 1      (0.36) (2.33) (0.17) (6.87)

Offspring 2      (0.402) (2.981) (0.149) (5.842)

### Heuristic

Heuristic crossover operator uses the fitness values of the two parent chromosomes to determine the direction of the search.

The offspring are created according to the equations:

$$\text{Offspring 1} = \text{BestParent} + r * (\text{BestParent} - \text{WorstParent})$$

$$\text{Offspring 2} = \text{BestParent}$$

Where  $r$  is a random number between 0 and 1.

It is possible that offspring 1 will not be feasible. This can happen if  $r$  is chosen such that one or more of its genes fall outside of the allowable upper or lower bounds. For this reason, heuristic crossover has a user defined parameter  $n$  for the number of times to try and find an  $r$  that results in a feasible chromosome. If a feasible chromosome is not produced after  $n$  tries, the worst parent is returned as offspring 1.



## Multi-Point Crossover (N-Point Crossover)

There are two ways in this crossover. One is even number of cross-sites and the other odd number of cross-sites. In the case of even number of cross-sites, cross-sites are selected randomly around a circle and information is exchanged. In the case of odd number of cross-sites, a different cross-point is always assumed at the string beginning

## Three Parent Crossover

In this crossover technique, three parents are randomly chosen. Each bit of the first parent is compared with the bit of the second parent. If both are the same, th bit is taken for the offspring otherwise; the bit from the third parent is taken for the offspring.

This concept is illustrated in Fig. 3.09

Parent 1	<table><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>	1	1	0	1	0	0	0	1
1	1	0	1	0	0	0	1		
Parent 2	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	0	1	1	0	1	0	0	1
0	1	1	0	1	0	0	1		
Parent 3	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr></table>	0	1	1	0	1	1	0	0
0	1	1	0	1	1	0	0		
Child	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	0	1	1	0	1	0	0	1
0	1	1	0	1	0	0	1		

Fig. 3.09. Three Parent Crossover

## Crossover with Reduced Surrogate

The reduced surrogate operator constrains crossover to always produce new individuals wherever possible. This is implemented by restricting the location of crossover points such that crossover points only occur where gene values differ.



## Shuffle Crossover

Shuffle crossover is related to uniform crossover. A single crossover position (as in single-point crossover) is selected. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled. This removes positional bias as the variables are randomly reassigned each time crossover is performed.

## Precedence Preservative Crossover (PPX)

PPX was independently developed for vehicle routing problems by Blanton and Wainwright (1993) and for scheduling problems by Bierwirth et al. (1996). The operator passes on precedence relations of operations given in two parental permutations to one offspring at the same rate, while no new precedence relations are introduced. PPX is illustrated in below, a problem consisting of six operations A-F.

The operator works as follows:

- A vector of length Sigma ( $\sigma$ ), sub  $i=1$  to  $m_i$ , representing the number of operations involved in the problem, is randomly filled with elements of the set 1,2.
- This vector defines the order in which the operations are successively drawn from parent 1 and parent 2.
- We can also consider the parent and offspring permutations as lists, for which the operations 'append' and 'delete' are defined.
- First we start by initializing an empty offspring.



- The leftmost operation in one of the two parents is selected in accordance with the order of parents given in the vector.
- After an operation is selected it is deleted in both parents.
- Finally the selected operation is appended to the offspring.
- This step is repeated until both parents are empty and the offspring contains all operations involved.
- Note that PPX does not work in a uniform-crossover manner due to the 'deletion-append' scheme used. Example is shown in Fig. 3.11.

Parent Permutation 1	A B C D E F
Parent Permutation 2	C A B F D E
Select Parent no. (1/2)	1 2 1 1 2 2
Offspring Permutation	A C B D F E

Fig. 3.11. Precedence Preservative Crossover (PPX)

### Ordered Crossover

Ordered two-point crossover is used when the problem is of order based, for example in U-shaped assembly line balancing etc. Given two parent chromosomes, two random crossover points are selected partitioning them into a left, middle and right portion. The ordered two-point crossover behaves in the following way: Child 1 inherits its left and right section from parent 1, and its middle section is determined by the genes in the middle section of parent 1 in the order in which the values appear in parent 2. A similar process is applied to determine child 2. This is shown in Fig. 3.12.



Parent 1     

4	2	1	3	6	5
---	---	---	---	---	---

Parent 2     

2	3	1	4	5	6
---	---	---	---	---	---

Child 1       

4	2	3	1	6	5
---	---	---	---	---	---

Child 2       

2	3	4	1	5	6
---	---	---	---	---	---

Fig. 3.112. Ordered Crossover

### Partially Matched Crossover (PMX)

PMX can be applied usefully in the TSP. Indeed, TSP chromosomes are simply sequences of integers, where each integer represents a different city and the order represents the time at which a city is visited. Under this representation, known as permutation encoding, we are only interested in labels and not alleles. It may be viewed as a crossover of permutations that guarantees that all positions are found exactly once in each offspring, i.e. both offspring receive a full complement of genes, followed by the corresponding filling in of alleles from their parents.

PMX proceeds as follows:

1. The two chromosomes are aligned.
2. Two crossing sites are selected uniformly at random along the strings, defining a matching section.
3. The matching section is used to effect a cross through position-by-position exchange operation.



4. Alleles are moved to their new positions in the offspring.

The following illustrates how PMX works

- Consider the two strings shown in Fig 3.13.
- Where the dots mark the selected cross points.
- The matching section defines the position-wise exchanges that must take place in both parents to produce the offspring.
- The exchanges are read from the matching section of one chromosome to that of the other.
- In the example, the numbers that exchange places are 5 and 2, 6 and 3, and 7 and 10.
- The resulting offspring are as shown in Fig. 3.14.

Name	9 8 4 . 5 6 7 . 1 3 2 1 0	Allele	1 0 1 . 0 0 1 . 1 1 0 0
Name	8 7 1 . 2 3 1 0 . 9 5 4 6	Allele	1 1 1 . 0 1 1 . 1 1 0 1

Fig. 3.13. Strings given

Name	9 8 4 . 2 3 1 0 . 1 6 5 7	Allele	1 0 1 . 0 1 0 . 1 0 0 1
Name	8 1 0 1 . 5 6 7 . 9 2 4 3	Allele	1 1 1 . 1 1 1 . 1 0 0 1

Fig. 3.14. Partially matched crossover

## Crossover Probability

The basic parameter in crossover technique is the crossover probability ( $P_c$ ). Crossover probability is a parameter to describe how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made



from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!). Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old population survive to next generation.

### 3.3.3 Mutation

A common view in the GA community, dating back to Holland's book *Adaptation in Natural and Artificial Systems*, is that crossover is the major instrument of variation and innovation in GAs, with mutation insuring the population against permanent fixation at any particular locus and thus playing more of a background role. This differs from the traditional positions of other evolutionary computation methods, such as evolutionary programming and clearly versions of evolution strategies, in which random mutation is the only source of variation. However, the appreciation of the role of mutation is growing as the GA community attempts to understand how GAs solve complex problems. Some comparative studies have been performed on the power of mutation versus crossover; for example, Spears (1993) formally verified the intuitive idea that, while mutation and crossover have the same ability for disruption of existing schemas, crossover is a more robust constructor of new schemas. Mühlenbein (1992), on the other hand, argues that in many cases a hill climbing strategy will work better than a GA with crossover and that "the power of mutation has been underestimated in traditional genetic algorithms." It is not a choice between crossover or mutation but rather the balance among crossover, mutation, and selection that is all important. The correct balance also depends on details of the fitness function and the encoding. Furthermore, crossover and mutation vary in relative usefulness over the course of a run. Precisely how all this happens still need to be elucidated. In my opinion, the most promising prospect for producing the right balances over the course of a run is to find ways for the GA to adapt its own mutation and crossover rates during a search. Some attempts at this will be described below.



Mutation is traditionally seen as a background operator, responsible for reintroducing inadvertently lost gene values (alleles), preventing genetic drift, and providing a small element of random search in the vicinity of the population when it has largely converged. Though it is widely held that crossover is the main force leading to a thorough search of the problem space, examples in nature however show that asexual reproduction can evolve sophisticated creatures without crossover; for example bdelloid rotifers. Biologists indeed, see mutation as the main source of raw material for evolutionary change.

Schaffer et al (SCLD 1989) did a large experiment to determine optimum parameters for GAs. They found that naïve evolution (just selection and mutation) performs a hill climb like search which can be powerful without crossover.

Mutation is applied to each offspring individual after crossover. It randomly alters each gene with a small probability (typically 0.001). Mutation provides a small amount of random search, and helps ensure that no point in the search space has a zero probability of being examined.

Mutation is an important part of the genetic search, in the sense that it helps prevent the population from stagnating at a local optima. Mutation is intended to prevent the search falling into a local optimum of the state space. That is mutation helps maintain genetic diversity from one generation of a population of chromosomes to the next. Mutation alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With the new gene values, the GA may be able to ~~arrive at~~ better solution than was previously possible. Mutation operators are of many types with the commonest ones being:

- Flip Bit (Flipping)



- Boundary
- Non-Uniform
- Uniform
- Gaussian
- Interchanging
- Reversing

The operators are selected based on the way chromosomes are encoded.

## Flip Bit

The mutation operator simply inverts the value of the chosen gene. i.e. 0 goes to 1 and 1 goes to 0.

This mutation operator (flip bit) can only be used for binary genes.

Consider the two original off-springs selected for mutation.

Original Offspring 1	1 1 0 1 1 1 1 0 0 0 0 1 1 1 1 0
Original Offspring 2	1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 0

Fig. 3.15. Flip Bit mutation

Invert the value of the chosen gene as 0 to 1 and 1 to 0.

The mutated off-spring produced are:

Mutated Offspring 1	1 1 0 0 1 1 1 0 0 0 0 1 1 1 1 0
Mutated Offspring 2	1 1 0 1 1 0 1 0 0 1 1 0 1 0

Fig. 3.16. Flip Bit mutation



## **Boundary**

This mutation operator replaces the value of the chosen gene with either the upper or lower bound for that gene (chosen randomly). This mutation operator can only be used for integer and float genes.

## **Non-Uniform**

The mutation operator increase the probability such that the amount of the mutation will be close to 0 as the generation number increase. The Non-uniform mutation operator prevents the population from stagnating in the early stages of the evolution and then allows the genetic algorithm to fine-tune the solution in the later stages of evolution. The mutation operator can only be used for integer and float genes just as the boundary operator.

## **Uniform**

The mutation operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. Just like boundary and non-uniform mutation operators this operator can also be only used for integer and float genes.

## **Gaussian**

This mutation operator adds a unit Gaussian distributed random value to the chosen gene. The new value is clipped if it falls outside of the user-specified lower or upper bounds for that gene. This operator is used only for integer and float genes.



Interchanging

Two random positions of the strings are chosen and teh bits corresponding to those positions are interchanged. This is shown in Fig. 3.17.

Parent	1 0 1 1 0 1 0 1
Child	1 1 1 1 0 0 0 1

Fig. 3.17 Interchanging

Reversing

A random position is chosen and the bits next to that position are reversed and child chromosome is produced. This is shown in Fig. 3.18.

Parent	1 0 1 1 0 1 0 1
Parent	1 0 1 1 0 1 1 0

Fig. 3.18. Reversing

Mutation Probability

The important parameter in the mutation technique is the mutation probability ( $P_m$ ). The mutation probability decides how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, it is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extreme.



### 3.4 REPLACEMENT

One of the most important stages of any breeding cycle is replacement which happens to be the last stage in the cycle. Once an offspring have been produced, there should be replacement which determines the current members of the population, if any should be replaced by the new offspring (solution). Two parents are drawn from a fixed size population, they breed two children, but not all four can return to the population, so two must be replaced. The technique used to decide which individual stay in a population and which are replaced in on a par with the selection in influencing convergence. Basically, there are two kinds of methods for maintaining the population; generational updates and steady state updates.

The basic generational update scheme consists in producing  $N$  children from a population of size  $N$  to form the population at the next time step (generation), and this new population of children completely replaces the parent selection. Clearly this kind of update implies that an individual can only reproduce with individuals from the same generation. Derived forms of generational update are also used like  $(\lambda + \mu)$ -update and  $(\lambda, \mu)$ -update. This time from a parent population of size  $\mu$ , a little of children is produced of size  $\lambda \geq \mu$ . Then the  $\mu$  best individuals from either the offspring population or the combined parent and offspring populations (for  $(\lambda, \mu)$ -and  $(\lambda + \mu)$ -update respectively), form the next generation.

In a steady state update, new individuals are inserted in the population as soon as they are created, as opposed to the generational update where as entire new generation is produced at each time step. The insertion of a new individual usually necessitates the replacement of another population member. (it leads to very strong selection pressure), or as the oldest member of the population, but those method are quite radical: Generally steady state updates use an ordinal based method for both the selection and the replace-



ment, usually a tournament method. Tournament replacement is exactly analogous to tournament selection except the less good solutions are picked more often than the good ones. A subtle alternative is to replace the most similar member in the existing population.

### **3.4.1 Random Replacement**

The children replace two randomly chosen individuals in the population. The parents are also candidates for selection. This can be useful for continuing the search in small populations, since weak individuals can be introduced into the populations

### **3.4.2 Weak Parent Replacement**

In weak parent replacement, a weaker parent is replaced by a strong child. With the four individuals only the fittest two, parent or child, return to population. This process improves the overall fitness of the population when paired with a selection technique that selects both fit and weak parents for crossing, but if weak individuals are discriminated against in selection the opportunity will never arise to replace them.

### **3.4.3 Both Parents**

Both parents replacement is simple. The child replaces the parent. In this case, each individual only gets to breed once. As a result, the population and genetic material moves around but leads to a problem when combined with a selection technique that strongly favours fit parents: the fit breed and then are disposed of.



### 3.5 SEARCH TERMINATION (Convergence Criteria)

The various stopping condition are listed as follows:

- **Maximum generations** - The genetic algorithm stops when the specified number of generations have evolved.
- **Elapsed time** - The genetic process will end when a specified time has elapsed.  
**Note:** If the maximum number of generation is reached before the specified time has elapsed, the process will end.
- **No change in fitness** - The genetic process will end if there is no change to the populations best fitness for a specified number of generations.  
**Note:** If the maximum number of generation has been reached before the specified number of generation with no changes is reached, the process will end.
- **Stall generations** - The algorithm stops if there is no improvement in the objective function for a sequence of consecutive generations.
- **Stall time limit** - The algorithm stops if there is no improvement in the objective function during an interval of time in seconds equal to **Stall time limit**.

The termination or convergence criterion finally brings the search to a halt. The following are the few methods of termination techniques.

### 3.6 HOW GENETIC ALGORITHMS WORK

Although genetic algorithms are simple to describe and program, their behaviour can be complicated, and many open questions exist about how they work and for what types of problems they are best suited. Much work has been done on the theoretical foundations of GAs. The traditional theory of GAs assumes that , at a very general level



of description, GAs work by discovering , emphasizing, and recombining good building blocks of solutions in a highly parallel fashion. The idea here is that good solutions tend to be made up of good building blocks, combinations of bit values that confer higher fitness on the strings in which they are present. Holland (1975) introduced the notion of *schemas* (or *schemata*) to formalize the informal notion of building blocks. A schema is a set of bit strings that can be described by a template made up of ones, zeroes, and asterisks; the asterisks representing wild cards (don't cares). For example, the schema  $H = 1 * * * * 1$  represents the set of all 6-bit strings that begin and end with 1 where  $H$  stands for "hyperplane".  $H$  is used to denote schemas because schemas define hyperplanes; planes of various dimensions in the  $l$  dimensional space of length- $l$  bit strings. The strings that fit this template (e.g., 100111 and 110011) are said to be *instances* of  $H$ . The schema  $H$  is said to have two defined bits (non-asterisks) or, equivalently, to be of *order* 2. Its *defining length* (the distance between its outermost defined bits) is 5. Here the term schema is used to denote both a subset of strings represented by such a template and the template itself.

### 3.7 WHEN TO USE A GENETIC ALGORITHM

The GA literature describes a large number of successful applications, but there are also many cases in which GAs perform poorly. Given a particular potential application, how do we know if GA is good method to use? There is no rigorous answer, though many researchers share the intuitions that if the space to be searched is known not to be perfectly smooth and uni-modal (consists of a single smooth hill), or is not well understood, or if the fitness function is noisy, and if the task does not require a global optimum to be ~~found~~, that is, if quickly finding a sufficiently good solution is enough, a GA will have a good chance of being competitive with or surpassing other weak methods. If a space is not large, then it can be searched exhaustively, and one can be sure that the



best possible solution has been found, whereas a GA might converge on a local optimum rather than on the globally best solution. If the space is smooth or uni-modal, a gradient-ascent algorithm such as steepest-ascent hill climbing will be much more efficient than a GA in exploiting the space's smoothness. If the space is well understood, search methods using domain-specific heuristics can often be designed to outperform any general-purpose method such as a GA. If the fitness function is noisy (e.g., if it involves taking error-prone measurements from a real-world process such as the vision system of a robot), a one-candidate-solution-at-a-time search method such as simple hill climbing might be irrecoverably led astray by the noise, but GAs, since they work by accumulating fitness statistics over many generations, are thought to perform robustly in the presence of small amounts of noise. These institutions, of course, do not rigorously predict when a GA will be an effective search procedure competitive with other procedures. A GA's performance will depend very much on details such as the method for encoding candidate solution, the operators, the parameter settings, and the particular criterion for success.

### 3.8 BUILDING BLOCK HYPOTHESIS

A genetic algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks (BBs).

The building block hypothesis is one of the most important criteria of how a genetic algorithm work. The importance of building blocks (BBs) and their role in the working of GAs have long been recognised (Holland, 1975; Goldberg, 1989). Furthermore, the following six conditions for a GA success have been proposed (Goldberg, Deb and Clark, 1992).

- Identify GAs which are the processing-building blocks.
- Ensure an adequate initial supply of raw BBs.



- Ensure growth of superior BBs.
- Ensure the mixing of BBs.
- Ensure good decisions among competing BBs and
- Solve problems with bounded BB difficulty.

One of the important conditions is to make sure that the GA is well supplied with a sufficient supply of the BBs required to solve a given problem. It is also equally important that proportion of the good ones in the population grow.

The first and the second task, that is guaranteeing the increase in market share of good BBs in a population has been recognised by Goldberg, Sactry and Laloza, (2001). The usual approach in achieving this is the schema theorem (Holland and Dejong (1997)).

### 3.9 THE SCHEMA THEOREM

Considering proportionate selection, single-point crossover, and no mutation the schema theorem may be written as follows

$$E(m(H, t + 1)) \geq m(H, t) \frac{f(H, t)}{\bar{f}(t)} \left\{ 1 - P_c \frac{\delta(H)}{L - 1} \right\}$$

where  $m(H, t + 1)$  is the expected number of individuals that represent the schema  $H$  at generation  $t + 1$ ,  $m(H, t)$  is the number of individuals that represent the schema  $H$  at counts generation  $t$ ,  $f(H, t)$  is the average fitness value of the individual containing schema  $H$  at generation  $t$ ,  $\bar{f}(t)$  is the average fitness of the population at generation  $t$ ,  $P_c$  is the crossover probability,  $\delta(H)$  is the defining length defined as the distance between the outer-most fixed positions of the schema and  $L$  is the string length.



Inspection of the schema theorem and an analysis of proportionate selection and single-point crossover indicates that the term  $m(H, t) \frac{f(H, t)}{f(t)}$  account for the selection and the term  $\{1 - P_c \frac{\delta(H)}{L-1}\}$  account for crossover operation.

It should be noted that the term representing the selection operator is exact and inequality occurs due to crossover operation. Some factors like crossover between identical individuals (self-crossover) are neglected.

The Schema theorem tells us that the proportion of schemata increases when they have above average fitness and relatively low crossover disruption.

However, the schema theorem as given by the equation above is restricted to proportionate selection and one-point crossover. This concern can be eliminated by identifying the characteristic form of schema theorem and substituting appropriate terms for other selection schemes and genetic operators. However, the generalized schema theorem can alternatively be written in the form

$$E(m(H, t + 1)) \geq \frac{f(H, t)}{f(t)} m(H, t) \{1 - P_c \frac{\delta(H)}{L-1}\} (1 - P_m)^{0(H)}$$

where  $(1 - P_m)^{0(H)}$  account for mutation operations and  $0(H)$  is the number of fixed bits in the schema.

These particular schemata are called building blocks and its applications are as follows

- It provides some tools to check whether a given representation is well-suited to a GA.
- The analysis of nature of the good schemata gives few ideas on the efficiency of genetic algorithm.



### 3.10 NO-FREE-LUNCH THEOREM

The No-Free-Lunch theorem states that without any structural assumption on a search or optimization problem, no algorithm can perform better than blind search.

To achieve a performance evaluation for an algorithm, it is not sufficient to demonstrate its better performance on a given set of functions. Instead of this, the diversity of an algorithm should be considered. The total number of possible algorithms as well should be computed and compared with the number of algorithms instances that a random search or a population based algorithm can have.

The question now is, how many different algorithms can be provided by such an algorithm class, and how does this number behave with respect to the total number of possible algorithms?

The answer gives us a ranking for algorithms according to their smaller or larger number of instances. It comes out that by such a ranking, random search is worst, while evolutionary approaches are (at least theoretically) able to provide any search sequence that is possible which implies that, population-based algorithms are principally able to cover the set of all possible algorithms.

The No-Free-Lunch theorem also provides information on the following:-

- the geometric interpretation of what it means for an algorithm to be well matched to a problem.
- brings insights provided by information theory into the search procedure.
- it provides that independent of the fitness function one cannot (without prior domain knowledge) successfully choose between two algorithms based on their previous behaviour.



### 3.11 DISTINCTION BETWEEN GENETIC ALGORITHMS WITH OTHER OPTIMIZATION TECHNIQUES

Genetic Algorithm differs substantially from all traditional search and optimization methods.

The four most significant differences are:

- It operates with coded versions of the problem parameters rather than parameters themselves i.e., GA works with the coding of solutions and not with the solution itself.
- Almost all conventional optimization techniques search from a single point but Genetic Algorithms always operate on a whole population of points (strings) i.e., it uses population of solutions rather than a single solution from searching. This plays a major role to the robustness of genetic algorithms. It improves the chance of reaching the global optimum and also helps in avoiding local stationary point.
- It uses fitness function for evaluation rather than derivatives. As a result, they can be applied to any kind of continuous or discrete optimization problem. The key point to be performed here is to identify and specify a meaningful decoding function.
- It uses probabilities transition operators while conventional methods for continuous optimization apply deterministic transition operators i.e., GAs do not use deterministic rules.



### 3.12 THE FLOYD-WARSHALL ALGORITHM

The Floyd-Warshall Algorithm compute the all pairs shortest path matrix. It uses a vectorised version of the Floyd-Warshall function. As in the dynamic programming algorithm, we assume that the graph is represented by an  $n \times n$  matrix with the weights of the edges.

$$w_{ij} = \begin{cases} 0 & \text{if } i = j \\ w_{ij} & \text{if } i \neq j \text{ and } (i, j) \in E \\ \infty & \text{if } i \neq j \text{ and } (i, j) \notin E \end{cases}$$

Output Format: an  $n \times n$  distance  $D = [d_{ij}]$  where  $d_{ij}$  is the distance from vertex  $i$  to  $j$ .

The objective is to find an estimate for the infinitesimal values using the Algorithm.

The algorithm for the Floyd-Warshall is as follows

Floyd-Warshall (w,n)

```
{for  $i = 1$  to  $n$  do           initialise
    for  $j = 1$  to  $n$  do
        {  $D^0[i, j] = w[i, j];$ 
           $pred[i, j] = nil;$  }
for  $k = 1$  to  $n$  do           Dynamic programming
    for  $i = 1$  to  $n$  do
        for  $j = 1$  to  $n$  do
            if ( $d^{k-1}[i, k] + d^{k-1}[k, j] < d^{k-1}[i, j]$ )
                 $d^k[i, j] = d^{k-1}[i, k] + d^{k-1}[k, j];$ 
                 $pred[i, j] = k;$ 
            else  $d^k[i, j] = d^{k-1}[i, j];$ 
return  $d^n[1..n, 1..n]$  }
```



## Chapter 4

# GENETIC ALGORITHM MODEL FOR TRAVEL SALESMAN PROBLEM (TSP)

In travelling salesman problem, salesman travels  $n$  cities and returns to the starting city with the minimal cost, he is not allowed to cross the city more than once.

The Travelling salesman problem (TSP) can be solved using genetic algorithm (GA) because the cities are random. The goal is to find the shortest distance between  $N$  different container locations. The path that the salesman takes is called a tour.

Operations of Zoomlion Ghana Limited (Kumasi) has been studied carefully. The operations of this company is to empty containers at eight (8) different sites in Kumasi (i.e. Subin sub-metro) starting from their station (Asokwa) to the Kuwait dump site popularly known as the Oti dump site with pick-ups and delivery.

This operations of the company can be modelled as a Travelling Salesman Problem (TSP). This is because all containers picked up at a site must be replaced with a new



one and each truck can pick only one container at a time.

A data was collected from Zoomlion Ghana Limited which has been used to create a set of routes on which the company must use to minimize the total travelling distance of the vehicles.

Testing every possibility for  $N$  city tour would be  $N!$ . This implies testing 8 container locations and their central packing space (i.e. Asokwa) making it 9 city tour, we would have to measure  $9! = 362880$  different tours. Calculating 362880 different tours for it fittest to determine the minimum distance would take years.

However genetic algorithm can be used to find a solution in the shortest possible time, although it might not find the best solution, it can find a near perfect solution for a 100 city tour in less than a minute.

There are couples of basic steps to solving the travelling salesman problem using GA which has been discussed below.

## 4.1 MODEL

The company uses two trucks with a specific capacity of one container at a time per truck. The trucks are all located at their central parking space at Asokwa. The eight (8) specific locations of the containers are;

- KATH Quarters
- Okomfo Anokye Teaching Hospital
- Amakom Market
- Amakom Division



- Central prisons
- 4BN Barracks
- Central Market
- Labour

However, all containers pick-up are to be emptied at only one disposal facility, Landfill site (Atonsus Kuwait)

The schedule of the two trucks and their assigned routes as partitioned are as follows. DS represents disposal site at Atonsus Kuwait (Oti landfill site)

Truck 1: Asokwa → 4BN → DS → KATH → DS → KATH Q → Central P → 4BN → Asokwa

Truck 2: Asokwa → Labour → DS → Amakom M → DS → Asafo → DS → Labour → Asokwa

The total distance travelled by both trucks in emptying all eight (8) containers was found to be 98.42 km, with 55.28 km for truck 1 and 43.14 km for truck 2.

## 4.2 DATA

Matlab is used to plot the graph of all the 9 cities using their local grid points (coordinates), Table 4.00, as shown in Fig. 4.01



653319.761	739756.815
651753.257	740794.647
651302.778	739970.992
652188.778	739970.953
652445.851	740402.992
654557.989	739511.174
654169.042	739547.516
651271.065	740208.777
651286.921	740414.581
651286.921	740414.579

Table 4.00: (X,Y) Coordinates

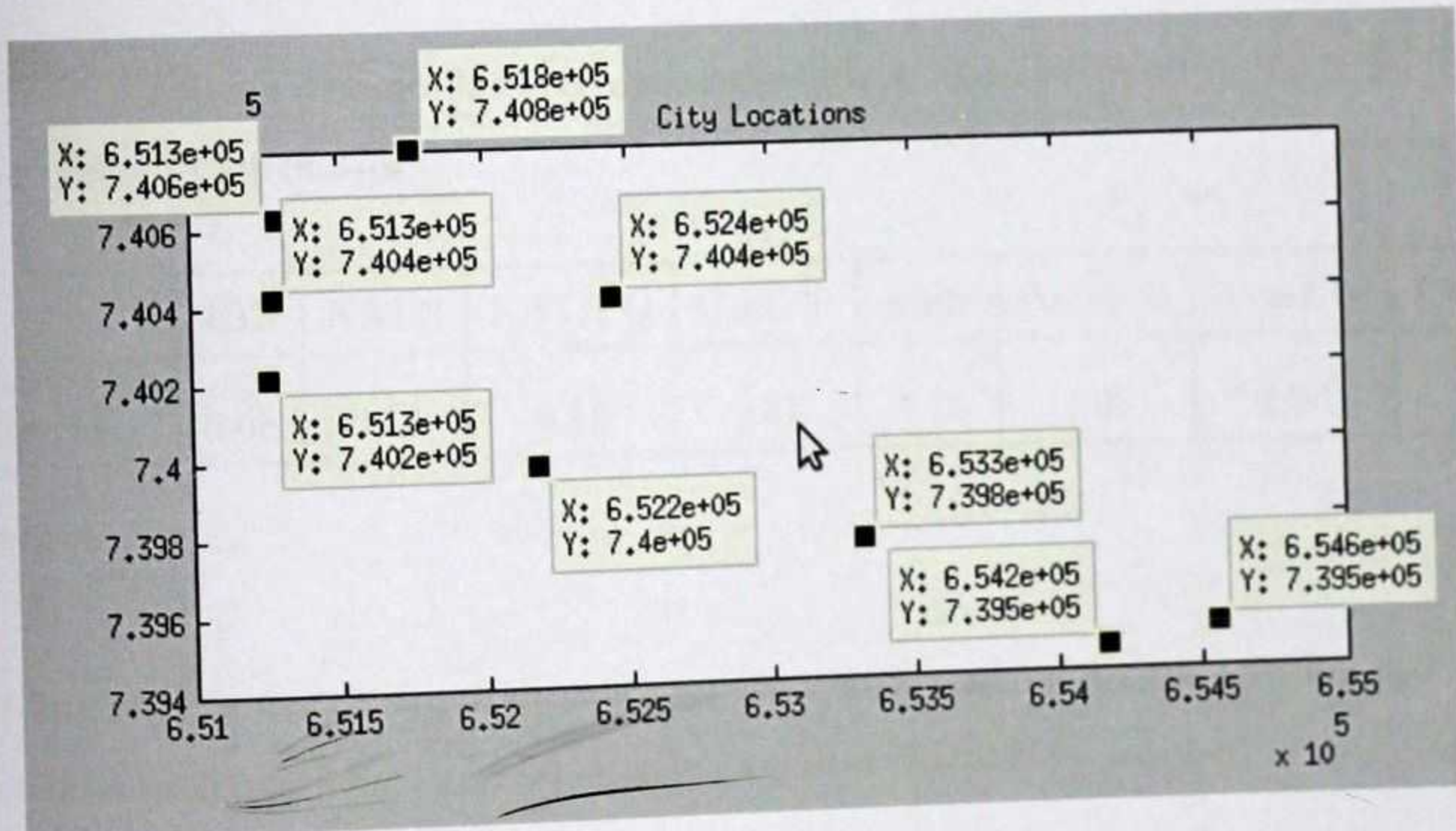


Fig. 4.01: Graph Showing the Coordinates



### 4.3 ENCODING

Permutation encoding is used. Numbers are assigned to all the 9 cities as shown below.

City 1  $\Rightarrow$  4BN

City 2  $\Rightarrow$  Komfo Anokye Teaching Hospital (KATH)

City 3  $\Rightarrow$  KATH Quarters

City 4  $\Rightarrow$  Central Prisons

City 5  $\Rightarrow$  Asafo Market

City 6  $\Rightarrow$  Amakom Division

City 7  $\Rightarrow$  Amakom Market

City 8  $\Rightarrow$  Labour

City 9  $\Rightarrow$  Central Station, Asokwa (i.e. tour starts from city with index 9)

Table 4.2 shows distance between disposable site at Atonsui Kuwait to the cities (container sites) in kilometres.

	4BN	KATH	KATH Q	Cent. P	Asafo	Amak D	Amak M	Labour	Asok
Disposable site (DS)	6.08	6.10	6.18	5.47	4.76	4.98	4.90	4.88	3.00

Table 4.01

Distance square matrix from all container sites in kilometres to the parking space of the trucks (starting point) and other container sites that will be required in partitioning the tour is shown in Table. 4.02



CITY	1	2	3	4	5	6	7	8	9
1	-	0.48	0.54	1.08	2.37	$\infty$	$\infty$	$\infty$	3.81
2	0.48	-	0.1	0.73	1.96	$\infty$	$\infty$	$\infty$	3.80
3	0.54	0.1	-	0.82	1.85	$\infty$	$\infty$	$\infty$	3.78
4	1.08	0.73	0.82	-	1.36	2.14	2.56	1.28	2.93
5	2.37	1.96	1.85	1.36	-	1.15	1.57	0.30	1.98
6	$\infty$	$\infty$	$\infty$	2.14	1.15	-	0.42	1.30	1.97
7	$\infty$	$\infty$	$\infty$	2.56	1.57	0.42	-	0.88	2.03
8	$\infty$	$\infty$	$\infty$	1.28	0.30	1.30	0.88	-	2.05
9	3.81	3.8	3.78	2.93	1.98	1.97	2.03	2.05	-

Table 4.02

The main objective is to find a set of least cost routes such that all containers are emptied. Since we cannot work with infinitesimal values in the distance matrix, the Floyd-Washall's algorithm is used to find approximate distances between the cities. The distance matrix after Floyd-Washall has been applied is shown in Table 4.03 and the corresponding distance square matrix is plot using Matlab as shown in Fig. 4.02



CITY	1	2	3	4	5	6	7	8	9
1	0	0.48	0.54	1.08	2.37	3.22	3.24	2.36	3.81
2	0.48	0	0.1	0.73	1.96	2.87	2.89	2.01	3.80
3	0.54	0.10	0	0.82	1.85	2.96	2.98	2.10	3.78
4	1.08	0.73	0.82	0	1.36	2.14	2.56	1.28	2.93
5	2.37	1.96	1.85	1.36	0	1.15	1.57	0.30	1.98
6	3.22	2.87	2.96	2.14	1.15	0	0.42	1.30	1.97
7	3.24	2.89	2.98	2.56	1.57	0.42	0	0.88	2.03
8	2.36	2.01	2.1	1.28	0.30	1.30	0.88	0	2.05
9	3.81	3.80	3.78	2.93	1.98	1.97	2.03	2.05	0

Table 4.03

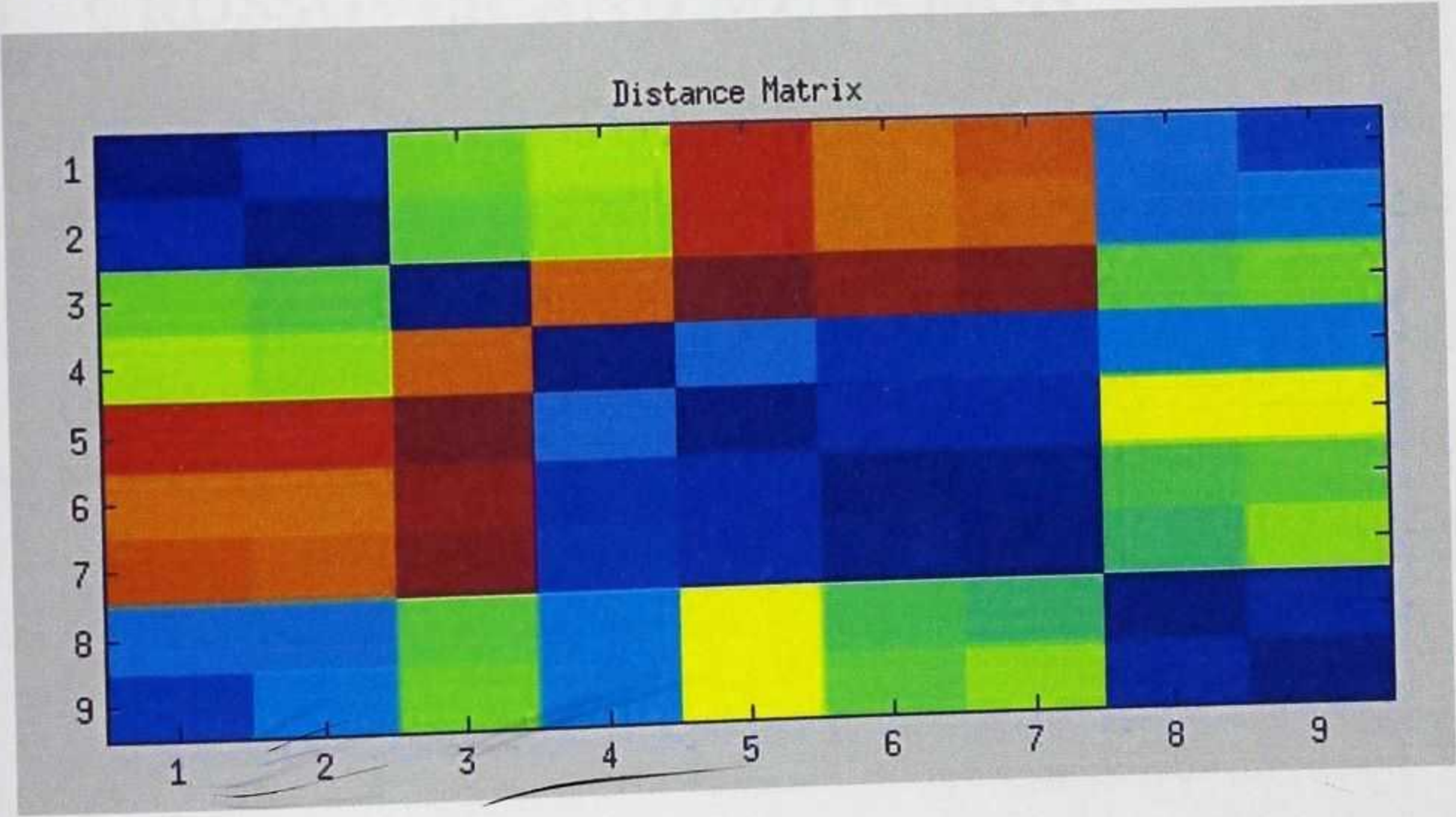


Fig. 4.02



## 4.4 INITIAL POPULATION

We create a group of many random tours in what is called a initial population. Population is a combination of chromosomes . We present the population as array of 1 2 3 4 5 6 7 8 9 chromosomes which represent all the different container locations and the central packing space as 9.

For each chromosome we calculate the length that is coded into it, actually this is the fitness of the tour. Fitness function is nothing but the minimum cost. Initially the fitness function is set to the maximum value and for each tour the cost is calculated and compared with the fitness function. The new fitness value is assigned to the minimum cost. Initial population is randomly chosen and taken as the parent.

## 4.5 CROSSOVER AND MUTATION

The two main problems for using GA to solve the TSP are choosing the proper methods of crossover and mutation that is used to combine the two parent tours to make the child tours. The cyclic crossover is used.

Given a random population of 8 7 9 1 2 3 4 5 chromosome. This means that we start from the central packing space we go to container site 1 to site 2 to site 3 to site 4 and to site 5 or from 9 we go to site 7 to site 8.

Unlike other methods of ~~crossover~~, cyclic crossover do not require crossover point. We choose the first gene from one of the parent chromosomes. If our parents are

Parent 1  $\Rightarrow$  1 2 3 9 4 5 6 7 8

Parent 2  $\Rightarrow$  8 4 6 9 1 2 3 5 7



Say we pick 1 from parent 1

Child  $\Rightarrow$  1 \* \* \* \* \*

We must pick every element from one of the parents and place it in the position it was previously in. Since the first position is occupied by 1, the number 8 from parent 2 cannot go there. So we must now pick the 8 from parent 1

Child  $\Rightarrow$  1 \* \* \* \* \* 8

This forces us to put the 7 in position 4 as in parent 1.

Child  $\Rightarrow$  1 \* \* \* 4 \* \* 7 8

since the same set of position is occupied by 1, 4, 7, 8 in parent 1 and parent 2, we finish by filling in the blank positions with the elements of those positions in parent 2.

Thus

Child 1  $\Rightarrow$  1 5 2 9 4 3 6 7 8

and we get child 2 from the complement of child 1. This type of crossover ensures that newly created chromosome is legal. A chromosome is legal if it is constructed according to the requirements of the salesman problem. In this crossover we notice that it is possible for us to end up with offspring being the same as the parents. This is not a problem since it will usually occur if parents have high fitness, in this case it could still be a good chance.



To solve the problem of not getting trapped in a local optimum we could use mutation. Due to the randomness of the process we will occasionally have chromosomes near a local optimum but not near the global optimum. Therefore the better the fitness the less chance of hiding the global optimum. So mutation is a completely random way of getting to a possible solution that would otherwise not be found.

Mutation is performed after crossover. The mutation index must decide whether to perform mutation on this child chromosome or not. We then choose a point to mutate and switch that point. For instance we had

Child  $\Rightarrow$  1 2 3 9 4 5 6 7 8

If we decide to choose the mutation point to be gene 2 and 7, the child would become

Child  $\Rightarrow$  1 7 3 9 4 5 6 2 8

We simply switched the places of genes 2 and 7. Another mutation that takes place is inverting a sub-tour in our child chromosome. So we have the chromosome

Child  $\Rightarrow$  1 2 3 9 4 5 6 7 8

And choose the same mutation points 2 and 7. The sub-tour between these two points is switched in reverse order

Child  $\Rightarrow$  1 7 3 9 4 5 6 2 8

After the mutation process the program makes a strict verification of the chromosome.



All the non legal chromosome are ignored.

The idea of the travelling salesman problem is to find a tour of a given number of cities, visiting each city once and returning to the starting city where the length of this tour is minimized.

The product finds a solution to the travelling salesman problem. For this purpose we use cities, chromosomes and populations, where our cities are the various container sites, chromosomes are the individual tours and the population is the combination of all the individual tours, i.e.  $9! = 362880$  tours.

Each city (container site) is situated on coordinates (x,y) on the map. Fig. 4.01 shows the coordinates of all the cities (container sites). In the working process a defined number of cities are being created. Then the program solves the travelling salesman problem for these cities.

## 4.6 FITNESS FUNCTION

The purpose of the fitness is to decide if a chromosome (tour) is good and how good it is. In the travelling salesman problem, the criteria for good chromosome (tour) is the length of the chromosome. The longer the tour that is coded, the better the chromosome. Calculation takes place during the creation of the chromosomes. Each chromosome is created and then its fitness function is calculated. The length of the tour is measured in pixels by the scheme of the tour.

$$\text{fitness tour} = \sum_{i=1}^n d_i$$



where  $n$  is the number of cities (container sites) and  $d_i$  is the distance between a city and a dump side (DS).

Matlab code is used to find the optimal route (tour) which is given as

7 6 9 4 1 2 3 5 8

and its corresponding graph shown in Fig. 4.03

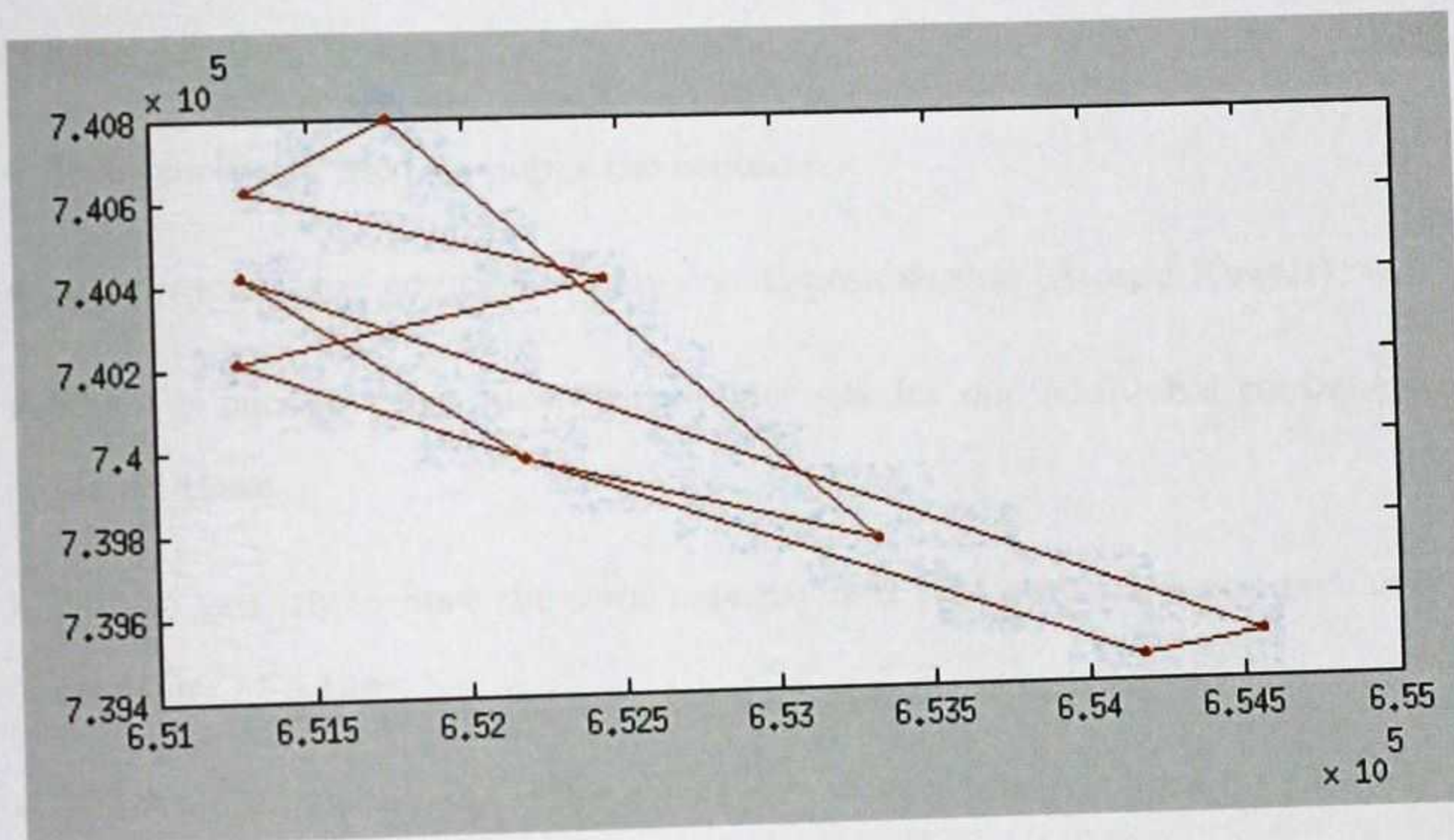


Fig. 4.03

Bearing in mind that the Zoomlion Ghana Limited uses two cars for picking up and delivery, the optimal route is rearranged for the two cars as represented below

Route 1: 7 6 9

Route 2: 4 1 2 3 5 8

Rearranged

9 6 7 9

9 4 1 2 3 5 8 9



## 4.7 OPTIMAL ROUTE WITH DISPOSABLE SITES

The optimal routes after they have been rearranged is as follows. This is done together with the disposal sites, represented by DS. Route 1:  $9 \rightarrow 6 \rightarrow \text{DS} \rightarrow 7 \rightarrow \text{DS} \rightarrow 9$

Route 2:  $9 \rightarrow 4 \rightarrow \text{DS} \rightarrow 1 \rightarrow \text{DS} \rightarrow 2 \rightarrow \text{DS} \rightarrow 3 \rightarrow \text{DS} \rightarrow 5 \rightarrow \text{DS} \rightarrow 8 \rightarrow \text{DS} \rightarrow 9$

Total distance for route 1 is calculated to be 18.72 km and total distance for route 2 is 66.22 km, hence the total distance of the two trucks is 84.94 km.

The fitness tour was calculated based on the following assumptions being used by Zoomlion Ghana Limited, Kumasi.

- Two trucks are used to empty the containers.
- All containers are emptied at only one disposable side (Atonsui Kuwait).
- There is enough space at each container site for one additional container to be placed there.
- All the two trucks have the same capacity and that each truck can pick only one container at a time.
- There are no traffics and other constraints after a tour has been established.

Then the optimal tour from the population depends on

- The shortest distance from the starting point to any of the pick-up location (container site).
- All the distances from the disposal site to the cities (pick-up) locations gives the minimum fitness value.



## 4.8 PERFORMANCE COMPARISON

Samson et al (2012) used the Ant colony algorithm and came out with four different feasible partitions for the 8 container location with each partition providing two routes.

Partition 1: 4BN  $\rightarrow$  KATH  $\rightarrow$  KATH Q  $\rightarrow$  Cent P  $\rightarrow$  Asokwa  
Asafo  $\rightarrow$  Amakom D  $\rightarrow$  Amakom M  $\rightarrow$  Labour  $\rightarrow$  Asokwa

Partition 2: 4BN  $\rightarrow$  KATH  $\rightarrow$  KATH Q  $\rightarrow$  Asafo  $\rightarrow$  Asokwa  
Cent P  $\rightarrow$  Amakom D  $\rightarrow$  Amakom M  $\rightarrow$  Labour  $\rightarrow$  Asokwa

Partition 3: 4BN  $\rightarrow$  KATH  $\rightarrow$  KATH Q  $\rightarrow$  Asokwa  
Cent P  $\rightarrow$  Asafo  $\rightarrow$  Amakom D  $\rightarrow$  Amakom M  $\rightarrow$  Labour  $\rightarrow$  Asokwa

Partition 4: 4BN  $\rightarrow$  KATH  $\rightarrow$  KATH Q  $\rightarrow$  Cent P  $\rightarrow$  Asafo  $\rightarrow$  Asokwa  
Amakom D  $\rightarrow$  Amakom M  $\rightarrow$  Labour  $\rightarrow$  Asokwa

Each pair of the routes under a single partition was run with the ant colony algorithm to obtain the total distances for TSP tours.



Partition 1	1	53.52 km	96.52 km
	2	43.00 km	
Partition 2	1	50.20 km	94.60 km
	2	44.40 km	
Partition 3	1	44.28 km	98.20 km
	2	53.92 km	
Partition 4	1	61.14 km	94.60 km
	2	33.46 km	

Table 4.04

Hence they concluded that an optimal routing which reduces the total distance of Zoomlion Ghana Limited, Kumasi is achieved to be Partition 2 or 4. This reduces the total distance by 3.82 km. However using the genetic algorithm, it has been proven that the total distance to be covered by the two trucks is 84.94 km which reduces the total distance travelled by the company by 13.48 km.



## Chapter 5

# CONCLUSION AND RECOMMENDATION

### 5.1 CONCLUSION

Genetic algorithms can be applied to solve combinatorial optimization problems (COPs) such as TSP. Genetic algorithms can find optimal solutions among the search space with the operators like crossover and mutation. They are not instantaneous, but can perform an excellent search. In this work, Genetic algorithm is tested to find the optimal route for the TSP which shows the superiority of Genetic Algorithm to Ant Colony Algorithm.

It is also proven that if the company in retrospective uses this work, they would be able to reduce their operational distance by 13.48 km thereby reducing their cost of fuelling their trucks which will intend reduce the cost of operations of the company. We are of the view that this work if adopted would increase the profit margin of the company and as well help the company to improve remuneration of all staff members of the company.



## 5.2 RECOMMENDATION

As an efficient optimization tool for combinatorial optimization problems, Genetic algorithm is very useful for solving problems which can be modelled as the TSP, thereby finding the optimal distance. In light of this capacity of the genetic algorithm, it is recommended that the GA should be used to solve Travelling Salesman Problem (TSP) instead of other traditional heuristic methods.

The following recommendations should also be considered by the company;

- It is advised that one personnel should not be allowed to ply only one route to ensure fairness.
- It is also recommended that the operations of Zoomlion i.e. (pick ups and delivery) should be done after 10:00 pm where there will be no traffic congestions.
- In future work, due to the success of the GA in this research, it is recommended that Genetic Algorithm should be considered for waste management problems in Ghana at large.



# REFERENCE

- [1] A. K. M. Khaled, AHSAN Talukder, TAIBUN Nessa, and KAUSHIK Roy, *Reduction of Search Space by Applying Controlled Genetic Operators for Weight Constrained Shortest Path Problem*. World Academy of Science, Engineering and Technology, pp. 207-211. (2005)
- [2] ANNE. Venables and GRACE Tan . *A 'Hands on' Strategy for teaching Genetic Algorithms to Undergraduates*. Journal of Information Technology Education, pp. 250-261. (2007)
- [3] ASHIS Ghosh and SATCHIDANANDA Dehuri, *Evolutionary algorithms for Multi-Criterion Optimization: A survey*. International Journal of Computing and Information Sciences, vol.2, pp. 38-57, (2004)
- [4] ALTSHULER E. E. *Design of a vehicular antenna for GPS/Iridium using a genetic algorithm*, IEEE Transaction on Antennas and propagation, 48(6), pp. 968-972, (2000)
- [5] BÄCK, T., and HOFFMEISTER, F. *Extended selection mechanism in genetic algorithms*. In R. K. Belwe and L. B. Booker, eds., Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann. (1991).
- [6] BÄCK, T., HOFFMEISTER, F., and SCHWEFEL, H. P. *A survey of evolution strategies*. In R. K. Belew and L. B. Booker, eds., Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan, (1991).



- [7] CHANG Wook Ahn and RAMAKRISHNA, R.S. *A genetic algorithm for shortest path routing problem and the sizing of populations*, IEEE Transactions on Evolutionary Computation, 6(6), pp. 566-579, (2002).
- [8] CHIA-FENG Juang. *A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms*, IEEE Transactions on Fuzzy Systems, 10(2), pp. 155-170, (2002).
- [9] DAVID B. Fogel, Lawrence J. Fogel. *An Introduction to Evolutionary Programming. Artificial Evolution* pp. 21-33, (1995)
- [10] DE JONG K. A. *Genetic algorithms are not functions optimizers*. In L. D. Whitley, ed., Foundations of Genetic algorithms 2. Morgan Kaufmann, (1993).
- [11] DIMEO R., and LEE K. Y. *Boiler-turbine control system design using a genetic algorithm*, IEEE Transactions on Energy Conversion, 10(4), pp. 752-759., (1995).
- [12] DO BOMFIM A. L. B., TARANTO G. N. and FALCAO D. M. *Simultaneous tuning of power system damping controllers using genetic algorithms*, IEEE Transactions on Power Systems, 15(1), pp. 163-169, (2000).
- [13] ERIC Krevise Prebys. *The genetic algorithm in Computer Science*. MIT undergraduate Journal of Mathematics, pp.165-170.
- [14] EUI-YOON Chung, KYU-PHIL Han, KUN-WOEN Song, SEOK-JE Cho, and YEON-HO Ha *Stereo matching using genetic algorithm with adaptive chromosomes*. The Journal of the pattern recognition society, vol.34 pp. 1-4, (2001).
- [15] FOGEL, D. B., and ATMAR, W., eds., *Proceedings of the second on Evolutionary Programming*. Evolutionary Programming Society, (1993).
- [16] GEN M., RUNWEI Cheng and DINGWEI Wang. *Genetic algorithms for solving shortest path problems*, IEEE Transactions on Evolutionary Computation, pp. 401-406, (1997).



- [17] KANNAN G., SASIKUMAR P. and DEVIKA K., *A genetic algorithm approach for solving a closed loop supply chain model: A case of battery recycling* Journal of Applied Mathematical Modelling, vol.34 pp. 655-670, (2010).
- [18] GHOSH S. C., SINHA B. P. and DAS N. *Channel assignment using genetic algorithm based on geometric symmetry*, IEEE Transactions on Vehicular Technology, 52(4), pp. 860-875, (2003).
- [19] GOLDBERG, D. E. *A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing*. Complex Systems 4, pp. 445-460, (1990).
- [20] GOLDBERG, D. E., DEB, K., and KORB, B. *Messy genetic algorithms revisited: Studies in mixed size and scale*. Complex Systems 4. pp. 415-444, (1990).
- [21] GOLDBERG, D. E. *Genetic Algorithms and Walsh Functions: Part I, A gentle introduction*. Complex Systems 3: pp. 129-152, (1998).
- [22] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading MA: Addison-Wesley, (1989).
- [23] GOLDBERG, D. E. *Simple Genetic Algorithms and the minimal deceptive problem*. In L. D. Davis, ed., Genetic Algorithms and Simulated Annealing. Morgan Kaufmann, (1987).
- [24] HAUPT, R. L. *An introduction to genetic algorithms for electromagnetic*, IEEE Antennas and Propagation Magazine, 37(2), pp. 7-15, (1995).
- [25] HONG Y. Y. and LI C. *Genetic algorithms Based Economic Dispatch for Cogeneration Units Considering Multiplant Multibuyer Wheeling*, IEEE Power Engineering Review, 22(1), pp. 66-69, (2002).



- [26] IVO F. Sbalzarini, SIBYLLE Müller and PETROS Koumoutsakos *Multiobjective optimization using evolutionary algorithms*. Proceedings of the summer program, Centre for Turbulence Research, pp. 63-71, (2000).
- [27] JANG-SUNG Chun, HYUNG-KYO Jung and SONG-YOP Hahn. *A study on comparison of optimization performances between immune algorithm and other heuristic algorithms*, IEEE Transactions on Magnetics, 34(5), pp. 2972-2975, (1998).
- [28] JENSEN M. T. *Generating Robust and flexible job shop schedules using genetic algorithms*, IEEE Transactions on Evolutionary Computation, 7(3), pp. 275-288, (2003).
- [29] JONES E. A. and JOINES W. T. *Design of Yagi-Uda antennas using genetic algorithms*. IEEE Transactions on Antennas and Propagation, 45(9), pp. 1386-1392, (1997).
- [30] KRISTINSSON K. and DUMONT G. A. *System identification and control using genetic algorithms*, IEEE Transactions on Systems, Man and Cybernetics, 22(5), pp. 1033-1046, (1992).
- [31] KROHLING R. A. and REY J. P. *Design of optimal disturbance rejection PID controllers using genetic algorithms*, IEEE Transactions on Evolutionary Computation, 5(1), pp. 78-82, (2001).
- [32] LIENIG, J. (1997). *A parallel genetic algorithm for performance-driven VLSI routing*, IEEE Transactions on Evolutionary Computation, 1(1), pp. 29-39.
- [33] MORI, N., YOSHIDA, J., TAMAKI, H. and NISHIKAWA, H. K. *A thermochemical selection rule for the genetic algorithm*, IEEE International Conference of Evolutionary Computation, 1995, pp. 188-193, (1996).



- [34] OZPINECI B., TOLBERT L. M. and CHIASSEON J. N. *Harmonic optimization of multilevel converters using genetic algorithms*, IEEE 35th Annual Power Electronics Specialists Conference, vol.5, pp. 3911-3916, (2004).
- [35] PARVIZ Ajideh *Schema theory-based pre-reading tasks: A neglected essential in the ESL Reading Class*. Journal of the reading matrix, vol.3, pp. 1-5, (2003).
- [36] PO-HUNG Chen and HONG-CHAN Chang. *Genetic aided scheduling of hydraulically coupled plants in hydro-thermal coordination*, IEEE Transactions on Power Systems, 11(2), pp. 975-981, (1996).
- [37] QING-FU Zhang and YIU-WING Leung. *An orthogonal genetic algorithm for multimedia multicast routing*, IEEE Transactions on Evolutionary Computation, 3(1), pp. 53-63, (1999).
- [38] RAYMER M. L., PUNCH W. F., GOODMAN E. D., KUHN L. A., and JAIN, A. K. *Dimensionality reduction using genetic algorithms*, IEEE Transactions on Evolutionary Computation, 4(2), pp. 164-171, (2000).
- [39] RAYMOND Chiong, and OOI Koon Beng. *A comparison between Genetic algorithms and Evolutionary Programming based on Cutting Stock Problem*. Engineering Letters, 14:1, (Advance online publication), pp. 1-14, (2007).
- [40] SIVANANDAN S. N., and DEEPA S. N., *Introduction to Genetic Algorithms*. Springer, pp. 39-71, (2008).
- [41] SHIMAMOTO, N., HIRAMATSU, A. and YAMASAKI, K. *A dynamic routing control based on a genetic algorithm*, IEEE International Conference on Neural Networks, 1993, vol.2, pp. 1123-1128, (2000).
- [42] SUCKLEY D. *Genetic algorithm in the design of FIR filters*, IEEE Proceedings Circuits, Devices and Systems, 138(2), pp. 234-238, (1991).



- [43] TANG K. S., MAN K. F., KWONG S and HE Q. *Genetic algorithms and their applications*, IEEE Signal Processing Magazine, 13(6), pp. 22-37, (1996).
- [44] TIMO Eloranta and ERKKI Mäkinen, *TimGA: A genetic algorithm for Drawing Undirected Graphs*. Divulgacioncs Mathematicas, vol.9 pp. 155-171, (2001).
- [45] WEI Song, CHENG Hua Li, SOON Cheol Park *Genetic algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures*. Journal of Expert Systems with Applications, vol. 36, pp. 9095-9104, (2008).



## Appendix A

# FLOW CHART SHOWING GENETIC ALGORITHM MODEL FOR TSP

