**SAVITRIBAI PHULE PUNE UNIVERSITY**

**A PRELIMINARY PROJECT REPORT ON**

**“Optimization of travelling salesman problem using Genetic Algorithm”**

**SUBMITTED TOWARDS THE PARTIAL FULFILMENT OF THE REQUIREMENTS OF**

**Academic Year: 2019-20**

**By:**

**Aditi Deshpande (BECOA125)**

**Anand Kulkarni (BECOA160)**

**Under The Guidance of**

**Dr. Anuradha Thakare**

****

**DEPARTMENT OF COMPUTER ENGINEERING,**

**PCET’S PIMPRI CHINCHWAD COLLEGE OF ENGINEERING**

Sector No. 26, Prabhakaran, Nigdi,

Pune - 411044

****

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

**CERTIFICATE**

This is to certify that, the project entitled

**“Optimization of travelling salesman problem using Genetic Algorithm**”

is successfully carried out as a mini project successfully submitted by following students of “PCET's Pimpri Chinchwad College of Engineering, Nigdi, Pune-44**”.**

**Aditi Deshpande :BECOA125**

**Anand Kulkarni :BECOA160**

**Under the guidance of Prof. Dr. Anuradha Thakare** In the partial fulfilment of the requirements for the B.E. (Computer Engineering)

**Dr. Anuradha Thakare**

**Project Guide**

Place:Pune

Date:

**ABSTRACT**

Traveling salesman problem (TSP) is a well-known NP-hard problem in many real-world applications. The aim of TSP is to find a complete, minimal-cost tour when a salesman is required to visit each of  given cities once and only once . So far, TSP has often been a touchstone for new strategies and algorithms proposed to solve combinatorial optimization problem.

Many methods have been developed for solving TSP, including exact algorithms and approximate algorithms. The exact algorithms are carried out to find the optimal solution from all valid solutions in a number of steps. But, because of exponential complexity, they are always infeasible if the scale of TSP becomes large.

Genetic algorithm (GA) is a global search algorithm appropriate for problems with huge search, for example, TSP, in which the initial population decides iterations, the crossover realizes the construction of the offspring, and the mutation operator maintains the diversity of the individuals. So far, there is a lot of literature to improve the effectiveness of crossover and mutation . Besides, as the first step of any GA, how to initialize an efficient population plays an important role in the process of solving a problem based on GA.

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter** | **Contents** | | **Page No.** |
| 1. |  | Introduction |  |
|  | *a.* | Problem Defination | 5 |
|  | *b.* | Motivation | 5 |
|  | *c.* | Objective | 5 |
|  | d. | Scope | 5 |
| 2. |  | Literature Survey | 6 |
| 3. |  | Analysis |  |
|  | a. | Mathematical Model | 7 |
|  | b. | Requirment Analysis | 7 |
|  | c. | Analysis Model | 8 |
| 4. |  | Requirment Specification |  |
|  | a. | User Interface | 9 |
|  | b. | S/W Interfaces | 9 |
|  | c. | H/W Interfaces | 9 |
| 5 |  | System Design |  |
|  | a. | System Architecture | 10 |
|  | c. | Flow Chart | 10 |
|  | e. | Dataflow Diagram | 11 |
|  | f. | Activity Diagram | 11 |
| 6 |  | Project Implementation |  |
|  | a. | Overview of Project Modules | 12 |
|  | b. | Tool and Technologies | 12 |
|  | c. | Algorithm Details | 12 |
|  | d. | Source Details | 16 |
| 7 |  | Software Testing | 17 |
| 7 |  | Results |  |
|  | a. | Outcomes | 20 |
| 8 |  | Conclusion |  |
|  | a. | Conclusion | 21 |
|  | b. | Furture work | 21 |
|  | c. | Application | 21 |
| 9 |  | Refrences | 22 |

Chapter 1: Introduction

**a) Problem Statement**

The Travelling Salesman Problem is one of the best known NP-hard problems, which means that there is no exact algorithm to solve it in polynomial time. The minimal expected time to obtain optimal solution is exponential. TSP is defined as a permutation problem with the objective of finding the path of the shortest length (or the minimum cost). TSP can be modelled as an undirected weighted graph, such that cities are the graph’s vertices, paths are the graph’s edges, and a path’s distance is the edge’s length. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex only once. Often, the model is a complete graph. If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour. Mathematically, it can be defined as given a set of n cities, named {c1, c2,…cn}, and permutations, σ1, ..., σn!.

**b) Project Idea**

To apply Genetic Algorithm technique to determine the optimal(path with least distance) path to be travelled by the travelling salesman with various combinations generated by the Genetic Algorithm.

**c) Motivation**

TSP known to be the NP- Hard, it is difficult to find the optimal solution with least complexicity. Genetic Algorithms (GA) can eliminate the least optimal solutions by making use of the fitness function which reduces the complexicity and improve performance as well. In this project, we will use GA to select the best optimal solution for the travelling salesman problem.

**d) Scope**

The existing system contains the solution with higher complexity and requires more time .Use of Genetic Algorithm makes the task efficient by the use of the fitness fitness function thus eliminates the least optimal path this improves the efficiency of the Algorithm.

Chapter 2: Literature Survey

Oliviu Matei proposed the solution for the Generalized Traveling Salesman Problem GTSP. GTSP has many application areas in science and engineering.The author used a local- global technique to solve Generalized Traveling Salesman Problem. Author proposed an efficient Genetic Algorithm to solve GTSP. The proposed algorithm use the techniques of local and global connections by considering nodes sets (clusters) and find a minimum.

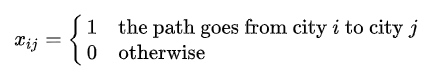
distance by including exactly one node from each cluster. Author simulates the problem on various TSP instances for TSPLIB and compared the result of proposed algorithm with some other existing algorithms. Gohar Vahdati et al. [2] proposed a hybrid search algorithm with Hopfield neural network (HNN) and Genetic algorithm (GA). The proposed algorithm has both the advantages of HNN and GA that can explore the search space and exploit the best solution. HNN is a very nice and efficient technique to solve TSP. Experimental results demonstrate that the proposed algorithm does not get stuck at a local optimum.

Chapter 3: Analysis

1. **Mathematical Model**

### Miller-Tucker-Zemlin formulation

Label the cities with the numbers 1, …, *n* and define:

{\displaystyle x\_{ij}={\begin{cases}1&{\text{the path goes from city }}i{\text{ to city }}j\\0&{\text{otherwise}}\end{cases}}}

For *i* = 1, …, *n*, distance from city *i* to city *j*. Then TSP can be written as the following integer linear programming problem:

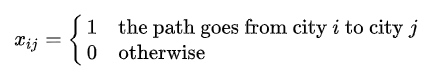
{\displaystyle {\begin{aligned}\min &\sum \_{i=1}^{n}\sum \_{j\neq i,j=1}^{n}c\_{ij}x\_{ij}\colon &&\\&x\_{ij}\in \{0,1\}&&i,j=1,\ldots ,n;\\&u\_{i}\in \mathbf {Z} &&i=2,\ldots ,n;\\&\sum \_{i=1,i\neq j}^{n}x\_{ij}=1&&j=1,\ldots ,n;\\&\sum \_{j=1,j\neq i}^{n}x\_{ij}=1&&i=1,\ldots ,n;\\&u\_{i}-u\_{j}+nx\_{ij}\leq n-1&&2\leq i\neq j\leq n;\\&0\leq u\_{i}\leq n-1&&2\leq i\leq n.\end{aligned}}}

The first set of equalities requires that each city is arrived at from exactly one other city, and the second set of equalities requires that from each city there is a departure to exactly one other city.

{\displaystyle nk\leq (n-1)k,}

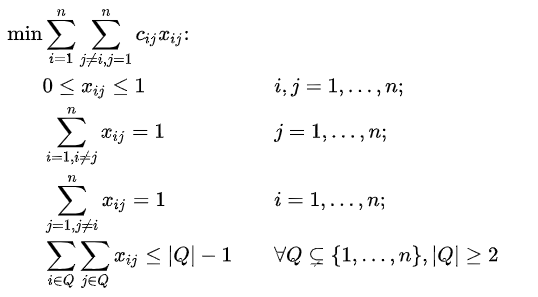
### Dantzig-Fulkerson-Johnson formulation

Label the cities with the numbers 1, …, *n* and define:

{\displaystyle x\_{ij}={\begin{cases}1&{\text{the path goes from city }}i{\text{ to city }}j\\0&{\text{otherwise}}\end{cases}}}

Take {\displaystyle c\_{ij}} to be the distance from city *i* to city *j*. Then TSP can be written as the following

integer linear programming problem:

{\displaystyle {\begin{aligned}\min &\sum \_{i=1}^{n}\sum \_{j\neq i,j=1}^{n}c\_{ij}x\_{ij}\colon &&\\&0\leq x\_{ij}\leq 1&&i,j=1,\ldots ,n;\\&\sum \_{i=1,i\neq j}^{n}x\_{ij}=1&&j=1,\ldots ,n;\\&\sum \_{j=1,j\neq i}^{n}x\_{ij}=1&&i=1,\ldots ,n;\\&\sum \_{i\in Q}{\sum \_{j\in Q}{x\_{ij}}}\leq |Q|-1&&\forall Q\subsetneq \{1,\ldots ,n\},|Q|\geq 2\\\end{aligned}}}

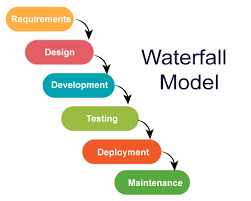
The last constraint of the DFJ formulation ensures that there are no sub-tours among the non-starting vertices, so the solution returned is a single tour and not the union of smaller tours. Because this leads to an exponential number of possible constraints,

1. **Requirment Analysis**

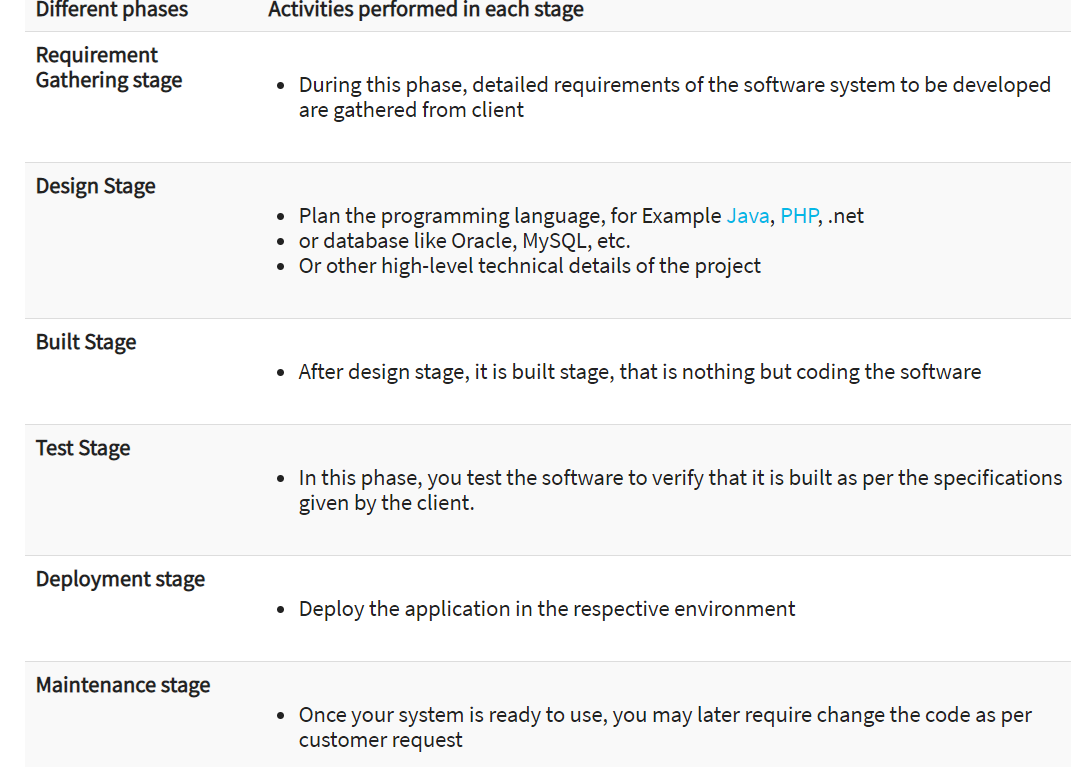
Travelling salesman problem involves information of certain cities through which the salesman has to travel. Calculation of Euclidean distance in between different cities can be maintained using the matrix.

1. **Analysis Model**

The best approach for analyzing the system is the Waterfall model it is most widely used model for analyzing the system at various aspects.



**WATERFALL MODEL** is a sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases. Each phase is designed for performing specific activity during the SDLC phase.

****

Chapter 4: Requirment Specification

1. **User Interfaces**

Since the TSP problem is implemented using java technology the most appropriate library/package to be used for User Inter5face designing is the Java SWING.

Java Swing is a lightweight Graphical User Interface (GUI) toolkit that includes a rich set of widgets. It includes package lets you make GUI components for your Java applications, and It is platform independent.

The Swing library is built on top of the Java Abstract Widget Toolkit (**AWT**), an older, platform dependent GUI toolkit.

**b) Software Interfaces**

The software interfaces involves one of the efficient programming language Java.

Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)

It is one of the most popular programming language in the world

It is easy to learn and simple to use

It is open-source and free

It is secure, fast and powerful

It has a huge community support (tens of millions of developers)

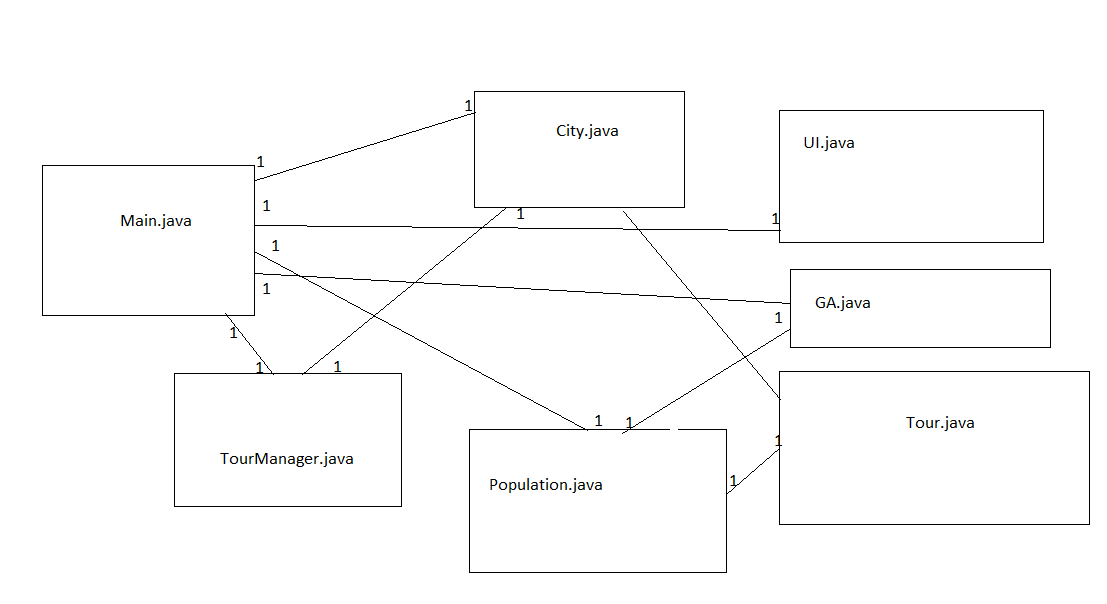
Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs.

1. **Hardware Requirment**

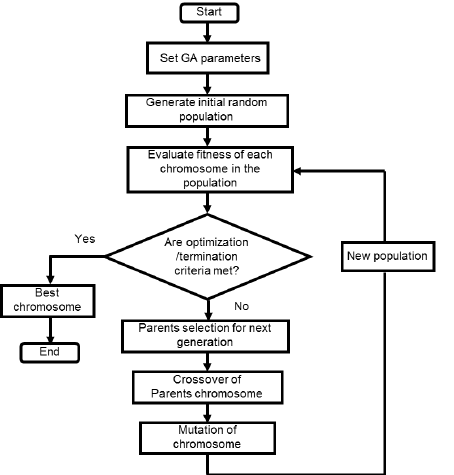
Processor: Intel(R) Core(TM)i3-2370M CPU @ 2.40GHz 2.40GHz, RAM: 4.00 GB System Type: 64-bit or 32-bit operating system.

Chapter 5: System Design

1. **System Architecture**



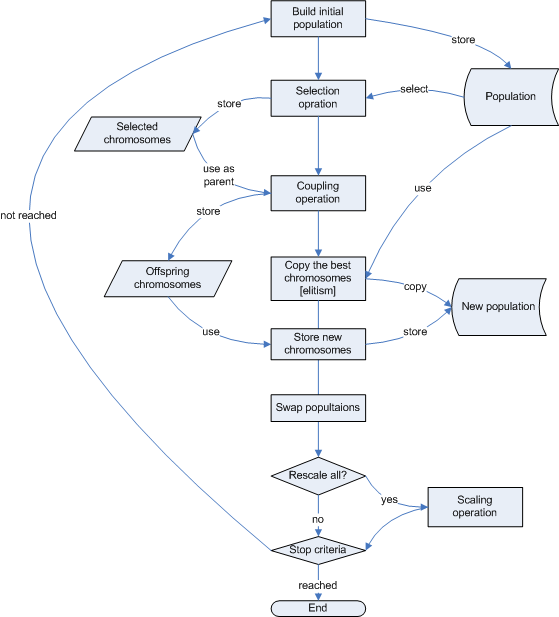
**b)Flowchart**



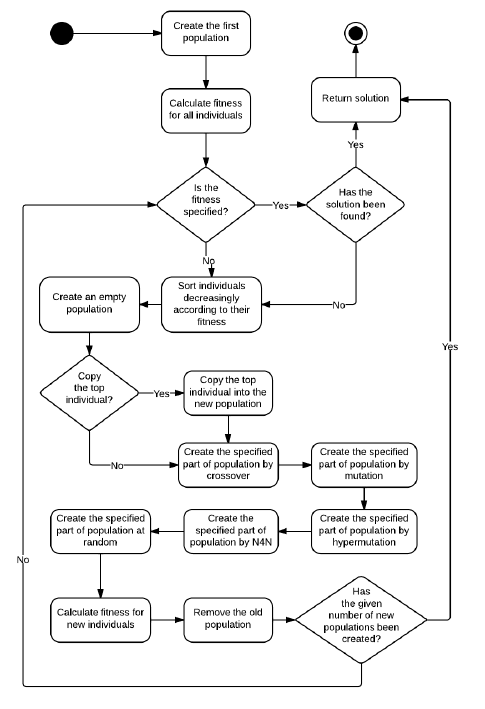
Here cities are taken as genes, string generated using these characters is called a chromosome, while a fitness score which is equal to the path length of all the cities mentioned, is used to target a population.

Fitness Score is defined as the length of the path described by the gene. Lesser the path length fitter is the gene. The fittest of all the genes in the gene pool survive the population test and move to the next iteration. The number of iterations depends upon the value of a cooling variable. The value of the cooling variable keeps on decreasing with each iteration and reaches a threshold after a certain number of iterations.

1. **Dataflow Diagram**



**c)Activity Diagram**



Chapter 6: Project Implemantation

1. **Tools and Technologies**

The tools involved in developing the system includes Java IDE Netbeans,Star UML for developing UML Diagrams such as activity ,dataflow diagrams.

The technologies evolved are Java programming language along with various UI libraries such as swing.

1. **Algorithmetic Details**

### Outline of the Basic Genetic Algorithm

1. **[Start]**Generate random population of *n* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)*of each chromosome *x* in the population
3. **[New population]**Create a new population by repeating following steps until the new population is complete
   1. **[Selection]**Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
   2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
   3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
   4. **[Accepting]** Place new offspring in a new population
4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step **2**

Basic Terminology

A genetic algorithm (or GA) is a search technique used in computing to find true or

approximate solutions to optimization and search problems.

(GA)s are categorized as global search heuristics.

(GA)s are a particular class of evolutionary algorithms that use techniques inspired by

evolutionary biology such as inheritance, mutation, selection, and crossover (also

called recombination).

The evolution usually starts from a population of randomly generated individuals and

happens in generations.

In each generation, the fitness of every individual in the population is evaluated,

multiple individuals are selected from the current population (based on their fitness),

and modified to form a new population.

Terminologies associated with Genetic Algorithm are:

Individual - Any possible solution

Population - Group of all individuals

Fitness – Target function that we are optimizing (each individual has a fitness)

Trait - Possible aspect (features) of an individual

Genome - Collection of all chromosomes (traits) for an individual.

In the following implementation, cities are taken as genes, string generated using these characters is called a chromosome, while a fitness score which is equal to the path length of all the cities mentioned, is used to target a population.

Fitness Score is defined as the length of the path described by the gene. Lesser the path length fitter is the gene. The fittest of all the genes in the gene pool survive the population test and move to the next iteration.

The number of iterations depends upon the value of a cooling variable. The value of the cooling variable keeps on decreasing with each iteration and reaches a threshold after a certain number of iterations.

# **Initialization**

After getting how to represent each individual, next is to initialize the population by selecting the proper number of individuals within it.

# **Selection**

Next is to select a number of individuals from the population in the mating pool. Based on the previously calculated fitness value, the best individuals based on a threshold are selected. After that step, we will end selecting a subset of the population in the mating pool.

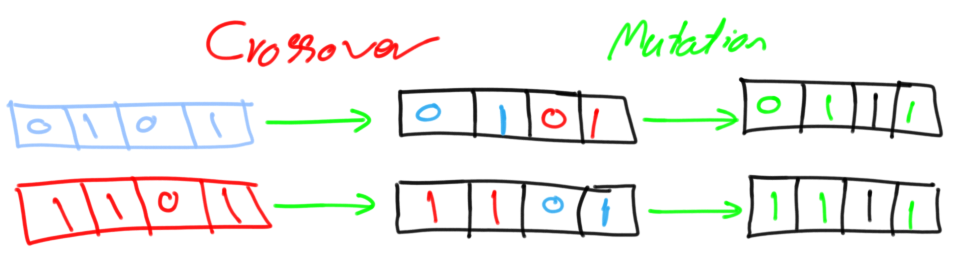
# **Variation Operators**

Based on the selected individuals in the mating pool, parents are selected for mating. The selection of each two parents may be by selecting parents sequentially (1–2, 3–4, and so on). Another way is random selection of the parents.

For every two parents selected, there are a number of variation operators to get applied such as:

1. Crossover (recombination)
2. Mutation

Figure 4 gives an example for these operators.



**Figure 4. Crossover and mutation.**

# **Crossover**

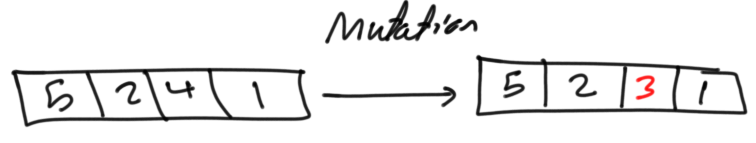
Crossover in GA generates new generation the same as natural mutation. By mutating the old generation parents, the new generation offspring comes by carrying genes from both parents. The amount of genes carried from each parent is random. Remember that GA is random-based EA. Sometimes the offspring takes half of its genes from one parent and the other half from the other parent and sometimes such percent changes. For every two parents, crossover takes place by selecting a random point in the chromosome and exchanging genes before and after such point from its parents. The resulting chromosomes are offspring. Thus operator is called single-point crossover.

Note that crossover is important and without it, the offspring will be identical to its parent.

# **Mutation**

Next variation operator is mutation. For each offspring, select some genes and change its value. Mutation varies based on the chromosome representation but it is up to you to decide how to apply mutation. If the encoding is binary (i.e. the value space of each gene have just two values 0 and 1), then flip the bit value of one or more genes.

But if the gene value comes from a space of more than two values such as 1,2,3,4, and 5, then the binary mutation will not be applicable and we should find another way. One way is by selecting a random value from such set of values as shown in figure 5.



**Figure 5. Mutation by randomly updating some genes.**

Note that without mutation the offspring will have all of its properties from its parents. To add new features to such offspring, mutation took place. But because mutation occurs randomly, it is not recommended to increase the number of genes to be applied to mutation.

The individual after mutation is called mutant.

How the mutation works?

Suppose there are 5 cities: 0, 1, 2, 3, 4. The salesman is in city 0 and he has to find the shortest route to travel through all the cities back to the city 0. A chromosome representing the path chosen can be represented as:  


This chromosome undergoes mutation. During mutation, the position of two cities in the chromosome is swapped to form a new configuration, except the first and the last cell, as they represent the start and endpoint.  


Original chromosome had a path length equal to **INT\_MAX**, according to the input defined below, since the path between city 1 and city 4 didn’t exist. After mutation, the new child formed has a path length equal to **21**, which is a much-optimized answer than the original assumption. This is how the genetic algorithm optimizes solutions to hard problems.

ADVANTAGES

1. Easy to understand.

2. Unlike older AI systems, the GA‘s do not break easily even if the inputs changed slightly. 3. GA can be employed for a wide variety of optimization problems.

4. GA performs very well for large scale optimization problems which may be very difficult or impossible to solve by other traditional methods.

DISADVANTAGES

1. Sometimes have trouble finding the exact global optimum because there is no guaranty to find best solution.

2. It requires large number of fitness function evaluations depending on the number of individuals and the number of generations.

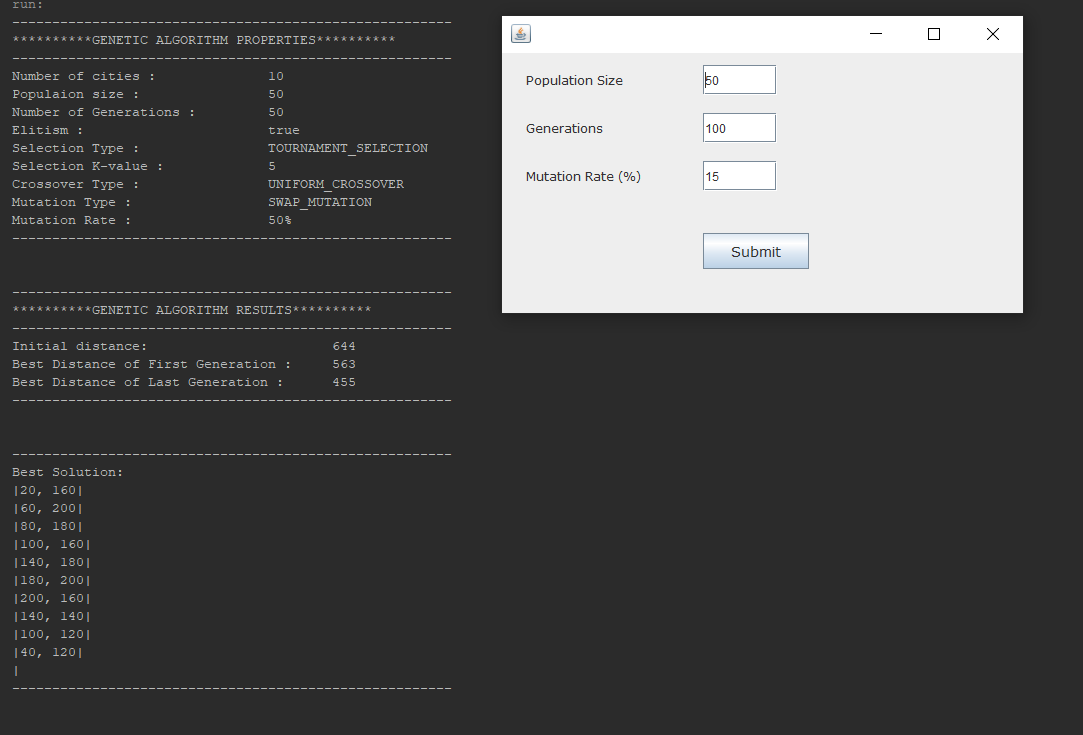
3. GA may take long time to evaluate the individuals.

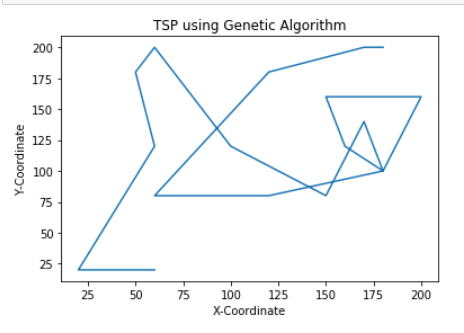
1. **Source code Details**

The source code that implemented on Java has two packages tspGA and TspGui. The files included in these packages are as follows: City.java models a city. TourManager.java holds the tour of the cities. Tour.java stores a candidate tour. Population.java manages the population of candidate tours. GA.java, it adds the cities and evolve a tour for travelling salesman problem.TspGuidisplays the main window where all the operations are performed by the user using graphical user interface.

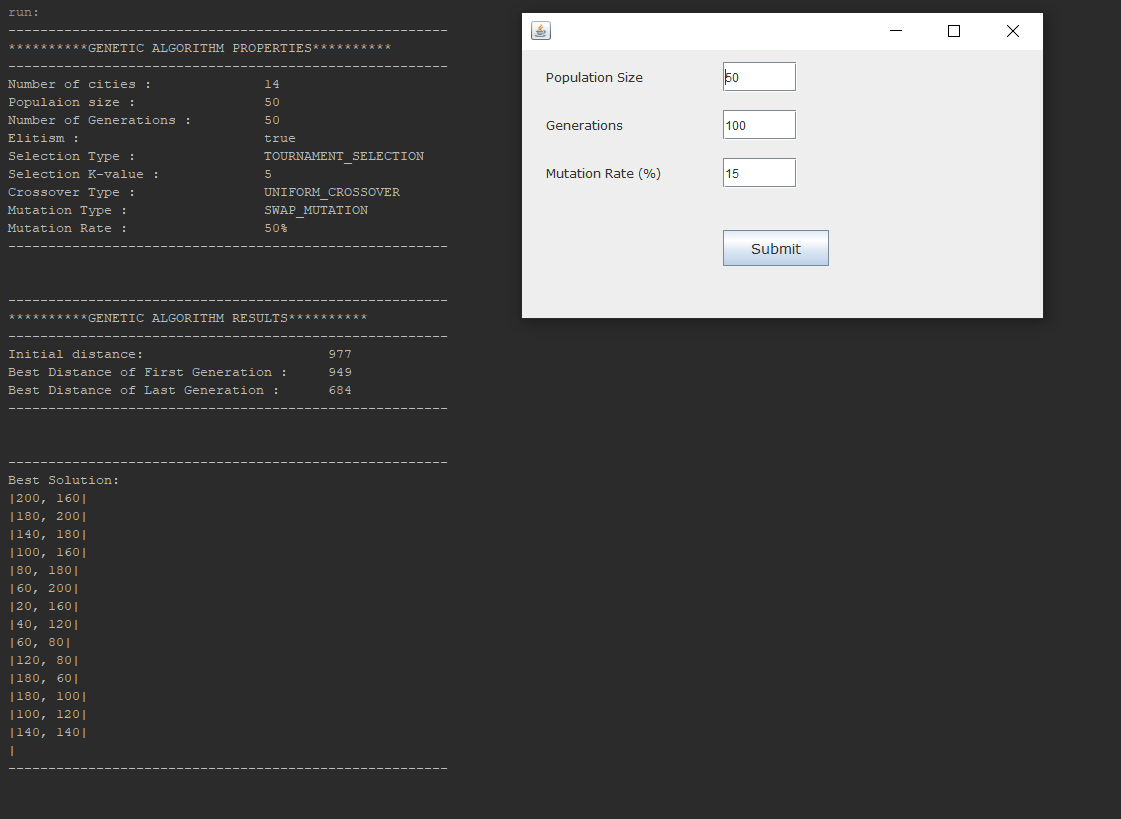
Chapter 7: Software testing

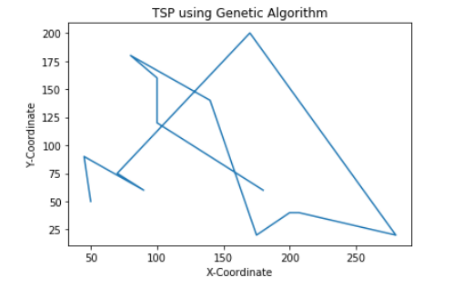
1. **Implementation for 10 Cities**

****

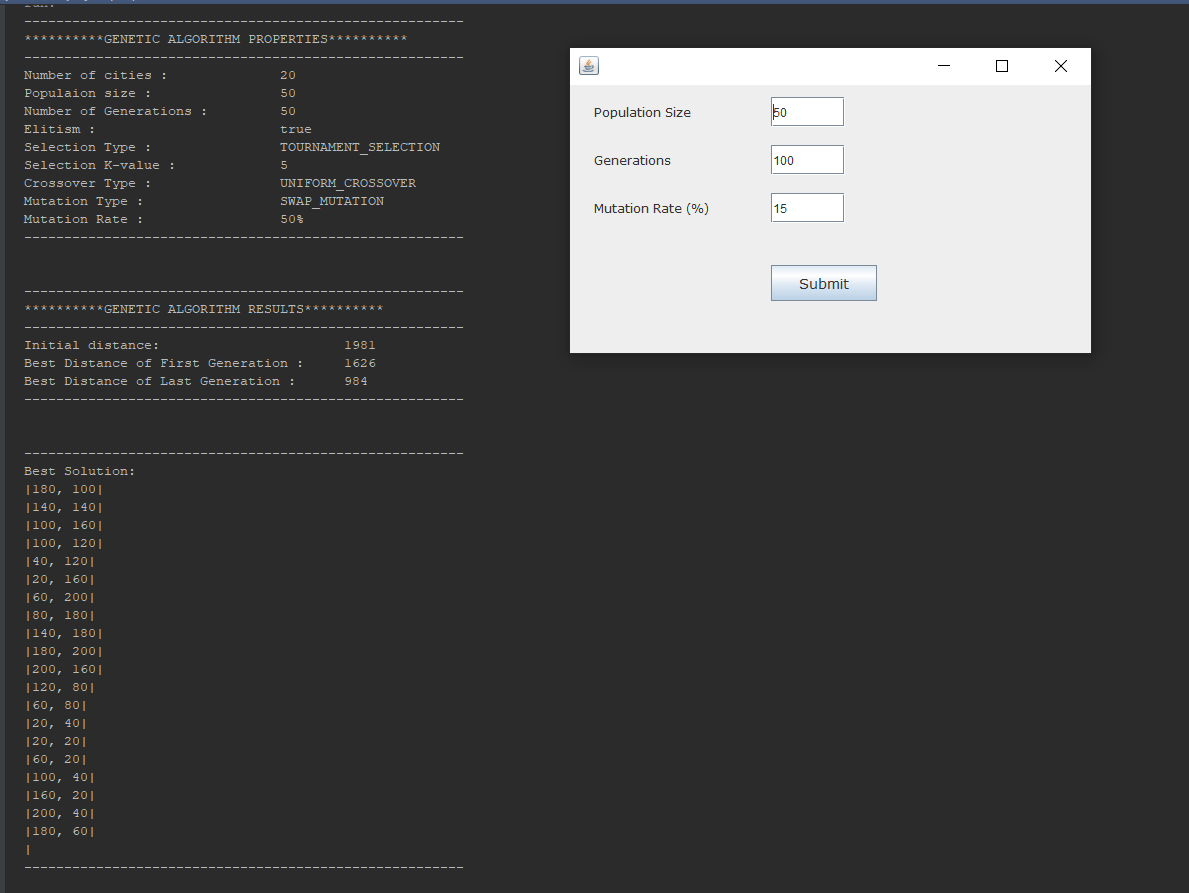


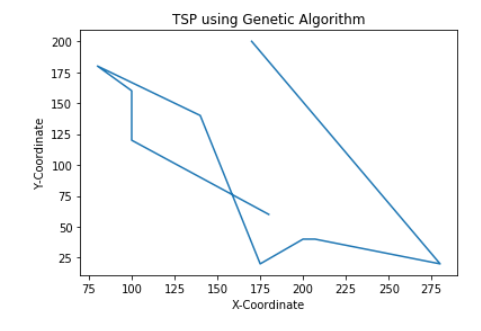
**Implentation for 15 Cities**

****



**Implemantation for 20 Cities**





Chapter 7: Outcomes

1. **Outcomes**

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Cities | Initial Distance | Best Distance of First Generation | Best Distance of Last Generation |
| 5 | 343 | 343 | 343 |
| 10 | 631 | 568 | 455 |
| 15 | 1207 | 1006 | 776 |
| 20 | 1981 | 1626 | 984 |
| 25 | 2593 | 2274 | 1163 |
| 30 | 2687 | 2607 | 1337 |
| 35 | 3095 | 2978 | 1388 |
| 40 | 3662 | 3245 | 1625 |
| 45 | 5237 | 5020 | 2883 |
| 50 | 6904 | 6476 | 3214 |

Chapter 8: Conclusion

1. **Conclusion**

Thus, Genetic algorithms depends very much on the manner the problem is fixed and which crossover and mutation methods are used. It works according to that and discover good solutions for the TSP. A number of genetic algorithm techniques have been analyzed and surveyed for solving TSP. Combining the knowledge from heuristic methods and genetic algorithms is a hopeful approach for solving the TSP. Although all the algorithms have their individual identity and produce solutions according to that, combining two or more algorithms will lead us to reach a best prominent and optimal solution for TSP.

1. **Furture work**

In this project, basic genetic algorithm is used to solve the Travelling Salesman Problem. As a future work, it can also be extended for hybrid genetic algorithm and parallel genetic algorithm. The performance between the two algorithms can also be analyzed based on the selection methods and crossover operators.

1. **Applications**

VEHICLE ROUTING:- Suppose that in a city ‗n‘ mail boxes have to be emptied everyday within a certain period of time, say one hour. The problem is to do the particular work with less number of vehicles in minimum time.

SCHOOL BUS ROUTING PROBLEM:- The objective of this problem is to obtain a bus loading pattern such that the number of routes is minimized.The total distance travelled by all buses is kept at minimum.

MISSION PLANNING PROBLEM:- The objective is to determine an optimal path for army men to accomplish the goals of the mission in the minimum possible time.

PRINTING PRESS SCHEDULING PROBLEM:- One of the major and primary applications of the mTSP arises in scheduling a printing press for a periodical with multi-editions. Here, there exist five pairs of cylinders between which the paper rolls and both sides of a page are printed simultaneously. There exist three kind of forms, namely 4-, 6- and 8-page forms, which are used to print the editions. The scheduling problem consists of deciding which form will be on which run and the length of each run. In the mTSP vocabulary, the plate change costs are the inter-city costs.

Chapter 8: Refrences

[1] VarshikaDwivedi, TarunaChauhan, SanuSaxena and PrincieAgrawal, ―Travelling Salesman Problem using Genetic Algorithm‖, National Conference on Development of Reliable Information Systems, Techniques and Related Issues (DRISTI) 2012.

[2] Naveen kumar, Karambir and Rajiv Kumar, ―A Genetic Algorithm Approach to Study Travelling Salesman Problem‖, Journal of Global Research in Computer Science, Volume 3, No. 3, March 2012, ISSN-2229-371X.

[3] Omar M. Sallabi and Younis EI-Haddad, ―An Improved Genetic Algorithm to Solve the Traveling Salesman Problem‖, World Academy of Science, Engineering and Technology Volume 3, 2009.

[4] ChetanChudasama, S. M. Shah and Mahesh Panchal, ―Comparison of Parents Selection Methods of Genetic Algorithm for TSP‖, International Conference on Computer Communication and Networks CSI- COMNET-2011.

[5] A. Arananayakgi, ―Reduce Total Distance and Time UsingGenetic Algorithm in Traveling Salesman Problem‖,International Journal of Computer Science & EngineeringTechnology, ISSN : 2229-3345, Vol. 5 No. 08 Aug 2014.

[6] Mouhammd Al kasassbeh, Ahmad Alabadleh and TahsenAl-Ramadeen, ―Shared Crossover Method for SolvingTraveling Salesman Problem‖, IJICS Volume 1, Issue 6September 2012. [7] http://www.slideshare/genetic algorithm

[7] Jenna Carr(2014),‖An Introduction to Genetic Algorithms‖.

[8] Elaine Rich & Kevin Knight(1991), Artificial Intelligence, Tata McGraw Hill Edition.

[9] E.L. Lawler et al. The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley(1986).

[10]Hahsler, Michael; Hornik, Kurt (2007), ‖TSP Infrastructure for the Traveling Salesman Problem.

[11] E. Balas (1989). ‖The Prize Collecting Traveling Salesman Problem,‖ Networks 19, 621-636.

[12] E.L. Lawler, J.K. Lenstra, A.H.G. RinnooyKan, and D.B. Shmoys, eds. (1985). The Travelling Salesman Problem, John Wiley, Chichester. [17] P Merz, B Freisleben,1997, Evolutionary Computation, IEEE International Conference on 1997.