USING GENETIC ALGORITHM
TO
MODEL THE SHORTEST PATH WITHIN TWENTY CITIES

BY

OTAREN FREDRICK ENOMA
(PG/PSC0909858)

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF PHYSICAL SCIENCES
UNIVERSITY OF BENIN, BENIN CITY,
NIGERIA.

APRIL, 2020

USING GENETIC ALGORITHM
TO
MODEL THE SHORTEST PATH WITHIN TWENTY CITIES

BY

OTAREN FREDRICK ENOMA
(PG/PSC0909858)

A RESEARCH WORK SUBMITTED IN THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTERS OF SCIENCE DEGREE IN COMPUTER SCIENCE (M.Sc.) OF UNIVERSITY OF BENIN,
BENIN CITY, NIGERIA.

MARCH, 2021

DECLARATION

I, Otaren Fredrick Enoma with Matriculation Number PG/PSC0909858, hereby declare that this project work entitled "USING GENETIC ALGORITHM TO MODEL THE SHORTEST PATH WITHIN TWENTY CITIES" which is being submitted by me in partial fulfilment of the requirements for the award of the Masters of Science Degree in computer Science is an authentic report of the work carried out during the 2017-2018 academic session. I further undertake that the matter embodied in the report have not been submitted previously by me to any other University or institution for the award of any degree or diploma.

_____ _____

Otaren Fredrick Enoma **Date**
PG/PSC0909858

CERTIFICATION

This Is To Certify That This Project Work Titled "USING GENETIC ALGORITHM TO MODEL THE SHORTEST PATH WITHIN TWENTY CITIES" Was Carried Out By OTAREN Fredrick Enoma With Matriculation Number PG/PSC0909858 In The Department Of Computer Science, Faculty Of Physical Sciences, University Of Benin, Benin City, Nigeria, Under My Supervision.

Professor A.A Imianvan **Date**
Project Supervisor

Professor Frank Amadin **Date**

Head of Department (HOD)

APPROVAL

The research work is hereby approved as it satisfies the academic requirements in respect of seminar work prescribed for the Masters of Science degree in Computer Science of the Department of Computer Science, Faculty of Physical Science, University of Benin.

_____ _____

Dr. F. O. Chete PhD **Date**
Programme Co-ordinator

_____ _____

Professor Frank .I. Amadi **Date**
Head of Department (HOD)

DEDICATION

This research work is dedicated to Almighty God for his infinite love and mercies and to all lovers of trending technologies.

ACKNOWLEDGEMENTS

The starting and finishing of the project were possible by the contribution and dedication of many people. I want to thank God for life and courage to embark on this project work and to see the end of it. More so, the immense contribution of my supervisor professor A. A. Imianvan is highly appreciated.

My appreciation goes to the following lecturers Dr. S.S. Diode (HOD), Prof E.A Onibere, Prof (Mrs.) V.V.N Akwukwuma, Prof (Mrs.) F. Egbokare, Prof G.O. Ekuobase, Prof (Mrs.) A.O. Egwali, Prof. F.I. Amadi, Dr. K.C Ukaoha, Eng. Dr. F.A.U. Imouokhome, Dr. (Mrs.) S. Konyeha, Dr. (Mrs.) V.I. Osunbor, Dr. F.O. Chete (Programme Coordinator), Mr. E.E. Obasohan, Mr. S.O.P Oliomogbe, Mr. N.E.O. Agbonlahor, Mr. E. Nwelih, Mrs. A.R. Osiobaifo, Dr. F.O. Oliha, Dr. E.P. Ebietomere, Dr. Mrs. R.O. Osasere, Mr. E.C. Igodan, Dr. J.C. Obi, Mr. K.O. Otokiti, Mr. I. Enabulele, Mr. E. Obayagbona, Miss. I. Osiosefe, Mr. F. Osagie, Mrs. T. Agenmomen and special thanks to other staff in the department of computer science for their immense support.

I express great appreciation to my beautiful and lovely wife who has been a source of encouragement to me and a very warm welcome to my children, Joshua and Caleb, daisy and Deborah, to my friends, well-wishers and my classmates for their encouragement and support.

TABLE OF CONTENTS

ABSTRACT

In this era, the best problem solving method is needed in all field irrespective of the complexity or simplicity of the problem. Researchers and developers are doing their best to make software's and machines more potent and intelligent. This is the advantage of artificial intelligent in developing solutions to searching algorithms that are potent and optimal. The most potent highly developed investigate method in Artificial Intelligence is the genetic algorithm. Genetic algorithm was developed to get best result to a known difficulty premised on inheritance, collection, crossover, mutation and further method. It has been proven that genetic algorithm is the most potent, impartial optimization method for analyzing a solution with large space.

this research have been able to define what is genetic algorithm, how it differs from other existing traditional search optimization method, review of ten (10) traditional techniques of finding the best route in a given network. Also the design of genetic algorithm, it's implementation on finding the best route within 20 cities (point) which is invariably the travelling salesman problem (TSP), and areas of application of application of genetic algorithms. The best route is invariably the shortest path.

CHAPTER ONE

1.0 INTRODUCTION

Adaptive heuristics investigate algorithm which is built on the development thoughts of usual collection and genetic is known as genetic algorithms. The working principles of Genetic algorithm are design toward imitating procedures in existing system needed for evolution. The principle was based on Charles Darwin law of common (natural) selection known as the survival of the fittest. Therefore random investigations in a précised infinite distance (space), to find a solution to a problem is represented.

It was first introduced in the 1975 by John Holland. Within several fields in the world of engineering, Gas has been extensively studied, experimented and applied. Another way of finding alternative solution to a problem is to use genetic algorithm. Also it consistently performs better than many traditional techniques in most of the problem link. Finding optimum parameters are some of the real world problems which might be difficult for traditional method but can be best be solve by Gas. Gas has been erroneously regarded as a functional optimizer because of its excellent performance in optimization. Genetic algorithm was developed like a calculating resemblance of flexible structures. It was mirror (modeled) freely on the ideology of development through likely selection, using a large number of persons which undergoes collection during the face of operators like recombination (crossover) and mutation. Individuals are evaluated using a fitness function and success in reproduction varies with fitness. Many scientist have done their best to developed living programs, this programs only try to manifest the characteristics and behaviors of a living organism in order to exist as a life. Though it has been suggested that a program would eventually evolved into a real life, the suggestion may look absorb presently but certainly not impossible if at a fast rates technology progress continues. In my opinion, it is worth taking

time out to discuss how Gas is connected to life and watch if such predictions are unrealistic or baseless.

1.2 AIMS AND OBJECTIVES

Nature has inspired several human inventions. Two notable examples are the artificial neural network and genetic algorithm.

Genetic algorithms search for evolution, which starts from a set of initial solutions or hypothesis where successive generations of solutions are generated. The aim is to understand what genetic algorithm is, genetic algorithm origin, various processes of genetic algorithm and its advantage over every traditional search method. Most precisely ,to find the best and shortest route within several points (cities), beginning from a point and visiting each point once and return to the point of start.

1.3 THE PURPOSE OF THE STUDY IS:

- To enable us understand the basic concepts and working principles of genetic algorithms.
- To highlights the relevance of genetic algorithm in soft computing
- To highlights the benefits and challenges of genetic algorithms
- To highlights how genetic algorithms (Gas) differ from other search method (Existing AI) such as the traditional search optimization methods and the edge it has over them.
- To enable us find the best possible and shortest route within twenty (20) cities without having to visit a city twice.

1.4 SCOPE OF THE STUDY

A genetic algorithm (GAs) has been used in science and technology as computational models of as adaptive algorithms and natural evolutionary systems for optimization problems.

Genetic algorithm operates on evolving population of artificial agents or organisms. Each agent is composed of genotype and phenotype. The scope of this study defines what genetic algorithm is, merits and demerits of genetic algorithm. It also covers different existing traditional search method, design and analysis of existing AI (traditional search method), design and analysis of genetic algorithm. Constraints of the traditional search method was identify, evaluation among conventional search method and genetic algorithm, the hedge genetic algorithm have over conventional search method, The implementation of genetic algorithm in finding the shortest and optimal route within twenty (20) cities (points), beginning from a point, transversing each point once and returning back to the starting point.

1.5 SIGNIFICANT OF THE STUDY

It is well know that using conventional AI systems does no generate stable solutions like the generic algorithm. Several has been done by scientist since the inception of evolution, computation in the 1950s and the 1960s to determine the effectiveness of genetic algorithm in solving problems compared to other conventional AI systems. There are several reasons while genetic algorithm is better than and preferred above other traditional search method, some of the reasons are:

- It is more robust (strong), very easy to understand and it practically does not demand the knowledge of mathematics
- Unlike other AI systems, the genetic algorithm is tough and does not break easily even if the input changes slightly or the presence of reasonable noise.

- Also while performing in large state space, multi-modal state space or n-dimensional surface, a genetic algorithm offers significant benefits over many other typical search optimization techniques like linear programming, heuristic, depth-first, breath-first.
- Genetic algorithm are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, the solution one might not otherwise find in a life .Also, it can solve every optimization problem which can be describe with the chromosome encoding, it can also solve problems with multiple solutions.

CHAPTER 2

LITERATURE REVIEW OF GENETIC ALGORITHMS

2.1 INTRODUCTION

In many researched field, the use and application of Artificial Intelligence (AI) approach has increased tremendously in areas such as system manufacturing design and operation scheduling, Noor 2007. It is imperative to assess the basis of artificial intelligence (AI) and genetic algorithms generally. Very importantly while genetic algorithm (Gas) has been applicable in this study work. This chapter discussed genetic algorithm in details and identify the different AI tools.

2.2 ARTIFICIAL INTELLIGENCE (AI):

In 1956 Mc carthy define artificial intelligence (AI) as a discipline in science and engineering that makes clever machine. Also Rich and Knight 1991 explain AI as the learning on using computers to execute task better than the way it is been tackled by people. A detail definition of AI was given by Webster (2003) which define it as an area of computer science that compose of the capacity of a machine to mimic clever human behavior. The branches of AI includes the following, artificial neural network (ANA), expert system (ES), fuzzy logic (fL), hybrid system and genetic algorithms (GA). Gas is be further expatiated below because it was used during the research.

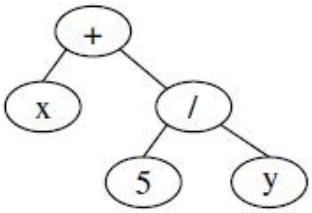
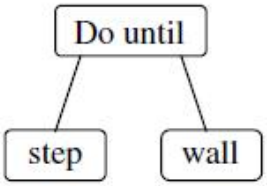
2.3 GENETIC ALGORITHMS (GA)

A methodology that is motivated by Darwin hypothesis of development is genetic algorithms. Similarly it can be said that in Gas, problems are solved by evolutionary process and the best solution (survivor) is the final result. It can also be alleged a solution progress. The exchange of genetic information among different genomes plays a tremendous role in the evolution of

higher living organisms, **Christian Jacob, in illustrating Evolutionary Computation with Mathematica, 2001**. Below is a concise report of the natural development procedure that will aid our indebt understanding of genetic algorithm (Ga). Naturally most existing organisms fundamentally comprises of units (cells), each unit (cell) comprises of a bunch of chromosomes. In turn, every chromosome being DNA string that aid as a representation for all organisms. The collection of genes is known as chromosomes, where a chunk of DNA defines all gene and encodes a specific protein. A quality is encoded by each gene , color of the eyes for instance. Different traits settings could have the possibility of been brown, blue or black. The settings are known as alleles, the particular spot of every genetic material (gene) in a chromosome is recognized as locus. A chromosome is s a complete set of genome comprises of a complete set of chromosomes. A distinct batch of genes specifies a genotype. The genotype is solely liable for the following birth organisms phenotype development, the diverse uniqueness (intellectual and bodily) such as the color of the eyes, intelligent intensity etc. the practice through which latest chromosomes are produced is identified as duplication. Crossover and mutation is very vital in the formation of latest chromosomes and it is the first thing that occurs. Genes from the parent chromosome are rejoin to produce latest chromosomes during crossover. Mutation is another vital operator that occurs all through duplication. Basically, a little amend is incorporated into the essentials of DNA during mutation. These changes were as a result of errors encountered when repetition genes from parents, obitko 1998. The compute of the strength of organisms is called survival; Holland 1975, initially established a style that consists of a series of procedures for Gas, that is pursue to to progress from single generation to another. fresh chromosomes are reproduced by operators like alteration (mutation) and crossover in both generation. The suitability or performance of each chromosome is considered by some strength value. The basis of selection of chromosomes into the subsequently generation is the fitness value. An operator

in which two diverse chromosome recombine their component to form offspring is called crossover. Mutation also is an operator where a gene of chromosome randomly selected is altered. Since we before now know that the choice of chromosome in the subsequently generation is based on the strength value, we can say that a chromosome that is fitter has further chance of getting chosen in the subsequently generation. The selection build on fitness guarantee that the chromosomes that are fittest stay alive throughout generation, where also the lowest fit becomes destroyed. Gas fundamental requirement are a fitness function that will determine chromosome fitness, an encoding standard to encode the result of a problem, determining the diverse measures and constraint for the paramount value, most importantly is the suitable merging of the crossover and alteration (mutation) operators. Gas have the capacity to powerfully execute in the development of an best result, the encoding of a problem solution in its implementation is the major difficulty that is encountered as there can arise an absolute amend in the form of a problem due to improper coding **Obitko 1998 and Negnevitsky 2002**. Encoding is the initial stride in realization of gas, which is the illustration of a difficulty result (chromosome). The problem to be solve is dependently mainly on the encoding. Below are the lists of diverse encoding method specified in table **Noor 2009**. Success has be achieve as a result of been used, Obitko1998.

Table 2.1: Diverse Problem Encoding Techniques Noor 2007

S/No.	Encoding technique	Example
1	Binary encoding	Chromosome 1 :1011001011001010111100101 Chromosome 2 : 1111111000001100000011111
2	Permutation encoding	Chromosome 1: 8 9 7 4 6 2 3 5 1 Chromosome 2 : 2 3 1 4 8 5 6 7 9
3	Value encoding	Chromosome 1: 5.3243 1.2324 2.3293 0.4556 2.4545 Chromosome 2 : HDIERJFDJDLDFABDJEIFLFEGT Chromosome 3 : (right), (back), (left), (back), (forward)
4	Tree encoding	Chromosome A  (+ x (/ 5 y)) Chromosome B  (do until step wall)

Following encoding with the production of a primary population randomly the subsequently move is to select the chromosomes will partake as parents in crossover. Carryout this selection is the major problem. According to Darwin's development hypothesis, the chromosome that is suitable survives during generations which may likely partake during crossover and produce offspring, "several of the famous practice of selection are tournament selection, roulette wheel selection, steady state selection and rank selection, **Noor 2007**. Following that chromosomes have been selected for crossover how to accomplish crossover process so with the aim of genes from two parents will be recombine to create children, chance selection and then replace of genes is the most common way of doing this, as shown below.

('|' is the crossover point):

Chromosome A=10011 | 10101110111

Chromosome B=11001 | 01010011010

Child A=10011 | 01010011010

Child B=11001 | 10101110111

There are several ways of carrying out crossover. The type of encoding being used is solely dependent on the type of crossover that is chosen, and as such it can be quite complicated sometimes. The performance of Gas can definitely improve by a appropriate collection of the type of Crossover for a certain problem. After solving the subject of crossover, mutation is the subsequently move. The core aim of implementing alteration (mutation) is to stimulate a definite point of variety into population so that Genetic Algorithm (GA) can be barred from being spellbound into a limited best **Obitko1998**.

It has be previously point out, that throughout mutation a minor adjust is formed in the genetic makeup of a chromosome. Mutation operator aimlessly change an offspring resulted from crossover. Mutation can be done by binary encoding as shown below,' preferred bit for mutation are revealed in bold' the type of encoding being used define how to perform mutation just like in crossover .

(Bits selected for mutation are shown in bold)

Chromosome A=1100111000010010

Chromosome B=1101101111110110

Mutated chromosome A=1101111000010010

Mutated chromosome B=1101100111110110

Similar to crossover the choice on how to execute mutation, also hang on the kind of encoding that is applied. "Crossover and mutation rates are the two basic parameters of GA" Obitko 1998.

The quantity of period crossover of chromosome is to be in a single generation is determined by the crossover rate. Choice of crossover pace range from zero (0%) to one hundred (100%). Chromosome in the subsequently generation shall be correct replica of chromosomes in the present generation if the crossover rate is 0%. All chromosomes in the population of subsequent generation shall determine the outcome of crossover amid any two chromosome of the present generation if the crossover rate is 100%. It is assumed that the purpose of crossover is that offspring formed all through the development would contain better part of their parents and execute better when liken to them. It is design that all through collection, a little component of the population in the present generation do get elected in the subsequently

generation. The number of genes within a population in one generation that will be mutated is known as mutation rate.

As seen earlier, an operator that produce a definite stage of multiplicity in a population is called mutation, and therefore genetic algorithm (GA) is prohibited from being locked into limited best.

Similarly “selection of alteration (mutation) rate is a fragile decision, **Noor 2007** Gas will be converted into a kind of random search by a high mutation rate , as such the characteristics of evolution is lost. There is the tendency of Gas converging into a local optimum if the mutation is very low. Population size is one important additional parameters of gas, the entire figure of chromosome in a populace in one generation is known as population size. Population size provides gas the searching region where the best result search is carried out, this make population size important, there is the tendency of gas being trapped on a local optimum if it is allotted a lesser searching region which could restrict gas penetrating capacity, this will occur if there are very few chromosome in a population. If the chromosome is many, it means gas is given a searching region that is large, this will certainly slow it downward by escalating its calculation efforts, another challenge is selecting a reasonable population size. According to **Obitko1998** using an extremely huge population size will not resolve the problem promptly as liken to normal size population.

2.4 MERITS AND DEMERITS OF GAS

Some of the attractive benefits and challenges of genetic algorithm at the end of this discussion are as follows

Merits:

1. All optimization which can be explained with the chromosome encoding can be solved.
2. Problems amid numerous solutions can be solved.
3. Multi dimensional, non differential, non continous and non parametrical problems can be resolve since genetic execution method does not rely on the error surface.
4. It is very simple to comprehend the scheme of gas and it does not need the information of mathematics.
5. It is extremely simple to transfer genetic algorithm to accessible simulations and representation.

Disadvantages:

1. Some optimization dilemma known as deviation problems won't be resolve by process of genetic algorithms. This arises due to badly recognized fitness functions which generate awful chromosome chunk in spite of the fact that only good chromosome blocks cross-over.
2. That a Genetic Algorithm (GA) will locate a common most favorable is not sure, repeatedly it happens when the populations have a group of focus.
3. Compare to further artificial intelligence (AI) practice, the genetic algorithm (GA) cannot promise consistent optimization answer times. still yet, the distinction amid the lesser and the fastest optimization reply time is a lot superior than with usual gradient

scheme. This ill-fated genetic algorithm quality limits the genetic algorithms' use in actual time relevance.

4. The usage of Genetic Algorithm (Gas) in power which are carry out in valid time are restricted due to casual results and convergence, in addition this implies that the whole population is developing, for this cannot be alleged for an individual inside this population. Consequently, it is awkward to employ Genetic Algorithms (GA) for on-line controls in real systems with no examining them foremost on a simulation replica.

ANALYSIS OF EXISTING AI (CONVENTIONAL SEARCH METHOD) FOR DETERMINIG THE SHORTEST PATH.

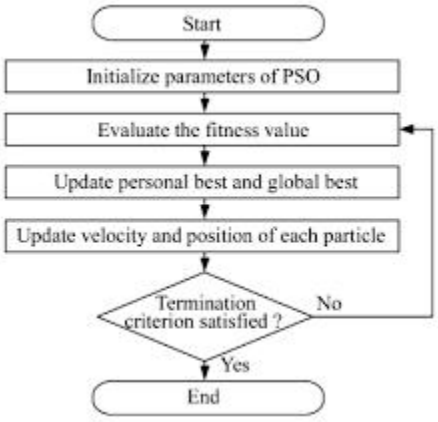
The target of many researched effort over the years has being to uncover the shortest path over a network. Different algorithm and a good sum of empirical conclusion with admiration to feat have resulted as a result of research effort, unfortunately when face with the dilemma of evaluating shortest path on valid road network, preceding study does not provides an apparent track for picking an algorithm. Most of the direct (shortest) path algorithms computational testing has been build on aimlessly produced network that possibly will not have the quality of authentic road network. **F. Benjamin Zhan with Charles E. Noon 1998.** The direct (shortest) route from single spot (node) to a different spot that is recognized requires the efficient management of network. It is very important to be ready to find alternatives route throughout the network in case there is several part of the direct (shortest) path that is damage or active. In this work, ten (10) shortest path algorithms were review, among the ten (10) algorithms reviewed; genetic algorithm was used for the implementation of finding the shortest path within twenty (20) cities.

PARTICLE SWARM OPTIMIZATION (PSO)

It is very hard to find a route from a graph, several algorithms exist plus being accessible to resolve this difficulty. at hand can be several part from a distinct source to a certain target in a graph, however to find an optimum path of minimum cost is very difficult. Swarm intelligence comes from the word Artificial Intelligence (AI) where the activities of ants , insets or birds e.t.c are analyze, we attempt to use this activities in computer discipline after analyzing it to explain diverse optimization issues. Optimization techniques introduce by **Kenedy with Eberhart in 1995** and stimulated by common activities of fish school or birds flocking. at hand are group of element identify as swarm in Pso that travel in search region so as to locate the direct (shortest) path from basis to target. All particle has a recent spot (the

spot of particle continuation in present time) , private best spot (the spot each particle pass through at present), universal spot (best spot of several particle within every particles in seek region), velocity (each particle individual velocity). Every element (particle) tends to shift in the direction of the particle which is nearer to the solution, the entire particle end to a certain solution which is best result after time. In PSO , in the search region which is identify as (particles) each sing solution is a bird, the entire particles have strength value which the strength task evaluates to be enhanced, it also include velocity that control the flying of the particle. class of irregular particles (solutions) initialized PSO and afterward search for the optima by generating renew. every one of the particle is restructured by the subsequent two best values at each iterations. The best result (fitness) it has attained so far is the first one, it is called Pbest. The finest value attain thus far by every particles in the population is another finest value that is follow by the particle swarm optimizer, universal best is the finest value and it is called Gbest.

When an element (particle) acquire fraction of the population as its ground neighbors, the greatest (best) value is a limited best and is called *lbest*.



Flowchart of the PSO algorithm.

Good success rate can be achieved in finding the optimum path using PSO based approach. It can also discover quicker sub-optimal pathway with tall conviction for the entire verified

network. It has been experimented that this algorithm outperformed those of only just reported genetic algorithm build approach for this process.

DIJKSTRA'S ALGORITHM perhaps the most prominent algorithms in computer discipline. In 1956 the Dutch computer researcher **Edsger Dijkstra conceived it and released it in 1959** to uncover the direct (shortest) path in a graph whose ends (edges) are all real numbers (non negative) values. This algorithm is still being used after fifty years in sector such as link state routing. Other scientists have work on it to build further advance path discovery algorithms specifically A*. The algorithm discover the pathway with lowest rate for a specified source vertex (node) in the diagram (graph) that is the direct {shortest} path involving that vertex and all other vertex. Cost of direct (shortest) path from a single height (vertex) to a single target height (vertex) can also be found by ending the algorithm once the direct (shortes)t path to the target vertex have been determined. If the vertex of a diagram (graph) symbolize cities and the ends (edge) path cost symbolize driving distance amid duo of cities joined by a straight road, for example; dijkstra algorithms can be apply to locate the direct (shortest) route amid a single city and several other city. Let the **origin node** be the node at the commencement of the path. Let the space (**distance) of node** Y be the distance from the **origin node** to Y. Dijkstra's algorithm will allot some primary space (distance) values and will try to advance them gradually.

1. A space (distance) value is allotted to every node: position it to zero for the starting point origin) node and to infinity for every additional nodes.

2. the entire nodes should be marked as isolated. place initial node as existing.

3. Consider every one of the isolated (unvisited) existing node neighbors and evaluate their space (distance) unproven. For instance, if existing node A has space (distance) of 6, with end (edge) linking it with a further node B, has length 2, the space (distance) to B through A will

be 6 + 2 = 8. If this space (distance) is less than the formerly documented space (distance), replace the distance.

4. Mark the entire neighbors of the existing node as visited. A visited node will not be examine again; its space (distance) documented is ultimate and smallest.

5. Stop when the entire nodes have been visited, else position the isolated node with the least distance (from the early node, bearing in mind all nodes in the graph) as the next “current node”.

FLOYD–WARSHALL ALGORITHM in a weighted diagram (graph), an algorithm for detecting direct (shortest) paths by useful (positive) or contrary (negative) end (edge) weights but with no contrary (negative) cycles is known as Floyd warshall algorithm.. The algorithm is furthermore recognized as **Floyd's algorithm**, the **Roy–Warshall algorithm**, the **Roy–Floyd algorithm**, or the **WFI algorithm**. It is a type of dynamic programming, Robert Floyd in 1962 developed the algorithm in its present form. However it’s exactly the similar algorithms formerly released by **Bernard Roy in 1959 with Steven Warshall in 1962 , Kenneth 2003.** .rather than determining a path from a particular establish node to every other nodes or a distinct target node, every direct (shortest) paths, are calculated within a sole loop from every node to all others. The algorithm liken during the diagram (graph) all probable paths amid each pairs of vertices. This is made by steadily enhancing an estimate of the direct (shortest) path amid two vertices pending when the estimation is best possible. a matrix regularly in a two size (dimensional) array is essential to be formed by this algorithms. The matrix will be n*n size provided there are (n) vertices in the network,. Every line (row) in the matrix signify a initial vertex in the diagram (graph), even as every column in the matrix signify an ending point in the diagram (graph). If there is a end (edge) amid a starting point i and ending point j in the diagram (graph), the cost of this end (edge) is located in point (i,j) of the matrix. If there is no end (edge) that is frankly connecting two vertices, an infinite

(an extremely big value) is positioned in the (i,j) point of the matrix to specify that it is not viable to openly move from i to j.

Floyd-Warshall's algorithm has a time complexity of O(n3), which is corresponding to performing Dijkstra's algorithm n times. Though, Floyd is typically quicker than when performing Dijkstra's algorithm for every node.

The lengths (summed weights) of direct (shortest) paths amid all pairs of vertices will be established by a distinct execution of the algorithm. While it does not revisit details of the paths themselves, it is likely to rebuild the paths with easy adjustment to the algorithm. Editions of the algorithm can also be employ for detecting the transitive closure of a relation , or (in connection with the Schulze voting system) widest paths amid every pairs of vertices in a weighted diagram(graph).

JOHNSON'S ALGORITHM

Is a direct (shortest) path algorithm that deals with various kinds of shortest part problems. It is the process of finding scattered (sparse), edge-weighted, directed diagram (graph) amid pairs of their vertices. The every pairs direct (shortest) path problem acquire a diagram (graph) that have vertices and ends (edges), and it generate the direct (shortest) path amid every pairs of vertices in that diagram (graph). Johnson's algorithm is exactly linked to the Floyd-warshall algorithm; notwithstanding Floyd-warshall is mainly capable for crowded (dense) graphs (with numerous edges) and Johnson's algorithms is best active for scattered (sparse) graph (with little edges) this algorithm was entitled after **Donald B Johnson's that first released it in 1977.** To iteratively apply Dijkstra's algorithm to each of the diagram (graph) vertices so that the direct (shortest) path amid any pairs of vertices can be created, is the basic idea of this algorithm. Johnson's algorithm is exciting because two other direct (shortest) path algorithms is employ as subroutines. In order to reweight the input diagram (graph) to get rid of negative ends (edges) and identify negative cycles, it make use of Bellman Ford, it then

uses Dijkstra's algorithm in this new altered graph to evaluate the direct (shortest) path amid pairs of vertices. The set of shortest path in the original graph is the algorithm output. There are three main steps in Johnson's algorithm;

i. i. a vertex that is new is added to the diagram (graph) which is linked by (edges) of zero weight to every additional vertices in the diagram (graph).

ii. Reweighting process that eliminates negative weight edges are experienced by all edges

iii. The vertex that is added from step one is remove and in the graph Dijkstra's algorithm is created on all node.

A SUBGOAL METHOD

A transitional condition of the original result to a difficulty is known as subgoal method. In a road traffic network shortest path search problem, the link or node that is positioned amid the source and target position amid which a direct (shortest) path is to be identify could be the Subgoal. With a strong understanding of Subgoal position, the challenge of detecting the (shortest) path from the source node to the target can be reduce into fragments of two or additional problems. Given that there is one subgoal node for example, the solution to the initial (original) problem can be gotten by resolving two alternate problems; detecting direct (shortest) path from the original node to the Subgoal joint (node) is one, whereas the other one is to locate the direct (shortest) path from the Subgoal joint (node) to the target joint (node) **Bander and White 1991 with Dillenburg and Nelson 1995.** The strength of the subgoal techniques is depended on the figure and locations of the Subgoal joint (nodes). The further normal they are scattered amid the initial (origin) node and end joint(node) when the Subgoal is more; the calculation cutback will be very large in a normal condition. Though the computational competence of the subgoal process is clear, the punishment of this seek out decrease is that the result possibly will not be most favorable, i.e., the path moving down the subgoal joint (node) possibly will not be the authentic direct (shortest) path. In other words

relevance of this process depend on whether or not such significant knowledge is accessible. In a road network routing surroundings, probable subgoals comprise bridges, halfway stops, driver's first choice, etc.

GENETIC ALGORITHMS (GA)

Several methods exist in resolving the direct (shortest) path in a network. The disparity amongst these algorithms is connected with their accuracy and speed.. The connection among the majority of these algorithms is the ability to execute on the vector diagram (graphs). a kind of algorithm that is employ to resolve many problems is the **Genetic algorithm (GA)**. The genetic algorithm (GA) is an optimization and seek method,(**Horowitch and Sahani 1978 with Horst and paradalos, 1995, Haupt and Haupt, 2004)** base on the ideology of natural selections and genetics (**Golberg 1989 and Scrinivas, and Patnaik, 1994**). This process was instituted by John Holland, **Holland 1975** with his colleagues. A genetic algorithm was supposedly established to be a strong seek technique **(Golberg, 1989)**. Genetic algorithm begins with an arbitrary population of compressed (encoded) candidate result known as chromosomes. It develop the population towards an best result Through a crossover process and mutation operator,. To estimate the strength (fitness) of every candidate result in the present population is the first step, and to choose the most fit candidate result to be parents of the subsequent generation of candidate solution. Following been chosen for recombination, parents are crossed (using crossover operator) and mutated (using mutation operator) to produce offspring (**Holland1975 and Whitley 1994**). The most fit individual (parents) and the latest child (offspring) form a fresh population from which the practice is repeated to generate fresh populations (**Hosseinali and Alesheikh 2008 with Godefroid and khurshid 2004**).

A* ALGORITHM is among the numerous seek algorithms that get hold of an input, guess a figure of probable path to yield result. In computer discipline, A* (as A star) is a computer algorithm that is extensively utilize in path discovering and diagram (graph) transversal, the method of mapping capable path amid points called nodes. **In 1964 Nils Nilsson** discovers a heuristic build strategy to boost the swiftness of Dijkstra's algorithm. This algorithm was originally named A1**. In 1967 Bertram Raphael** produce remarkable progress upon this algorithm, but fall short to prove the best. He named this algorithm A2. Afterward, **Peter E. Hart in 1968** establishes a case that proves A2 was best possible when using a reliable heuristic with merely minor amends. His attestation of the algorithm also incorporated a piece that demonstrates that the latest A2 algorithm was the best algorithm likely given the clause. He forename the algorithm in kleene star syntax to be the algorithm that begins with A and consists of all feasible version number or A*. A* algorithm is a diagram (graph/tree) seek algorithm that locate a path from a known primary node to a specified goal node, it apply a "heuristic estimate" h(x) that gives an approximation of the finest route that pass through that node. It visits the nodes in accordance of this heuristic approximate. It accompanies the manner of best first search. A* algorithm has been utilize in quite a lot of network difficulties, for illustration the implementation of the A* algorithm in Euclidean networks has been well acknowledged. **Golden et al. (1978)** experimentally instituted that the A* algorithm increase less than 10% of the nodes that would be stretched by a LS (link state) algorithm. **Sedgewick et al. 1986** establish that the A* finds the shortest path in numerous Euclidean diagram (graphs) with an usual calculation attempt O*(n)*, compared to O*(n* log *n)* necessary by the LS algorithm. Of newly, **Nicosia et al. 2003** establish a universal estimate property of A* which permit an answer being found with a known estimate relation: if the practical (heuristic) function *e(i)* is not itself a lower bound, but there exists $k>1$ such that *e/k* is a lower bound, then the result generated by A* is less than or equal to *k*-times the value of an most favorable

result. The authentic performance of the A* algorithm in a moving network depend principally on the value of the practical (heuristic) function *F(i)* or *e(i,d)* utilize. A* employ a best-first search and locate a least-cost path from a known primary node to single goal node (out of single or additional probable goals). As A* traverses the diagram (graph), it trail a path of the least recognized approach (heuristic) cost, maintaining a sorted priority queue of alternate path segments down the way. Associated with greedy best-first search but is more correct since A* takes into notice the nodes that have previously been traversed. A* figure the least-cost path to the node which makes it a best first search algorithm.

Uses the formula f(x) = g(x) + h(x) where

g(x) is the entire distance from the primary position to the present location.

h(x) is the heuristic function that is used to approximate distance from the current location to the goal state. This function is distinct because it is a mere evaluation rather than an exact value. The more precise the approach the better, faster the goal state is reach and with more precision.

f(x) = g(x) + h(x) this is the present estimate of the direct (shortest) path to the goal.

THE HIERARCHICAL SEARCH STRATEGY is well recognized in the discipline of AI and also recognized as a brooding (abstraction) problem resolving approach. **Plolya 1945** is likely the foremost to discover the seek idea in the perspective of road network. The thought was more look at by **Sacerdoti 1974, Korf (1987)** and scores of other researchers. The crucial thought following the hierarchical seek is that to successfully get a result to difficult problem, the seek process must primarily center on the fundamental attributes of the problem devoid of bearing in mind the lesser level information, and then finalized the facts later. Such practice is related to how a driver actually locates a route amid two positions on a map. usually, the driver foremost seek out the key roads in the vicinity adjacent to the starting

point position and target position, and then locate the entrance roads to the key road from both starting point and target position. Road networks frequently inherit hierarchical topology and can thus be easily transmit into a hierarchical illustration. For instance, transportation system (network) links are explicitly planned to meet definite travel functions. Freeways and key arterials are planned to supply for elongated expanse activities while limited streets and collectors are apply generally for land entrance. Thus, the roadway links could simply be categorized as claimed by their functions, which then can be apply in the search course. **Chou et al. (1978)** anticipated using link length as a key reason and remove the protracted links to form an advanced sub-network and faction the shorter links with their nodes into lesser-level sub-networks. **Liu (1997) and Jagadeesh et al. 2002** suggested combining every one of the roads into two levels base on their feature such as speed limit and number of lanes. Direct (Shortest) path search in a hierarchical network could be executed in diverse ways. **Shapiro et al. 1992 and Car and Frank 1994** anticipated an approach that starts by initially finding two direct (shortest) paths from the starting point and target nodes to the closest entry points to the subsequently higher level and then calculating the shortest path amid these entry points employing a usual LS algorithm. This can be illustrated by using a simple two level hierarchical network. The seek begin from the lesser level network where the starting point node and target node are situated. The shortest path trees are build ahead from the starting point node and backward from the target node on the lower level network. The seek out on the lower level network (i.e., land access links) are enclosed by the links (or nodes) which also dwell in the higher level network. Once the search trees come in contact the higher level network (that is, the next entry nodes are establish), an LS (link state) algorithm is called to discover the direct (shortest) path amid the entry points. This practice possibly will work well for long distance trips, but will castigate short and medium distance trips as the nearest entry points possibly will not be the finest position to link the starting point and target on the higher

level. **Liu (1997) identified** this problem and projected an algorithm that considers every one of the access points to the key roads adjoining the starting point and target. The hierarchical algorithm judge a two-level hierarchy and begins with covering the low-level sub-grid networks where the starting point and target nodes are positioned to the higher level network before calling an common direct (shortest) path search algorithm. Comparable to **Liu (1997) Jagadeesh et al. 2002** also consider a two-level hierarchical network. Their algorithm starts with building an augmented network by merging the higher level network with a set of model links (also called logical links) that connect the starting point and target nodes to the equivalent entry nodes. The cost of every model link is the smallest travel time of the resultant travel path recognized by a short path algorithm. Their algorithm also embrace a network pruning system, related to the branch pruning practice, to further decrease the calculation time. One subject inherited with a hierarchical search algorithm is that it frequently does not permit any shortcuts such as moving from one arterial road to a different by employing a residential road. In a traffic network there frequently exist several kinds of shortcuts and some of them possibly will be inevitable. For instance, it is regularly compulsory to swing amid two parallel freeways by moving along an arterial road which join them. With such condition, the seek procedure talk about will ignore the valid direct (shortest) path (the path which has a shortcut).

hierarchical algorithm are though a function of the quantity of network topology (layers and links), seek rules integrated and trip length.

BELLMAN–FORD ALGORITHM an algorithm which calculate direct (shortest) paths from a sole starting place vertex to every one of the other vertices in a weighted digraph**[Bang-Jensen & Gutin (2000)].** The algorithm was initially anticipated by **Alfonso Shimbel 1955,** but is pretty label after **Richard Bellman and Lester Ford Jr,** who produce it in 1956 and 1958, independently**. Schrijver (2005)**

] the same algorithm was also published in 1957 by **Edward F. Moore** , and for this basis it is occasionally term the **Bellman–Ford–Moore algorithm**.

This algorithm is sluggish than Dijkstra's algorithm for the similar difficulty (problem), but extra flexible, as it is competent of managing diagram (graphs) where some of the ends (edge) weights are negative figures. Bellman-Ford Algorithm is extremely unique for its cleanness and depth. The single-source problem is solve by The Bellman-Ford algorithm in the common case, in which ends (edges) can have negative weights and the diagram (graph) is directed. If the diagram (graph) is undirected, it will have to be modified by adding two ends (edges) in every direction to make it directed **(Chumbley A. https://brilliant.org/wiki/)**. Bellman-Ford has the characteristics of detecting negative weight cycles accessible from the starting place, which denote that no direct (shortest) path exists. If a negative weight set (cycle) exist, a path could move considerably on that set (cycle), reducing the path cost too. When there is no negative weight set (cycle), then Bellman-Ford will return the weight of the direct (shortest) path along with the path alone. Similar to Dijkstra's algorithm, Bellman–Ford advance by composure, in which the estimate to the proper distance are substituted by better ones until they finally attain the result. The estimated distance to every vertex in both algorithms is constantly an overvalue of the right space, and is substituted by the slightest of its previous value and the length of a recently establish path. conversely, Dijkstra's algorithm make use of a priority queue to greedily choose the next joint (vertex) that have not been refined, and carry out this calm process on all of its outgoing edges.

This algorithm has the advantages

(i) minimization of cost

(ii) performance Maximization

(iii) traffic splitting is allowed

Disadvantages

(i) In Routing Information Protocol weightings are not put into thought

(ii) sluggish answer to change in the n/w topology purpose are (i) apply in distance-vector routing protocols (ii) network reduction resources (iii) Information Routing

K-Shortest Path

The **k shortest path routing algorithm:** an expansion algorithm of the direct (shortest) path routing algorithm in a known network. It is occasionally important to have additional path amid two nodes in a recognized network. Whilst there are further limitations, other paths diverse from the direct (shortest) path can be calculated. Individual can apply direct (shortest) path algorithms such as Dijkstra's algorithm or Bellman Ford algorithm To discover the direct (shortest) path and broaden them to discover additional path. The k direct (shortest) path routing algorithm is an overview of the direct (shortest) path problem. The algorithm does not only locate the direct (shortest) path, but also k − 1 extra paths in non-decreasing sequence of value. The number of direct (shortest) paths to locate is k. The difficulty can be limited to have the k direct (shortest) path with no loops (loop less k shortest path) or with loop.

There have been several documents produced on the k direct (shortest) path routing algorithm difficulty **ever since 1957**. Some major works not just on locating the sole direct (shortest) path amid a pair of nodes, but pretty citing a series of the k direct (shortest) paths, were executed amid the **1960s till 2001**. Nearly all of the latest study has continually been on the relevance of the algorithm and its variation. A manuscript on figurative calculation of k-

direct (shortest) paths and related dealings with the stochastic practice algebra tool was **produce in 2010 By Michael Gunter et al.** the slightest value route from a precise node to a different known node is the direct (shortest) path along a network, and this path will in general be the preferred route amid those two nodes. it is crucial to establish the next direct (shortest) path if the direct (shortest) path amid two nodes is not accessible for some cause,. a third path may be required If this too is not obtainable,. The sequence of paths thus derived are jointly known as the k-shortest paths (KSP) and it signify the first, second, third,…, kth paths classically of slightest length from single node to a different. This algorithm is therefore broadly apply in the discipline of telecommunications, operation research, computer discipline and transportation discipline. **W. Brander with M. C. Sinclair made** a relative study of the k-shortest path algorithms. We represent the network as a diagram (graph) G = (V, E) where the fixed (finite) set of n nodes is V or joint (vertices) V(G) and E is a fixed (finite) set of m ends (edges) (i.e. links or arcs) E(G) that bond the nodes. The work presented was motivated by the yearning to discover a swift algorithm to calculate the KSPs amid nodes in a network. The two unique algorithms: Yen and Lawler

K-shortest path algorithm is a simplification of the direct (shortest) path algorithm. K-shortest path is utilize in a range of areas like sequence alignment problem in molecular bioinformatics, robot motion planning, path seeking in gene network where rate to analyze paths play a crucial task.

CONSTRAINTS OF EXISTING AI (TRADITIONAL SEARCH METHOD)

Genetic algorithm (GA) is one of the mainly all-encompassing and superior developed heuristic seek procedure in artificial intelligence (AI). Genetic algorithm (GA) is build to locate the best optimized solution for a certain problem premised on inheritance, mutation, selection and some other techniques.

SOME OF THE CONSTRAINTS OF CONVENTIONAL SEARCH METHODS WHICH MAKE GENETIC ALGORITHM MORE PREFERABLE ARE AS FOLLOW

1. The conventional method work with sole point, while Genetic Algorithm (GA) works amid a population of points.
2. The conventional scheme works with the solution themselves while genetic algorithm work with coding of solution set.
3. Conventional search method use deterministic system while genetic algorithm employ probabilistic transition set of laws, etc.

2.9 SUMMARY

Genetic algorithm (GA) is the shortest idea of the practice of ordinary reproduction which comprises the procedures of crossover, alteration (mutation) and selection of chromosomes from single generation to a different one. The initial difficulty that needs to be attempt is illustration and encoding, following the solving of a problem with the aid of genetic algorithm (GA). Since a number of encoding practice is real, therefore an appropriate method must be elected. Subsequently is to choose proper crossover and alteration (mutation) procedure which should be in agreement to the encoding practice, previously preferred. Developing a suitable fitness function which would calculate the strength of all he chromosomes in a population is another important aspect. These fitness values are dependable for the choice or refusal of a chromosome in the subsequently generation as the best (fittest) chromosomes have superior likelihood of survival into the subsequently

generation as relate to their weaker equal. The key genetic algorithm (GA) constraints are the mutation and crossover rates, number of generations and population size. These constraints are fundamental because they decide the superiority of the solution.

CHAPTER THREE

3.0 ANALYSIS OF THE PROPOSED AI (GENETIC ALGORITHMS)

A breed of calculation models encourage by development is known as Genetic algorithm (GA) .a likely result to a particular problem is encoded by this algorithm in a data like form chromosome and apply the operators if recombination to this structures to the extent to protect important details. The scopes of difficulty to which Gas have being tried are entirely broad. Natural selection and ordinary genetics is the search mechanism Gas is actually based on. As a result of the robustness of Gas , it is widely used to resolve optimization problems. In the past Gas has been used to resolve optimization problems in industrial engineering. There are five major elements that makes up a Gas.

I representing the solution genetically

II mechanism well defined for initial population generation.

iii appropriate function to estimate the worth of the solution.

Iv operators alike (crossover and mutation)

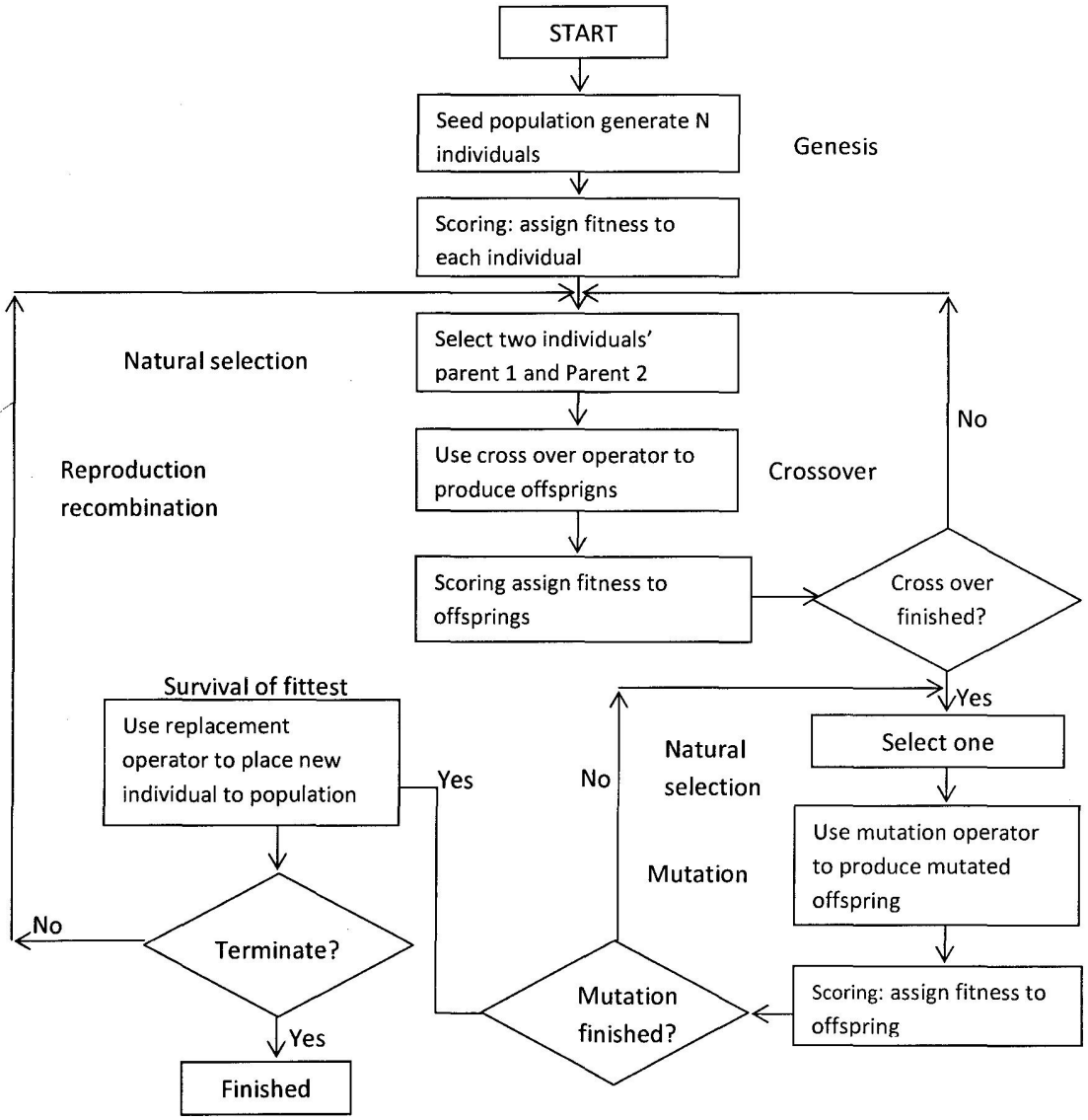
Gas begins its implementation accompanied by a population of chromosomes randomly. The well-known composition are evaluated and the allocated reproductive chance in a way that the chromosomes has a preferable representation of solution to the chosen problem and are allocated more chance to generate chromosomes with a inferior solutions. We identify the integrity of a solution with admiration to the recent population.

3.1. COMPARISON OF GAS WITH TRADITIONAL METHODS

The disparity between genetic algorithm (Gas) and further traditional explore method will be discuss;

Function value that is coded defines Gas instead of the actual value themselves. Suppose we are to realize the highest of a function F(X1, X2) with two (2) unknown the values x1, x2 will not be dealt with directly by the Gas, but with strings in which the values are encoded. We can use strings that represent binary values. A set of population of points are used by Gas to carry out a search , not on the problem space with just a single point. Noises spaces filled with local optimum points can be search by gas due to its strength. Genetic algorithms (GA) glance at diverse regions of the problem space one time and used the complete information to guide rather than depending on a distinct point to explore the point. Payoff information is used by the Gas to direct themselves during the problem scope. Several techniques used for searching also needs information to guide themselves . derivatives are required by hill climbing for example . some degree of fitness of a ppoint in the space is the only information a genetic algorithms needs. This measure of fitness is sometime called purpose function rate. Once the recent degree of a integrity (goodness) is known by the Gas , searching for the optimum can continue using this information. We assume that Gas is not deterministic but probabilistic in nature. It is as a result of techniques randomly used by the Gas. Gas has a major feature of been parallel, it was estimated by Holland that Gas process n strings at every generation, in certainty Gas process n^3 valuable substrings.

FLOW CHART FOR GENETIC PROGRAMMING



RC chakraborty (2010)

Fig. genetic algorithm program flowchart

3.2 GENETIC ALGORITHM DESIGN

Several choices have to be made when designing a Gas for a given application. The nature of the problem will define the type of encoding to be used. Representation of non bit string is very common and it includes forms of integer or floating point values. When the set of the alleles expands in a situation where the floating point number is the string, chromosomes set becomes greater. Recent Gas uses diverse representation approach to make sure that possible chromosome is a resemblance of the set of possible solutions of the problems. There is a choice to make after defining after defining your type of encoding method to be used, this include fitness function form, size of population, operators such as crossover and mutation and their rates. Types of evolutionary schemes to be used, and real condition for stopping or restarting. Combination of problem specific modeling, experience and using different evolution scheme and parameters are the usual design approach. Below is the usual design of a Gas using complete replacement and genetic operator.

1. Start; population of n chromosome are randomly generated (solution suitable for the problem).
2. New population; latest population are formed by continuing the above steps until ther is completion of the new population.
3. a. selection: from a population, two parent chromosome are preferred found on their strength, (when strength is better , ability of been preferred is high).

   b. crossover: the parents are crossed to generate latest offspring using a crossover probability. Offspring is the precise duplicate of parents if there was no crossover executed.

   c. mutation: new offspring at each locus are mutated using a mutation possibility.

   d. Accepting: offspring are deposited inside the latest population.

4. Test; when the end clause is fulfilled, in the population currently let the best solution be returned as you stop.

5. loop; means we can go to step two (2).

3.4 DESIGN METHODOLOGY OF GENETIC ALGORITHM

Like a calculation (computational) intelligence process, Gas is a seek techniques use to locate fairly accurate solutions to combinatorial optimization difficulties in computer science. Gas as optimization practice is premised on Darwin law of natural development (evolution) which is the survival of the fittest. To assume a solution foremost along with merging the fit solution to produce a fresh production of solutions that should be superior to the past's generations is the basic idea of Gas. Mutation of random element is included to report for the intermittent failure. The procedure of Gas is as follow;

Encoding : performance in Gas that are problem definite that transfer the result of a problem into chromosome is known as encoding. The different techniques used for encoding in gas are binary encoding, tree encoding, value encoding, and permutation encoding.

Binary encoding: this kind of encoding is very common, the records value is changed into binary strings, it gives many chromosome possibly that has tiny amount of alleles. We represent a chromosome in binary strings as follows;

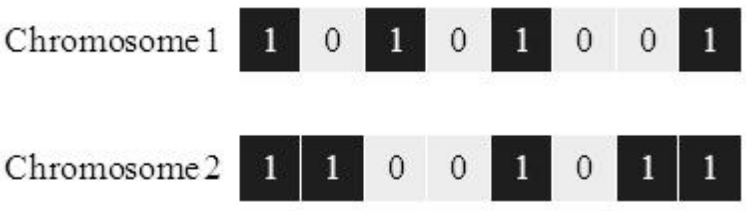
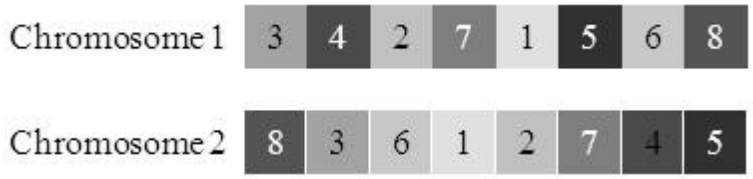


Figure 1.

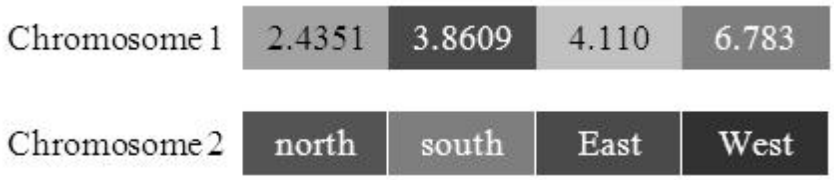
3.4.1 Permutation encoding

This kind of encoding is best suited for queing or ordering mix-up (problem). Permutation encoding is applied in challenging optimization mix-up (problems) like the Travelling Salesman Problem (TSP). Every chromosomes is a string of number in a run in this kind of encoding.



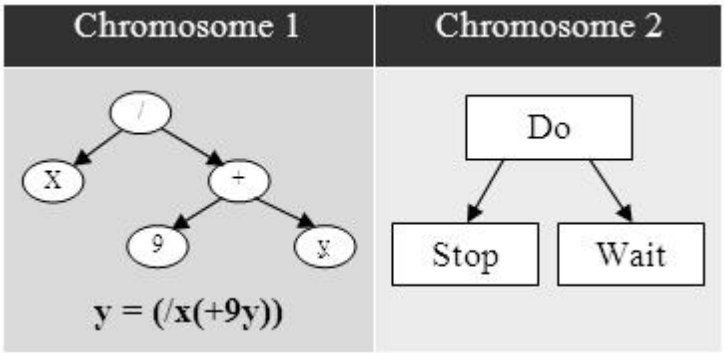
3.4.1.1 Value encoding

Is a technique where every chromosomes is a strings of some value and is used where we required some complicated values. It can be in form of numbers, characters with real numbers to some complicated objects. in Figure 3.



Tree Encoding

Growing terminology or programs such as genetic programming is best used by tree encoding. all chromosomes is a tree of a few objects, function or command in language programming.

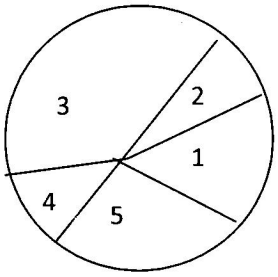


3.4.2 Reproduction (or selection)

In a population that is new, operator that makes additional copies of superior strings is called reproduction. good string are selected in a population to form mating pool by the reproduction in order to preserve the generation of population that is new. Reproduction of individuals is very important in the present population. Common used selection operators are;

3.4.2.1 Roulette-wheel selection:

Well recognized as proportionate reproduction, it is a Gas that is utilize for choosing helpful likely solution for recombination. The probability of any personality been preferred is relative to its fitness superior or lower than its competitor strength (fitness). Calculation of individual fitness is done as you spin the wheel n^3 times, selecting the string choosen by the pointer in an instance each time. The frequently employ reproduction operator is the proportionate reproduction. It is a genetic algorithm for picking likely functional result for recombination's, the probability of an individual to be elected is relative to its strength bigger or lower than its competitor fitness



Example

Population	Fitness
1	25.0
2	5.0
3	40.0
4	10.0
5	20.0

3.4.2.2 Tournament Selection

Is a techniques used to select an individual in a Gas from a population of individuals. Several tournament are run among few individuals (chromosomes) randomly choosen from the population. In each tournament, the winner (the most fit) is chosen for crossover. Pressure for selection can easily be adjusted by altering the size of tournament. If the size of tournament is large , individuals that are weak have less chance of been chosen.

Example

- known population and fitness
- choose B and D
- B wins
- Probability

Individual	Fitness	Probability
A	5	1/25 = 4%
B	20	9/25=32%
C	11	7/25=28%
D	8	5/25=20%
E	6	3/25=12%

3.4.2.3 Rank selection

In these techniques, likelihood of choice is proportional to relative strength instead of absolute fitness. There is really no different whether the most fit candidates ten (10) times more fit than the next fit or(0.001%). Probability of selection might be same in both cases, the most important thing is the ranking relative to other individuals. Premature convergence is avoided by rank pressure of selection for differentials of large fitness occurs in recent generation

3.4.2.4 Steady state selection

Simple form of Gas where two parents selected are crossed, producing mutated offspring that are inserted into the latest population. The latest population at every generation entirely consists of offsprings that was produced by means of parents in the past generation (while the offspring and parents possibly will look alike). Some scheme like the elitist schemes describe successive generations to some degree, overlap some parts of the past's generation is kept in the latest population. The generation gap is used to describe the fraction of fresh individuals at every generation, (Dejong 1975). Only few individuals are replaced in each generation . in steady state selection , offspring from crossover and mutation of individuals fitness are used to replace a little quantity of the slightest robust individuals. Evolving rule base system such as (system classifier, Holland 1986) are often used by steady state, Gas in which recall what has previously been learnt and incremental learning is important, also problems are solved collectively by members of the population..

3.4.2.5 Elitism

Is a selection method that pressure Genetic algorithm as to maintain good numbers of the finest individuals in every generation. Best individuals are save and the process of selection is improved. Monotonically, quality of best solution increases overtime in each generation with

no elitist selection, best individuals can be possibly lost due to pressure of crossover, mutation and selection.

3.4.2.6 Boltzmann Selection

Is a method that is related to simulated annealing in which temperature that continuously varies controls selection rate according to fixed plan. When the temperature is high it imply that pressure of choice is little (i.e. the chance of every individual reproducing is reasonable.).the pressure of selection gradually increase as we gradually lowered the temperature which allows the Gas to narrow closely to the part of the investigate that is best, while maintaining suitable level of multiplicity. A typical implementation is that each individual be assign an expected value exp val $(i,t) = \frac{e^{f(i)/T}}{\left(e^{f(i)/T}\right)t}$

The temperature is T, and the population time is <> t

3.4.3 Crossover:

An operator that is use to merge two strings to obtain a superior and strong offspring is called crossover operator. It is an operator in gas that mates (combine) two parents (chromosomes) to generate a latest offspring. If the new chromosome takes the finest character from every one of the parents, it possibly will be superior to both of the parents. Throughout development, crossover takes place according to crossover probability that is defined. Genes are collected from parents chromosome to create offspring that is new by crossover. The following are the process of crossover;

3.4.3.1 One-point crossover

It is a crossover operator that is executed by indiscriminately choosing a point of crossing through the string (chromosome) along with exchanging on the right part of the crossing position all bits as shown below.

Chromosome 1 (parent) 01101100 chromosome 2 (offspring) 01111001

Chromosome 2 (parent) 01111001 chromosome 2 (offspring) 01101100

Before crossover after crossover

The symbol, a perpendicular line, 1 is the preferred crossover point

3.4.3.2 Two-point crossover

an operator that pick two crossover spot contained by a chromosome randomly an interchange two (2) parent chromosome, to create two fresh offspring's between these points..

Parent1 1101100100110110 **Parent 2** 11011110000111110

Interchanging the parents chromosomes produce this offspring

Offspring 1 110110010011 0110 **Offspring 2** 110110010011 0110

3.4.3.3 Arithmetic crossover

An operator that merge two parent chromosome vector linearly to generate new offspring of two by means of the equations.

Offspring 1 = a* parent 1+ (1-a) * parent 2

Offspring 2 = (1 parent 1+a* parent2

The weighty random factor is a, chosen before each crossover operation

3.4.3.4 Heuristics crossover operator

In heuristics crossover, the direction of search is determined by the fitness value of the two chromosomes. This equation generate the offsprings

Child (Offspring) 1 = Best parent +r* (Best parent - worst parent)

Child (Offspring) 2 = Best parent

The value of r is a chance number involving 0 and 1

3.4.4 Mutation

Once crossover is carry out, mutation is carried out. Alteration Mutation is a genetic operator that is utilizes to sustain genetic variety from single production of population of chromosomes to the subsequent. Mutation takes place all through development base on a user-definable mutation chance, typically position to moderately little value, say 0.01 a fine initial option. Mutation change one or extra gene values in a chromosome from its early state, this can produce totally fresh gene values being join to the gene pool with the latest gene values, the genetic algorithm may perhaps be able to land at superior solution than was beforehand possible.

Mutation is an significant component of the genetic seek, it aid to avoid the population from idle at any limited optima, mutation is planned stop the seek (search) declining into a confined best of the state region.

The alteration (mutation) operators are of many types

FlipBit, Boundary, Non-uniform, Uniform, and Gaussian, the operators are chosen premised on the mode chromosomes are encoded.

i. **Flip Bit**: This alteration (mutation) operator plainly reverses the value of the elected gene i.e. 0 goes to 1 and 1 goes to 0. This mutation operator can merely be used for binary genes. reflect on the two unique offsprings preferred for mutation

ii. initial offspring 1 1101111000011110

initial offspring 2 1101100100110110

The altercated (mutated) offspring are

Altercated (Mutated) offspring 1 1100111000011110

Altercated (Mutated) offspring 2 1101100100110100

iii. **Boundary:** This alteration (mutation) operator substitute the value of the preferred gene with any of the greater or lesser bound for that gene (selected aimlessly). this alteration (mutation) operator

iv. be capable of use by integer and float genes

v. **Non-uniform**: This alteration (mutation) operator raises the prospect so that the sum of the alteration (mutation) will be near 0 as the generation digit is enlarged. This mutation operator stop the population from stalling during starting phase of the development afterward permit the genetic algorithm to fine tune the result in the afterward phase of development

vi. **Uniform**: in this mutation operator, the same random value preferred amid the users precise upper and lower bound for that gene replaces the value of the chosen gene. This alteration (mutation) operator can merely be apply for integer and float gene.

vii. **Gaussian**: This alteration (mutation) operator insert a unit Gaussian dispersed arbitrary value to the preferred gene. The value of the latest gene is truncated if it falls exterior of the user précised lesser or higher bound for that gene. This operator can merely be employ for integer and float genes.

3.4.5 Genetic Algorithm and Modeling

Modeling the difficulty (problem) of the travelling salesman to get a solution is the major focus of genetic algorithms. Genetic algorithm is commonly and widely known and used to model the problem of travelling salesman to get a solution.

3.4.6 Using genetic algorithm to model the travelling salesman difficulty (problem) (TSP)

Finding a tour of specified amount of cities is the fundamental thought of the travelling salesman problem, visiting exactly once each city and coming back to the starting city when we minimize the tour length. The origin of the travelling sales man problem was from Euler in 1759 whose challenge was to move about to all position on a chessboard just once a knight. The travelling salesman problem first came into limelight in a manuscript that was written by german salesman Bf Voigt in 1832 on ways to flourish as a travelling salesman. He mentioned the travelling salesman though not by the name, but he suggested that to cover several locations as probable devoid of having to visit whichever location twice is the major side of scheduling a tour. The only guaranteed way currently to optimally resolve the travelling salesman problem of whichever size by listing every probable tour and searching for the tour with minimum cost is the Gas.

CHAPTER FOUR

IMPLEMENTATION OF GENETIC ALGORITHM

Optimization method based on normal development is known as genetic algorithms. This involves the survival of the fittest inspiration that is used in a investigate algorithm which decides the method of searching that do not have to to look at each potential result in the viable area to attain a fine result. Natural procedure of evolution is what genetic algorithm is based on. The most fit individuals are expected to survive and mate in nature, it thus means that since the subsequently production were breeds from healthy parents, they ought to be fitter and healthier. This same principle is functional to a difficulty by assuming a solution initially and next combining the fit solution to produce new generations which ought to be better than the past's generation. A random mutation factor is also integrated to account for the numerous misfortunes in nature. Below are steps of genetic algorithms which was discus in chapter three; Encoding, Evaluation, Crossover, Mutation, Decoding. Various problems such as the Np-Hard is solve by this optimization technique. Though this technique cannot give the best solution but it does give fine estimate in a realistic amount of time.

TYPES OF OPTIMIZATION PROBLEMS WHERE GENETIC ALGORITHM IS IMPLEMENTED ARE AS FOLLOWS;

4.1 KNAPSACK PROBLEMS

This problem involve; where we have numbers of objects to be picked and placed into a knapsack. The aim is to optimize object picked via mostly valuable, that compute from sum value of item with utmost weight to be picked. This kind of problems occurs in logistic industry for resourceful transportation of result.

4.2 FACILITY PROBLEMS

The problems include; where there is an important need to place a series of facility in a strategic place. The purpose is toward optimizing areas that comprise least rate for users, set multi-facilities where diverse types of facilities exist to decrease the total rate and most planned area of need so as to exploit market prospect. It is frequently use for building city community facilities or shopping complex.

4.3 BIN PACKING PROBLEM

This problem is comparable to knapsack problem. The aim is to look for the optimization after a group of objects with diverse characters (weight, volume and other dimension) filled into several bin; to maximize the number bins used, genetic algorithms is used. Problem such as information filled into a server has to be likely low, if initial bin of space is not entirely filled yet, it will be able to store data into the left over.

4.4 PRISONER'S DILEMMA

A further important use of genetic algorithms will be the prisoner'sdilemma. The prisoner's dilemma is competitions in which two prisoners are detained in detach cells unable to interact. Both of the prisoners is ask to desert and let down the other and must decide to do so instead of partnering several prisoners. assuming one of the prisoner flaw, he will receive five (5) points why the next prisoner receive nothing. Each of the prisoners will receive one (1) point if they together defect. Both of the players will receive three (3) point if they both cooperate.

4.5 TRAVELING SALESMAN PROBLEM (TSP)

The basic thought behind the Traveling Salesman Problem (TSP) is to uncover a tour of a specified figure of cities, visit every one of the city precisely once each city and coming back to the early city (position) wherever the duration of this tour is reduced. Genetic Algorithms may possibly be applied to resolve this problem. While at hand are number of cities which the salesman needs to pass through with least cost of distance, it is optimized. This kind of problem occurs in areas such as circuit design, routing for transportation and others.

Implementation of genetic algorithm is commonly employed in solving the travelling salesman problem, shall be considered.

4.5.1 Implementation of genetic algorithm (GA) to resolve the Travelling Salesman difficulty (Problem) is discussed.

Genetic algorithm (GA) can be used in much less time to realize a solution. Though it might not realize the finest solution, but it can realize a close excellent solution for 100 city tour in fewer than a minute. Within are essential procedures to solving the TSP using a genetic algorithm (GA) premised on Darwin law of Natural Selection, the algorithm develop through three operators

once an early population is aimlessly generated;

1. Selection that is equivalent to survival of the fittest (best individual);

2. Crossover that involves union amid two individuals

3. Mutation that defines arbitrary (random) modifications.

1. Selection Operator

- Preferable individuals are giving preference, enabling them to move on their genes to the subsequently generation.
- Fitness defines integrity of both individual.
- objective function or subjective judgment determines Fitness
- **2. Crossover Operator**
- factors that differentiate genetic algorithm (GA) from other optimization approach ;
- selection operator is employ to choose Two individuals the population
- it randomly choose A crossover spot next to the bit strings
- The value of the duo (two) strings is swapped

If S1=000000 and S2=111111 and the crossover point is 2 then S1'=110000 and S2=001111

S1	0 0	0 0 0 0
S2	1 1	1 1 1 1

S1’	1 1	0 0 0 0
S2’	0 0	1 1 1 1

- The duo recent offspring that is produced from the union be place into the subsequent production (generation) of population.
- The action is predicted to produce even superior individuals By recombining segment of fine individuals

3. Mutation Operator

. A portion of the new individuals with some low probability will have some of their bits flipped.

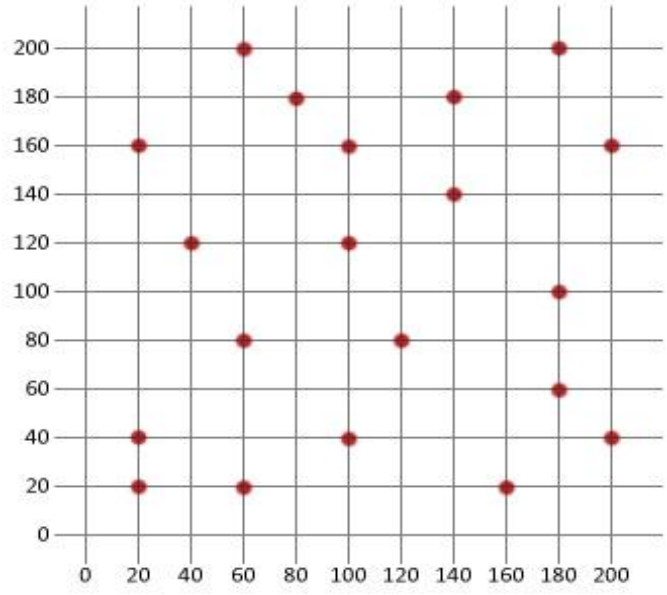
- Maintaining variation inside the population and stall early intersection is it sole purpose..
- an aimless walk in the search region is induces by Mutation alone
- parallel, noise-tolerant, hill-climbing algorithms are created by Mutation and selection (without crossover)

Before	01111110
After	11111110

Mutation operation

Employing genetic algorithm to resolve the traveling salesman difficulty (problem)

Just before obtaining detail knowledge of what the Traveling Salesman Problem (TSP) is, the reason it is a difficult problem, let us examine a practical illustration of the problem. Suppose you are a salesman and you have been given a map as seeing e one below.



In the map above, you can see that there is a sum of twenty (20) points (locations) and you were inform it is your profession to move to once every one of these locations to make a vend (sale) and return to your starting point. The shortest distance in the course of your travel becomes your real route. Before you commence your journey you may need to plan a route in order to reduce your time of travel. Humans are very excellent on this, a reasonably route can effortlessly be work out devoid of needing to perform more than look at the map. Once a route we feel is best is located, how do we investigate if it is truly the best route? It might look impossible

Let's consider a related map with just three (3) locations in lieu of the primary twenty (20) to identify why it is not easy to prove the best possible route. From the three possible locations on the map we foremost have to choose an initial location. We would have an option of two

(2) cities meant for the next location, and afterward there is presently 1 city left to select to conclude our path. It therefore means at hand are 3 x 2 x 1 diverse routes to make a choice from.

There are only six (6) different routes to choose from In this particular example, in this case of presently three (3) locations, it is very easy to determine every one of the six (6) routes to get the direct path. If you have knowledge of mathematics you know what the problem is here. The factorial of the amount of locations to visit is the number of likely routes; problem by means of factorials is that they develop in magnitude quickly!

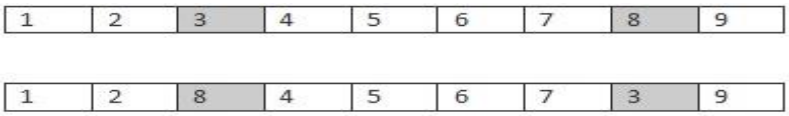
For instance, the factorial of eight (8) is 40320 for, that of nine (9) is 362880, but the factorial of twenty (20) is extremely huge, 2432902008176640000.

Assuming we desire to get the direct route for our initial map of twenty (20) locations, we need to estimate 2432902008176640000 diverse routes! With present computing ability this seems impossible, and even for difficult problems, it is nearly impossible. Genetic algorithm is the best method that can be use to generate optimal solution even in less than a minute in this kind of large problem.

4.6 HOW THE GENETIC ALGORITHM WORKS

It is important we put up A Genetic Algorithm (Gas) in a specific way to find a result to the travelling salesman problem. A route where each position is incorporated at least only on one occasion needs to be represented by a good solution. it will be invalid and waste of computational time calculating distance of a route containing a particular location more than on one occasion before completely miss a position. Special type of crossover and mutation techniques are desired to guarantee the genetic algorithms meets such condition. The mutation techniques must have the ability of shuffling the route, it must never include or take

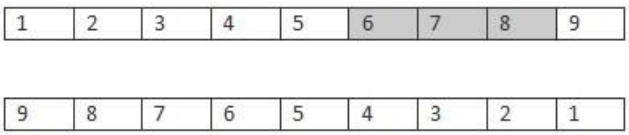
away a location from the route else it will generate an incorrect solutions. Swapped mutation is one unique form of mutation we possibly will use in this case. Two positions in the route are chosen at random with swap mutation and their points are merely exchange. if we relate swap mutation to the above list, [1,2,3,4,5] For example, we may finish up with [1,2,5,4,3], positions 3 and 5 were reverse creating a fresh list with same values, in a diverse form.



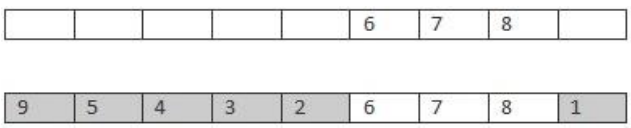
Pre-existing values are only swapped by using swapped mutation, a list of missing or duplicate values can never be created when compared with the initial, and this is accurately what we want for the Traveling Salesman Problem (TSP).

There is need to decide a crossover technique that can enforce same control. Ordered crossover is one crossover technique that is able to produce a suitable path. We choose a subset from the initial parent and add that subset to the offspring in this crossover method; any value that is missing is then added to the offspring from the next parents so as to to find them. Let's consider the above examples,.

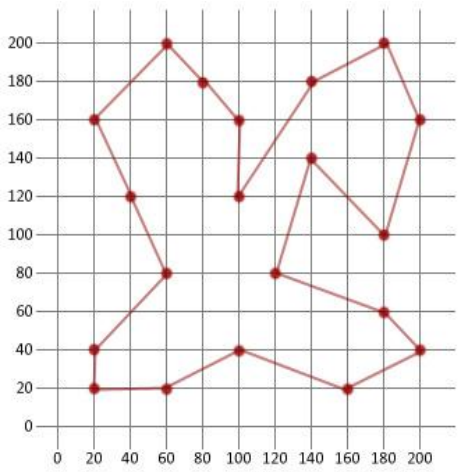
Parents



Offspring



A breaking up of a route is taking from the first parent (678) and attached to the offspring route. The misplaced route location is added from the second parent orderly. Nine (9) is the first location in the next parent path that is not within the offspring path, it is therefore included in the first available position, eight (8) is the subsequent point in the parent path which is within the offspring path, it is therefore omitted. This development will continue pending when there are no more unfilled values in the offspring. When correctly implemented, a route that contains every of the positions with no position lost or duplicated. The diagram below indicates the best route as the final result when genetic algorithms are implemented.



4.7 APPLICATIONS OF GENETIC ALGORITHM

The Genetic algorithm elasticity has led to control of this field in various applications over diverse industries. Below are some of the areas of applications

4.7.1 Applications of Evolvable Hardware

Genetic algorithms are manipulated to produce electronic in this field. Stochastic operators are exploited by genetic algorithms models to promptly obtain latest sketch build on the past

designs. The useful design which is compulsory by the inventor will be achieve as the model keeps developing as it keep running in its surroundings factors. A robot with ability of controlling fixed Gas to regenerate its design after some faults due to surroundings challenges such as electromagnetic waves which can cause malfunction in its normal design is an example of such a reconfigurable model. However the design of the robot will be alter by the robot to a latest report if it encounters a situation where it needs more functionality to execute its task.

4.7.2 Robotics

Robotics prompted inventors and engineers to test and figure out the vital aspect of a machine (robot) such as hardware component and software planning to develop an efficient and more comprehensive robot. To sketch a new robot that suits the new objectives, all the mentioned activities are required to be performing again. Many of these extra sketch requirements can be deleted by manipulating genetic algorithms. The ability to promptly initiate a collection optimal sketch which can be used for specific task and activities is provided by genetic algorithms. This move will result in the initiation of robots that can execute several tasks and processed more complete applications. Direction-finding is another aspect of using gas in robotics; optimize solutions for direction-finding is provided by Gas by which the robot can reach to its vital destination without being missing or striking another object in the location.

4.7.3 Engineering design

The design of latest engineering copy is compound and the process is time absorbing, but to design a best copy that uses least measures to deliver the highest output is more complex. A great attempt and experience is required to perfectly complete the task. The functionality of Gas is very relevant here. Computer base engineering design applications is integrated by Gas. This application is able to examine various aspect of engineering design concept when

initiating a new sketch for a given problem by following such strategy. This approach not only provides the vital sketch but also support the required inventor to know the weakness and the probable failure points of the sketch. In many engineering industries, such approach is currently being used in areas like aerospace, civil, automotive, robotics, electronics etc.

4.7.4 Data Encryption

Genetic Algorithm is being used in the field of cryptology to produce a latest advanced encryption via the Gas operators of Crossover and alteration (Mutation). A collection of algorithm that hold undisclosed keys for the coding of data or content into secret writing (cipher text) and decrypt (decodes) them back into their primary state is known as cryptosystem..

4.7.5 Computer Gaming

The rival of the human participant in the field of gaming is often a structure of further Artificial Intelligence (AI) that includes Genetic Algorithm. Procedures that have been utilized and also organized via GA to make certain that Artificial Intelligence (AI) is able to discover and advance from the past experience. AI tends to avoid repeating past mistakes with the help of the learning techniques which therefore boost the playability of the game. The human player gets more experience as they are vital to change their strategies from occasionally. A state whereby the human participant finds a pattern of moves which definitely guarantee success is avoided by this process, which implies the game does not pose any challenge any longer.

The characteristics of bending easily without breaking of Genetic Algorithm (Gas) have led to the used of this field in different applications over diverse industries. Some of these applications are discussed below

CHAPTER FIVE

5.0 SUMMARY

This project work was aimed to site the application and connection of computing and biology based on Darwin theory of natural selection. Also this study focuses on the different existing AI i.e. the traditional search optimization method and the genetic algorithm, merits and demerits of genetic algorithms, why genetic algorithm was the most preferred methods by many scientists in solving optimization problems, common among them be the Travelling Salesman Problem (TSP). The following conditions were established in this study:

i. Artificial Intelligence (AI) is the field of Computer discipline that integrating computer and biological intelligence
ii. That genetic algorithm and traditional search method are subfield of AI
iii. That Darwin theory of evolution is an AI methodology that inspires genetic algorithms.
iv. That the sustained existence of the most fit amid individuals over successive generations for resolving problems is simulated by genetic algorithm.
v. That genetic algorithm has operators that it uses to maintain genetic diversity. The operators are Reproduction (selection), Crossover (i.e. recombination) and Mutation
vi. That genetic algorithm is better than and preferred above other traditional search method in that it is more robust, very easy to understand

5.1 CONCLUSION

It is simple to relate genetic algorithms to a large array of problems such as optimizing problems ranging from the Travelling Salesman Problems (TSP), scheduling and layout problems and early model knowledge, in several problems the result might be fine but reduced on others. The algorithm could be slow if mutation was used, crossover makes the algorithm to be significantly faster. More specifically, genetic algorithm is likened to hill climbing search and extremely related to a sampled beam search. There is a trouble with local maxima associated with all hill climbing algorithms. Within genetic problem, limited domains are those individuals that get trapped with a good but not best fitness measure. Little change (mutation) of any kind gives bad fitness but crossover can help them get out of a restricted domain. Since change (mutation) is a random procedure, there is a possibility of having an unexpected big mutation to help these individuals out of this situation. These individuals will by no means get out; it is their children (offspring) that gets out of limited domain. A major distinction between hill climbing and Genetic Algorithms is that it's usually a fine idea in genetic algorithm (Gas) to fill the limited domain with individuals. Above all, Gas has fewer problems with limited domain than with reverse transmission neural system.

5.2 RECOMMENDATION

Developments of forms of genetic algorithms (Gas) to adapt to various precise tasks will be witnessed in the future. The very theory of genetic algorithms (Gas) that it is unaware of the problem domain as used, to solve problems might be defied. Latter on the future we will realize that Gas can become even more powerful by this practice.

APPENDIX

Creating the Genetic Algorithm

In literature of the traveling salesman problem since locations are normally denoted as cities, and routes are referred to as tours, we shall adopt the standard naming conventions in our code.

To start, let's create a class that can encode the cities.

City.java

```
/*
* City.java
* Models a city
*/
package tsp;


public class City {
    int x;
    int y;

    // Constructs a randomly placed city
    public City(){
        this.x = (int)(Math.random()*200);
        this.y = (int)(Math.random()*200);
    }
```

```
// Constructs a city at chosen x, y location
public City(int x, int y){
    this.x = x;
    this.y = y;
}

// Gets city's x coordinate
public int getX(){
    return this.x;
}

// Gets city's y coordinate
public int getY(){
    return this.y;
}

// Gets the distance to given city
public double distanceTo(City city){
    int xDistance = Math.abs(getX() - city.getX());
    int yDistance = Math.abs(getY() - city.getY());
    double distance = Math.sqrt( (xDistance*xDistance) + (yDistance*yDistance) );

    return distance;
}
```

```
    @Override
    public String toString(){
        return getX()+", "+getY();
    }
}
```

Now we can create a class that holds all of our destination cities for our tour

TourManager.java

```
/*
* TourManager.java
* Holds the cities of a tour
*/

package tsp;

import java.util.ArrayList;

public class TourManager {

    // Holds our cities
    private static ArrayList destinationCities = new ArrayList<City>();

    // Adds a destination city
```

```
    public static void addCity(City city) {
        destinationCities.add(city);
    }

    // Get a city
    public static City getCity(int index){
        return (City)destinationCities.get(index);
    }

    // Get the number of destination cities
    public static int numberOfCities(){
        return destinationCities.size();
    }
}
```

Next we need a class that can encode our routes, these are generally referred to as tours so we'll stick to the convention.

Tour.java

```
/*
* Tour.java
* Stores a candidate tour
*/
```

```
package tsp;

import java.util.ArrayList;
import java.util.Collections;

public class Tour{

    // Holds our tour of cities
    private ArrayList tour = new ArrayList<City>();
    // Cache
    private double fitness = 0;
    private int distance = 0;

    // Constructs a blank tour
    public Tour(){
        for (int i = 0; i < TourManager.numberOfCities(); i++) {
            tour.add(null);
        }
    }

    public Tour(ArrayList tour){
        this.tour = tour;
    }

    // Creates a random individual
```

```
public void generateIndividual() {
    // Loop through all our destination cities and add them to our tour
    for (int cityIndex = 0; cityIndex < TourManager.numberOfCities(); cityIndex++) {
      setCity(cityIndex, TourManager.getCity(cityIndex

}
    // Randomly reorder the tour
    Collections.shuffle(tour);
}

// Gets a city from the tour
public City getCity(int tourPosition) {
    return (City)tour.get(tourPosition);
}

// Sets a city in a certain position within a tour
public void setCity(int tourPosition, City city) {
    tour.set(tourPosition, city);
    // If the tours been altered we need to reset the fitness and distance
    fitness = 0;
    distance = 0;
}

// Gets the tours fitness
public double getFitness() {
    if (fitness == 0) {
```

```
        fitness = 1/(double)getDistance();
    }
    return fitness;
}

// Gets the total distance of the tour
public int getDistance(){
    if (distance == 0) {
        int tourDistance = 0;
        // Loop through our tour's cities
        for (int cityIndex=0; cityIndex < tourSize(); cityIndex++) {
            // Get city we're travelling from
            City fromCity = getCity(cityIndex);
            // City we're travelling to
            City destinationCity;
            // Check we're not on our tour's last city, if we are set       our
            // tour's final destination city to our starting city
            if(cityIndex+1 < tourSize()){
                destinationCity = getCity(cityIndex+1);
            }
            else{
                destinationCity = getCity(0);
            }
            // Get the distance between the two cities
            tourDistance += fromCity.distanceTo(destinationCity);
```

```
            }
            distance = tourDistance;
        }
        return distance;
    }

    // Get number of cities on our tour
    public int tourSize() {
        return tour.size();
    }

    // Check if the tour contains a city
    public boolean containsCity(City city){
        return tour.contains(city);
    }

    @Override
    public String toString() {
        String geneString = "|";
        for (int i = 0; i < tourSize(); i++) {
            geneString += getCity(i)+"|";
        }
        return geneString;
    }
}
```

We also need to create a class that can hold a population of candidate tours

Population.java

```
/*
* Population.java
* Manages a population of candidate tours
*/

package tsp;

public class Population {

    // Holds population of tours
    Tour[] tours;

    // Construct a population
    public Population(int populationSize, boolean initialise) {
        tours = new Tour[populationSize];
        // If we need to initialise a population of tours do so
        if (initialise) {
            // Loop and create individuals
            for (int i = 0; i < populationSize(); i++) {
                Tour newTour = new Tour();
                newTour.generateIndividual();
```

```
            saveTour(i, newTour);
        }
    }
}

// Saves a tour
public void saveTour(int index, Tour tour) {
    tours[index] = tour;
}

// Gets a tour from population
public Tour getTour(int index) {
    return tours[index];
}

// Gets the best tour in the population
public Tour getFittest() {
    Tour fittest = tours[0];
    // Loop through individuals to find fittest
    for (int i = 1; i < populationSize(); i++) {
        if (fittest.getFitness() <= getTour(i).getFitness()) {
            fittest = getTour(i);
        }
    }
    return fittest;
```

```
    }

    // Gets population size
    public int populationSize() {
        return tours.length;
    }
}
```

Next, the let's create a GA class which will handle the working of the genetic algorithm and evolve our population of solutions.

GA.java

```
/*
* GA.java
* Manages algorithms for evolving population
*/

package tsp;

public class GA {

    /* GA parameters */
    private static final double mutationRate = 0.015;
    private static final int tournamentSize = 5;
    private static final boolean elitism = true;
```

```
// Evolves a population over one generation
public static Population evolvePopulation(Population pop) {
    Population newPopulation = new Population(pop.populationSize(), false);

    // Keep our best individual if elitism is enabled
    int elitismOffset = 0;
    if (elitism) {
        newPopulation.saveTour(0, pop.getFittest());
        elitismOffset = 1;
    }

    // Crossover population
    // Loop over the new population's size and create individuals    from
    // Current population
    for (int i = elitismOffset; i < newPopulation.populationSize(); i++) {
        // Select parents
        Tour parent1 = tournamentSelection(pop);
        Tour parent2 = tournamentSelection(pop);
        // Crossover parents
        Tour child = crossover(parent1, parent2);
        // Add child to new population
        newPopulation.saveTour(i, child);
    }
```

```
    // Mutate the new population a bit to add some new genetic material
    for (int i = elitismOffset; i < newPopulation.populationSize(); i++) {
        mutate(newPopulation.getTour(i));
    }

    return newPopulation;
}

// Applies crossover to a set of parents and creates offspring
public static Tour crossover(Tour parent1, Tour parent2) {
    // Create new child tour
    Tour child = new Tour();

    // Get start and end sub tour positions for parent1's tour
    int startPos = (int) (Math.random() * parent1.tourSize());
    int endPos = (int) (Math.random() * parent1.tourSize());

    // Loop and add the sub tour from parent1 to our child
    for (int i = 0; i < child.tourSize(); i++) {
        // If our start position is less than the end position
        if (startPos < endPos && i > startPos && i < endPos) {
            child.setCity(i, parent1.getCity(i));
        } // If our start position is larger
        else if (startPos > endPos) {
            if (!(i < startPos && i > endPos)) {
```

```
                child.setCity(i, parent1.getCity(i));
            }
        }
    }

    // Loop through parent2's city tour
    for (int i = 0; i < parent2.tourSize(); i++) {
        // If child doesn't have the city add it
        if (!child.containsCity(parent2.getCity(i))) {
            // Loop to find a spare position in the child's tour
            for (int ii = 0; ii < child.tourSize(); ii++) {
                // Spare position found, add city
                if (child.getCity(ii) == null) {
                    child.setCity(ii, parent2.getCity(i));
                    break;
                }
            }
        }
    }
    return child;
}

// Mutate a tour using swap mutation
private static void mutate(Tour tour) {
    // Loop through tour cities
```

```
    for(int tourPos1=0; tourPos1 < tour.tourSize(); tourPos1++){
        // Apply mutation rate
        if(Math.random() < mutationRate){
            // Get a second random position in the tour
            int tourPos2 = (int) (tour.tourSize() * Math.random());

            // Get the cities at target position in tour
            City city1 = tour.getCity(tourPos1);
            City city2 = tour.getCity(tourPos2);

            // Swap them around
            tour.setCity(tourPos2, city1);
            tour.setCity(tourPos1, city2);
        }
    }
}

// Selects candidate tour for crossover
private static Tour tournamentSelection(Population pop) {
    // Create a tournament population
    Population tournament = new Population(tournamentSize, false);
    // For each place in the tournament get a random candidate tour and
    // add it
    for (int i = 0; i < tournamentSize; i++) {
        int randomId = (int) (Math.random() * pop.populationSize());
```

```
            tournament.saveTour(i, pop.getTour(randomId));
        }
        // Get the fittest tour
        Tour fittest = tournament.getFittest();
        return fittest;
    }
}
```

Now we can create our main method, add our cities and evolve a route for our travelling salesman problem.

TSP_GA.java

```
/*
* TSP_GA.java
* Create a tour and evolve a solution
*/

package tsp;

public class TSP_GA {

    public static void main(String[] args) {

        // Create and add our cities
        City city = new City(60, 200);
        TourManager.addCity(city);
```

```
City city2 = new City(180, 200);
TourManager.addCity(city2);
City city3 = new City(80, 180);
TourManager.addCity(city3);
City city4 = new City(140, 180);
TourManager.addCity(city4);
City city5 = new City(20, 160);
TourManager.addCity(city5);
City city6 = new City(100, 160);
TourManager.addCity(city6);
City city7 = new City(200, 160);
TourManager.addCity(city7);
City city8 = new City(140, 140);
TourManager.addCity(city8);
City city9 = new City(40, 120);
TourManager.addCity(city9);
City city10 = new City(100, 120);
TourManager.addCity(city10);
City city11 = new City(180, 100);
TourManager.addCity(city11);
City city12 = new City(60, 80);
TourManager.addCity(city12);
City city13 = new City(120, 80);
TourManager.addCity(city13);
City city14 = new City(180, 60);
```

```
TourManager.addCity(city14);
City city15 = new City(20, 40);
TourManager.addCity(city15);
City city16 = new City(100, 40);
TourManager.addCity(city16);
City city17 = new City(200, 40);
TourManager.addCity(city17);
City city18 = new City(20, 20);
TourManager.addCity(city18);
City city19 = new City(60, 20);
TourManager.addCity(city19);
City city20 = new City(160, 20);
TourManager.addCity(city20);

// Initialize population
Population pop = new Population(50, true);
System.out.println("Initial distance: " + pop.getFittest().getDistance());

// Evolve population for 100 generations
pop = GA.evolvePopulation(pop);
for (int i = 0; i < 100; i++) {
    pop = GA.evolvePopulation(pop);
}

// Print final results
```

```
        System.out.println("Finished");
        System.out.println("Final distance: " + pop.getFittest().getDistance());
        System.out.println("Solution:");
        System.out.println(pop.getFittest());
    }
}
```

Output:

Initial distance: 1996

Finished

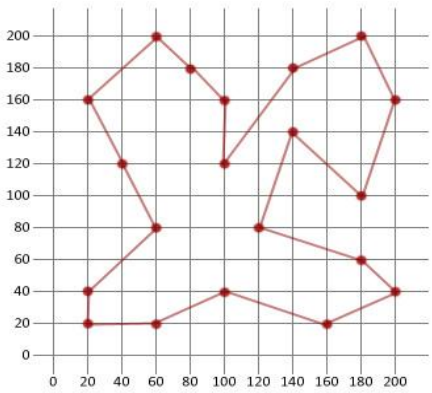
Final distance: 940

Solution:

|60, 200|20, 160|40, 120|60, 80|20, 40|20, 20|60, 20|100, 40|160, 20|200, 40|180, 60|120, 80|140, 140|180, 100|200, 160|180, 200|140, 180|100, 120|100, 160|80, 180|

As you may see in just 100 generations we were able to find a route just over twice as good as our original and probably pretty close to optimum.

Final Results:



REFERENCES

McCarthy, J. 1960. "Recursive functions of symbolic expressions and their computation by machine". (Retrieve 2019)

Christian Jacob, 2001. Illustrating Evolutionary Computation with Mathematical. (Retrieve 2020)

John Holland, 1975 Adaptation in Natural and Artificial Systems. (Retrieve 2019)

Noor **2007** - National Acad Sciences, Simulations of biological evolution, in which computers are used to evolve systems toward a goal, often require many generations to achieve even simple goals

Obitko 1998, Introduction to genetic algorithms, Gzech Technical University. (Retrieve 2019)

Michael Negnevitsky 2002, Artificial Intelligence A Guide to Intelligent Systems Second Edition. (Retrieve 2020)

Webster& Watson, 2003; **Artificial intelligence**: a survey on evolution, models, applications and future trends. (Retrieve 2020)

Geetha, R.R.; Bouvanasilan, N.; Seenuvasan,**V.A** perspective view on Travelling Salesman Problem using genetic algorithm" Nature & Biologically Inspired Computing, 2009. NaBIC 2009, World Congress. (Retrieve 2019)

Prof. Dr. R. Safaric & Dr. A. Rojko, "Genetic code of the parent and the offspring before and after the cross-over", Maribor December 2006,

Mariusz Głabowski, Bartosz Musznicki, Przemysław Nowak , and Piotr Zwierzykowski. 2013. "Efficiency Evaluation of Shortest Path Algorithms".

Huang, B, Wu, Q, and Zhan, F.B. 2007. "A shortest path algorithm with novel heuristics for dynamic transportation networks.

Sturtevant, N. R., and R. Geisberger. 2010, A Comparison of High-Level Approaches for Speeding Up Pathfinding. In Proceedings of the Sixth Conference on Artificial Intelligence and Interactive Digital Entertainment.

Rehab F. Abdel-Kader, 2011, an improved discrete PSO with GA operators for efficient QoS- multicast routing, International Journal of Hybrid Information Technology.

Davoud Sedighizadeh and Ellips Masehian, 2009, Particle Swarm Optimization Methods, Taxonomy and application, International Journal of Computer Theory and Engineering.

Hemalatha, S, and Valsalal, P. 2012. Identification of Optimal Path in Power System Network Using Bellman Ford Algorithm.

Andrew, S, and David, J. 2011, Computer network.

Peter Hofner, and Bernhard Moller. 2012." Dijkstra, Floyd and Warshall meet Kleene".

Walter .V. M. Kada and H. Chen 2006. Shortest path analysis in raster maps for pedestrian navigation in location based systems.

Li, Y., R. He and Y. Guo, 2006. Faster genetic algorithm for network path.