KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, KUMASI

COLLEGE OF SCIENCE

DEPARTMENT OF MATHEMATICS

# A GENETIC ALGORITHM MODEL FOR VEHICLE ROUTING PROBLEM (VRP)

By

OWUSU-HEMENG ANDREW (BSc. MATHEMATICS)

A thesis submitted to the Graduate Studies Kwame Nkrumah University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Philosophy (Mathematical Statistics)

October 14, 2014

# 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.

| | | |
|---|---|---|
| Owusu-Hemeng Andrew | ..................... | .................. |
| Student | Signature | Date |

Certified by:

| | | |
|---|---|---|
| Dr. J. Ackora-Prah | ..................... | .................. |
| Supervisor | Signature | Date |

Certified by:

| | | |
|---|---|---|
| Prof. S. K. Amponsah | ..................... | .................. |
| Head of Department | Signature | Date |

## Dedication

This work is dedicated to my wonderful, beloved and caring mother Owusu Vida, supportive brother Owusu Bright and to all those who helped me finished this work in one way or the other.

# Abstract

Genetic algorithms provide a search technique used in computing to find true or approximate solution to optimization and search problems. In this work, Genetic algorithm is tested to find the optimal route for the Vehicle Routing Problem (VRP). The Vehicle Routing Problem (VRP) is a complex combinatorial optimization problem that belongs to the NP-complete class. Due to the nature of the problem it is not possible to use exact methods for large instances of the VRP.

Genetic Algorithms are used to model the Vehicle Routing Problem which shows the superiority of Genetic Algorithm over the company's normal route. Matlab simulations was carried out to find the optimal route of Amponsah Efah Pharmaceutical limited from its main depot after the production stage in Adum, Kumasi.

## Acknowledgements

I would not have been able to complete this work without the encouragements, trust and support of my supervisor, Dr. J. Ackora - Prah, for introducing me to the field of Genetic Programming and for making me want to be a part of it. I also want to thank Apati Justice Kwame and Owusu Richard for their careful editing.

# Contents

# List of Figures

# Chapter 1

# Introduction

Evolutionary Algorithms (EAs) are computer programs that attempt to solve complex problems i.e., Combinatorial Optimization Problems (COPs) by mimicking the processes 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 consideration. Over recent years the application of EAs to COPs in computational chemistry has become common place and there is now a growing collection of published applications. One of the numerous algorithms proposed for applying to these COPs is Genetic Algorithm.

Genetic algorithms (GAs) are search methods based on principles of natural selection and genetics (Fraser, 1957; Bremermann, 1958; Holland, 1975). GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles. For example, in a problem such as the traveling salesman problem, a chromosome represents a route, and a gene may represent a city. GAs work with coding of parameters, rather than the parameters themselves. Genetic algorithms are often viewed as function optimizer, although the range of problems to which genetic algorithms have been applied are quite broad.

## 1.1 Background

Many human inventions were inspired by nature. One example is Genetic Algorithm (GA) which is inspired by Charles Darwin's theory about evolution - "the survival of the fittest". In nature, competition among individuals for scanty resources according to the principle of selection (survival of the fittest) results in the fittest individuals dominating over the weaker ones to reach certain remarkable tasks. An implementation of genetic algorithm begins with a population of (typically random) chromosomes.

John Holland's pioneering book Adaptation in Natural and Artificial Systems (1975, 1992) showed how the evolutionary process can be applied to solve a wide variety of problems using a highly parallel technique that is now called the genetic algorithm. This particular branch of Evolutionary Algorithms (EAs) was inspired by the way living things evolved into more successful organisms in nature. Before applying the genetic algorithm to the problem, the user designs an artificial chromosome of a certain fixed size and then defines a mapping (encoding) between the points in the search space of the problem and instances of the artificial chromosome. GAs search by simulating evolution, starting from an initial set of solutions or hypotheses, and generating successive "generations" of solutions. One then evaluates these structures and allocated reproductive opportunities in such a way that these chromosomes which represent a better solution to the target problem are given more chances to 'reproduce' than those chromosomes which are poorer solutions. The 'goodness' of a solution is typically defined with respect to the current population.

Changes occur during reproduction. The genetic algorithm (GA) transforms a population (set) of individual objects, each with an associated fitness value, into a new generation of the population using the Darwinian principle of reproduction and survival of the fittest and analogs of naturally occurring genetic operations such as crossover (sexual recombination) and mutation. Each individual in the population represents a possible solution to a given problem. The genetic algorithm attempts to find a very good (or best) solution to the problem by genetically breeding the population of individuals over a series of generations.

The genetic algorithm differs from other search methods in that it searches among a population of points, and works with a coding of parameter set, rather than the parameter values themselves. It also uses objective function information without any gradient information. The transition scheme of the genetic algorithm is probabilistic, whereas traditional methods use gradient information. Because of these features of genetic algorithm, they are used as general purpose optimization algorithm. They also provide means to search irregular space and hence are applied to a variety of function optimization, parameter estimation and machine learning applications.

The VRP is an NP-hard combinatorial optimization problem which is usually very time consuming or even impossible and only relatively small instances can be solved to optimality. To this day, it seems that no exact algorithm is capable of consistently solving instances in excess of 50 customers.Many versions of the Vehicle Routing Problem have been described. The Vehicle Routing Problem (VRP) is one of the most popular problems in combinatorial optimization, and its study has given rise to several exact and heuristic solution techniques of general applicability. It generalizes the Traveling Salesman Problem (TSP) and is therefore NP-hard. A recent survey of the VRP can be found in the first six chapters of the book edited by Toth and Vigo (2002a).

The VRP is often defined under capacity and route length restrictions. When only capacity constraints are present the problem is denoted as Classical Vehicle

Routing Problem (CVRP). The Vehicle Routing Problem is discussed here and can in a simplified way be described as follows: A fleet of vehicles is to serve a number of customers from a central depot. Each vehicle has limited capacity and each customer has a certain demand. A cost is assigned to each route between every two customers and the objective is to minimize the total cost of travelling to all the customers. All the itineraries start and end at the depot and they must be designed in such a way that each customer is served only once and just by one vehicle. Several other extensions have also been studied; the vehicle fleet may be heterogeneous, vehicles may perform both collections and deliveries on the same route, some vehicles may be unable to visit certain sites, some customers may require several visits over a given time period, there may exist more than one depot, deliveries may be split among.

Real life Vehicle Routing Problems are usually so large that exact methods cannot be used to solve them. For the past two decades, the emphasis has been on metaheuristics, which are methods used to find good solutions quickly.This is due to the fact that sharp lower bounds on the objective value are hard to derive, which means that partial enumeration based exact algorithms (using branch-and-bound or dynamic programming) will have a slow convergence rate. Since exact approaches are in general inadequate, heuristics are commonly used in practice.

## 1.2 Statement of Problem

After drug production stage at Amponsah Efah Pharmaceutical Limited, issues of transportation always crop up due to the cost involve in its drug distribution to the various wholesale points. This has been a challenge for some time now since the various wholesale points are randomly located and choice of infeasible route lead to high transportation cost.

## 1.3 Objectives

In order to resolve the above stated problem, it is the objective of this work;

- To minimize the total distribution distance of the vehicle by applying Genetic Algorithm (GA) to generate the optimal route for Amponsah Efah Pharmaceutical Limited.

## 1.4 Methodology

To accomplish the above stated objective, Genetic algorithm was applied to the secondary data acquired from the company.

## 1.5 Organization of Study

The background of this research, objectives, and brief methodology of the genetic algorithm is introduced in the first chapter.
The second chapter reviews Evolutionary Algorithm (EA), some heuristics methods and other earlier research carried out in Genetic Algorithm.
For the chapter three, the methodology of the genetic algorithm is discussed in details together with a review of the Vehicle Routing Problem (VRP) and in chapter four, a presentation is done on the application of GA to VRP.
Finally, the conclusion and recommendation based on this research conducted is talked about in chapter five.

# Chapter 2

# Literature Review

## 2.1 Review of Related Work

Kannan, Sasikumar and Devika (2010) recast a Genetic Algorithm approach for solving a closed supply chain model. A case 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 and for smaller size problem the computational results obtained through Genetic Algorithm are compared with the solutions obtained by GAMS optimization software.

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 algorithm (Tim GA) which nicely draws undirected graphs of moderate sizes. The aesthetic criteria used are the number of edge crossing, even distribution of nodes, and edge length deviation. Eloranta and Makinen (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.

Various applications of Genetic Algorithms to the problem of image segmentation are explored by Keri Woods (2007) on his paper Genetic Algorithm: 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 optimizing 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 recast of genetic algorithms and the Evolution of Neutral Networks for language processing by Jaine T. (2009) presents ways in which he have used GAs to find which Neutral Network (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 evolution 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 presenting words and sentences at the output layer is examined with binary and floating point schemes.

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 Shimanoto 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.

Takagi-Sugeno-Kang (TSK) type recurrent fuzzy network is proposed by Juang (2002), which develops from a series of 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.

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 algorithm 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.

Scheduling of hydraulically coupled plants can be approximated by genetic algorithms. An effective approach was suggested by Chen and Chang (1996) to 24 hours 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 model.

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 coeficient for penalty functions and determination of gene variables using genetic algorithms was discussed. The IEEE 30 and IEEE 188 bus system were used as test system to illustrate the applicability of the proposed method.

Genetic algorithms applied to scheduling and optimization of refinery operations was discussed by Oliveira, Almeida and Hamacher (2008). This paper presents a Genetic Algorithm-based method to optimize the production schedule of the fuel oil and asphalt section in a petroleum refinery. Two Genetic Algorithm models were developed to establish the sequence and size of all production shares.
It shows the development of a methodology that applies GA to solve the scheduling problem of fuel oils and asphalt. In this study, the proposed GA aims at minimizing the demand that cannot be supplied, minimizing the production that cannot be allocated to the tanks, and minimizing the number of operational changes.
Two GA models were proposed to solve the optimization of a lot-sizing and sequencing problem in a multi-product plant with two-stage serial machines Direct and Indirect representation. The first uses a direct representation of the production schedule, dividing the scheduling horizon into discrete intervals of one hour which achieves some interesting results, which are further improved with the use of the NM operator. The second model uses an indirect representation which must be decoded into a production scheduling and achieved outstanding performance levels concerning demand fulfillment and production allocation; a satisfactory performance level (according to the refinery's real production scheduling) was

observed when it comes to operational mode change minimization. They also showed that the implementation of the modified EM method allowed the GA to find better solutions due to weight updates during the evolutionary process avoiding meeting some specific objectives while neglecting others. The obtained results confirm that the proposed Genetic Algorithm models, associated with the multi-objective energy minimization method, are able to solve the scheduling problem, optimizing the refinery's operational objectives.

Within the classical resource-constrained project scheduling problem (RCPSP), the activities of a project have to be scheduled such that the makespan of the project is minimized. Considering Resource-constrained project scheduling problem (RCPSP) with makespan minimization as objective Hartmann (1998) propose a new genetic algorithm approach to solve this problem and compared it to two genetic algorithm concepts. While our approach makes use of a permutation based genetic encoding that contains problem-specific knowledge, the other two procedures employ a priority value based and a priority rule based representation, respectively.

The representation is based on a precedence feasible permutation of the set of the activities. The genotypes are transformed into schedules using a serial scheduling scheme. Among several alternative genetic operators for the permutation encoding, a ranking selection strategy was chosen, a mutation probability of 0.05, and a two point crossover operator which preserves precedence feasibility. The initial population was determined with a randomized priority rule method. In order to evaluate the approach, two GA concepts which make use of a priority value and a priority rule representation, respectively was compared with. As further benchmarks, seven other heuristics known were considered.

The outcome reveals that the procedure is the most promising genetic algorithm to solve the RCPSP since in-depth computational study revealed that their GA outperformed the other GAs as well as the other approaches. Finally, computational study show that genetic algorithm yields better results than several heuristic procedures presented in the literature.

The accuracy of the localized face center coordinates and orientation has a heavy influence on the recognition performance so, finding the exact location of face after localization algorithm is crucial. In view of this Kanan and Moradi (2005) introduced a new method using genetic algorithms (GA's) for face localization: A genetic Algorithm based Method for Face Localization and pose Estimation. Face images often have a background that can affect on the face localization algorithm. In their algorithm, input image is first enhanced by means of histogram equalization.
Then connected components are determined by applying a region growing algorithm (coarse segmentation), followed by computing the fit ellipse for face area and at least exact location of face is found by genetic algorithms method.
Then the best-fit ellipse for face area is computed. We have used genetic algorithms to find the best location (includes the best orientation and the best position) of face in image. To check the utility of the proposed algorithm, experimental studies were carried out on the ORL database images.

Many automated analytical techniques such as Curie-point pyrolysis mass spectrometry (Py-MS) have been used to identify bacterial spores giving use to large amounts of analytical data. Hence the rapid identification of Bacillus spores and bacterial identification are paramount because of their implications in food poisoning, pathogenesis and their use as potential biowarfare agents. In view of this, Correa and Goodacre (2011) proposed A genetic algorithm-Bayesian network approach for the analysis of metabolomics and spectroscopic data: application to the rapid identification of Bacillus spores and classification of Bacillus species.
They developed a novel genetic algorithm-Bayesian network algorithm that accurately identifies sand selects a small subset of key relevant mass spectra (biomark-

ers) to be further analysed which once identified, it becomes a subset of relevant biomarkers used to identify Bacillus spores successfully and to identify Bacillus species via a Bayesian network model specifically built for this reduced set of features.
This final compact Bayesian network classification model is parsimonious, computationally fast to run and its graphical visualization allows easy interpretation of the probabilistic relationships among selected biomarkers. In addition, they compared the features selected by the Genetic Algorithm-Bayesian Network (GA-BN) approach with the features selected by partial least squares-discriminant analysis (PLS-DA). The classification accuracy results show that the set of features selected by the GA-BN is far superior to PLS-DA.

Several heuristic approaches for the flowshop scheduling problem have been developed. In recent years, meta-heuristic approaches, such as Simulated Annealing, Tabu Search, and Genetic Algorithms, have become very desirable in solving combinatorial optimization problems because of their computational performance.
In a paper, which introduced the fundamental model and described a GA-based heuristic for solving the flowshop scheduling problems: A Model To Study Genetic Algorithm For The Flowshop Scheduling Problem proposed by Tyagi and Varshney (2012). Many scheduling problems are NP-hard problems. For such NP-hard combinatorial optimization problems, heuristics play a major role in searching for near-optimal solutions. In this GA-based heuristic, a different parameter set was generated for the Genetic operators which protect the best schedule which has the minimum make-span, at each generation and then transfer this schedule to the next population with no change. This operation enables us to choose the higher crossover and mutation probability pc = 1 (crossover probability) and pm = 0.05 (mutation probability). This increase the diversity of the population to get a better solution and also show the excellent performance of the LOX operator. Their heuristic was compared with the NEH (Nawaz, Enscore, Ham) Algorithm

which is the most popular heuristic in the literature. The computational experience shows that the Genetic Algorithm approach provides competitive results for flowshop scheduling problems.

Alba and Troya (2009)proposed a paper on A Survey of Parallel Distributed Genetic Algorithms.
In this work we have presented the fundamental syntax and semantics of sequential genetic algorithms (GAs). The paper deals with very popular extensions of these algorithms known as parallel distributed GAs, in which many subalgorithms run in parallel with sparse migrations of strings.
A structured and extensive overview on the more important and up-to-date PGA systems is discussed. In it, much of the existing software and criteria for their classification is used. In addition, we present in the paper useful technical information about PGAs relating operators, structured-population paradigms, and parameters guiding the parallel search. We have included a brief theoretical foundation of a distributed GA to make the paper relatively self-contained. In particular, we have offered a location of PGAs in the taxonomy of search techniques, a nomenclature revision, algorithmic descriptions of techniques, future trends, a classification of a large portion of the existing software, open questions relating generational versus steady-state evolution modes and heterogeneous versus homogeneous parallel algorithms, and many other minor details and major concepts relating parallel GAs in general. Our main interest has been in parallel distributed GAs, since the impact of the research in this kind of algorithms is a priori larger than for other kinds of parallel genetic algorithms.
We are especially concerned with offering useful and rigorous material that could help new and expert practitioners. Although our overview is obviously not complete, it represents a good starting point to conduct future research in this domain or to make new applications by using parallel distributed GAs.

Recovery of used products has become increasingly important recently due to economic reasons and growing environmental or legislative concern. Product recovery, which comprises reuse, remanufacturing and materials recycling, requires an efficient reverse logistic network. In view of this a paper: An Optimization Model for Reverse Logistics Network under Stochastic Environment Using Genetic Algorithm was proposed by Hosseinzadeh and Roghanian (2012). One of the main characteristics of reverse logistics network problem is uncertainty that further amplifies the complexity of the problem. The degree of uncertainty in terms of the capacities, demands and quantity of products exists in reverse logistics parameters. With consideration of the factors noted above, this paper proposes a probabilistic mixed integer linear programming model for the design of a reverse logistics network. The demand of manufacturing centers and recycling centers are regarded as random variables. This probabilistic model is first converted into an equivalent deterministic model. In this paper multi-product, multi-stage reverse logistics network problem which consider the minimizing of total shipping cost was proposed and fixed opening costs of the disassembly centers and the processing centers in reverse logistics. Then, we propose priority based genetic algorithm to find reverse logistics network to satisfy the demand imposed by manufacturing centers and recycling centers with minimum total cost under uncertainty condition. Finally the proposed model was applied to the hypothetical problem. And then, computing results show that it can obtain solutions for reverse logistics network design problem with some stochastic parameters. In fact, this type of network design problem belongs to the class of NP-hard problems.

After suitable modifications, genetic algorithms can be a useful tool in the problem of wavelength selection in the case of a multivariate calibration performed by PLS. Unlike what happens with the majority of feature selection methods applied to spectral data, the variables selected by the algorithm often correspond to well-defined and characteristic spectral regions instead of being single variables scattered throughout the spectrum. This leads to a model having a better predictive ability than the full-spectrum model; Application of genetic algorithm-PLS for feature selection in spectral data sets by Leardi (2000), furthermore, the analysis of the selected regions can be a valuable help in understanding which the relevant parts of the spectra are.

Nowadays, spectral data are perhaps the most common type of data to which chemometric techniques are applied. Owing to the development of new instrumentation, data sets in which each object is described by several hundreds of variables can be easily obtained. Methods such as partial least squares (PLS) or principal component regression (PCR), being based on latent variables, allow one to take into account the whole spectrum without having to perform a previous feature selection.

The present study shows that the GA can be a good method for feature selection in spectral data sets. The results obtained on five different data sets demonstrate that the predictive ability of the models obtained with the wavelengths selected by the algorithm is very often much better, and anyway never worse, than the predictive ability of the full spectrum. Another relevant point is that the selected variables almost always clearly identify spectroscopically relevant regions.

In this paper, the authors discussed the problem of selecting suppliers for an organisation, where a number of suppliers have made price offers for supply of items, but have limited capacity. Selecting the cheapest combination of suppliers is a straight forward matter, but purchasers often have a dual goal of lowering the number of suppliers they deal with. This second goal makes this issue a bicriteria problem - minimisation of cost and minimisation of the number of suppliers. Hence a Genetic Algorithm for a Bicriteria Supplier Selection Problem by Weintraub and Basnet (2005) presented a mixed integer programming (MIP) model for this scenario in which Quality and delivery performance are modeled as constraints. Smaller instances of this model may be solved using a MIP solver, but large instances will require a heuristic. To this a multi-population genetic algorithm for generating pareto-optimal solutions of the problem was presented. The performance of this algorithm is compared against MIP solutions and Monte Carlo solutions.

The contributions of this paper are two-fold: the inclusion of number of selected suppliers as a criterion in supplier selection in a MIP model and the development and testing of a multi-population genetic algorithm for the generation of Pareto-optimal solutions. Three algorithms were presented to generate the Pareto-optimal solutions.

The exact (MIP) algorithm solved problems with up to 50 suppliers within reasonable time, but not higher sized problems. The performance of multi-population genetic algorithm was close to the MIP algorithm results. The genetic algorithm performed better than Monte Carlo optimization, particularly in regard to the number of solutions generated. In regard to the purchase cost of the solutions, the genetic algorithm performed better for all but the largest problems for which the performances of the two algorithms were almost even.

Computer software marketed by companies such as the Heat Transfer Research Institute (HTRI), HTFS, and B-JAC International are used extensively in the thermal design and rating of HEs. A primary objective in HE design is the estimation of the minimum heat transfer area required for a given duty, as it governs the overall cost of the HE. However, because the possible design configurations of heat transfer equipment are numerous, an exhaustive search procedure for the optimal design is computationally intensive. Tayal, Fu, and Diwekar (2009) presented a paper: Optimal Design of Heat Exchangers: A Genetic Algorithm Framework for solving the combinatorial problem involved in the optimal design of HEs. The problem is posed as a large-scale, combinatorial, discrete optimization problem involving a blackbox model. This paper demonstrates the first successful application of genetic algorithms to optimal HE design with a black-

box model. It also incorporates methodologies to avoid process infeasibilities and design vibrations. In addition this paper compares the performance of SA and GAs in solving this problem and presents strategies to improve the performance of the optimization framework.
From this study, it was concluded that (1) The optimal design obtained using combinatorial algorithms such as GAs and SA significantly improves base-case designs; (2) these algorithms also result in considerable computational savings compared to an exhaustive search; and (3) GAs have an advantage over other methods in obtaining multiple solutions of the same quality, thus providing more flexibility to the designer.

A vehicle designer can do little to improve road surface roughness, so designing a good suspension system with good vibration performance under different road conditions becomes a prevailing philosophy in the automobile industry. The ride quality of a vehicle is significantly influenced by its suspension system, the road surface roughness, and the speed of vehicle. A paper, Research on Suspension System Based on Genetic Algorithm and Neural Network Control by Tang and Guo (2009) showed a five degree-of-freedom half body of vehicle suspension system model which can describe both the vertical movement and the pitching movement of the body, what's more, it can demonstrate the effect of the passenger, which makes it to be a relatively ideal model for suspension dynamic. In this work, the specified half body vehicle model with passenger involving five degree of freedom is presented to achieve the excellent ride comfort and drive stability of the system description. Genetic algorithm and neural network control are used to control the suspension system. The desired objective is proposed as the minimization of a multi-objective function formed by the combination of not only sprung mass acceleration, pitching acceleration, suspension travel and dynamic load, but also the passenger acceleration.
The model is assumed to have five masses attached with linear springs and non-

linear dampers. It is also assumed that the system does not vibrate in lateral direction, only oscillates in vertical and longitudinal directions. Furthermore, the tires are assumed not losing the contact with the road surface. Approaches are presented for suspension design which uses genetic algorithm and neural network control algorithm. It is obvious from the response plots that vehicle body vertical acceleration, passenger response and pitching angular response decreased compared with the passive suspension system, which naturally brings ride comfort. And the suspension travel and dynamic load reduced compared with the passive suspension system, which indicates that the proposed controller proves to be effective in the stability improvement of the suspension system. A mechanical dynamic model of the five degree of freedom half body of vehicle suspension system is also simulated and analyzed by using software Adams. Simulation results demonstrate that the proposed active suspension system proves to be effective in the ride comfort and drive stability enhancement of the suspension system.

People often need to make decisions based on different kinds of information, but the explosion of information is hard to handle and reading everything may be very time consuming. Various kinds of summaries (e.g.: titles, abstracts, keywords, outlines, previews, reviews, biographies and bulletins) help reduce this problem. Summarizing Jewish Law Articles Using Genetic Algorithms is a paper by HaCohen-Kerner, Malin, Chasson (2005). This paper describes the first summarization model for texts in Hebrew. The summarization is done by extraction of the most relevant sentences. The introduction of summaries offers the readers the option whether or not to read the entire text. In addition, summaries can serve as brief substitutes of full documents.
Automatic text summaries can be produced with two main approaches: Natural Language Processing (NLP) and information extraction (IE). Three machine learning methods have been tried: perceptron learning, Naive Bayesian learning, and genetic algorithm. The best results have been achieved by the genetic

algorithm. To the best of our knowledge, this model is also the first to use successfully genetic algorithm for sentence extraction. This model belongs to the sentence extraction approach. That is, it selects the most important sentences from the article and proposes them as a summary. In contrast to many summarization models that were designed and checked mostly for English articles taken from magazines and newspapers, the model deals with articles referring to Jewish law written in Hebrew.

Using Genetic Algorithm for Distributed Generation Allocation to Reduce Losses and Improve Voltage Profile by Sedighizadeh, and Rezazadeh (2008) is a paper that presents a method for the optimal allocation of Distributed generation in distribution systems. In this paper, the aim would be optimal distributed generation allocation for voltage profile improvement and loss reduction in distribution network. Genetic Algorithm (GA) was used as the solving tool, which referring two determined aim; the problem is defined and objective function is introduced. Considering to fitness values sensitivity in genetic algorithm process, there is needed to apply load flow for decision-making. Load flow algorithm is combined appropriately with GA, till access to acceptable results of this operation. It was implemented on part of Tehran electricity distributing grid. The resulting operation of this method on some testing system is illuminated improvement of voltage profile and loss reduction indexes.
The impact of DG in system operating characteristics, such as electric losses, voltage profile, stability and reliability needs to be appropriately evaluated. The installation of DG units at non-optimal places can result in an increase in system losses, implying in an increase in costs and, therefore, having an effect opposite to the desired. As a contribution to the methodology for DG economical analysis, in this paper it is presented an algorithm for the allocation of generators in distribution networks, in order to voltage profile improvement and loss reduction in distribution network. The Genetic Algorithm is used as the optimization tech-

nique.
In this paper the results of application of GA algorithm to the optimal allocation of DGs in distribution network is presented. The Khoda Bande Loo distribution test feeder in Tehran has been solved with the proposed algorithm and, the simple genetic algorithm.

Brain Computer Interfaces (BCIs) measure brain signals of brain activity intentionally and unintentionally induced by the user, and thus provide a promising communication channel that does not depend on the brain's normal output pathway consisting of peripheral nerves and muscles. Ghanbari et al, (2012) paper on Brain Computer Interface with Genetic Algorithm Genetic Algorithm to select the effective number of electrodes and Redundancy Reduction.
BCI operation depends on the interaction of two adaptive controllers, the user, who must maintain close correlation between his or her intent and these phenomena, and the BCI, which must translate the phenomena into device commands that accomplish the user's intent. They might also control a neuroprosthesis that provides hand grasp to those with mid-level cervical spinal cord injuries.
With adequate recognition and effective engagement of these issues, BCI systems could provide an important new communication and control option for those with disabilities that impair normal communication and control channels. They might also provide to those without disabilities a supplementary control channel or a control channel useful in special circumstances.
Their paper is on one hand to reduce the redundancy and on the other hand to increase the BCI speed and making use of it in real time form. Hence, the linear filtering method is applied to Artifact removal which is a relatively simple method with fairly low complexity computations.
Since the EEG is non-stationary in general, it is most appropriate to use time-frequency domain methods like wavelet transform (WT) as a mean for feature extraction. The simulation results confirm this fact that the Genetic algorithm

is applied in order to choose the best features from the feature space as well as the best channels from the many channels that have been used. In this case, the increased number of electrodes causes a non-linear increase in computational complexity (decrease transfer rate). To overcome these problems in this article evolutionary intelligent method for selecting the effective number of electrodes and redundancy reduction was used. One of the main privileges of the mixed methods used in this paper is that, the redundant data are removed by the selection power of the genetic algorithm. This fact reduces the data dimension and reduced time response of system significantly.

## 2.2 Evolutionary Algorithms

### 2.2.1 Introduction

Evolutionary algorithms (EAs) are based on the process of Darwin's theory of evolution. Evolutionary algorithms (EAs) use biologically derived techniques such as inheritance, mutation, natural selection and recombination.

In 1882, Charles Darwin defined natural selection or survival of the fittest as the preservation of favorable individual differences and variations and the destruction of those that are injurious. In nature, features that make an individual more suited to compete are preserved when reproducing and the weakening features are eliminated. This process is called evolution, where features are controlled by units called genes which form sets called chromosomes. Over generations, the fittest individuals survive and their fittest genes are transmitted to their descendants during a sexual recombination process called crossover. To their basic components one can subsume population (set of solutions), chromosomes (individuals), fitness of the chromosomes, process of reproduction (selection of parents and children generation), replacement (death of the individuals) and generation completion.

Typically evolution starts from a population of completely random individuals (solutions), represented by chromosomes, and happens in generations. Traditionally, solutions occur as binary strings of zeros (0's) and ones (1's), but different encodings are also possible. Each individual is characterized by its fitness. Each generation is defined by population size, as well as the birth and death processes. In every generation, multiple individuals are stochastically selected from the current population, and next-modified through mutation or recombination to form a new population, which becomes current in the following iteration of the algorithm. Solutions which form the offspring are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is motivated by a hope, that the new population will be better than the old one. In such a manner, an approximation algorithm evolves towards better solutions. The procedure stops when the desired stopping criterion, like number of populations or improvement of the best solution, is reached. As a result of this simulated evolution one obtains highly evolved solution to the original problem, that is, the best chromosome picked out of the final population.

The main purpose of evolutionary algorithms is to imitate this evolutionary process in computers.

The general procedure of EAs is as follows:

1. Initialize the population by randomly selecting or generating a set of potential solutions (also called chromosomes or individuals). Such individuals evolve during several generations (i.e., they constitute off-springs) through steps 2 to 4 below;

2. Evaluate each individual in the population by calculating its fitness;

3. Reproduce selected individuals to form a new population (best individuals are kept, while the others are discarded);

4. Perform evolutionary operations, such as crossover and mutation, on the population according to pre-specified probabilities. Crossover exchanges

the genetic material of a pair of individuals to create the population of the next generation, while mutation randomly changes a gene of a chromosome;

5. Loop to step 2 until some condition is met (e.g., reaching a specified number of generations).

An idea of evolutionary algorithms for combinatorial optimization problems, inspired by Darwin's theory of evolution, was introduced by John Holland in 1975. Categorically, three of the nearly contemporaneous sources of the evolutionary algorithms (EA) have been kept alive over three decades and experienced an amazing increase of interest during the last fifteen years. Two of them are lying in the United States of America, the source of evolutionary programming (EP) in San Diego (Fogel, 1962; Fogel et al., 1966), the source of genetic algorithms (GA) in Ann Arbor (Holland, [1962, 1975]). Evolution strategies (ES), the third main variant of EA, were founded by students at the Technical University of Berlin (TUB) (Rechenberg, [1965, 1971]; Schwefel, [1965, 1975]).

Evolutionary algorithms have been generally applied in the recent decade to different disciplines, such as DSS research, scheduling, engineering, chemistry, health, management and finance (Chambers 2001; Carlsson and Turban 2002; Osyczka 2002; Marczyk, 2004; Mora et al. 2006). They are also increasingly applied to the economics field. Excellent surveys and discussion of relevant issues about the applications in economics can be found in Dawid (1999); Arifovic (2000); Tsang, Lsasi and Quintana (2009); Safarzynska and Bergh (2009). Below is a summary of the economics areas of applications and a number of examples.
The string representation in GAs has been successfully used to code consumer preferences (Aversi et al. 1997); production functions (Birchenhall, Kastrinos and Metcalfe 1997); pricing strategies (Curzon Price 1997) and production rules in cobweb models (Dawid and Kopel 1998; Frenke 1998).
In addition, genetic programming has been used to develop an optimal price-setting rule (Dosi et al. 1999) and an optimal trading rule (Allen and Karjalainen

1999).

Furthermore, Hidalgo et al. (2008) used an EA to find correct parameters for technical indicators applied to interpret stock market trending and investing decisions. Lately, Jina, Tsang and Li (2009) applied a constraint handling EA to search equilibriums for bargaining problems. Safarzynska and Bergh (2009) showed that applications of evolution strategies in economics are rare; therefore, this paper contributes to utilizing evolution strategies in the field of economics.

### 2.2.2 Evolutionary Strategies

Evolutionary strategies (ES) were developed in Germany by Ingo Rechenberg and H.P. Schwefel in the 1960's. It imitates mutation, selection and recombination by using normally distributed mutations, a deterministic mechanism for selection (which chooses the best set of offspring individuals for the next generation ) and a broad repertoire of recombination operators. The primary operator of the ES is Mutation. There are two variants of selection commonly used in ES. In the elitist variant, the $'\rho'$ parent individual for new generation are selected from the set of both $'\rho'$ parent and $'\vartheta'$ offspring at old generation. This is called Plus Strategy $\rho + \vartheta$.

Additionally, each individual contains a number of strategy parameters, these being the variances and covariances of the object variables (the covariances are optional, but when used are normally defined using the rotation angles of the covariance matrix). The strategy parameters are used to control the behavior of the mutation operator and are not required when decoding an individual. The recombination operator produces one child and requires two parents for each object variable and strategy parameter in the child.
Historically, the same parents are used to generate all object variables in the child, then the parents are re-selected for each strategy parameter. The parents

are selected randomly from the current population (i.e., there is no selection pressure at this point).

Mutation, which is the main operator in the ES acts upon strategy parameters as well as object variables. The mutation operator first perturbs the strategy parameters. The object variables are then mutated using the resulting probability distribution defined by the modified strategy parameters. This special mutation operator allows the ES to evolve good strategy parameters for the problem and has been termed self-adaptation.

Historically ESs were designed for parameter optimization problems. The encoding used in an individual is therefore a list of real numbers: these are called the object variables of the problem. Like the GA, EAs run until some termination criteria are satisfied. However, in ESs remains an attractive alternative to GAs, especially in the field of parameter optimization, where in model systems they appear to outperform GAs.

An ES-algorithm as developed by Rechenberg and Schwefel can be described briefly as follows

1. A current population of m individuals is randomly initialized.
2. Fitness scores are assigned to each of the m individuals.
3. $l$ new offspring are generated by recombination from the current population.
4. The $l$ new offspring are mutated.
5. Fitness scores are assigned to the l new offspring.
6. A new population of m individuals is selected, using either $.m; l/\rho$ or $.mCl/\rho$ selection.
7. The new population becomes the current population.
8. If the termination conditions are satisfied exit, otherwise go to step 3.

### 2.2.3 Evolutionary Programming

Evolutionary programming is an optimization strategy that is based on the stochastic modification of a set of trial solutions. Evolutionary Programming (EP) was developed by L. J. Fogel et al. (1966) in the USA. This technique is especially well suited for combinatorial problems and situations where the fitness landscape has many local minima. EP (which is a Stochastic Optimization Strategy) is a useful method of optimization when other techniques such as gradient descent or direct analytical discovery are not possible.

Illustrates of the form of an EP scheme

1. A current population of m individuals is randomly initialized.
2. Fitness scores are assigned to each of the m individuals.
3. The mutation operator is applied to each of the m individuals in the current population to produce m offspring.
4. Fitness scores are assigned to the m offspring.
5. A new population of size m is created from the m parents and the m offspring using tournament selection.
6. If the termination conditions are satisfied exit, otherwise go to step 3.

An important point is that the strings do not have to be of a fixed length, they could mutate into longer or shorter forms. If you look at the algorithm, you can see why this is - there is no crossover operation. All in all, this means that system representation in ES or EP can be direct and simple.
However, not using crossover also has one major disadvantage - that of speed, mutation is a slow way to search for good solutions.

## 2.3 Genetic Algorithm

The term genetic algorithms, almost universally abbreviated nowadays to GAs, are search methods based on principles of natural selection and genetics (Fraser, 1957; Bremermann, 1958; Holland, 1975). They are powerful and widely applicable stochastic search technique and optimization methods based on the principles of genetics, natural selection and natural evaluation.

A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (i.e., minimizes the cost function). The method was developed by John Holland (1975) over the course of the 1960s and 1970s and finally popularized by one of his students, David Goldberg (at the University of Michigan), who was able to solve a difficult problem involving the control of gas-pipeline transmission for his dissertation (Goldberg, 1989). John Holland, whose book *Adaptation in Natural and Aritificial Systems* of 1975 and Goldberg (with his successful applications and excellent book (1989)) was instrumental in creating what is now a flourishing field of research and application that goes much wider than the original GA.

GA is a method for moving from one population of "chromosomes" (e.g., strings of ones and zeros, or "bits") to a new population by using a kind of "natural selection" together with the genetics-inspired operators of crossover, mutation, and inversion. Each chromosome consists of "genes" (e.g., bits), each gene being an instance of a particular "allele" (e.g., 0 or 1). For example, in a problem such as the traveling salesman problem, a chromosome represents a route, and a gene may represent a city.

In contrast to traditional optimization techniques, GAs work with coding of parameters, rather than the parameters themselves. The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones.

Crossover exchanges subparts of two chromosomes, roughly mimicking biological recombination between two single-chromosome (”haploid”) organisms; mutation randomly changes the allele values of some locations in the chromosome; and inversion reverses the order of a contiguous section of the chromosome, thus re-arranging the order in which genes are arrayed.

Although there are a variety of operators such as crossover, mutation and inversion as defined above, the two main operators used is:-

- Crossover, which creates new individuals by combining parts from two individuals like the bit-string crossover in which two strings are used as parents and new individuals are formed by swapping a sub-sequence between the two strings

- Mutation, which creates new individuals by making changes in a single individual like the bit-flipping mutation, in which a single bit in the string is flipped to form a new offspring.

Genetic algorithm maintains a population of individuals, say $P(t)$, for generation $t$. Each individual represents a potential solution to the problem at hand. Each individual is evaluated to give some measure of its fitness. Some individuals undergo stochastic transformations by means of genetic operations to form new individuals. The new individuals, called offspring $C(t)$, are then evaluated. A new population is formed by selecting the more fit individuals from the parent population and offspring population.

After several generations, genetic algorithm converges to the best individual, which hopefully represents an optimal or suboptimal solution to the problem.

The general structure of the Genetic algorithms is as follow:

```
Begin
    {
    t = 0;
    Initialise $P(T)$;
    Evaluate $P(t)$;
    While (not termination condition) to
    Begin
        {
        Apply crossover and mutation to $P(t)$ to yield $C(t)$;
        Evaluate $C(t)$;
        Select $P(t+1)$ from $P(t)$ and $C(t)$;
        $t = t + 1$ ;
        }
End
```

Another important concept of GAs is the notion of population. Unlike traditional search methods, genetic algorithms rely on a population of candidate solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms. For example, small population sizes might lead to premature convergence and yield substandard solutions. On the other hand, large population sizes lead to unnecessary expenditure of valuable computational time.

## 2.4 Genetic Programming (GP)

The Genetic Programming (GP) is an EC technique which was developed in 1992 by Koza John. Genetic programming is a collection of methods for the automatic generation of computer programs that solve carefully specified problems, via the core, but highly abstracted principles of natural selection. In a sentence, it is the compounded breeding of (initially random) computer programs, where only the relatively more successful individuals pass on genetic material (programs and program fragments) to the next generation. Genetic Programming represents a special type of genetic algorithm in which the structures that undergo adaptation are not data structures, but hierarchical computer programs of different shapes and sizes. In GP, the individuals do not represent the solution of a given problem. Now, they represent algorithms/procedures to solve such problem. Thus, the GP find out the best procedure to solve a problem.

In GP, the individuals are defined by a function set (subprograms, mathematical functions, etc.) and a terminal set (constants, variables, etc.). The GP process starts by creating an initial population of randomly-generated programs and continues by producing new generations of programs based on the Darwinian principle of *"the survival of the fittest"*. The automatically-generated computer programs are expressed as function composition, and the main breeding operations are *reproduction* and *cross-over*.

By reproduction we mean that a program from generation $i$ is copied unchanged within generation $i+1$, while the cross-over takes two parent-programs from generation $i$, breaks each of them in two components, and adds to generation $i+1$ two children programs that are created by combining components coming from different parents.

In order to create a GP-based application, the user has to specify a *set of building blocks* based on which the population of programs is constructed, and an *evaluation function* that is used to measure the fitness of each individual program.

There are two types of primitive elements that are used to build a program: *terminals* and *functions*. Both terminals and functions can be seen as LISP-functions, the only difference between them consisting of the number of arguments that they are taking: terminals are not allowed to take arguments, while functions take at least one argument. The *individuals* generated by the GP system represent computer *programs* that are built by *function composition* over the set of terminals and functions.

Consequently, GP imposes the *closure property* (Koza 1994): any value returned by a function or a terminal must represent a valid input for any argument of any function in the function set.

As we have already mentioned, the GP problem specification must include a domain-specific *fitness evaluation function* that is used by the GP system to estimate the "fitness" of each individual of a generation. More specifically, the fitness function takes as input a GP-generated program P, and its output represents a measure of how appropriate P is to solve the problem at hand. Both cross-over and reproduction are performed on randomly chosen individuals, but they are biased for highly fit programs. Such an approach has two major advantages: on one hand, the "highly fit" bias leads to the potentially fast discovery of a solution, while on the other hand, GP is capable of avoiding local minima by also using in the breeding process individuals that are less fit than the "best" offsprings of their respective generations.

Normally, the GP uses two operators: crossover and *mutation.* The crossover operator exchanges two sub-trees from two tree randomly selected. In this way, two new trees are created. The mutation operator randomly selects a tree and creates a new tree by taking a sub-tree and replacing it by other one, which is randomly generated. These operators preserve the syntactic constraints of the models.

This model is easy to implement in GP, through the utilization of the ADF (Automatic Definition Function) technique. This extension of GP permits to define functions to evolve in parallel with the main procedure. These functions can be called by other functions, or by the main procedure, during the evolution.

Rather than blindly searching the fitness space, or searching from randomly initialized states, genetic programming attempts to extract the useful parts of the more successful programs and use them to create even better solutions. How does the system know which parts of a program are useful, and how to combine them to form more fit solutions? By randomly selecting parts of the more successful programs, and randomly placing those parts inside other successful programs. Genetic programming relies upon the fitness function to tell if the new child received something useful in the process. Often the child is worse for its random modification, but often enough the right code is inserted in the right place, and fitness improves.

Given the programming language such as Lisp and a fitness metric, the steps executed by a genetic programming algorithm are straightforward:

- **Initial Population:** With an algorithm that allows random generation of code, an initial population of potential solutions can be generated. All will be quite inept at solving the problem, as they are randomly generated programs. Some will be slightly better than others, however, giving evolution something to work with.

- **Fitness Ranking:** Via the fitness metric, the individual programs are ranked in terms of ability to solve the problem.

- **Selection:** The closer (better) solutions are selected to reproduce because they probably contain useful components for building even better programs.

- **Mating:** At random, chunks of those selected programs are excised, and placed inside other programs to form new candidate solutions. These "children" share code from both parents, and (depending on what code was selected) may exhibit hybrid behavior that shares characteristics of both.

- **Mutation:** To simulate genetic drift/stray mutation, many genetic programming systems also select some of the more fit programs and directly duplicate them, but with a few of their statements randomly mutated.

- **Repetition until Success:** From here, the process starts over again at Fitness Ranking, until a program is found that successfully solves the problem.

Not every child will be more fit than its parent(s), and indeed, a very large percentage will not be. The expectation, however, is that some children will have changes that turn out to be beneficial, and those children will become the basis of future generations.

Note that the random makeup of the initial population has a large effect on the likelihood that GP will find a successful program. If a single run does not succeed after a reasonable period of time, often it will succeed if it is restarted with a new random population.
These steps constitute the core of most genetic programming systems, though all systems tweak, or completely change, many aspects of these steps to suit the theoretical interests pursued by their designers. The field is still young and there is no standard of what a genetic programming system must include, or how it must proceed from step to step.

## 2.5 Other Related Methods

### 2.5.1 Simulated Annealing

Robust probabilistic optimization method mimicking the solidification of a crystal under slowly decreasing temperature.
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 of 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 temperature and then is allowed to cool slowly so that its structure is frozen at the crystal configuration of lowest energy. The slower the energy, the more perfect is the crystal formed. By cooling, complex physical systems natural 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 behavior 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 Gibs law.

Gibs law gives this probability as:

$$p = e^{\frac{z}{k}} \tag{2.1}$$

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
$T \leftarrow$ 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 Algorithm. Both Genetic Algorithm 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.

### 2.5.2 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 search methods consists of choosing randomly a solution in the neighborhood of 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 minimum. 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 optimum 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 very easy to implement to achieve fairly good solution faster.

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

Stochastic Hill Climbing Algorithm

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

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

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

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

```
1. $p^* \leftarrow create$ (Implicitly, $p^*x \leftarrow gpm(p^* \cdot g)$ )
2. while terminating criterion is not satisfied do
3. $p_{new}x \leftarrow gpm(p_{new} \cdot g)$
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 Algorithm

Genetic Algorithms are a family of computational models inspired by evolution or in other words, they are search algorithms that are based on concepts of natural selection and natural genetics (Fraser, 1957; Bremermann, 1958; Holland, 1975). Thus GA is a stochastic global search method that mimics the metaphor of natural biological evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information. Genetic algorithm was developed to simulate some of the processes observed in natural evolution, a process that operates on chromosomes (organic devices for encoding the structure of living being). GAs operate on a number of potential solutions, called a population, consisting of some encoding of the parameter set simultaneously and applying the principle of survival of the fittest to produce (hopefully) better and better approximations to a solution. Genetic algorithms are often viewed as function optimizer, although the range of problems to which genetic algorithms have been applied are quite broad.

An implementation of genetic algorithm begins with a population of (typically random) chromosomes. A new population is created by allowing *parent solutions* in one generation to produce *offspring*, which are included in the next generation. A *'survival of the fittest'* principle is applied to ensure that the overall quality of solutions increases as the algorithm progresses from one generation to the next.

The overall structure of genetic algorithm is as follows (Gen and Cheng (200), Goldberg (1989)):

1. Selection
2. Crossover
3. Mutation

In GA, each individual represents a potential solution to the problem at hand. Each individual is evaluated to give some measure of its fitness. Some individuals undergo stochastic transformations by means of genetic operations to form new individuals. There are two types of transformation:

- Crossover, which creates new individuals by combining parts from two individuals.
- Mutation, which creates new individuals by making changes in a single individual.

A general framework and a possible implementation of a genetic algorithm for a permutation scheduling problem is given below.

- **Initialisation:** Choose initial population $P$ containing $q$ solutions to be the current population (randomly generate $q$ permutations also called *strings*).
- **Evaluation:** Compute a fitness value for each solution of $P$ (compute the value $F(S)$ for each solution $S$).
- Reproduction- Use fitness values to select solutions from $P$ to form a mating pool (select $q/2$ best permutations).
- **Regeneration:** Apply *crossover, mutation* and any other selected operations to solutions of the mating pool to form a new population ($q/2$ new permutations are obtained and replace $q/2$ worst permutations in the population).

- **Termination test:** Test whether the algorithm should terminate. If it terminates, output the best solution generated; otherwise, return to the evaluation step.

In the initialization step, we create a population by generating q random permutations. A non-negative fitness function $F(S)$ is used in the evaluation step.

In the reproduction step, a mating pool of size $q/2$ is created. To apply the crossover operation in the regeneration step, solutions in the mating pool are randomly partitioned into pairs. With probability $p$cross, each pair undergoes a crossover; otherwise, the pair is unchanged. Under a crossover operation, the two solutions, which we refer to as parents, combine to produce two *offspring*, each containing some characteristics of each parent. The hope is that one of the offspring will inherit the desirable features of each parent to produce a good quality solution. A mutation operation is applied to solutions before placing them into the new population, each element of each string is selected with probability $p$mut to be perturbed. E.g., if a job of a string is selected for mutation, then it is swapped with another randomly selected job in the same string (which yields a neighbour in the swap neighbourhood).

As a termination test, a time limit is set and the algorithm terminates when this limit is exceeded.

Thus the major steps involved are the generation of a population of solutions, finding the objective function and fitness function and the application of genetic operators. These aspects are described briefly below.

```
    Begin GA
$g = 0$ {generation counter}
Initialization
Evalaution population $P(g)$ { i.e., compute fitness value}
While not done do
$g = g + 1$
Select $P(g)$ from $P(g-1)$
Crossover $P(g)$
Mutate $P(g)$
Evaluate $P(g)$
End
End GA
```

For the basic GA operations: One generation is broken down into a selection phase and recombination phase. Strings are assigned into adjacent slots during selection.

An important characteristic of genetic algorithm is the coding of variables that describes the problem. The most common coding method is to transform the variables to a binary string or vector; GAs perform best when solution vectors are binary. If the problem has more than one variable, a multi-variable coding is constructed by concatenating as many single variables coding as the number of variables in the problem. Genetic Algorithm processes a number of solutions simultaneously.

At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

The genetic algorithm deifiers from other search methods in that it searches among a population of points, and works with a coding of parameter set, rather than the parameter values themselves. The transition scheme of the genetic algorithm is probabilistic, whereas traditional methods use gradient information. Because of these features of genetic algorithm, they are used as general purpose optimization algorithm. They also provide means to search irregular space and hence are applied to a variety of function optimization, parameter estimation and machine learning applications.

Genetic algorithms work on two types of spaces alternatively: Coding space and solution space, or in other words, genotype space and phenotype space. Genetic operators (crossover and mutation) work on genotype space, while evolution and selection work on phenotype space. The selection is the link between chromosomes and the performance of decoded solutions. The mapping from genotype space to phenotype space has a considerable influence on the performance of genetic algorithms.

The genetic algorithms provide a directed random search in complex landscapes. There are two important issues with respect to search strategies: exploration (investigate new and unknown areas in search space) and exploitation (make use of knowledge of solutions previously found in search space to help in find better solutions). This can be done by making genetic operators perform essentially a blind search; with a hope that selection operators direct the genetic search toward the desirable area of solution space.

Unlike simple neighbourhood search methods that terminate when a local optimum is reached, GAs are stochastic search methods that could in principle run for ever. In practice, a termination criterion is needed; common approaches are to set a limit on the number of fitness evaluations or the computer clock time, or to track the population's diversity and stop when this falls below a pre-set threshold. The meaning of diversity in the latter case is not always obvious, and it could relate either to the genotype or the phenotype, or even, conceivably, to the fitnesses, but the most common way to measure it is by genotype statistics. For example, we could decide to terminate a run if at every locus the proportion of one particular allele rose above 90%.

After several generations, genetic algorithm converges to the best individual, which hopefully represents an optimal or suboptimal solution to the problem.

## 3.2 Encoding

As for any search and learning method, the way in which candidate solutions are encoded is a central, if not the central, factor in the success of a genetic algorithm. Encoding is a process performed using bits, arrays, trees, numbers or list to represent individual genes. Most GA applications use fixed-length, fixed-order bit strings to encode candidate solutions. How to encode the solutions of the problem into chromosomes is a key issue when using genetic algorithms. One outstanding problem associated with encoding is that some individuals correspond to infeasible or illegal solutions to a given problem. This may become very severe for constrained optimization problems and combinatorial optimization problems. It must be distinguished between two concepts: Infeasibility and Illegality.

Infeasibility refers to the phenomenon that a solution decoded from chromosome lies outside the feasible region of given problem. Penalty methods can be used to handle infeasible chromosomes. One of these methods is by force genetic algorithms to approach optimal form both sides of feasible and infeasible regions.

Illegality refers to the phenomenon that a chromosome does not represent a solution to a given problem. Repair techniques are usually adopted to convert an illegal chromosome to legal one.

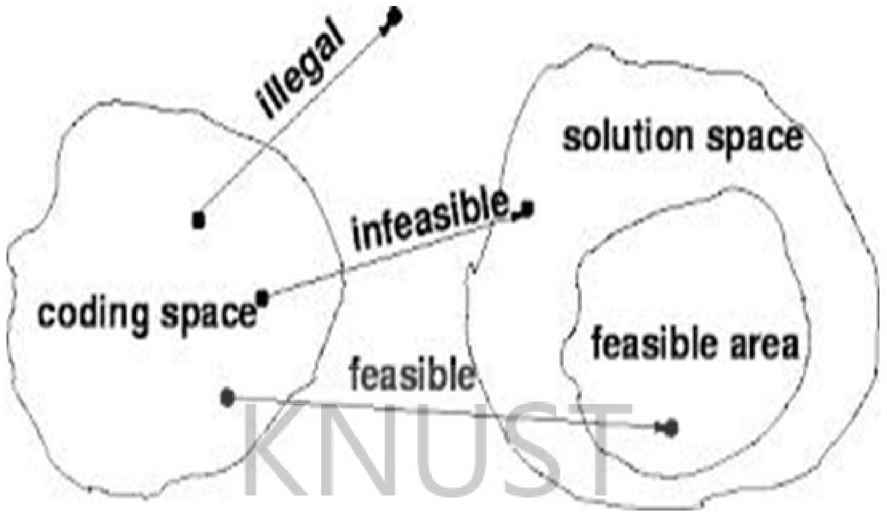


Figure 3.1: Infeasibility and Illegality of Encoding

Encoding can be adapted and one reason for adapting the encoding is that a fixed-length representation limits the complexity of the candidate solutions. For example, in the Prisoner's Dilemma example, Axel-rod fixed the memory of the evolving strategies to three games, requiring a chromosome of length 64 plus a few extra bits to encode initial conditions.

Various encoding methods have been created for particular problems to provide effective implementation of genetic algorithms. According to what kind of symbol is used as the alleles of a gene, the encoding methods can be classified as follows:

- Binary encoding
- Value encoding
- Permutation encoding
- Tree encoding
- Octal Encoding

### 3.2.1 Binary Encoding

Binary encodings (i.e., bit strings) are the most common encodings for a number of reasons. One is historical: in their earlier work, Holland and his students concentrated on such encodings and GA practice has tended to follow this lead. In Binary-coded strings having 1's and 0's are mostly used. Thus the data value is converted into binary strings. It gives many chromosomes with small number of alleles. The length of the string is usually determined according to the desired solution accuracy. Much of the existing GA theory is based on the assumption of fixed-length, fixed-order binary encodings.
A chromosome represented in a binary encoding is shown in Fig. 3.2

| | |
|---|---|
| Chromosome 1 | 1 1 0 1 0 0 1 0 1 1 |
| Chromosome 2 | 1 0 0 0 1 0 0 1 1 0 |

Figure 3.2: Binary Encoding

### 3.2.2 Value Encoding

For many applications, it is most natural to use an alphabet of many characters or real numbers to form chromosomes. In value encoding, every chromosome is a sequence of some values such as real numbers, characters or objects. Value encoding is best used for function optimization problems. It is often required to develop new GA operators specific for the problem in value encoding. It has been widely confirmed that value encoding perform better than binary encoding for function optimization and constrained optimizations problems. Examples include Kitano's many-character representation for graph-generation grammars, Meyer and Packard's real-valued representation for condition sets, Montana and Davis's real-valued representation for neural-network weights, and Schultz-Kremer's real-valued representation for torsion angles in proteins.
Holland's schema-counting argument seems to imply that GAs should exhibit worse performance on valued encoding than on binary encodings. Several em-

pirical comparisons between binary encodings and valued encodings have shown better performance for the latter. Valued encoding can be seen in Fig. 3.3 below.

| | | | | | |
|---|---|---|---|---|---|
| Chromosome 1 | 6.5434 | 1.7543 | 0.0012 | 2.9112 | 5.0654 |
| Chromosome 2 | Left | right | back | forward | centre |

Figure 3.3: Value Encoding

**Permutation Encoding**

Permutation encoding is best used for combinational optimization problems because the essence of this kind of problems is to search for the best permutation or combination of items subject to constrains. In permutation encoding, every chromosome is a string of numbers which represents number in a sequence as shown below in Fig. 3.4

| | |
|---|---|
| Chromosome 1 | 2 6 4 3 7 5 8 1 9 |
| Chromosome 2 | 9 6 7 8 3 4 2 5 1 |

Figure 3.4: Permutation Encoding

### 3.2.3 Tree Encoding

In tree encoding, every chromosome is a tree of some objects, functions or commands in programming languages. Tree encoding schemes, such as John Koza's scheme for representing computer programs, have several advantages, including the fact that they allow the search space to be open-ended (in principle, any size tree could be formed via crossover and mutation). This open-endedness also leads to some potential pitfalls. The trees can grow large in uncontrolled ways, preventing the formation of more structured, hierarchical candidate solutions. (Koza's (1992, 1994) "automatic definition of functions" is one way in which GP can be encouraged to design hierarchically structured programs.) Also, the resulting trees, being large, can be very difficult to understand and to simplify. Systematic experiments evaluating the usefulness of tree encodings and comparing them with other encodings are only just beginning in the genetic programming community. These are only the most common encodings; a survey of the GA literature will turn up experiments on several others.

Tree encodings such as those used in genetic programming automatically allow for adaptation of the encoding, since under crossover and mutation the trees can grow or shrink. Meyer and Packard's encoding of condition sets also allowed for individuals of varying lengths, since crossovers between individuals of different lengths could cause the number of conditions in a set to increase or decrease.

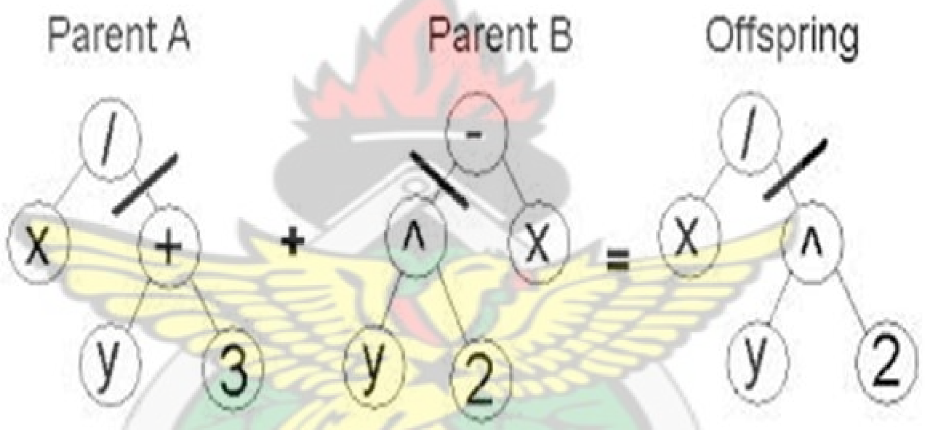


Figure 3.5: Tree Encoding

### 3.2.4 Octal Encoding

This encoding uses string made up of octal numbers (0-7)

| | |
|---|---|
| Chromosome 1 | 13578327 |
| Chromosome 2 | 26834425 |

Figure 3.6: Octal Encoding

## 3.3 The Fitness Function

The operation of fitness proportionate reproduction for the genetic programming paradigm is the basic engine of Darwinian reproduction and survival of the fittest. Fitness is the driving force of Darwinian natural selection and, likewise, of genetic algorithms. Having decoded the chromosome representation into the decision variable domain, it is possible to assess the performance, or fitness, of individual members of a population. This is done through an objective function that characterizes an individual's performance in the problem domain. Each individual in a population is assigned a fitness value derived from its raw performance measure given by the objective function or as a result of its interaction with the environment. This value is used in the selection to bias towards more fit individuals. Highly fit individuals, relative to the whole population, have a high probability of being selected for mating whereas less fit individuals have a correspondingly low probability of being selected. In the natural world, this would be an individual's ability to survive in its present environment.
The fitness function is an equation that is a function of including properties (genes) in each string (chromosome). The general form of this function depends on the studying problem and mentions the final goal of problem. Thus, the objective function establishes the basis for selection of pairs of individuals from the population with a probability according to their relative fitness, and recombine them together during reproduction to produce the next generation.
Note that the parents remain in the population while this operation is performed and therefore can potentially participate repeatedly in this operation (and other operations) during the current generation. Thus, the selection of parents is done with replacement (i.e. reselection) allowed.

In the case of a minimization problem, the most fit individuals will have the lowest numerical value of the associated objective function. This raw measure of fitness is usually only used as an intermediate stage in determining the relative performance of individuals in a GA.

## 3.4 Operators in Genetic Algorithm (GA)

In this section we describe some of the selection, crossover, and mutation operators commonly used in genetic algorithms.

## 3.5 Selection Processes in Genetic Algorithm (GA)

Selection is usually the first operator applied on a population after encoding. This is an operator that makes more copies of better strings in a new population. Selection is the process of determining the number of times, or trials, a particular individual is chosen for reproduction and, thus, the number of offspring that an individual will produce. It is the component which guides the algorithm to the solution by preferring individuals with high fitness over low-fitted ones. It can be a deterministic operation, but in most implementations it has random components. 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. In selection process a chromosome is selected according to it's objective function measurement (Biologist called it fitness function). This function can show some specific measurement such as utility, benefit or any other objectives that should be maximized or minimized. The useful chromosome to copy is determined according to the same functions.

The selection function identifies promising genetic material and determines how much of it should be present in the next generation. The genetic material that composes genotypes of higher quality tend to have a higher probability of finding itself reused in some form or other in the next generation. In most GP systems, selection is applied at the organism level and the term "mating pool" is taken to mean the storage containing the individuals selected for reproduction. While GP systems typically select genotypes, there are other possibilities. In particular, selecting whole sets of genotypes (such as schemata) or combinations of sets and individual genotypes is also possible. A *genotype selection scheme* determines the probability that a genotype will be selected for producing offspring by crossover or mutation. In order to search for increasingly fitter phenotypes, higher selection probabilities are assigned to the genotypes of better scoring phenotypes. The selection operator selects genotypes based on the general principle that the fitter the individual, the higher its probability of being selected for reproduction should be.

Selection might operate with or without replacement. With replacement, the solution that is added to the new parent population is kept in the combined population and a good solution can be chosen more than once for the new parent population. The term with replacement is used since the original genotype is available for further selection as additional genetic material is selected. The system maintains no memory of prior selection. Without replacement, the selected solution is placed in the parent population and removed from the combined population and each solution can only be selected once for the new parent population.

Selection provides the driving force in genetic algorithms. With too much force, genetic search will terminate prematurely. In this way, the selection directs the genetic search toward promising regions in the search space and that will improve the performance of genetic algorithms.

Many selection methods have been proposed, examined and compared. Just as the case for encodings, these descriptions do not provide rigorous guidelines for which method should be used for which problem; this is still an open question for GAs. (For more technical comparisons of different selection methods, see Goldberg and Deb 1991, Bäck and Hoffmeister 1991, and Hancock 1994.)

Some of the selection methods are as follows;

- Roulette Wheel Selection (Fitness-Proportionate Selection)
- Tournament selection
- Boltzmann selection
- Rank selection

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

Holland's original GA used fitness-proportionate selection, in which the "expected value" of an individual(i.e., the expected number of times an individual will be selected to reproduce) is that individual's fitness divided by the average fitness of the population. The most common method for implementing this is "roulette wheel" sampling. It is the simplest method that selects the best chromosome according to the ratio of each chromosomes fitness to sum of all fitness values related to all chromosomes (roulette-wheel selection (Holland, 1975; Goldberg, 1989).

Roulette wheel selection is most common selection method used in genetic algorithms for selecting potentially useful individuals (solutions) for crossover and mutation. In roulette wheel selection, as in all selection methods, possible solutions are assigned a fitness by the fitness function. This fitness level is used to associate a probability of selection with each individual. While candidate solutions with a higher fitness will be less likely to be eliminated, there is still a chance that they may be. We can force the property to be satisfied by applying a random experiment which is, in some sense, a generalized roulette game. At this game a roulette wheel is rolled around a central point and then a specific area is selected when it stops.

To produce a simple roulette wheel, the ratio of each string fitness to the sum of all fitness values in the population is calculated. This ratio is determined as an area of the roulette wheel by assigning each individual a slice of the circular "roulette wheel", the size of the slice being proportional to the individual's fitness value. Thus, the bigger the value, the larger the slice (segment) is. The wheel is spun $N$ times, where $N$ is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation. The process is repeated until the desired number of individuals is selected. Since individuals with higher fitness have more probability of selection, this may lead to biased selection towards high fitness individuals. It can also possibly miss the best individuals of a population. There is no guarantee that good individuals will find their self in the next generation. This method can be implemented as follows:

1. Sum the total expected value of individuals in the population. Call this sum T.

2. Choose a random integer r between 0 and T.

3. Loop through the individuals in the population, summing the expected values, until the sum is greater than or equal to r. The individual whose expected value puts the sum over this limit is the one selected.

With roulette wheel selection there is a chance some weaker solutions may survive the selection process; this is an advantage, as though a solution may be weak, it may include some component which could prove useful following the recombination process.

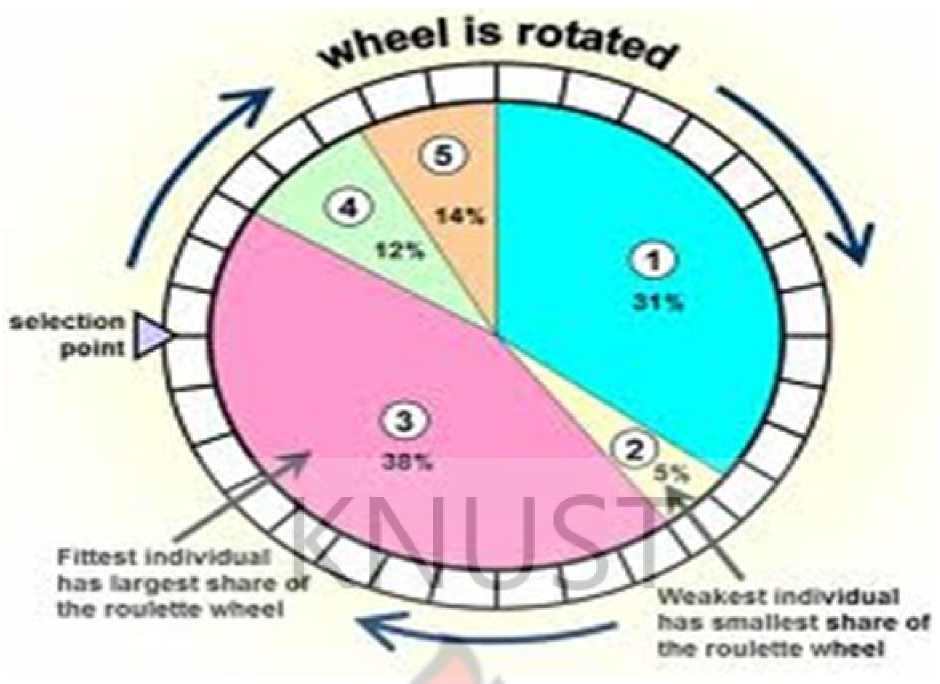

wheel is rotated
5
14%
4
12%
1
31%
selection point
3
38%
2
5%
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

Figure 3.7: Roulette Wheel

The average fitness of the population for $i^{th}$ generation in roulette wheel selection is calculated as

$$\overline{FRW_{ii}} = \frac{\sum_{j=1}^{N} FRW_j}{N}$$

where $i$ varies from 1 to $ngen$ and $j$ varies from 1 to $N$.

Therefore, the probability for selecting the $j^{th}$ string is

$$PRW_j = \frac{FRW}{\sum_{j=1}^{N} FRW}$$

where

$ngen \rightarrow$ total number of generations

$N \rightarrow$ total population size

$FRW_{i,j} \rightarrow$ fitness of $j^{th}$ individual in $i^{th}$ generation for roulette wheel selection

$FRW_j \rightarrow$ Average Fitness of the population in generation in Roulette Wheel Selection

To illustrate,

For example, consider a population containing four strings shown in the Table 3.1 below. Each string is formed by concatenating four substrings which represents variables A, B, C and D. Length of each string is taken as four bits. The first column represents the possible solution in binary form. The second column gives the fitness values of the decoded strings. The third column gives the percentage contribution of each string to the total fitness of the population. Then by "Roulette Wheel" method, the probability of candidate A being selected as a parent of the next generation is 28.09%. Similarly, the probability that the candidates B, C, D will be chosen for the next generation are 19.59, 12.89 and 39.43 respectively. These probabilities are represented on a pie chart, and then four numbers are randomly generated between 1 and 100. Then, the likeliness that the numbers generated would fall in the region of candidate B might be once, whereas for candidate A it might be twice and candidate D more than once and for candidate C it may not fall at all. Thus, the strings are chosen to form the parents of the next generation.

Table 3.1: Table of candidates A, B, C, D

| Candidate (A, B, C, D) | Fitness value | Percentage of total fitness |
|---|---|---|
| 1011 0110 1101 1001 | 109 | 28.09 |
| 0101 0011 1110 1101 | 76 | 19.59 |
| 0001 0001 1111 1011 | 50 | 12.89 |
| 1011 1111 1011 1100 | 153 | 39.43 |
| Total | 388 | 100 |

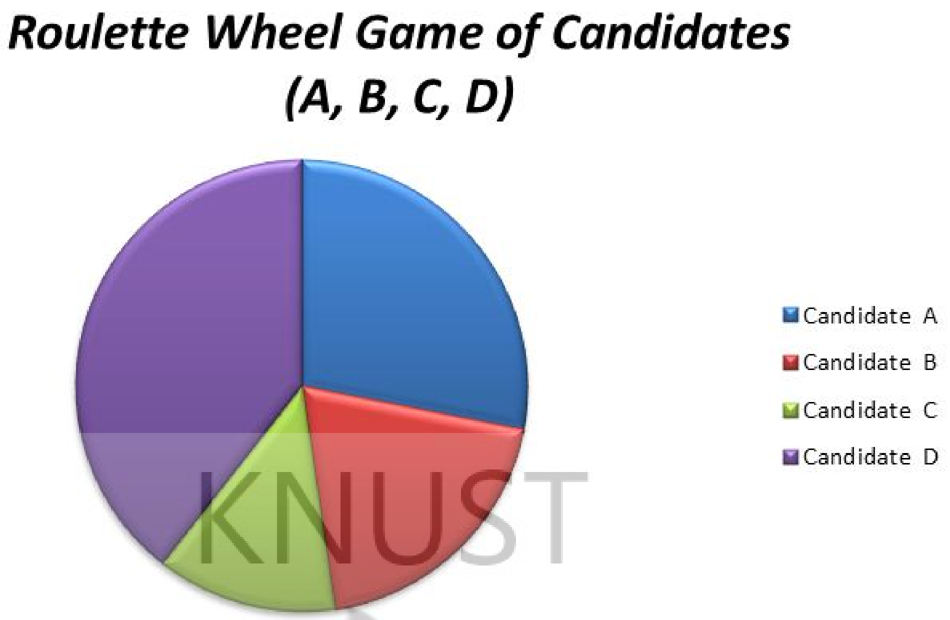


Figure 3.8: Roulette Wheel Game of Candidates A, B, C and D

### 3.5.2 Tournament Selection Method

The other alternative to strict fitness-proportional selection is *tournament selection* (Goldberg et al., 1989) in which a set of chromosomes is chosen and compared, the best one being selected for parenthood. In this method a group of individuals are chosen at random form and the individual with the highest fitness is selected for inclusion in the next generation. Selection pressure can be easily adjusted by changing the tournament size. If the tournament size is larger, weaker individuals are less likely to be selected. This process is repeated until the appropriate numbers of individuals are selected for the new generation.

In tournament selection a string is only selected when it successes to other competitors or on the other hand it's fitness is highest than the other competitors. The number of individual in the set is called the tournament size. In tournament selection, $s$ chromosomes are chosen at random (either with or without replacement) and entered into a tournament against each other. The fittest individual in the group of $k$ chromosomes wins the tournament and is selected as the parent. The most widely used value of $s$ is 2. Using this selection scheme, $n$ tournaments

are required to choose $n$ individuals. That is a common tournament size is 2, this is called binary tournament. By adjusting tournament size, the selection pressure can be made arbitrarily large or small. For example, using large tournament size has the effect of increasing the selection pressure, since below average individuals are less likely to win a tournament while above average individuals are more likely to win it.
During a tournament selection,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 selected again. An analysis of this method was presented by (Goldberg and Deb (1991)).

One potential advantage of tournament selection over all other forms is that it only needs a preference ordering between pairs or groups of strings, and it can thus cope with situations where there is no formal objective function at all - in other words, it can deal with a purely subjective objective function. However, we should point out again that tournament selection is also subject to arbitrary stochastic effects in the same way as roulette-wheel selection - there is no guarantee that every string will appear in a given cycle. Indeed, using sampling with replacement there is a probability of approximately that a given string will not appear at all. One way of coping with this, at the expense of a little extra computation, is to use a variance reduction technique from simulation theory.

### 3.5.3 Boltzmann Selection

*Boltzmann selection* is a method inspired by the technique of simulated annealing. In Boltzman selection, selection pressure is slowly increased over time to gradually focus the search. The inspiration comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. By analogy with this physical process, each step of this selection algorithm replaces a current individual by a random "nearby" solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter $T$ called the temperature. The temperature is gradually decreased during the process, creating a dependency such that the current solution changes almost randomly when $T$ is large, but increasingly "downhill" as (T) goes to zero. The allowance for "uphill" moves saves the method from becoming stuck at local minima.

A typical implementation is to assign to each individual $i$ an expected value, where $T$ is temperature and $t$ denotes the average over the population at time $t$. Experimenting with this formula will show that, as $T$ decreases, the difference in $ExpVal(i,t)$ between high and low fitnesses increases. The desire is to have this happen gradually over the course of the search, so temperature is gradually decreased according to a predefined schedule.
The selection probability is as follows for this method:

$$P_i = \frac{\exp(Bf_i)}{Z}$$

where B control the selection intensity and

$$Z = \sum_{j=1}^{n} \exp(bf_j)$$

Rogers and Prugel-Bennett proposed the selection intensity is nearly determined using B.
Fitness-proportionate selection is commonly used in GAs mainly because it was part of Holland's original proposal and because it is used in the Schema theorem, but, evidently, for many applications simple fitness-proportionate selection requires several "fixes" to make it work well.

*Proposed Annealed Selection*

The proposed selection approach is to move the selection criteria from exploration to exploitation so as to obtain the perfect blend of the two techniques. In this method, fitness value of each individual is computed. Depending upon the current generation number of genetic algorithm, selection pressure is changed and new fitness contribution, of each individual is computed. Selection probability of each individual is computed on the basis of . As the generation of population changes, fitness contribution changes and selection probability of each individual also changes.

The proposed blended selection operator computes fitness of individual depending on the current number of generation as under:

$$FX_i = \frac{FRW_i}{(ngen + 1) - nogen}$$

The probability for selecting the $i^{th}$ string is

$$PX_i = \frac{FX_i}{\sum_{i=1}^{N} FX_i}$$

where; $FX_i$ $\rightarrow$ Average Fitness of the population in generation in Proposed Blended Selection

$FRW_i \rightarrow$ Average Fitness of the population in $i^{th}$ generation in Roulette Wheel Selection

ngen $\rightarrow$ total number of generations

nogen $\rightarrow$ current number of generation

### 3.5.4 Rank Selection

In ranking selection, the individuals in the population are sorted from best to worst according to their fitness values. Each individual in the population is assigned a numerical rank based on fitness, and selection is based on this ranking rather than differences in fitness. Ranking selection sorts the genotypes of a generation according to the raw fitness score of their associated genotypes. The probability of a genotype being selected for reproduction depends only on its position in terms of fitness relative to the other genotypes and not on the actual fitness score.

The previous selection will have problems when the fitness differs very much. For example, if the best chromosome fitness is 90% of all the roulette wheel then the other chromosomes will have very few chances to be selected. Since rank selection first ranks the population and then every chromosome receives fitness from this ranking, the worst will have fitness 1, second worst 2 etc. and the best will have fitness $N$ (number of chromosomes in population). After this, all the chromosomes have the chance to be selected

Rank selection prevents too quick convergence, thus this method can lead to slower convergence because the best chromosomes do not differ so much from other ones and differs from roulette wheel selection in terms of selection pressure. Rank selection overcomes the scaling problems like stagnation or premature convergence when the selection has caused the search to narrow down too quickly. Ranking controls selective pressure by uniform method of scaling across the population.

In Rank Selection, sum of ranks is computed and then selection probability of each individual is computed as under:

$$rsum_i = \sum_{i=1}^{n} r_{i,j}$$

where $i$ varies from 1 to $ngen$ and $j$ varies from 1 to $N$.

$$PRANK_i = \frac{r_{i,j}}{rsum_i}$$

where;

$rsum_i \rightarrow$ sum of ranks in generation

$r_{i,j}^{th} \rightarrow$ rank of $j^{th}$ individual in $i^{th}$ generation for rank selection

$ngen \rightarrow$ total number of generations

The advantage of this method is that it can prevent very fit individuals from gaining dominance early at the expense of less fit ones, which would reduce the population's genetic diversity and might hinder attempts to find an acceptable solution.

The disadvantage of this method is that it required sorting the entire population by rank which is a potentially time consuming procedure.

## 3.6 Parameters for Genetic Algorithms

The operation of GAs begins with a population of a random string representing design or decision variables. The population is then operated by three main operators; reproduction, crossover and mutation to create a new population of points. GAs can be viewed as trying to maximize the fitness function, by evaluating several solution vectors. The purpose of these operators is to create new solution vectors by selection, combination or alteration of the current solution vectors that have shown to be good temporary solutions. The new population is further evaluated and tested till termination. If the termination criterion is not met, the population is iteratively operated by the above three operators and evaluated.

This procedure is continued until the termination criterion is met. One cycle of these operations and the subsequent evaluation procedure is known as a generation in GAs terminology. The operators are described as follows.

## 3.7 Crossover

After selection process, the basic operator for producing new chromosomes is the Crossover operator. In most GAs, individuals are represented by fixed-length strings and crossover operates on pairs of individuals (parents) to produce new strings (offspring) by exchanging segments from the parents' strings. Usually, chromosomes are randomly split and merged, with the consequence that some genes of a child come from one parent while others come from the other parents. One of the unique aspects of the work involving genetic algorithms (GAs) is the important role that Crossover plays in the design and implementation of robust evolutionary systems.
Crossover is a very powerful tool for introducing new genetic material and maintaining genetic diversity, but with the outstanding property that good parents also produce well-performing children or even better ones. Several investigations have come to the conclusion that crossover is the reason why sexually reproducing species have adapted faster than asexually reproducing ones.

It is intuitive from this construction that good sub-strings from parent strings can be combined to form a better child string, if an appropriate site is chosen. With a random site, the children strings produced may or may not have a combination of good sub-strings from parent strings, depending on whether or not the crossing site falls in the appropriate place. But this is not a matter of serious concern, because if good strings are created by crossover, there will be more copies of them in the next mating pool generated by crossover. It is clear from this discussion that the effect of crossover may be detrimental or beneficial. Thus, in order to preserve some of the good strings that are already present in the mating pool, all strings in the mating pool are not used in crossover.
Traditionally, the number of crossover points (which determines how many segments are exchanged) has been fixed at a very low constant value of 1 or 2.

Support for this decision came from early work of both a theoretical and empirical nature by (J. H. Holland, (1992)). In spite of this, other GAs problems were implemented using other types of crossover.

In this Section, a number of variations on crossover are described (Goldberg, 1989; Spears, 1997) and discussed and the relative merits of each reviewed.

- One-Point Crossover (Single-Point Crossover)
- Two-Point Crossover
- Multi Point Crossover (N Point Crossover)
- Uniform Crossover
- Three Parent Crossover
- Order Crossover
- Cycle Crossover (CX)
- Arithmetic Crossover
- Heuristic Crossover
- Partially Matched Crossover (PMX)
- Precedence Preservative Crossover (PPX)
- Crossover Probability

### 3.7.1 One-Point Crossover (Single Point Crossover)

A commonly used method for crossover is called one-point crossover. In this method, a single point crossover position (called cut point) is chosen at random and the parts of two parents after the crossover position (shown as vertical lines) are exchanged or swapped to form two offspring. Thus, the portion right of the selected site of these two strings is exchanged to form a new pair of strings. The new strings are thus a combination of the old strings. One-point crossover is more suitable when string length is small.

Then according to one-point crossover, if a random crossover point is chosen from left and segments are cut, we can see it as shown below:

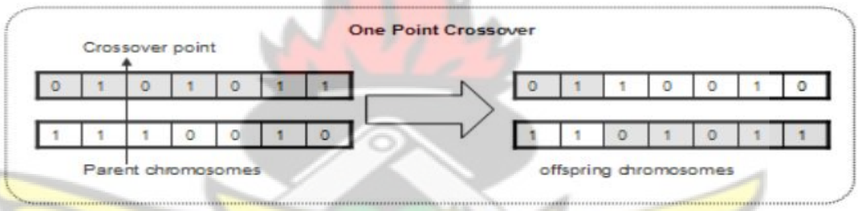


Figure 3.9: One-point crossover operation

### 3.7.2 Two-Point Crossover

To reduce bias and 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 to disrupt schemas with large likely defining lengths and can combine more schemas than one-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. As mentioned, the two-point crossover operator randomly selects points within chromosome then interchanges the two parent chromosomes between these two points to produce two new offspring for mating in the next generation as shown below:

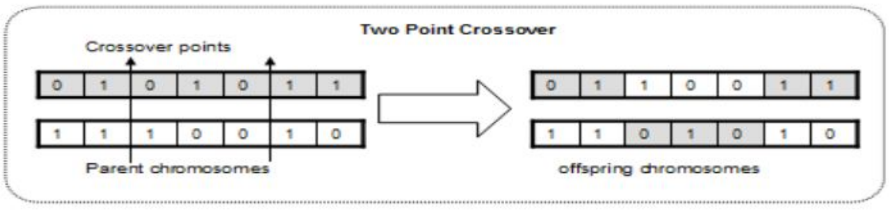


Figure 3.10: Two-point crossover

In figure 3.10, the arrows indicate the crossover points. Thus, the contents between these points are exchanged between the parents to produce new children for mating in the next generation.

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

Multi-point crossover is a generalization of single point crossover, introducing a higher number of cut-points. In this case multi positions are chosen at random and the segments between them are exchanged. Instead of only one, $N$ breaking points are chosen randomly. The section between the first allele position and the first crossover point is not exchanged between individuals. Every second section is swapped. Then, the bits between successive crossover points are exchanged between the two parents to produce two new offspring. The idea behind multi-point, and indeed many of the variations on the crossover operator, is that the parts of the chromosome representation that contribute to most of the performance of a particular individual may not necessarily be contained in adjacent substrings. Furthermore, the disruptive nature of multi-point crossover appears to encourage the exploration of the search space, rather than favoring the convergence to highly fit individuals early in the search, thus making the search more robust. Among this class, two-point crossover is particularly important.

### 3.7.4 Uniform Crossover

Another common recombination operator is uniform crossover (Syswerda, 1989; Spears and De Jong, 1994). The uniform crossover is a more general method. In uniform crossover, every allele is exchanged between the pair of randomly selected chromosomes with a certain probability, pe, known as the swapping probability. Usually the swapping probability value is taken to be 0.5. Uniform crossover does not use cut-points, but simply uses a global parameter to indicate the likelihood that each variable should be exchanged between two parents.

Uniform crossover, like multi-point crossover, has been claimed to reduce the bias associated with the length of the binary representation used and the particular coding for a given parameter set. This helps to overcome the bias in single-point crossover towards short substrings without requiring precise understanding of the significance of individual bits in the chromosome representation. Spears and De Jong, 1994 have demonstrated how uniform crossover may be parameterized by applying a probability to the swapping of bits. This extra parameter can be used to control the amount of disruption during recombination without introducing a bias towards the length of the representation used. When uniform crossover is used with real-valued alleles, it is usually referred to as *discrete recombination*.

In this method some independent genes (bits) is selected from each string stochastically and then are exchanged. However, for a uniform crossover, the following steps should be proceeded (Gen and Cheng, 2000):

1. Two chromosomes is selected by selection operator.
2. A part of parent chromosome is stochastically used to form a part of childes.
3. The second step is repeated until the total part of child is completed from.

Consider the following two parents, crossover mask and resulting offspring:

| | |
|---|---|
| Parent 1 | 1 0 1 1 0 0 0 1 1 1 |
| Parent 2 | 0 0 0 1 1 1 1 0 0 0 |
| Mask | 0 0 1 1 0 0 1 1 0 0 |
| Offspring 1 | 0 0 1 1 1 1 0 1 0 0 |
| Offspring 2 | 1 0 0 1 0 0 1 0 1 1 |

Figure 3.11: Uniform Crossover

Here, the first offspring, 1, is produced by taking the bit from 1 if the corresponding mask bit is 1 or the bit from 2 if the corresponding mask bit is 0.
Offspring 2 is created using the inverse of the mask or, equivalently, swapping P1 and P2.

### 3.7.5 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, the bit is taken for the offspring otherwise; the bit from the third parent is taken for the offspring.

| | |
|---|---|
| Parent 1 | 1 1 0 1 0 0 0 1 |
| Parent 2 | 0 1 1 0 1 0 0 1 |
| Parent 3 | 0 1 1 0 1 1 0 0 |
| Offspring | 0 1 1 0 1 0 0 1 |

Figure 3.12: Three Parent Crossover

### 3.7.6 Order Crossover (OX)

The original order crossover operator (which we refer to as order crossover) was developed by Davis, 1985. The offspring inherits the elements between the two crossover points, inclusive, from the selected parent in the same order and position as they appeared in that parent. The remaining elements are inherited from the alternate parent in the order in which they appear in that parent, beginning with the first position following the second crossover point and skipping overall elements already present in the offspring. Order Crossover is a fairly simple permutation crossover.

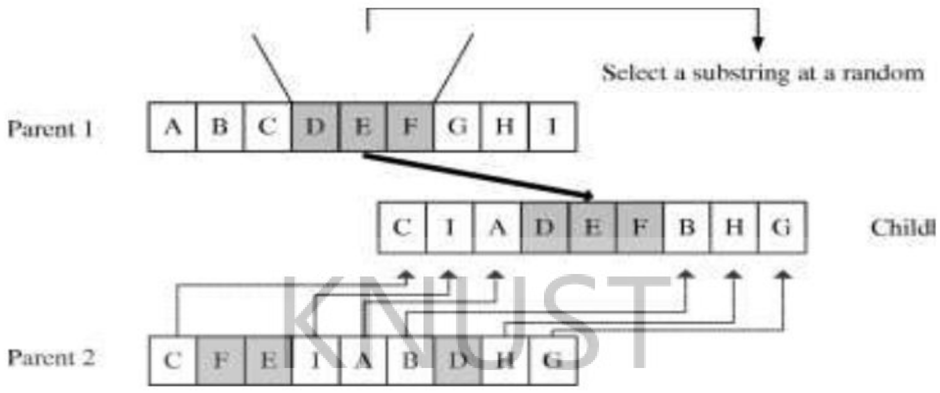


Figure 3.13: Order Crossover (OX)

### 3.7.7 Cycle Crossover (CX)

The Cycle Crossover (CX) proposed by (Oliver, 1987) builds offspring in such a way that each allele (and its position) comes from one of the parents. The Cycle Crossover operator identifies a number of so-called cycles between two parent chromosomes. Then, to form offspring 1, cycle one is copied from parent 1, cycle 2 from parent 2, cycle 3 from parent 1, and so on.

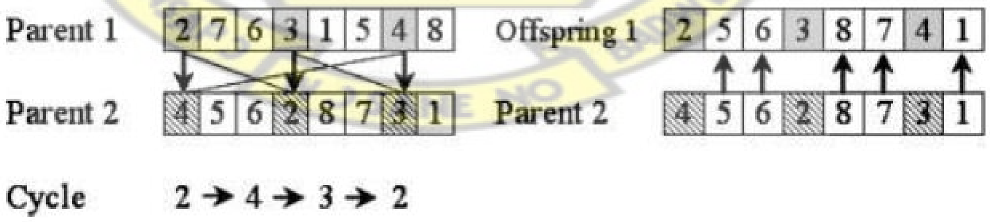


Figure 3.14: Cycle Crossover (OX)

### 3.7.8 Arithmetic Crossover

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

Offspring $1 = a *$ Parent $1 + (1 - a) *$ Parent 2

Offspring $2 = (1 - a) *$ Parent $1 + a *$ Parent 2

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 |

Figure 3.15: Arithmetic Crossover (Before)

Now applying the two equations and assuming that the weighing factor a=0.7, 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 |

Figure 3.16: Arithmetic Crossover (After)

### 3.7.9 Heuristic Crossover

Heuristic crossover operator uses the fitness value of two parent chromosomes to determine the direction of the search.
The offspring are created according to the equations:

Offspring $1 =$ Best Parent $+ r *$ (Best Parent $-$ Worst Parent )

Offspring $2 =$ Best Parent

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 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.

### 3.7.10 Partially Matched Crossover (PMX)

Apart from always generating valid offspring, the PMX operator (Goldberg and Lingle, 1985) also preserves orderings within the chromosome. In PMX, two parents are randomly selected and two random crossover sites are generated. Alleles within the two crossover sites of a parent are exchanged with the alleles corresponding to those mapped by the other parent. It can be said that it is a crossover of permutations that guarantees that all positions are found exactly once in each offspring, ie 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 illustrate how PMX works.

- Consider the two strings shown below.
- Where the dots marl 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 below:

```
Strings given
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

Partially matched crossover
Name   9 8 4 . 2 3 1 0 . 1 6 5 7   Allele   1 0 1 . 0 1 1 . 1 1 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
```

Figure 3.17: Partially Matched Crossover (PMX)

### 3.7.11 Precedence Preservative Crossover (PPX)

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

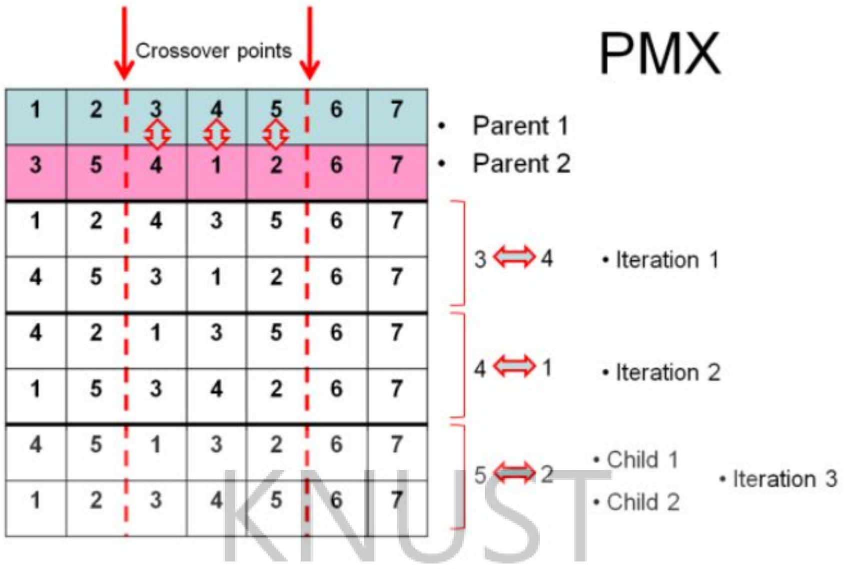


Figure 3.18: A PMX example

The operator works as follows:

- A vector of length Sigma ($\sigma$), sub $i = 1tomi$, 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. 3.19 below

| | |
|---|---|
| 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 |

Figure 3.19: Precedence Preservative Crossover (PPX)

### 3.7.12 Crossover Probability

The basic parameter in crossover technique is the crossover probability (Pc). 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 survives to next generation.

## 3.8 Mutation

If we use a crossover operator, such as one-point crossover, we may get better and better chromosomes but the problem is, if the two parents (or worse, the entire population) has the same allele at a given gene then one-point crossover will not change that. In other words, that gene will have the same allele forever. Mutation is designed to overcome this problem in order to add diversity to the population and ensure that it is possible to explore the entire search space. Hence mutation is employed to give new information to the population (uncover new chromosomes) and also prevents the population from becoming saturated with similar chromosomes, simply said to avoid premature convergence.

In natural evolution, mutation is a random process where one allele of a gene is replaced by another to produce a new genetic structure. Usually considered as a background operator, the role of mutation is often seen as providing a guarantee that the probability of searching any given string will never be zero and acting as a safety net to recover good genetic material that may be lost through the action of selection and crossover. This has the effect of tending to inhibit the possibility of converging to a local optimum, rather than the global optimum.

Given that mutation is generally applied uniformly to an entire population of strings, it is possible that a given binary string may be mutated at more than one point. With non-binary representations, mutation is achieved by either perturbing the gene values or random selection of new values within the allowed range.

Mutation adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima. It is an operator that introduces diversity in the population whenever the population tends to become homogeneous due to repeated use of reproduction and crossover operators. It may cause the chromosomes of individuals to be different from those of their parent individuals. The need for mutation is to create a point in the neighborhood of the current point, thereby achieving a local search around the current

solution.
For example, the following population having four eight bit strings may be considered:
01101011
00111101
00010110
01111100.
It can be noticed that all four strings have a 0 in the left most bit position. If the true optimum solution requires 1 in that position, then crossover operators described above will not be able to create 1 in that position. The inclusion of mutation introduces probability $pm$ of turning 0 into 1.
Where as the crossover operator recombines good sub-strings from good strings together, hopefully, to create a better sub-string, the mutation operator alters a string locally expecting a better string.

In evolutionary strategies, mutation is the primary variation/search operator. Unlike evolutionary strategies, mutation is often the secondary operator in GAs, performed with a low probability. In GA applications, mutation is randomly applied with low probability value called as mutation probability $P_m$, typically in the range 0.001 and 0.01, while for automated circuit design problems; it is usually between 0.3 and 0.8 and modifies elements in the chromosomes. The best mutation probability is 'application dependent'. Large mutation probability increases the probability that good schemata will be destroyed, but increase population diversity. A coin toss mechanism is employed; if random number between zero and one is less than the mutation probability, then the bit is inverted, so that zero becomes one and one becomes zero. This helps in introducing a bit of diversity to the population by scattering the occasional points. This random scattering would result in better optima, or even modify a part of genetic code that will be beneficial in later operations. On the other hand, it might produce a weak individual that will never be selected for further operations.

The commonest ones out of the many mutation operators are:

- Single Point Mutation
- Multi Point Mutation
- Bit Flip Mutation
- Interchanging Mutation
- Reversing Mutation
- Reorder Mutation
- Uniform Mutation
- Mutation Probability

### 3.8.1 Single Point Mutation

A commonly used method for mutation is called single point mutation. Single gene (chromosome or even individual) is randomly selected to be mutated and its value is changed depending on the encoding type used.

Consider two parents selected for mutation

| | |
|---|---|
| Parent 1 | 0 1 **1** 1 1 0 1 0 1 1 |
| Parent 2 | 1 1 1 0 0 1 **0** 0 1 0 |

Figure 3.20: Single Point Mutation (Before)

The mutated offspring produced after changing or inverting the value of the chosen gene as 0 to 1 and 1 to 0 are

| | |
|---|---|
| Offspring 1 | 0 1 **0** 1 1 0 1 0 1 1 |
| Offspring 2 | 1 1 1 0 0 1 **1** 0 1 0 |

Figure 3.21: Single Point Mutation (After)

### 3.8.2 Multi Point Mutation

Multi genes (chromosomes or even individuals) are randomly selected to be mutated and the values are changed depending on the encoding type used, as shown in Fig. 3.22 and Fig. 3.23.

Here are two parents selected for mutation.

| | |
|---|---|
| Parent 1 | 1 0 1 **1** 1 0 0 **1** 0 1 |
| Parent 2 | 1 **0** 1 1 0 0 **1** 0 0 1 |

Figure 3.22: Multi Point Mutation

The mutated offspring produced are

| | |
|---|---|
| Offspring 1 | 1 0 1 **0** 1 0 0 **0** 0 1 |
| Offspring 2 | 1 **1** 1 1 0 0 **0** 0 0 1 |

Figure 3.23: Multi Point Mutation

### 3.8.3 Bit-Flip Mutation (Flipping)

In bit-flip mutation, each chosen bit in a parent binary string (chromosome) is changed (a 0 is converted to 1, and a 1 is converted to 0) to produce offspring. A parent mutation chromosome is randomly generated. For a 1 in mutation chromosome, the corresponding bit in parent chromosome is flipped ( 0 to 1 and 1 to 0) offspring chromosome is produced. This is illustrated in the Fig. 3.24.

| | |
|---|---|
| Parent | **1** 1 1 **0** 1 1 0 **0** 0 1 |
| Mutation chromosome | **0** 1 0 **1** 1 0 0 **1** 0 0 |
| Offspring | **0** 1 1 **1** 1 1 0 **1** 0 1 |

Figure 3.24: Bit-Flip Mutation

### 3.8.4 Interchanging Mutation

Two random positions of the string are chosen and the bits corresponding to those positions are interchanged. This is shown in Fig. 3.25.

| | |
|---|---|
| Parent | 1 **0** 1 1 0 **1** 0 1 |
| Offspring | 1 **1** 1 1 0 **0** 0 1 |

Figure 3.25: Interchanging Mutation

### 3.8.5 Reversing Mutation

A random position is chosen and the bits next to that position are reversed and child chromosome is produced. This is shown in Fig. 3.26.

| | |
|---|---|
| Parent | 0 1 1 0 1 1 **0** 1 0 1 |
| Offspring | 0 1 1 0 1 1 0 **0** **1** **0** |

Figure 3.26: Reversing Mutation

### 3.8.6 Reorder Mutation

This swaps the positions of pair of bits or genes which are selected randomly to increase diversity in the decision variable space.

### 3.8.7 Uniform Mutation

The mutation operator replaces the value of the chosen gene with a uniform random value selected between the user-specific upper and lower bounds for that gene.

### 3.8.8 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 permutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, if is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extreme.

## 3.9 Replacement

Replacement, which is the last stage of any breeding cycle tends to be the most important stage. Replacement determines the current members of the population, ie old parents or offspring been produced should be replaced by new offspring if any. 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. 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 from 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 offspring and parent populations (for $(\lambda, \mu)$- and $(\lambda + \mu)$-update respectively), from 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 an entire new generation is produced at each time step. The insertion of a new individual usually necessitates the replacement of another population member. The individual to be deleted can be chosen as the worst member of population, (it leads to a 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 replacement, 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 subtile alternative is to replace the most similar member in the existing population.

When selecting which members of the old population should be replaced the most apparent strategy is to replace the least fit members deterministically. Thus, for an individual to survive successive generations, it must be sufficiently fit to ensure propagation into future generations.

Some of the most common replacement techniques are outlined below.

- Random Replacement
- Weak Parent Replacement
- Both Parent Replacement

### 3.9.1 Random Replacement

The children replace two random 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 population.

### 3.9.2 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 the population. This process improves the overall fitness of the population when paired with a selection technique that selects both fit and weak parent for crossing, but if weak individuals and discriminated against in selection, the opportunity will never raise to replace them.

### 3.9.3 Both Parents Replacement

Both parents replacement is when the child replaces the parent. In this case, each individual only gets to b reed 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 off.

## 3.10 Search Termination (Convergence Criteria)

Because the GA is a stochastic search method, it is difficult to formally specify convergence criteria. As the fitness of a population may remain static for a number of generations before a superior individual is found, the application of conventional termination criteria becomes problematic. A common practice is to terminate the GA after a pre-specified number of generations and then test the quality of the best members of the population against the problem definition. If no acceptable solutions are found, the GA may be restarted or a fresh search initiated.

The various stopping conditions are listed as follows:

- **Maximum generations**- the genetic algorithm stops when the specific number of generations have evolved.

- **Elapsed time**- The genetic process will end when a specific time has elapsed.

  **Note:** If the maximum number of generation has been reached before the specific time has elapsed, the process will end.

- **No change in fitness**- The genetic process will end if there is no change to the population's best fitness for a specific number of generations.

  **Note:** If the maximum number of generation has been reached before the specific number of generations with no changes has been reached, the process will end.

- **Stall generations**- The algorithm stops if there is no improvement in the subjective 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.

- **Best individual**- A best individual convergence criterion stops the search once the minimum fitness in the population drops below the convergence value. This brings the search to a faster conclusion guaranteeing at least one good solution.

- **Worst individual**- Worst individual terminates the search when the least fit individuals in the population have fitness less than the convergence criteria. This guarantees the entire population to be of minimum standard, although the best individual may not be significantly better than the worst.

In this case, a stringent convergence value may never be met, in which case the search will terminate after the maximum has been exceeded.

- **Sum of fitness**- In this termination scheme, the search is considered to have satisfaction converged when the sum of the fitness in the entire population is less than or equal to the convergence value in the population record. This guarantees that virtually all individuals in the population will be within a particular fitness range, although it is better to pair this convergence criteria with weakest gene replacement, otherwise a few unfit individuals in the population will blow out the fitness sum. The population size has to be considered while setting the convergence value.

- **Median fitness**- Here at least half of the individuals will be better than or equal to the convergence value, which should give a good range of solutions to choose from.

**A simple example of genetic algorithm for one generational cycle done by hand (Goldberg '89)**

For a better idea of what this section is about, let us have a look at a simple genetic algorithm and use it in an everyday application: searching for a maximum of the function $f(x) = x^2$, where $x$ can take values between 0 and 31. It is clear right away that the solution is the value $x = 31$, but this problem is simple enough that it shows nicely the genetic algorithm operation basics. General genetic algorithm starts with a randomly chosen population which is made up of binary series. The criterion function helps us determine their quality and based on this information, the number of copies which each individual subject will have at the construction of the next generation is set. Then, the acquired copies cross-over randomly between themselves and a random selection is made of the genetic string (chromosome) length which will cross-over or be switched (See Figure 3.27 below).
It can easily be seen that a good subject, which has many copies, has an advantage

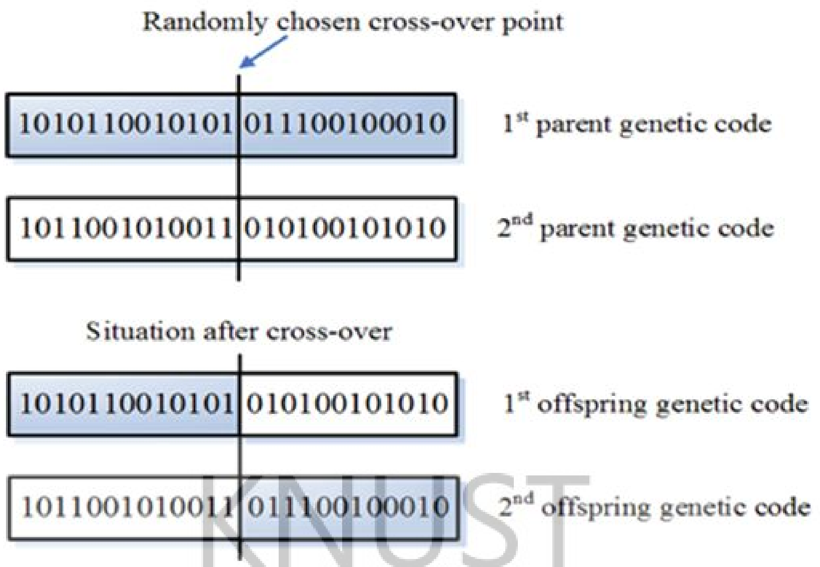


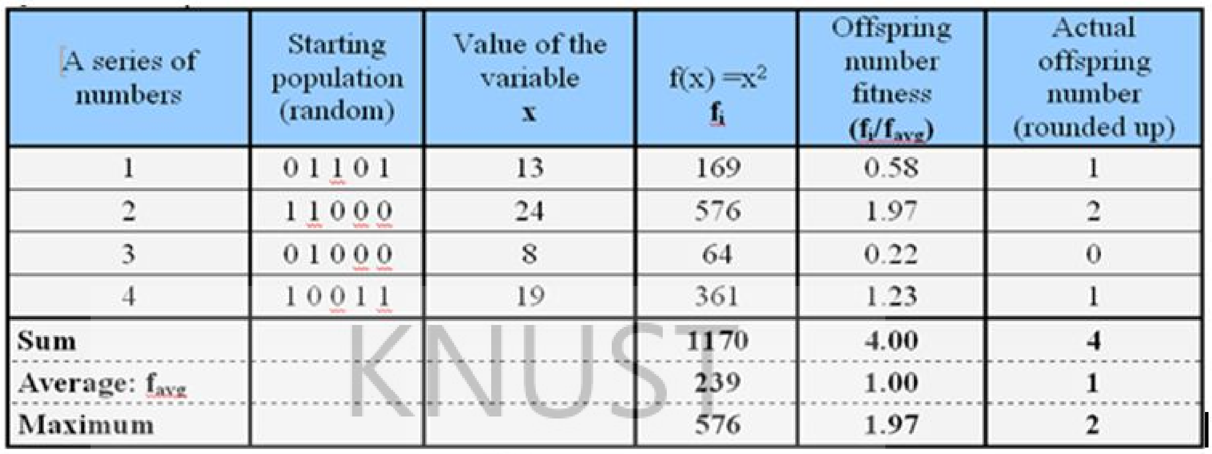
Figure 3.27: Genetic code of the parents and the offspring before and after the cross-over

in comparison with the weak one, because his genetic material can cross-over with more subjects than the genetic material of the weak ones. What follows is also the mutation for which we set the probability of 0.001 in this experiment. This means that there is a possibility of one in a thousand for each bit in the series to change from 0 to 1, or reverse. In our experiment, not even one bit mutated in the first generation. That is fine. If we take a look at the generation quality value, we can see that it has increased in the first generation already. We could continue with the process until we reached a subject with the quality 312 = 961, which would meet our criterion to stop searching for a maximum, or until the number of iterations would reach the value, prescribed in advance. That would mean that the algorithm did not find the solution; at least not a solution that would be as good as we determined it to be in our criterion in the beginning.

This execution of the genetic algorithm was simple, but it still shows the essential element of its operation, which is that the result of the optimisation is improving from generation to generation.

Table 3.2 shows an example of the characteristics (quality) computation of a randomly chosen starting population (first generation or the starting parents).

Table 3.2: The computation of a starting randomly chosen population characteristics



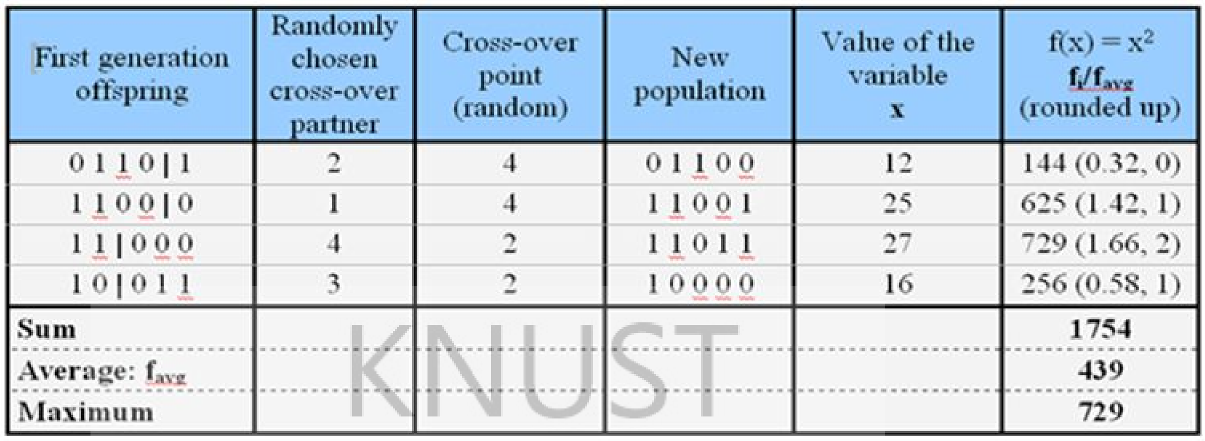
| A series of numbers | Starting population (random) | Value of the variable x | $f(x) = x^2$ $f_i$ | Offspring number fitness ($f_i/f_{avg}$) | Actual offspring number (rounded up) |
|---|---|---|---|---|---|
| 1 | 0 1 1 0 1 | 13 | 169 | 0.58 | 1 |
| 2 | 1 1 0 0 0 | 24 | 576 | 1.97 | 2 |
| 3 | 0 1 0 0 0 | 8 | 64 | 0.22 | 0 |
| 4 | 1 0 0 1 1 | 19 | 361 | 1.23 | 1 |
| **Sum** | | | **1170** | **4.00** | **4** |
| **Average: $f_{avg}$** | | | **239** | **1.00** | **1** |
| **Maximum** | | | **576** | **1.97** | **2** |

On the basis of the computed characteristics of the individual subjects, a selection is made and only the best subjects are allowed to continue with the cross-over (multiplication). Table 3.3 shows the cross-over of this first generation and the "birth" of the offspring and their characteristics (quality) computation with the criterion function $f(x) = x^2$. When we cross-over the parents (first column in Table 3.3), we get a new population with better characteristics (See Table 3.3 - columns 4-6). We assumed the value 0.001 for the mutation. Since there are four subjects in each generation, each of the subjects being five bits (binary places) long, the probability that one of them would mutate is 4*5*0.001=0.02. In the first step, none of the bits has mutated. We can keep computing in the same way until we reach the value $x = 31$ in just a few iterations.

Mutation for $f(x) = \max x^2$

It takes many more modifications and improvements for successful operation at solving a very complex problem, however, they are beyond this section.

Table 3.3: The presentation of the cross-over and the first generation offspring characteristics computation



| First generation offspring | Randomly chosen cross-over partner | Cross-over point (random) | New population | Value of the variable x | $f(x) = x^2$ $f_i/f_{avg}$ (rounded up) |
|---|---|---|---|---|---|
| 0 1 1 0 \| 1 | 2 | 4 | 0 1 1 0 0 | 12 | 144 (0.32, 0) |
| 1 1 0 0 \| 0 | 1 | 4 | 1 1 0 0 1 | 25 | 625 (1.42, 1) |
| 1 1 \| 0 0 0 | 4 | 2 | 1 1 0 1 1 | 27 | 729 (1.66, 2) |
| 1 0 \| 0 1 1 | 3 | 2 | 1 0 0 0 0 | 16 | 256 (0.58, 1) |
| **Sum** | | | | | **1754** |
| **Average: $f_{avg}$** | | | | | **439** |
| **Maximum** | | | | | **729** |

Table 3.4: The presentation of the mutation and the first generation offspring characteristics computation

| String No. | Offspring after xover | Offspring after mutation | $x$ value | Fitness $f(x) = x^2$ |
|---|---|---|---|---|
| 1 | 0 1 1 0 0 | **1** 1 1 0 0 | 26 | 676 |
| 2 | 1 1 0 0 1 | 1 1 0 0 1 | 25 | 625 |
| 3 | 1 1 0 1 1 | 1 1 0 1 1 | 27 | 729 |
| 4 | 1 0 0 0 0 | 1 0 **1** 0 0 | 18 | 324 |
| Sum | | | | **2354** |
| Average | | | | **588.5** |
| Max | | | | **729** |

## 3.11 When to use a Genetic Algorithm

The GA literature describes a large number of successful applications, but there are also many cases in which GA performs poorly. Given a particular potential application, how do you know if GA is a good method to use? There is no rigorous answer; though many researchers share the intuition that if the space to be searched is known not to be perfectly smooth and uni-modal (consist 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 search exhaustively, whereas a GA might converge on a local optimum rather than on the global 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 the GA in exploiting the space's smoothness. If the space is well understood, search methods using domain specific heuristics can often be design to outperform any general-purpose method such as a GA. If the fitness function is noisy (e.g., if it involves error-prone measurements from a real-world process such as 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 a 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.

## 3.12 Some uses of Genetic Algorithm

Genetic algorithms (GAs) are adaptive methods which may be used to solve search and optimization problems. The power of GAs comes from the fact that the technique is robust and can deal successfully with a wide range of problem areas, including those which are difficult for other methods to solve. Therefore, the main ground for GAs is in difficult areas where no such solving techniques exist. Even where existing techniques work well, improvements can be made by mixing them with GAs.
GAs in various forms are implemented to wide range of problems including the following:

- Optimization: GAs have been used in a wide variety of optimization tasks, including numerical optimization and combinatorial optimization problems such as circuit design and job shop scheduling.

- Automatic Programming: GAs have been used to evolve computer programs for special tasks and to design other computational structures cellular automata and sorting networks.
- Machine and robot learning: GAs have been used for many machine learning applications, including classification and prediction tasks such as the prediction of dynamical systems, weather prediction , and prediction of protein structure. GAs have also been used to design neural networks, and to evolve rules for learning classifier systems or symbolic production systems and to design and control robots
- Economic models: GAs have been used to model processes of innovation, the development of bidding strategies, and the emergence of economic markets.
- Immune system models: GAs have been used to model various aspects of the natural immune system including somatic mutation during an individual's lifetime and the discovery of multi-gene families during evolutionary time
- Ecological models: GAs have been used to model ecological phenomena such as biological arms races, host-parasite co-evolution, symbiosis, and resource flow in ecologies.
- Population genetics models: GAs have been used to study questions in population genetics, such as "under what conditions will a gene for recombination be evolutionarily viable (or practicable) ?"
- Interactions between evolution and learning: GAs have been used to study how individual learning and species evolution affect one another.
- Models of social systems: GAs have been used to study evolutionary aspects of social systems, such as the evolution of cooperation, the evolution of communication, and trail-following behaviour in ants.

This list is by no means exhaustive, but it gives a flavour of the kinds of things for which GAs have been used, both for problem-solving and for modelling.

## 3.13 Advantages and Disadvantages of Genetic Algorithms

Perhaps it is not obvious why such an algorithm should lead to accurate solutions for optimization problems.

**Advantages:**

- It can solve every optimisation problem which can be described with the chromosome encoding.
- It solves problems with multiple solutions.
- Since the genetic algorithm execution technique is not dependent on the error surface, we can solve multi-dimensional, non-differential, non-continuous, and even non-parametrical problems.
- Structural genetic algorithm gives us the possibility to solve the solution structure and solution parameter problems at the same time by means of genetic algorithm.
- Genetic algorithm is a method which is very easy to understand and it practically does not demand the knowledge of mathematics.
- Genetic algorithms are easily transferred to existing simulations and models.

**Disadvantages:**

- Certain optimisation problems (they are called variant problems) cannot be solved by means of genetic algorithms. This occurs due to poorly known fitness functions which generate bad chromosome blocks in spite of the fact that only good chromosome blocks cross-over.

- There is no absolute assurance that a genetic algorithm will find a global optimum. It happens very often when the populations have a lot of subjects.

- Like other artificial intelligence techniques, the genetic algorithm cannot assure constant optimisation response times. Even more, the difference between the shortest and the longest optimisation response time is much larger than with conventional gradient methods. This unfortunate genetic algorithm property limits the genetic algorithms' use in real time applications.

- Genetic algorithm applications in controls which are performed in real time are limited because of random solutions and convergence, in other words this means that the entire population is improving, but this could not be said for an individual within this population. Therefore, it is unreasonable to use genetic algorithms for on-line controls in real systems without testing them first on a simulation model.

## 3.14 Flow Charts Showing Genetic Application to VRP

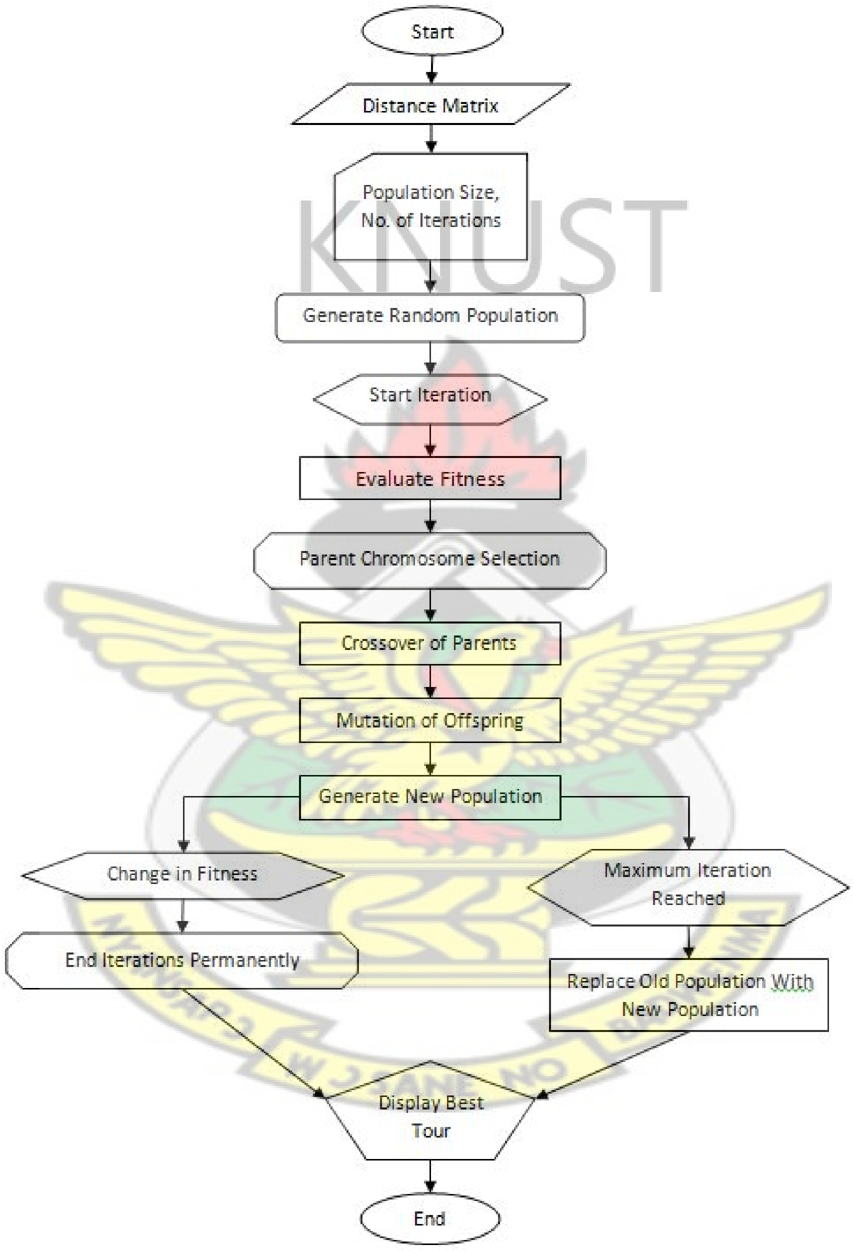

Start
Distance Matrix
Population Size, No. of Iterations
Generate Random Population
Start Iteration
Evaluate Fitness
Parent Chromosome Selection
Crossover of Parents
Mutation of Offspring
Generate New Population
Change in Fitness
Maximum Iteration Reached
End Iterations Permanently
Replace Old Population With New Population
Display Best Tour
End

Figure 3.28: The Flow Chart

## 3.15 The Vehicle Routing Problem

Vehicle Routing Problem (VRP) lies at the heart of distribution management. It is faced each day by thousands of companies and organizations engaged in the delivery and collection of goods or people. Because conditions vary from one setting to the next, the objectives and constraints encountered in practice are highly variable. Most algorithmic research and software development in this area focus on a limited number of prototype problems. By building enough flexibility in optimization systems one can adapt these to various practical contexts. Much progress has been made since the publication of the first article on the "truck dispatching" problem by Dantzig and Ramser (1959). Several variants of the basic problem have been put forward.

The VRP can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several customer demands, finds the set of routes with overall minimum route cost which service all the demands. All the itineraries start and end at the depot and they must be designed in such a way that each customer is served only once and just by one vehicle.

### 3.15.1 VRP in Real Life

The VRP is of great practical significance in real life. It appears in a large number of practical situations, such as transportation of people and products, delivery service and garbage collection. For instance, such a matter of course as being able to buy milk in a store arises the use of vehicle routing twice. First the milk is collected from the farms and transported to the dairy and when it has been put into cartons it is delivered to the stores. That is the way with most of the groceries we buy. And the transport is not only made by vehicles but also by plains, trains and ships. VRP is everywhere around.

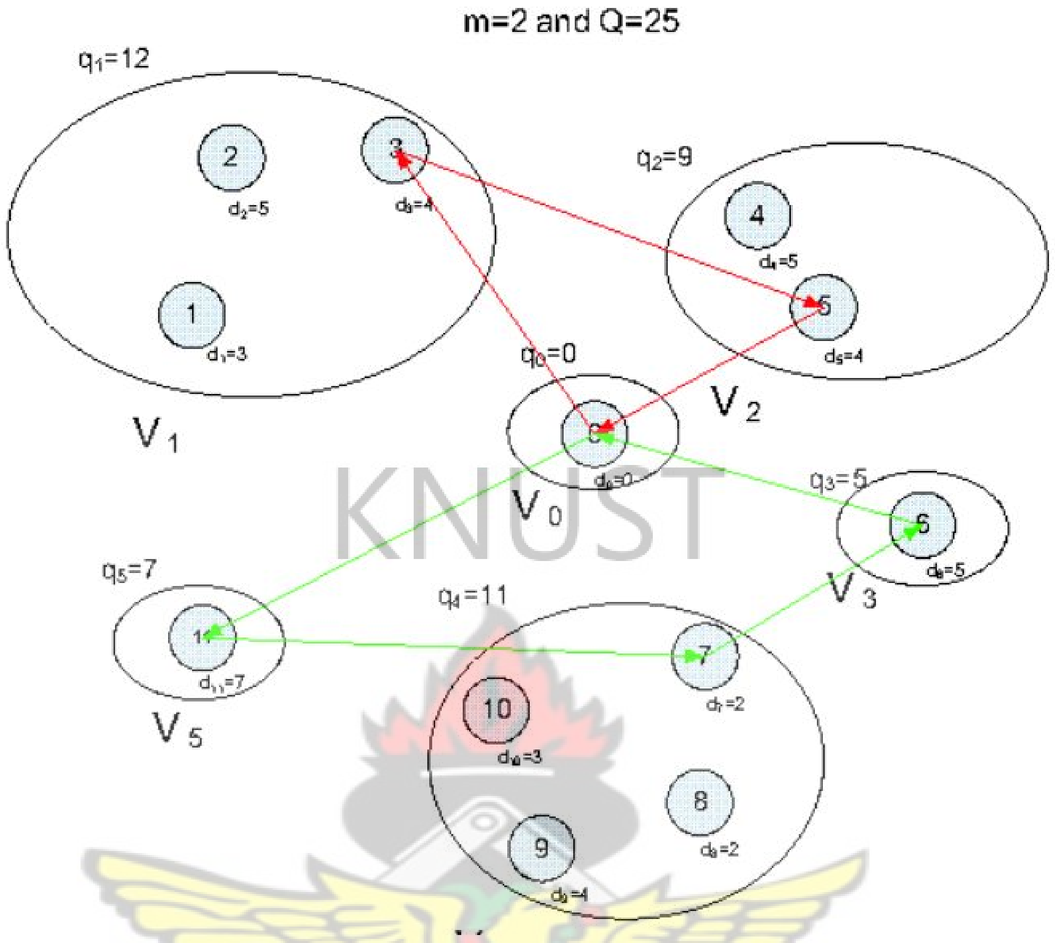


Figure 3.29: An illustration of a VRP

One can therefore easily imagine that all the problems, which can be considered as VRP, are of great economic importance, particularly to the developed nations. The economic importance has been a great motivation for both companies and researches to try to find better methods to solve VRP and improve the efficiency of transportation.

## 3.16 Solution Methods and Literature Review of VRP

In real life, VRP can have many more complications, such as asymmetric travel costs, multiple depots, heterogeneous vehicles and time windows, associated with each customer. These possible complications make the problem more difficult to solve. A little is considered in this project because the emphasis is rather on Genetic Algorithms.

The VRP is an NP-hard combinatorial optimization problem which is usually very time consuming or even impossible and only relatively small instances can be solved to optimality. Since exact approaches are in general inadequate for solving large instances, heuristics are commonly used in most approaches.

## 3.17 Heuristics For VRP

Heuristics are approximation algorithms that aim at finding good feasible solutions quickly. An impressive number of heuristics have been proposed for the VRP. In the following we separately review these two families of algorithms. Almost all of these methods were developed, described and tested for the symmetric VRP. In addition, since finding a feasible solution with exactly m vehicles is itself an NP-complete problem, almost all methods assume an unlimited number of available vehicles. However, it should be observed that many of the proposed methods may be quite easily adapted to take into account additional practical constraints, although these may affect their overall performance. They can be roughly divided into two main categories;

1. Classical Heuristics

2. Metaheuristics

### 3.17.1 Classical Heuristics

Several heuristics have been devised for the VRP, only some of which are sufficiently well known to be truly viewed as 'classical'. In the early classical heuristics, much of the emphasis was put on quickly obtaining a feasible solution and possibly applying to it a post optimization procedure. Using the classification proposed by Laporte and Semet (2002), we describe Classical Heuristics for VRP under these headings: The Classical Heuristics can be divided into three groups;

- Route Construction methods,
- Two phase methods
- Route Improvement methods

#### Route Construction Methods

Route construction methods were among the first heuristics for the CVRP and still form the core of many software implementations for various routing applications. These algorithms typically start from an empty solution and iteratively build routes by inserting one or more customers at each iteration, until all customers are routed. Construction algorithms are further subdivided into sequential and parallel, depending on the number of eligible routes for the insertion of a customer. Sequential methods expand only one route at a time, whereas parallel methods consider more than one route simultaneously. Route construction algorithms are fully specified by their three main ingredients, namely an initialization criterion, a selection criterion specifying which customers are chosen for insertion at the current iteration, and an insertion criterion to decide where to locate the chosen customers into the current routes. Route Construction methods gradually build a feasible solution by selecting arcs based on minimizing cost like

- Nearest Neighbour method
- Clarke and Wright based heuristic

**Nearest Neighbour Method**

In this algorithm the rule is always to go next to the nearest unvisited customer subject to the following restrictions:

- We start from the depot
- From each cluster is visited exactly one vertex (customer) and
- The sum of all the demands of the current tour (route) does not exceed the capacity of the vehicle Q.

If the sum of all the demands of a current tour (route) exceeds the capacity of the vehicle then we start again from the depot and visit next the nearest customer from an unvisited yet cluster. If all the clusters are visited, then the algorithm terminates.

**Clarke and Wright based heuristic**

The first and most famous heuristic of this group was proposed by Clarke and Wright (1964) and is based on the concept of *saving*, an estimate of the cost reduction obtained by serving two customers sequentially in the same route, rather than in two separate ones. If $i$ is the last customer of a route and $j$ is the first customer of another route, the associated saving is defined as $s_= c_{i0} + c_{0j} - c_{ij}$. If $s_{ij}$is positive, then serving i and j consecutively in a route is profitable. The Clarke and Wright algorithm considers all customer pairs and sorts the savings in non increasing order. Starting with a solution in which each customer appears separately in a route, the customer pair list is examined and two routes are merged whenever this is feasible. Generally, a route merge is accepted only if the associated saving is nonnegative but, if the number of vehicles is to be minimized, then negative saving merges may also be considered. The Clarke and Wright algorithm is inherently parallel since more than one route is active at any time. However, it may easily be implemented in a sequential fashion.

Given:

1. n customers with known locations and demands
2. identical delivery vehicles of known capacity
3. a distance or time limit for every vehicle route

Design vehicle routes so that the total length is minimized and

1. the requirements of all customers are satisfied,
2. the vehicle capacities are not exceeded, and
3. the distance or time limit is not violated.

**Clarke and Wright Savings**

Label the customers as cities $1, 2, \ldots, n$ and let the warehouse be city 0. Determine the costs $c_{ij}$ to travel between all pairs of cities and the warehouse $i = 0, 1, \ldots, n;\ j = 0, \ldots, n$.

1. Select the warehouse as the central city.
2. Calculate the savings $s_{ij} = c_{i0} + c_{0j}cij$ for all pairs of cities (customers) $i, j\ \ (i = 1, 2, \ldots, n; j = 1, 2, \ldots, n; i \neq j)$.
3. Order the savings, $s_{ij}$, from largest to smallest.
4. Starting with the largest savings, do the following:
   (a) If linking cities $i$ and $j$ results in a feasible route, then add this link to the route; if not, reject the link.
   (b) Try the next savings in the list and repeat (a). Do not break any links formed earlier. Start new routes when necessary. Stop when all cities are on a route.

**Two Phase Methods**

Two-phase methods are based on the decomposition of the VRP solution process into the two separate subproblems:

1. Clustering: determine a partition of the customers into subsets, each corresponding to a route, and

2. Routing: determine the sequence of customers on each route.

In a cluster-first-route-second method, customers are first grouped into clusters and the routes are then determined by suitably sequencing the customers within each cluster. Different techniques have been proposed for the clustering phase, while the routing phase amounts to solving a TSP. Examples of the Two phase method are

- The Sweep Algorith

- The Fisher and Jaikumar Algorithm

These were selected partly because of their popularity, and also because they operate on vastly different principles.

**The Sweep Algorithm**

The sweep algorithm, due to Wren (1971), Wren and Holliday (1972), and Gillett and Miller (1974), is often referred to as the first example of cluster first-route-second approach. The algorithm applies to planar VRP instances.
The algorithm starts with an arbitrary customer and then sequentially assigns the remaining customers to the current vehicle by considering them in order of increasing polar angle with respect to the depot and the initial customer. As soon as the current customer cannot be feasibly assigned to the current vehicle, a new route is initialized with it. Once all customers are assigned to vehicles, each route is separately defined by solving a TSP.
This heuristic is of the type "clustering first, route later". Assume the customers

are points in a plane with euclidean distances as costs. The distance between $(x_i, y_i)$ and $(x_j, y_j)$ is

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

**i.** Compute the polar coordinates of each customer with respect to the depot. Sort the customers by increasing polar angle.

**ii.** Add loads to the first vehicle from the top of the list as long as the capacity allows. Continue with the next vehicle until all customers are included. Now the customers have been clustered by vehicles.

**iii.** For each vehicle, optimize its route by a suitable TSP method. Graphically, in step I we rotate a ray centered at the depot. The starting angle can be chosen arbitrarily.

**The Fisher and Jaikumar Algorithm**

The Fisher and Jaikumar (1981) algorithm solves the clustering step by means of an appropriately defined Generalized Assignment Problem (GAP) which calls for the determination of a minimum cost assignment of items to a given set of bins of capacity Q, and where the items are characterized by a weight and an assignment cost for each bin. Each vehicle is assigned a representative customer, called a seed, and the assignment cost of a customer to a vehicle is equal to its distance to the seed. The GAP is then solved, either optimally or heuristically, and the final routes are determined by solving a TSP on each cluster.

Fisher-Jaikumar Generalized Assignment based algorithm

- Step 1 (seed selection): select a seed $j_k$ in $V$ for each cluster $k = 1, \dots, K$
- Step 2 (allocation of customers to seed): compute $c_{ik}$ the cost of allocating

customer i to k as the cost of inserting $i$ in the route $0 - j_k - 0$

$$c_{ik} = \min\ \{c_{0i} + c_{ijk} + c_{jk0},\ c_{0jk} + c_{jki} + c_{i0}\} - (c_{0jk} + c_{jk0})$$

- Step 3 (generalized assignment): Solve a GA problem with costs $c_{ij}$, weights for the customers $d_i$, and vehicle capacity $Q$
- Step 4 (TSP solution): solve a TSP for each cluster found

A different family of two-phase methods is the class of so-called petal algorithms. These generate a large set of feasible routes, called petals, and select the final subset by solving a set partitioning model. Foster and Ryan (1976) and Ryan et al. (1993) have proposed heuristic rules for determining the set of routes to be selected, while Renaud et al. (1996b) have described an extension that considers more involved configurations, called 2-petals, consisting of two embedded or intersecting routes. The overall performance of these algorithms is generally superior to that of the sweep algorithm.

**Route Improvement methods**

Local search algorithms are often used to improve initial solutions generated by other heuristics. Starting from a given solution, a local search method applies simple modifications, such as arc exchanges or customer movements, to obtain neighbor solutions of possibly better cost.
If an improving solution is found, it then becomes the current solution and the process iterates; otherwise a local minimum has been identified.
A large variety of neighborhoods are available. These may be subdivided into intra-route neighborhoods, if they operate on a single route at a time or inter-route neighborhoods if they consider more than one route simultaneously.
The most common neighborhood type is the $\lambda$-opt heuristic of Lin (1965) for the TSP, where $\lambda$ edges are removed from the current solution and replaced by $\lambda$ others. The computing time required to examine all neighbors of a solution is

proportional to $n\lambda$. Thus, only $\lambda = 2$ or 3 are used in practice. As an alternative, one can use restricted neighborhoods characterized by subsets of moves associated with larger $\lambda$ values, such as Or-exchanges (Or, 1976).

The advantage of the classical heuristics is that they have a polynomial running time, thus using them one is better able to provide good solutions within a reasonable amount of time. On the other hand, they only do a limited search in the solution space and do therefore run the risk of resulting in a local optimum.

### 3.17.2 Metaheuristics

Metaheuristics are more effective and specialized than the classical heuristics. They combine more exclusive neighbourhood search, memory structures and recombination of solutions and tend to provide better results, e.g. by allowing deterioration and even infeasible solutions. However, their running time is unknown and they are usually more time consuming than the classical heuristics. Furthermore, they involve many parameters that need to be tuned for each problem before they can be applied. Several metaheuristics have been applied to the VRP. With respect to classical heuristics, they perform a more thorough search of the solution space and are less likely to end with a local optimum. These can be broadly divided into three classes:

1. Local Search, including
   - Simulated Annealing (SA)
   - Deterministic Annealing (DA)
   - Tabu Search (TS)
2. Learning Mechanisms, including
   - Neural Networks (NN)
   - Ant Colony Optimization (ACO)

3. Population Search, including

    - Adaptive Memory Procedures (AMP)
    - Genetic Algorithm (GA)

The best heuristics often combine ideas borrowed from different metaheuristic principles. Recent surveys of VRP metaheuristics can be found in Gendreau et al. (2002), Cordeau and Laporte (2004), and Cordeau et al. (2005).

The algorithms SA, DA and TS move from one solution to another one in the neighbourhood until a stopping criterion is satisfied. The fourth method, AS, is a constructive mechanism creating several solutions in each iteration based on information from previous generations. NN is a learning method, where a set of weights is gradually adjusted until a satisfactory solution is reached. Finally, GA maintain a population of good solutions that are recombined to produce new solutions. Compared to best-known methods, SA, DA and AS have not shown competitive results and NN are clearly outperformed. TS has got a lot of attention by researches and so far it has proved to be the most effective approach for solving VRP. Many different TS heuristics have been proposed with unequal success.

**Local Search**

Local search algorithms explore the solution space by iteratively moving from a solution $x_t$ at iteration t to a solution xt+1 in the neighborhood $N(x_t)$ of $xt$ until a stopping criterion is satisfied. If $f(x)$ denotes the cost of solution $x$, then $f(x_{t+1})$ is not necessarily smaller than $f(x_t)$. As a result, mechanisms must be implemented to avoid cycling.

A limited number of simulated annealing heuristics for the CVRP were proposed in the early 1990s. Osman's implementation (Osman, 1993) is the most involved and also the most successful. It defines neighborhoods by means of a 2-interchange scheme and applies a different rule of temperature changes. Instead of using a non increasing function, as do most authors in the field, Osman decreases $\theta_t$ continuously as long as the solution improves, but whenever $x_{t+1} = x_t t, \theta_t$ is either halved or replaced by the temperature at which the incumbent was identified. This algorithm succeeded in producing good solutions but was not competitive with the best tabu search implementations available at the same period.

In Simulated Annealing, a solution x is drawn randomly from $N(x_t)$.
If $f(x) \leq f(x_t)$, then $x_{t+1} := x$. Otherwise,

$$x_{t+1} = \begin{cases} x & \text{with probability } p_t \\ x_t & \text{with probability } 1 - p_t \end{cases}$$

where $p_t$ is a decreasing function of $t$ and of $f(x) - f(x_t)$. This probability is often equal to

$$p_t = \exp\left(1 - \frac{f(x) - f(x_t)}{\theta_t}\right)$$

where $\theta_y$ is the *temperature* at iteration $t$, usually defined as a non increasing function of t.

Deterministic Annealing (Dueck, 1990, 1993) is similar. There are two main versions of this algorithm: in a threshold-accepting algorithm, $x_{t+1} := x$ if $f(x) < f(x_t) + \theta_1$, where $\theta_1$ is a user controlled parameter; in record-to-record travel, a record is the best known solution $x*$, and $x_{t+1} := xiff(x_{t+1}) < \theta_2 f(x*)$, where $\theta_2$ is also user controlled.
Deterministic annealing was first applied to the VRP by Golden et al. (1998) and more recently by Li et al. (2005). The latter algorithm combines the record-to-record principle of Dueck (1993) with GTS. It works on a sparsified graph containing only a proportion $\alpha$ of the 40 shortest edges incident to each vertex,

where $\alpha$ varies throughout the algorithm. The algorithm is applied several times from three initial solutions generated by the Clarke and Wright (1964) algorithm, with savings $s_{ij}$ defined as $c_{i0} + c_{0j} - \lambda c_{ij}$, and $\lambda = 0.6, 1.4,$ and 1.6.
Neighbors are defined by means of intra- and inter-route 2-opt moves, and Non improving solutions are accepted as long as their cost does not exceed that of the incumbent by more than 1%. Whenever the solution has not improved for a number of iterations, a perturbation is applied to the best known solution to restart the search. This is achieved by temporarily moving some vertices to different positions.

In Tabu Search, in order to avoid cycling, any solution possessing some given attribute of $x_{t+1}$ is declared tabu for a number of iterations. At iteration t, the search moves to the best non-tabu solution $x$ in $N(x_t)$. These local search algorithms are rarely implemented in their basic version, and their success depends on the careful implementation of several mechanisms. The rule employed to define neighborhoods is critical to most local search heuristics. In simulated annealing several rules have been proposed to define Î¸t (see Osman, 1993). Tabu search relies on various strategies to implement tabu tenures (also known as short term memory), search diversification (also known as long term memory), and search intensification which accentuates the search in a promising region.
As written above, to date Tabu Search has been the best metaheuristic for VRP. The heuristic starts with an initial solution $x_1$ and in step t it moves from solution xt to the best solution $x_{t+1}$ in its neighbourhood N(xt), until a stopping criterion is satisfied. If $f(x_t)$ denotes the cost of solution $xt$, $f(x_{t+1})$ does not necessarily have to be less than $f(x_t)$. Therefore, a cycling must be prevented, which is done by declaring some recently examined solutions tabu or forbidden and storing them in a tabulist. Usually, the TS methods preserve an attribute of a solution in the tabulist instead of the solution itself to save time and memory.

**Learning mechanisms**

A limited number of heuristics based on learning mechanisms have been proposed for the VRP. None of the known neural networks based methods is satisfactory, and the early ant colony based heuristics could not compete with the best available approaches.

**Neural Networks** are models composed of richly interconnected units through weighted links, like neurons in the brain. They gradually construct a solution through a feedback mechanism that modifies the link weights to better match an observed output to a described output. In the field of vehicle routing neural network models called the elastic net and the self-organizing map are deformable templates that adjust themselves to the contour of the vertices to generate a feasible VRP solution. An example is provided by Ghaziri (1993).

**Ant Colony** algorithms (see Dorigo et al., 1999) also use a learning mechanism. They are derived from an analogy with ants which lay some pheromone on their trail when foraging for food. With time more pheromone is deposited on the more frequented trails. When constructing a VRP solution a move can be assigned a higher probability of being selected if it has previously led to a better solution in previous iterations.

**Population Search**

Population search algorithms operate on several generations of solution populations. Population search consists of maintaining a pool of good parent solutions and recombining them to produce offspring.

The **Adaptive Memory Procedure (AMP)** put forward by Rochat and Taillard (1995) constitutes a major contribution to the field of metaheuristics. An adaptive memory is a pool of good solutions which is updated by replacing its worst elements with better ones. In order to generate a new solution, several solutions are selected from the pool and recombined. In the context of the VRP, vehicle routes are extracted from these solutions and used as the basis of a new solution. The extraction process is applied as long as it is possible to identify routes that do not overlap with previously selected routes. When this is no longer possible, a search process (e.g., tabu search) is initiated from a partially constructed solution made up of the selected routes and some unrouted customers. Any solution constructed in this fashion replaces the worst solution of the pool if it has a better cost.

In **Genetic Algorithm (GA)**, it is common to repeat the following operation $k$ times: extract two parent solutions from the populations to create two offspring using a crossover operation, and apply a mutation operation to each offspring; then remove the $2k$ worst elements from the population and replace them with the $2k$ offspring.

### 3.17.3 Attributes of Good VRP Heuristics

Four attributes of good VRP heuristics Vehicle routing heuristics, as are most heuristics, are usually measured against these criteria: accuracy, speed, flexibility and simplicity.

**Accuracy**

Accuracy measures the degree of departure of a heuristic solution value from the optimal value. Since optima and sharp lower bounds are usually unavailable in the case of the VRP, most comparisons have to be made with best known values.

**Speed**

Just how important is computation speed in vehicle routing? It all depends on the planning level at which the problem is solved and on the degree of accuracy required. At one extreme, real-time applications such as express courier pickup and delivery or ambulance redeployment require fast, sometimes almost instantaneous, action.

**Flexible**

A good VRP heuristic should be flexible enough to accommodate the various side constraints encountered in a majority of real-life applications. Experiences suggest that an efficient way of handling side constraints in a local search process is through the use of two objectives. The first, $F(x)$, computes the routing cost of solution $x$. The second, $F'(x)$, is the sum of $F(x)$ and weighted penalty terms associated with violations of each side constraint.

**Simplicity**

Simplicity Several VRP heuristics are rarely implemented because they are just too complicated to understand and to code.

# Chapter 4

# A Genetic Algorithm Method for Vehicle Routing Problem (VRP)

## 4.1 The Model

The most general version of VRP is the Capacitated Vehicle Routing Problem, which will be referred to as just VRP from now on. The VRP consists of designing a set of at most m delivery or collection routes with the following assumptions:

1. Each route starts and ends at the depot
2. Each customer is visited exactly once by exactly one vehicle
3. The total demand of each route does not exceed the capacity of the vehicle
4. The total routing cost is minimized.

The model for VRP has the following parameters:

- $n$ is the number of customers
- $K$ denotes the capacity of each vehicle
- $d_i$ denotes the demand of customer $i$
- $c_{ij}$ is the cost of travelling from customer $i$ to customer j

All parameters are considered non-negative integers. A homogeneous fleet of vehicles with a limited capacity K and a central depot, with index 0, makes deliveries to customers, with indices 1 to n. The problem is to determine the exact tour of each vehicle starting and ending at the depot. Each customer must be assigned to exactly one tour, because each customer can only be served by one vehicle.

The sum over the demands of the customers in every tour has to be within the limits of the vehicle capacity. The objective is to minimize the total travel cost. That could also be the distance between the nodes or other quantities on which the quality of the solution depends, based on the problem to be solved. Hereafter it will be referred to as a cost.

The mathematical model is defined on a graph $(N, A)$, Let $G(N, A)$ be a graph where $N = \{n_0, n_l, \ldots, n_n\}$ is a node set. Node $n_0$ represents a depot, while the remaining nodes correspond to customers, thus node set $N$ corresponds to the set of customers $C$ from 1 to $n$ in addition to the depot number 0. With $A = \{(n_i, n_j) : n_i, n_j \in N, i \neq j\}$ as an arc set. The arc set $A$ consists of possible connections between the nodes. A connection between every two nodes in the graph will be included in $A$ here. With each arc $(i, j) \in A$ are associated a cost matrix $(c_{ij})$ and a travel time matrix $(t_{ij})$. It is assumed that the cost is symmetric, i.e. $c_{ij} = c_{ji}$, and also that $c_{ii} = 0$. If these matrices are symmetrical, as is commonly the case, then it is standard to define the VRP on an undirected graph $G = (N, E)$, where $E = \{(n_i, n_j) : n_i, n_j \in N, i < j\}$ is an edge set. The number of vehicles is either known in advance or treated as a decision variable. Each customer has a non-negative demand $d_i$ and a service time $t_i$. A fleet of $V$ identical vehicles of capacity $K$ is based at the depot.

The decision variables are $X_{ij}^v$ and $d_i^v$

$$X_{ij}^v = \begin{cases} 1, & \text{if vehicle } v \text{ drives from node } i \text{ to node } j \\ 0, & Otherwise \end{cases} \tag{4.1}$$

$$d_i^v = \text{ fraction of customer's demand } i \text{ delivered by vehicle } v \tag{4.2}$$

The objective function of the mathematical model is

$$\min \sum_{v\in V} \sum_{(i,j)\in A} c_{ij} X_{ij}^{v} \qquad (4.3)$$

$$\begin{aligned}
\text{Subject to} \quad \sum_{v\in V}\sum_{j\in N} X_{ij}^{v} &= 1 \quad \forall i \in C && (4.4)\\
\sum_{i\in V_l} d_i &= 1 \quad \text{for } l = 1,\dots,k && (4.5)\\
\sum_{i\in C} d_i \sum_{j\in N} X_{ij}^{v} &\leq K \quad \forall v \in V && (4.6)\\
\sum_{j\in C} X_{0j}^{v} &= 1 \quad \forall v \in V && (4.7)\\
\sum_{i\in N} X_{ik}^{v} - \sum_{j\in N} X_{kj}^{v} &= 0 \quad \forall k \in C \text{ and } \forall v \in V && (4.8)\\
\sum_{v\in V}\sum_{j\in N} X_{ij}^{v} &\geq 1 \quad \forall j \in N && (4.9)\\
d_i^{v} &\geq 0, \quad \forall i \in A \text{ and } \forall v \in V && (4.10)
\end{aligned}$$

$$X_{ij}^{v} \in \{0,1\}, \quad \forall (i,j) \in A \text{ and } \forall v \in V \qquad (4.11)$$

Constraint Explanations are as follows:

- Constraint 4.4 is to make sure that each customer is assigned to exactly one vehicle.
- Constraint 4.5 guarantees that the total demand of each customer will be fulfilled.
- In equation 4.6 the capacity constraints is stated. The sum over the demands of the customers within each vehicle v has to be less than or equal to the capacity of the vehicle. This guarantees that the vehicle capacity will not be exceed by the sum of the demands.

- Equations 4.7 and 4.8 are about entrance and exit flows. Firstly, each vehicle can only leave the depot once. Secondly, the number of vehicles leaving every customer k and entering the depot must be equal to the number of vehicles leaving.

- Constraint 4.9 guarantees that each node will be visited at least once by at least one vehicle.

- Equation 4.10 guarantees the decision variable to be positive.

- Equation 4.11 guarantees the decision variables to be binary.

However, there is a lower bound on the number of vehicles, which is the smallest number of vehicles that can carry the total demand of the customers,

$$\frac{\sum_{i\in C} d_i \sum_{j\in N} X_{ij}^v}{K}$$

## 4.2 Computation and Results

Upon studying the operations of Amponsah Effah Pharmaceutical Limited (Kumasi) carefully, the operations of this company is to distribute their medicine after production to their nineteen (19) wholesale points starting from their depot in Adum to different cities with their delivery vehicle.

Since the wholesale points of the company are sited in random cities and the delivery vehicle have to distribute the medicines without passing through a specific route, their operations can be modeled by Vehicle Routing Problem.

A data was collected from Amponsah Effah Pharmaceutical Limited which has been used to create a set of routes on which the company uses to minimize the total distribution distance of the vehicle.

Testing every probability for N wholesale tour would be . This implies that testing 19 wholesale points including their main depot in Adum making it 20 tours, we would have to measure different tours. To calculate the fittest of tours for its 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 wholesale tour in less than a minute.

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

## 4.3 Model

Amponsah Efah Pharmaceuticals Limited has their main depot in Adum where it has all its vehicle located. A delivery van is used for the distributions in the northern sector to the depot. The nineteen (19) specific wholesale points for the northern sector including their depot Amponsah Efah located in different cities are:

| | |
|---|---|
| Action Pharmacy | Fredemens Pharmacy |
| Amponsah Efah | Nyankwa Pharmacy |
| Asempa Pharmacy | Kojach Pharmacy |
| Benita Pharmacy | Kojach Pharmacy Annex |
| Big Maron Pharmacy | Lansa Chemist |
| Concept Medical | Mensaf Chemist |
| Costa Pharmacy | Noble Chemist |
| Danni Herbal | Numens Chemist |
| Evergreen Pharmacy | Oson's Chemist |
| Panacea Pharmacy | Porter Pharmacy |

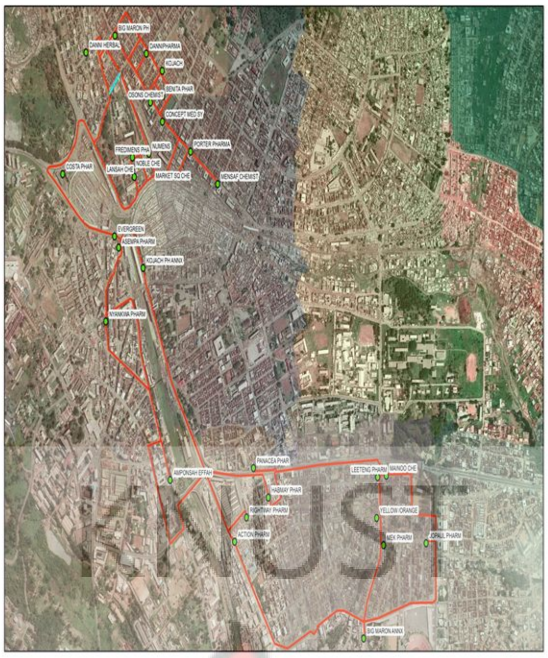


Figure 4.1: Geographical positions of Wholesale points

.

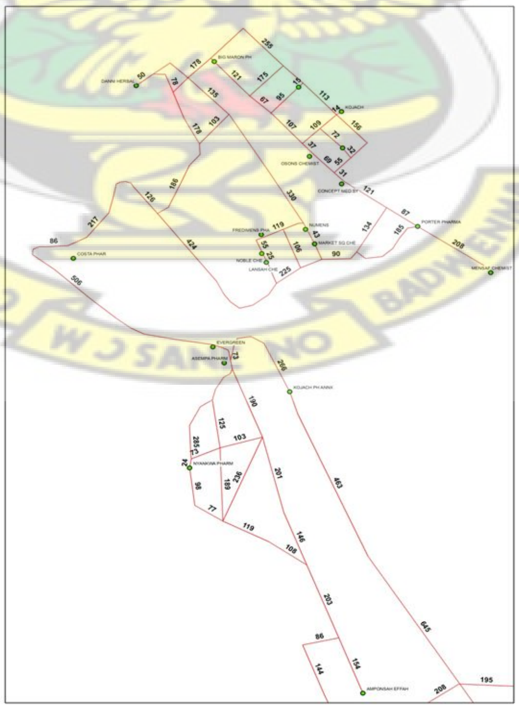


Figure 4.2: Geographical positions of Wholesale points with Distance

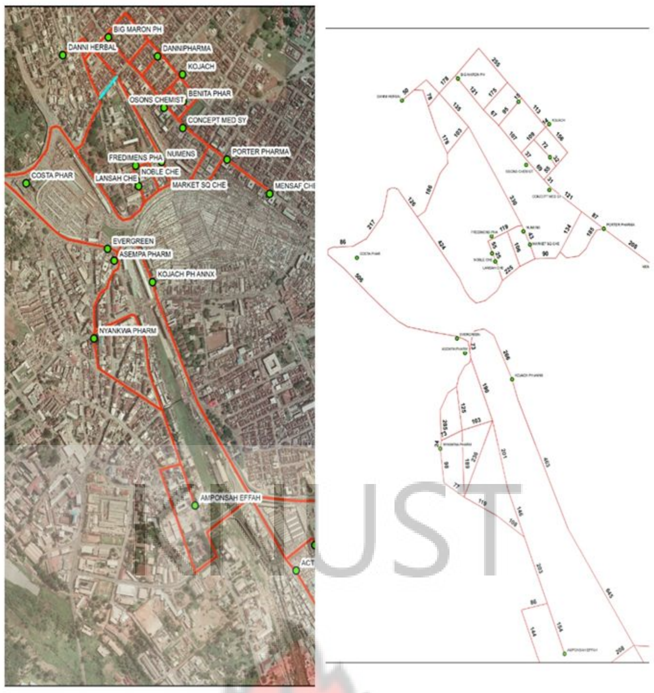


Figure 4.3: Joint representation of Fig. 4.1 and 4.2

The delivery van has to set off from the central depot point.

The schedule of the delivery van and its assigned route as partitioned is as follows:

Amponsah Effah → Action Pharmacy → Panacea Pharmacy → Nyankwah Pharmacy → Asempa Pharmacy → Kojach Pharmacy Annex → Evergreen Pharmacy → Costa Pharmacy → Lansah Chemist → Noble Chemist → Fredemens Pharmacy → Numens Pharmacy → Danna Herbal → Big Maron Pharmacy → Kojach Pharmacy → Benita Pharmacy → Oson's Pharmacy → Concept Medicals → Porter Pharmacy → Mensaf Pharmacy

The total distance travelled by the delivery van from the depot to all the nineteen (19) wholesale points and back to the depot was found to be 11336 meters (11.336km).

## 4.4 Data

In using their local grid points (coordinates) shown in Table 4.1, all the 19 wholesale points and their depot are plotted in a graph below using Matlab.

Table 4.1: Coordinates of Wholesale points

| Wholesale Point Name | $X_m$ | $Y_m$ |
|---|---|---|
| ASEMPA PHARMACY | 652315 | 740442 |
| BENITA PHARMACY | 652597 | 740991 |
| BIG MARON PHARMACY | 652292 | 741221 |
| CONCEPT MEDICAL | 652589 | 740905 |
| COSTA PHARMACY | 651964 | 740712 |
| DANNI HERBAL | 652110 | 741159 |
| EVERGREEN PHARMACY | 652289 | 740483 |
| FREDIMENS PHARMACY | 652402 | 740774 |
| KOJACH PHARMACY | 652589 | 741092 |
| LANSAH CHEMIST | 652413 | 740702 |
| MENSAF CHEMIST | 652936 | 740676 |
| NOBLE CHEMIST | 652403 | 740725 |
| NUMENS PHARMACY | 652505 | 740787 |
| NYANKWA PHARMACY | 652235 | 740171 |
| OSONS CHEMIST | 652514 | 740976 |
| PORTER PHARMACY | 652766 | 740795 |
| AMPONSAH EFFAH | 652639 | 739588 |
| ACTION PHARM | 653042 | 739361 |
| PANACEA PHARMACY | 653161 | 739632 |
| KOJACH PHARMACY ANNEX | 652468 | 740367 |

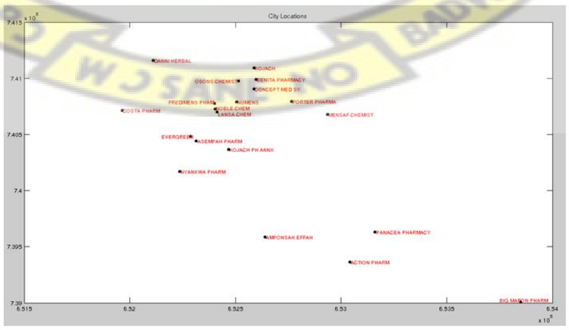


Figure 4.4: Graph Showing The Coordinates of Wholesale points

## 4.5 Encoding

Value encoding is used. Numbers are assigned to all the 20 points as shown below.
Let WP represent wholesale point

WP 1 $\Rightarrow$ Amponsah Efah

WP 2 $\Rightarrow$ Nyankwa Pharmacy

WP 3 $\Rightarrow$ Asempa Pharmacy

WP 4 $\Rightarrow$ Evergreen Pharmacy

WP 5 $\Rightarrow$ Kojach Pharmacy Annex

WP 6 $\Rightarrow$ Costa Pharmacy

WP 7 $\Rightarrow$ Lansah Chemist

WP 8 $\Rightarrow$ Noble Chemist

WP 9 $\Rightarrow$ Fredemens Pharmacy

WP 10 $\Rightarrow$ Numens Pharmacy

WP 11 $\Rightarrow$ Benita Pharmacy

WP 12 $\Rightarrow$ Porter Pharmacy

WP 13 $\Rightarrow$ Mensaf Pharmacy

WP 14 $\Rightarrow$ Concept Medicals

WP 15 $\Rightarrow$ Action Pharmacy

WP 16 $\Rightarrow$ Oson's Pharmacy

WP 17 $\Rightarrow$ Kojach Pharmacy

WP 18 $\Rightarrow$ Panacea Pharmacy

WP 19 $\Rightarrow$ Big Maron Pharmacy

WP 20 $\Rightarrow$ Danni Herbal

The distance matrix is shown below and the corresponding distance square matrix graph is plotted using Matlab as shown in Fig. 4.5

$$\left[\begin{array}{lrrrrrrrrrrrrrrrrrrrr}
 & WP1 & WP2 & WP3 & WP4 & WP5 & WP6 & WP7 & WP8 & WP9 & WP10 & WP11 & WP12 & WP13 & WP14 & WP15 & WP16 & WP17 & WP18 & WP19 & WP20 \\
WP1 & 0 & 759 & 894 & 967 & 1160 & 1400 & 2253 & 2278 & 2333 & 2521 & 2478 & 2753 & 2961 & 2823 & 2941 & 2923 & 3083 & 3211 & 2431 & 2321 \\
WP2 & 759 & 0 & 309 & 382 & 575 & 888 & 1741 & 1766 & 1821 & 2009 & 1966 & 2241 & 2449 & 2311 & 2429 & 2411 & 2571 & 2650 & 1859 & 1809 \\
WP3 & 894 & 309 & 0 & 73 & 266 & 579 & 1432 & 1457 & 1512 & 1700 & 1657 & 1932 & 2140 & 2002 & 2120 & 2102 & 2262 & 2341 & 1559 & 1500 \\
WP4 & 967 & 382 & 73 & 0 & 339 & 592 & 1359 & 1384 & 1439 & 1627 & 1584 & 1859 & 2067 & 1929 & 2047 & 2029 & 2189 & 2268 & 1555 & 1427 \\
WP5 & 1160 & 575 & 266 & 339 & 0 & 845 & 1612 & 1637 & 1692 & 1880 & 1837 & 2112 & 2320 & 2182 & 2300 & 2282 & 2442 & 2521 & 1816 & 1766 \\
WP6 & 1400 & 888 & 579 & 592 & 845 & 0 & 767 & 792 & 847 & 1035 & 992 & 1267 & 1475 & 1337 & 1455 & 1437 & 1597 & 1676 & 885 & 835 \\
WP7 & 2253 & 1741 & 1432 & 1359 & 1612 & 767 & 0 & 25 & 80 & 199 & 242 & 500 & 708 & 570 & 688 & 670 & 830 & 909 & 664 & 714 \\
WP8 & 2278 & 1766 & 1457 & 1384 & 1637 & 792 & 25 & 0 & 55 & 174 & 217 & 492 & 700 & 562 & 680 & 662 & 822 & 901 & 630 & 689 \\
WP9 & 2333 & 1821 & 1512 & 1439 & 1692 & 847 & 80 & 55 & 0 & 119 & 162 & 437 & 645 & 507 & 625 & 607 & 767 & 846 & 584 & 634 \\
WP10 & 2521 & 2009 & 1700 & 1627 & 1880 & 1035 & 199 & 174 & 119 & 0 & 43 & 318 & 526 & 388 & 506 & 488 & 160 & 239 & 465 & 515 \\
WP11 & 2478 & 1966 & 1657 & 1584 & 1837 & 992 & 242 & 217 & 162 & 43 & 0 & 275 & 483 & 345 & 463 & 445 & 605 & 684 & 508 & 558 \\
WP12 & 2753 & 2241 & 1932 & 1859 & 2112 & 1267 & 500 & 492 & 437 & 318 & 275 & 0 & 208 & 208 & 326 & 308 & 468 & 547 & 640 & 896 \\
WP13 & 2961 & 2449 & 2140 & 2067 & 2320 & 1475 & 708 & 700 & 645 & 526 & 483 & 208 & 0 & 416 & 534 & 516 & 676 & 755 & 848 & 1154 \\
WP14 & 2823 & 2311 & 2002 & 1929 & 2182 & 1337 & 570 & 562 & 507 & 388 & 345 & 208 & 416 & 0 & 118 & 100 & 260 & 339 & 432 & 738 \\
WP15 & 2941 & 2429 & 2120 & 2047 & 2300 & 1455 & 688 & 680 & 625 & 506 & 463 & 326 & 534 & 118 & 0 & 156 & 202 & 316 & 556 & 862 \\
WP16 & 2923 & 2411 & 2102 & 2029 & 2282 & 1437 & 670 & 662 & 607 & 488 & 445 & 308 & 516 & 100 & 156 & 0 & 160 & 239 & 332 & 638 \\
WP17 & 3083 & 2571 & 2262 & 2189 & 2442 & 1597 & 830 & 822 & 767 & 160 & 605 & 468 & 676 & 260 & 202 & 160 & 0 & 142 & 382 & 688 \\
WP18 & 3211 & 2650 & 2341 & 2268 & 2521 & 1676 & 909 & 901 & 846 & 239 & 684 & 547 & 755 & 339 & 316 & 239 & 142 & 0 & 270 & 576 \\
WP19 & 2431 & 1859 & 1559 & 1555 & 1816 & 885 & 664 & 630 & 584 & 465 & 508 & 640 & 848 & 432 & 556 & 332 & 382 & 270 & 0 & 306 \\
WP20 & 2321 & 1809 & 1500 & 1427 & 1766 & 835 & 714 & 689 & 634 & 515 & 558 & 896 & 1154 & 738 & 862 & 638 & 688 & 576 & 306 & 0
\end{array}\right]$$

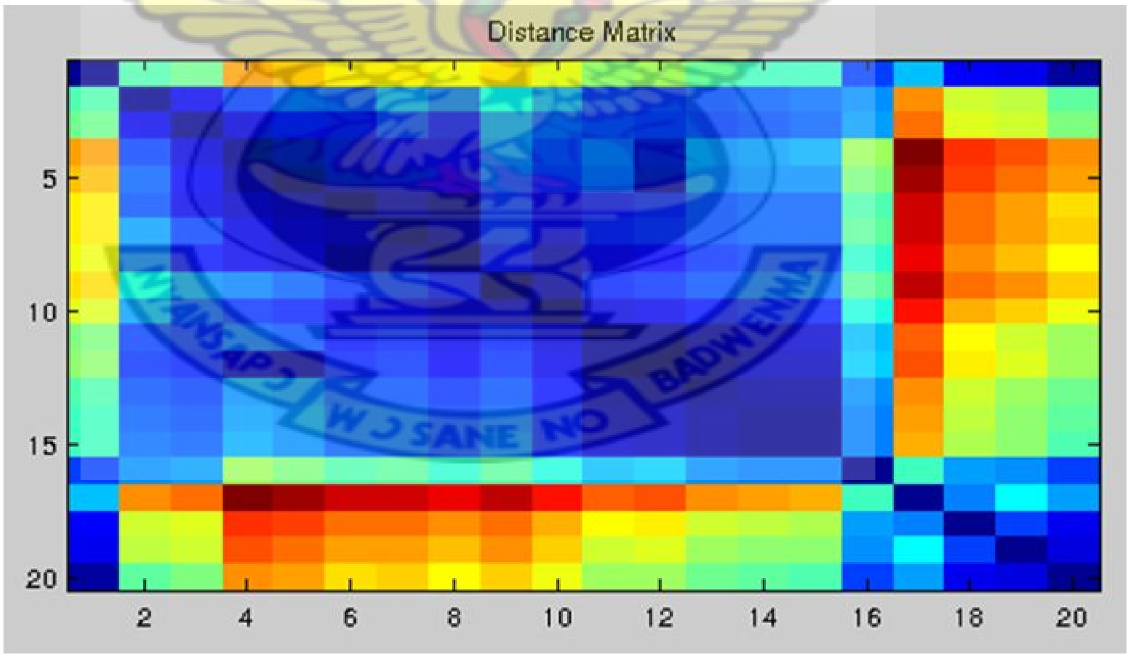


Figure 4.5: Plot of Distance square matrix

## 4.6 Initial Population

A group of many random tours called an initial population is created where a population is a combination of chromosomes. We represent the population as array of

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

chromosomes which represent all the different wholesale locations and the main depot as 20.

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.

## 4.7 Crossover and Mutation

There are two main problems associated with the use of GA to solve VRP. These problems are the choosing of proper methods of crossover and mutation that is used to combine the two parent tours to make the child tours. The two-point crossover is used.

Given a random population of

12 16 6 9 2 15 11 5 18 10 13 14 4 1 3 8 19 17 7 20

This means that we start from the depot 1, the van goes to wholesale point 3 to point 8 to point 19 to point 17 to point 7 to point 20 or from the depot 1, the van goes to in the opposite direction, thus from 1 it goes to wholesale point 4 to point 14 to point 13 to point 10 to point 18 to point 5 to point 11 to point 15 to point 2 to point 9 to point 6 to point 16 and then to point 12.

To reduce bias and the endpoint effect, two-point crossover is used in which two positions in the parents are chosen at random and the segments between them are exchanged.

If our parents with the two random points chosen are

Parent 1 $\Rightarrow$ 18 16 3 12 20 7 6 1 17 $\vdots$ 15 4 2 14 11 $\vdots$ 9 13 10 8 19 5

Parent 2 $\Rightarrow$ 12 13 3 19 7 10 1 11 18 $\vdots$ 16 2 15 6 4 $\vdots$ 8 20 9 11 14 5

Then after the crossover, the offspring produced will be

Offspring 1 $\Rightarrow$ 18 16 3 12 20 7 6 1 17 $\vdots$ 16 2 15 6 4 $\vdots$ 9 13 10 8 19 5

Offspring 2 $\Rightarrow$ 12 13 3 19 7 10 1 11 18 $\vdots$ 15 4 2 14 11 $\vdots$ 8 20 9 11 14 5

The problem of not being trapped in a local optimum could be solved by mutation after crossover. Due to the randomness of the process we will occasionally have chromosomes near a local optimum but not 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 offspring or not. We then choose a point to mutate and switch that point. Like we had

Offspring 1 $\Rightarrow$ 18 16 3 12 20 7 6 1 17 $\vdots$ 16 2 15 6 4 $\vdots$ 9 13 10 8 19 5

Offspring 2 $\Rightarrow$ 12 13 3 19 7 10 1 11 18 $\vdots$ 15 4 2 14 11 $\vdots$ 8 20 9 11 14 5

If we decide to choose the mutation point in offspring 1 to be 3 and 10, and that of offspring 2 to be 7 and 9, then the two offspring would become

Offspring 1 $\Rightarrow$ 18 16 10 12 20 7 6 1 17 $\vdots$ 16 2 15 6 4 $\vdots$ 9 13 3 8 19 5

Offspring 2 $\Rightarrow$ 12 13 3 19 9 10 1 11 18 $\vdots$ 15 4 2 14 11 $\vdots$ 8 20 7 11 14 5

The process makes a strict verification of the chromosome after the mutation process to ignore non legal chromosome.

The product finds a solution to the Vehicle Routing Problem. For this purpose of VRP of finding the minimum total tour, we use cities, chromosomes and populations, where our cities are the wholesale points, chromosomes are the individual tours and the population is the combination of all the individual tours, ie., 20!

Each wholesale point is situated on coordinates $(x, y)$ on the map. In the working process a defined number of wholesale points are being created. Then the program solves the vehicle routing problem foe these wholesale points in different cities.

## 4.8 Fitness Function

To decide if a chromosome (tour) is good and how good it is, is the purpose of the fitness function. The criteria for good chromosome (tour) in VRP is the length of the chromosome. Thus, the longer the chromosome that is coded, the better the chromosome. Calculation takes place during the creation of the chromosome. 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 wholesale points and is the distance between a wholesale point and the depot.

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

1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17 18 19 20 6

and its corresponding graph is shown below in Fig. ??

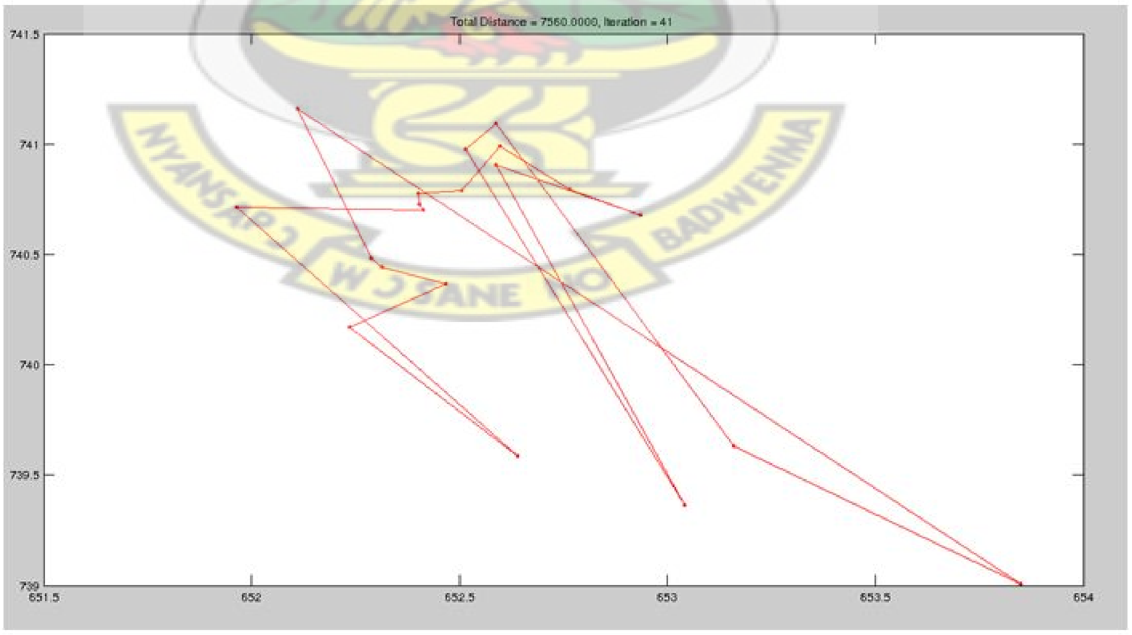


Figure 4.6: Optimal Route Tour

Bearing it in mind that Amponsah Efah Pharmaceuticals Limited uses one delivery van, the optimal route is rearranged for the one vehicle as represented below.

1 2 3 5 4 7 8 9 10 11 12 13 14 15 16 17 18 19 20 6

Rearranged

1 2 3 5 4 7 8 9 10 11 12 13 14 15 16 17 18 19 20 6 1

## 4.9 Optimal Route With Delivery Points

The optimal routes after they have been rearranged is as follows. This is done together with the main depot point as:
Amponsah Effah → Nyankwa Pharmacy → Asempa Pharmacy → Kojach Pharmacy → Evergreen Pharmacy → Kojach Pharmacy Annex → Evergreen Pharmacy → Lansa Pharmacy → Noble Chemist → Fredemens Pharmacy → Numens Pharmacy ⇒ Benita Pharmacy → Porter Pharmacy → Mensaf Pharmacy → Concept Medicals → Action Pharmacy → Oson's Pharmacy → Kojach Pharmacy → Panacea Pharmacy → Big Maron Pharmacy → Danni Herbal → Costa Pharmacy

Total distance for the delivery van (for the northern sector) is calculated to be 7560 metre (7.560km).

The fitness tour was calculated based on the following assumptions being used by Amponsah Efah Pharmaceutical Limited, Kumasi.

- One delivery van is used to make the distributions in the northern sector.
- The van picks up all the wholesale points demand from only one source which is the depot, Amponsah Efah Pharmaceutical Limited, Adum.
- The van is big enough to contain the requested demands of all the wholesale points in a single distribution without shortage for more.

- There are no traffic 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 which is the depot, to any of the wholesale points.
- All the distances from the depot to the wholesale point locations gives the minimum fitness value.

# Chapter 5

# Conclusion and Recommendation

## 5.1 Conclusion

In this work, Genetic algorithm is tested to find the optimal route for the VRP which shows the superiority of Genetic Algorithm over the company's normal route.

It is also proven that if Amponsah Effah Pharmaceutical Limited in retrospective uses this work, they would be able to reduce their operational distance by 3776m (3.7760km) thereby reducing their cost of fuelling their delivery vans which intend reduces 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 tool for combinatorial optimization problems, Genetic Algorithm is very useful for solving problems which can be modeled as the VRP, thereby finding the optimal distance. In light of this capacity of Genetic Algorithm, it is recommended that GA should be used to solve Vehicle Routing Problem (VRP) instead of other traditional heuristic methods.

With regards to our conclusion, we also recommend GA as a tool in seek for an optimal route for their vehicle since it will profit the company in the following areas:

- Increase in profit margin due to reduced cost as a result of reduced distance covered.

- Increase in incentives to achieve staffs satisfactions which will also aid in another increase in profit margin of the company.

# REFERENCES

A. Leon and J. Pozo, *Using a genetic algorithm to study properties of minimum energy states and geometrical frustration in artificial "spin ice" systems*, Journal of Magnetism and Magnetic Materials 320, pp. 210-216, (2008).

A. A. Ghanbari, A. Broumandnia, H. Navidi, A. Ahmadi, *Brain Computer Interface with Genetic Algorithm*, International Journal of Information and Communication Technology Research Vol. 2 No. 1, (2012).

A. Ghosh and S. Dehuri, *Evolutionary algorithms for Multi-Criterion Optimization: A Survey*, International Journal of Computing and Information Sciences, Vol. 2, pp. 35-62, (2004)

Arifovic, J., *Evolutionary algorithms in macroeconomic models.* Macroeconomic Dynamics 4 (5): pp. 373-414, (2000).

Aslaug Soley Bjarnadottir, *Solving the Vehicle Routing Problem with Genetic Algorithms*, Informatics and Mathematical Modelling, IMM Technical University of Denmark, DTU, (2004).

Birchenhall, C., N. Kastrinos, and J. Metcalfe, *Genetic algorithms in evolutionary modeling*, Journal of Evolutionary Economics 7 (4): pp. 375-393, (1997).

Bremermann, H. J., *The evolution of intelligence. The nervous system as a model of its environment*, Technical Report No. 1, Department of Mathematics, University of Washington, Seattle, WA, (1958).

Carlsson, C., and E. Turban, *DSS: Directions for the next decade*. Decision Support Systems 40 (1): pp. 1-8, (2002).

Cassaigne, N., Aversi, R., G. Dosi, G. Fagiolo, M. Meacci, and C. Olivetti, *Demand dynamics with socially evolving preferences.* IIIASA Working Paper, Luxemburg, Austria, (1997).

C. Darwin, *The Origin of Species*, Dent Gordon, London, (1973).

C. Basnet, A. Weintraub, *A Genetic Algorithm for a Bicriteria Supplier Selection Problem*,(2001)

Chia-Feng Juang. *A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithm*, IEEE Transactions on Fuzzy Systems, 10(2), pp. 155-170, (2002)

Chuan-Yin T. and Li-Xin G. *Research on Suspension System Based on Genetic Algorithm and Neural Network Control*, The Open Mechanical Engineering Journal, pp. 72-79, (2009).

Clarke, G., Wright, J.W. *Scheduling of vehicles from a central depot to a number of delivery points*. Operations Research 12, pp. 568-581, (1964).

Cordeau, J.-F., Laporte, G.,Mercier, A.. *An improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows*. Journal of the Operational Research Society 55, pp. 542-546, (2004).

Cordeau, J.-F., Gendreau,M., Hertz, A., Laporte,G., Sormany, J.-S.. *New heuristics for the vehicle routing problem*. In: Langevin, A., Riopel, D. (Eds.), Logistics Systems: Design and Optimization. Springer-Verlag, New York, pp. 279-297, (2005).

D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, (1995).

D. B. Fogel, *Evolving Artificial Intelligence*, Ph.D. Thesis, University of California, San Diego, CA, (1992).

D. E. Goldberg, *Sizing populations for serial and parallel genetic algorithms*. In Schaffer, J. D. (editor) Proceedings of the Third International Conference on Genetic Algorithms. San Mateo, CA: Morgan Kaufmann Publishers Inc. (1989).

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

D. Goldberg and R. Lingle, Alleles, *loci and traveling salesman problem*, In J.J. Grefenstette, editor, Proceedings of International Conference on GAs, Lawrence Erlbaum, (1985).

D. E. Goldberg, B. Korb, K. Deb, *Messy Genetic Algorithms: Motivation, Analysis, and First Results, Complex Systems 3*, pp. 493-530, (1989).

D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison Wesley Longman, Inc., ISBN 0-201-15767-5, (1989).

Dantzig, G.B., Ramser, J.M.. *The truck dispatching problem.* Management Science 6, pp. 81-91, (1959).

Dawid, H. *Adaptive learning by genetic algorithms: Analytical results and application to economic models.* Berlin: Springer, (1999).

Davis, L., *Applying algorithms to epistatic domains,* in: Proc. Int. Joint Conf. on Artifical Intelligence, pp. 162-164, (1985).

Davis, Lawrence (editor), *Genetic Algorithms and Simulated Annealing* London: Pittman, (1987).

Davis, Lawrence, *Handbook of Genetic Algorithms* New York: Van Nostrand Reinhold, (1991).

De Jong K., *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems,* PhD Dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, (1975).

Dosi, G., L. Marengo, A. Bassanini, and M. Valente., *Norms as emergent properties of adaptive learning.* Journal of Evolutionary Economics 9 (1): pp 5-26, (1999).

Dorigo, M., Di Caro, G., Gambardella, L.M. *Ant algorithms for discrete optimization.* Artificial Life 5, pp. 137-172, (1999).

Dueck, G. *New optimization heuristics, the great deluge algorithm and the record-to-record travel.* Technical report, IBM Germany, Heidelberg Scientific Center, (1990).

Dueck, G. *New optimization heuristics: The great deluge algorithm and the record-to-record travel.* Journal of Computational Physics 104, pp 86-92, (1993).

E. Alba and J. M. Troya, *A Survey of Parallel Distributed Genetic Algorithms,* Journal of Information and Operations Management ISSN: 0976?7754 & E-ISSN: 09767762 , Vol. 3, pp 38-42, (2012).

E. Correa, R. Goodacre, *A genetic algorithm-Bayesian network approachfor the analysis of metabolomics and spectroscopic data: application to the rapid identification of Bacillus spores and classification of Bacillus species,* Correa and Goodacre BMC Bioinformatics, (2011).

Fisher, M.L., Jaikumar, R.. *A generalized assignment heuristic for the vehicle routing problem.* Networks 11, pp. 109?124, (1981).

Foster, B.A., Ryan, D.M.. *An integer programming approach to the vehicle scheduling problem.* Operations Research 27, pp. 367-384, (1976).

Forrest, S., *Genetic algorithms: Principles of natural selection applied to computation,* Science 261: pp. 872-878,(1993).

F. Oliveira, M.R. Almeida and S. Hamacher, *Genetic algorithms applied to scheduling and optimization of refinery operations,* (2001).

Frenke, R., *Co-evolution and stable adjustment in the cobweb model,* Journal of Evolutionary Economics 8 (4): pp 383-406, (1998).

Fraser, A. S., *Simulation of genetic systems by automatic digital computers II: Effects of linkage on rates under selection,* Austral. J. Biol. Sci. 10: pp. 492-499, (1957).

Forel, D. B., and Atmar, W., eds., *Proceedings of the second on Evolutionary Programming*. Evolutionary Programming Society, (1993)

Fogel L. J., *Autonomous automata*. Industrial Research 4: pp. 14-19, (1962).

Fogel L. J., Owens A. J. and Walsh M. J., *Artificial Intelligence through Simulated Evolution*, Wiley, New York, (1966).

G. Kannan, P. Sasikumar and K. Devika, *A genetic algorithm approach for solving a closed loop supply chain model: A case battery recycling* Journal of Applied Mathematics, Vol. 34, pp. 655-670, (2010)

Gen, M. and Cheng, R., *Genetic algorithms and engineering optimization*, John Wiley, New York, (2000).

Ghaziri, H.. *Algorithms Connections of Optimisation Combination*. Thesis of doctorate, Ecole Polytechnic Federal of Lausanne, Switzerland, (1993).

Gillett, B.E., Miller, L.R.. *A heuristic algorithm for the vehicle-dispatch problem.* Operations Research 21, pp. 340-349, (1974).

Golden, B.L., Wasil, E.A., Kelly, J.P., Chao, I-M.. *Metaheuristics in vehicle routing.* In: Crainic, T.G., Laporte, G. (Eds.), Fleet Management and Logistics. Kluwer Academic, Boston, pp. 33-56, (1998).

Goldberg, D. E., Deb, K., and Korb, B. *Messy genetic algorithms revisited: Studies in mixed size and scale.* Complex Systems 4. pp. 410-470, (1990)

Goldberg, D. E. *Genetic Algorithms and the minimal deceptive problem.* In L. D. Davis, ed., *Genetic Algorithms and Simulated Annealing.* Morgan Kaufmann, (1987) Goldberg, D. E. *A note on Boltzman tournament selection for genetic algorithms and population-oriented simulated annealing.* Complex Systems 4, pp. 445-460.

Goldberg, D. E. *Genetic Algorithms and Walsh Functions: Part I, A gentle introduction.* Complex Systems 3: pp. 130-155 (1998).

Goldberg, D. E., *Genetic algorithms and Walsh functions: Part II, deception and its analysis,* Complex Syst. 3: pp. 153-171, (1989a).

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

Goldberg, D. E., Sastry, K. and Latoza, T., *On the supply of building blocks*, in: Proc. of the Genetic and Evolutionary Computation Conf., pp. 336-342, (2001).

Goldberg, D. E. and Lingle, R., *Alleles, loci, and the TSP,* in: Proc. 1st Int. Conf. on Genetic Algorithms, pp. 154-159, (1985).

Holland, John H., Escaping brittleness: *The possibilities of general-purpose learning algorithms applied to parallel rule-based systems.* In Michalski, Ryszard S., Carbonell, Jaime G. and Mitchell, Tom M. *Machine Learning: An Artificial Intelligence Approach*, Los Altos, CA: Morgan Kaufman, Vol. II. pp. 593-623, (1986).

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).

H.-P. Schwefel, *?Numerical Optimization of Computer Models?*, Wiley, Chichester, (1981).

H. G. Beyer and H. P. Schwefel, *Evolution strategies: A comprehensive introduction, Natural Computing 1*, pp. 3-52, (2002).

H. P. Schwefel. *Evolutions strategies and Numeric Optimization,* Dr.-Ing. Thesis, Technical University of Berlin, Department of Process Engineering, (1975).

H. R. Kanan, M. H. Moradi, *A Genetic Algorithm based Method for Face Localization and pose Estimation*, 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, (2005)

Hidalgo, J., F. Soltero, D. Bodas-Sagi, and P. Fernandez-Blanco, *Technical market indicators optimization using evolutionary algorithms.* Paper presented at the thirteenth IEEE International conference on evolutionary computation, Atlanta, Georgia, USA, (2008).

Holland J. H., *Outline for a logical theory of adaptive systems.* JACM 9: pp. 297-314, (1962).

I. Kara, T. Bektas, *Integer linear programming formulation of the generalized vehicle routing problem,* in Proc. of the 5-th EURO/INFORMS Joint International Meeting, (2003).

J. Blazewicz, M. Szachniuk and A. Wojtowicz, *Evolutionary algorithm for a reconstruction of NOE paths in NMR spectra of RNA chains,* Bulletin Of The Polish Academy Of Sciences Technical Sciences, Vol. 52, No. 3, pp. 8-33, (2004).

J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, (1992).

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

Jina, N., E. Tsang, and J. Li., *A Constraint-guided method with evolutionary algorithms for economic problems.* Applied Soft Computing 9 (3): pp. 924-935, ( 2009).

J. D. Schaffer and A. Morishima, *An adaptive crossover distribution mechanism* *for genetic algorithms* in Proceeding of the Second International Conference on Genetic Algorithms, pp. 36-40, (1987).

J.F. Goncalves, J.M. Mendes, and M.C.G. Resende. *A hybrid genetic algorithm for the job shop scheduling problem.* European Journal of Operational Research, Vol. 167, pp. 77-95, (2005).

Kinnear, Kenneth E. Jr. (editor), *Advances in Genetic Programming.* Cambridge, MA: MIT Press, (1994).

Koza, J. R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press, (1992).

Koza, J., *Genetic Programming II: Automatic Discovery of Reusable Programs,* The MIT Press,USA, (1994).

Krohling R. A. and Rey J. P. *Design of optimal disturbance rejection PID controllers using genetic algorithm,* IEEE Transactions on Evolutionary Computation, 5(1), pp. 78-82, (2001).

L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, New York, (1966).

Laporte, G., Semet, F. *Classical heuristics for the capacitated VRP.* In: Toth, P., Vigo, D. (Eds.), *The Vehicle Routing Problem.* SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, pp. 109-128, (2002).

Lin, S.. *Computer solutions of the travelling salesman problem.* Bell System Technical Journal 44, pp. 2245-2269, (1965).

Li, F., Golden, B.L.,Wasil, E.A.. *Very large-scale vehicle routing: New test problems, algorithms and results.* Computers & Operations Research 32, pp. 1165-1179, (2005).

Muhlenbein, H. and Schlierkamp-Voosen, D., *Predictive models for the breeder genetic algorithm: I. continous parameter optimization,* Evol. Comput. 1, pp. 25-49, (1993).

Mora, M., G. Forgionne, J. Gupta, L. Garrido, F. Cervantes, and O. Gelman, *A strategic review of the intelligent decision-making support systems research: The 1980-2004 period. In Intelligent decision-making support systems: Foundations, applications and challenges*, eds., pp. 441-462. Germany: Springer-Verlag, (2006)

Marczyk, A., *Genetic algorithms and evolutionary computation,* (2004).

M. Younes, M. Rahli and L. A. Koridak, *Economic Power Dispatch Using Evolutionary Algorithm,* Journal of Electrical Engineering, Vol. 57, No. 4, pp. 211-217, (2006).

M. Sedighizadeh, and A. Rezazadeh, *Using Genetic Algorithm for Distributed Generation Allocation to Reduce Losses and Improve Voltage Profile,* World Academy of Science, Engineering and Technology 37, (2008).

Manish C. T, Yan F., and Urmila M. D.,*Optimal Design of Heat Exchangers: A Genetic Algorithm Framework,* Ind. Eng. Chem. Res., 1999, 38 (2), pp. 456-467 (1998).

M. H. Khorshid and H. A. Hassan, *An Economic Optimization-Oriented Decision Support Model Based On Applying Single-Objective Evolutionary Algorithms To Computable General Equilibrium Models,* Working Paper No. 9, pp. 15-39, (2010).

M. H. Khorshid and H. A. Hassan, *An Economic Optimization-Oriented Decision Support Model Based On Applying Single-Objective Evolutionary Algorithms To Computable General Equilibrium Models,* Working Paper No. 9, pp. 4-12, (2010).

M. Hosseinzadeh and E. Roghanian, *An Optimization Model for Reverse Logistics Network under Stochastic Environment Using Genetic Algorithm,* International Journal of Business and Social Science Vol. 3, No. 12, (2012)

M. Kumar, M. Husian, N. Upreti and D. Gupta, *Genetic Algorithm: Reviewand Application,* International Journal of Information Technology and Knowledge Management, Vol. 2, No. 2, pp. 451-454, (2010)

N. Tyagi and R. G. Varshney, *A Model To Study Genetic Algorithm For The Flowshop Scheduling Problem,* Journal of Information and Operations Management, Vol 3, pp-38-42 (2012)

Oliver, J. M., Smith, D. J. and Holland, J. R. C., *A study of permutation crossover operators on the travelling salesman problem*, in: Proc. 2nd Int. Conf. on Genetic Algorithms, pp. 224-230, (1987).

Or, I. *Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking.* PhD thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, (1976).

Osman, I.H.. *Metastrategy simulated annealing and tabu search algorithms for the vehicle routingproblem.* Annals of Operations Research 41, pp. 421-451, (1993).

Osyczka, A. *Evolutionary algorithms for single and multi-criteria design optimization.* Germany: Physica Verlag, (2002).

OReilly, U.M. and Oppacher, F., *'Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing'*, Davidor, Y. ,Schwefel, H.P. and Manner, R. (Ed.), *Parallel Problem Solving in Nature* - PPSN III, Vol.866 of Lecture Notes in Computer Science, Springer-Verlag, Germany, pp. 397-406, (1994).

P.J.B. Hancock, *An empirical comparison of selection methods in evolutionary algorithms.* In T.C. Fogarty (ed.), *Evolutionary Computing*: AISB Workshop, Leeds, UK, April 1994; Selected Papers. Springer-Verlag, Berlin, pp. 80-94, (1994).

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)

R. Kumar, *Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms,* International Journal of Machine Learning and Computing, Vol. 2, No. 4, (2012).

R. A. Fisher. *The Design of Experiments,* Oliver and Boyd, Edinburgh, (1935).

R. Leardi, *Application of genetic algorithm,PLS for feature selection in spectral data sets,* Journal Of Chemometrics; Vol. 14, pp. 643-655, (2000).

Rechenberg I., *Cybernetic solution path of an experimental problem.* Royal Aircraft Establishment, Farnborough p. Library Translation 1122, (1965).

Rechenberg I., *Evolutions Strategy: Optimization technique System and Principle in Biological Evolution.* Dr.Ing. Thesis, Technical University of Berlin, Department of Process Engineering, (1971).

Renaud, J., Boctor, F.F., Laporte, G.. *An improved petal heuristic for the vehicle routing problem.* Journal of the Operational Research Society 47, pp. 329-336, (1996b).

Rochat, Y., Taillard, Ã.D.. *Probabilistic diversification and intensification in local search for vehicle routing.* Journal of Heuristics 1, pp. 147-167, (1995).

Ryan, D.M., Hjorring, C., Glover, F.. *Extensions of the petal method for vehicle routing.* Journal of Operational Research Society 44, pp. 289-296, (1993).

Safarzynska, K., and J. Bergh, *Evolutionary models in economics: A survey of methods and building blocks.* Journal of Evolutionary Economics, (2009)

S. Hartmann, *A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling,* (1998).

S. L. K. Pond, D. Posada, M. B. Gravenor, C. H. Woelk and S. D.W. Frost, *GARD: a genetic algorithm for recombination detection, Bioinformatics Applications Note,* Vol. 22, No. 24 , pp. 3096-3098, (2006).

S. N. Sivanandam, S. N. Deepa, *Introduction to Genetic Algorithms,* ISBN 978-3-540-73189-4 Springer Berlin Heidelberg New York, Springer-Verlag Berlin Heidelberg , pp. 39-71,(2008).

Schwefel H-P., *Kybernetische Evolution als Strategie der exprimentellen Forschung in der Stromungstechnik.* Masters thesis, Technical University of Berlin, Germany, (1965).

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).

Spears, W., *Recombination parameters, in: The Handbook of Evolutionary Computation,* T. Back, D. B. Fogel and Z. Michalewicz, eds, Chapter E1.3, IOP Publishing and Oxford University Press, Philadelphia, PA, pp. E1.3:1-E1.3:13, (1997).

Spears, W. M. and De Jong, K. A., *On the virtues of parameterized uniform crossover,* in: Proc. 4th Int. Conf. on Genetic Algorithms, (1994).

Syswerda, G., *Uniform crossover in genetic algorithms,* in: Proc. 3rd Int. Conf. on Genetic Algorithms, pp. 2-9,(1989).

Tsang, E., P. Lsasi, and D. Quintana. *A special session on evolutionary computation in finance and economics.* The annual congress on evolutionary computation, Norway, (2009).

Timo Eloranta and Erkki Makinen, TimGa: *A genetic algorithm for drawing Undirected Graphs.* Divulgacioncs Mathematics, Vol. 9, pp. 155-171, (2001).

T. Miquelez, E. Bengoetxea, P. Larranaga, *Evolutionary Computation Based On Bayesian Classifiers,* Int. J. Appl. Math. Comput. Sci., Vol. 14, No. 3, pp. 335-349, (2004).

Tank 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).

T. Back and Frank Hoffmeister. *Adaptive search by evolutionary algorithms,* In W. Ebeling, M. Peschel, and W. Weidlich, editors, Models of Selforganization in Complex Systems (MOSES), Akademie Verlag, Berlin, pp. 156-163, (1992).

Tanese, Reiko. *Distributed Genetic Algorithm for Function Optimization.* PhD. dissertation. Department of Electrical Engineering and Computer Science. University of Michigan. (1989).

T. Back and F.Hoffmeister, *Extended Selection Mechanisms in Genetic Algorithms,*ICGA4, pp. 92-99, (1991).

U. M. O'Reilly and F. Oppacher. *The troubling aspects of a building block hypothesis for genetic programming.* Technical Report 94-02-001, 1399 Hyde Park Road Santa Fe, New Mexico 87501-8943,USA, (1992).

U.M. O'Reilly. *An analysis of genetic programming.* Carleton University Ottawa, Ont., Canada, Canada,(1995).

Wren, *A. Computers in Transport Planning and Operation.* Ian Allan, London, (1971).

Wren, A., Holliday, *A. Computer scheduling of vehicles from one or more depots to a number of delivery points.* Operational Research Quarterly 23, pp. 333-344, (1972).

Y. HaCohen-Kerner, E. Malin, I. Chasson, *Summarizing Jewish Law Articles Using Genetic Algorithms.*

## REFERENCES

A, L. and Pozo, J. (2008). Using a genetic algorithm to study properties of minimum energy states and geometrical frustration in artificial ”spin ice” systems. *Journal of Magnetic and Madnetic Materials*, 320:pp. 210–216.