# GENETIC SEARCH FOR MULTI SCHOOL ROUTING

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Arjun Sivaram Kumar Ganesan

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

July 2011

Fargo, North Dakota

# North Dakota State University
Graduate School

Title

## GENETIC SEARCH FOR

## MULTI-SCHOOL ROUTING

By

**ARJUN SIVARAM KUMAR GANESAN**

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

**MASTER OF SCIENCE**

SUPERVISORY COMMITTEE:

North Dakota State University Libraries Addendum

To protect the privacy of individuals associated with the document, signatures have been removed from the digital version of this document.

Approved by Department Chair:

Date Signature

# ABSTRACT

Ganesan, Arjun Sivaram Kumar, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, July 2011. Genetic Search for Multi School Routing. Major Professor: Dr. Kendall Nygard.

In genetic search for multi school routing, a set of commonly shared school buses and vans with capacity, travel speed and travel time limitations are used to service students from their pick up locations to their respective schools. The goal is to reduce the number of buses used, the total travel time of the students and the total operating cost of the school buses. This paper focuses on the optimization of the routes using genetic algorithm that is widely applied for combinatorial optimization problems including vehicle routing problems with time windows. The proposed genetic algorithm uses the genetic representation of routes as genes, which is then combined as chromosomes, applies appropriate genetic operators and performs chromosome evaluation. The progress of the solution is also displayed in a GUI window and the results are saved in the database for further analysis. A random set of several test problems were used to evaluate the performance of the proposed approach and encouraging results were obtained.

# ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to Dr. Kendall Nygard for his guidance, encouragement and continued support throughout this research.

I would also like to thank Dr. William Perrizo, Dr. Simone Ludwig and Dr. Jing Shi for their valuable comments and suggestions. I also express my thanks and appreciation to my family for their motivation. Lastly, I am thankful to all colleagues and friends who made my stay at NDSU a memorable and valuable experience.

# DEDICATION

To

*Appa, Amma, Sireesha and Anisha Kutti*

# TABLE OF CONTENTS

# TABLE OF CONTENTS (CONTINUED)

# TABLE OF CONTENTS (CONTINUED)

# LIST OF TABLES

Table Page

# LIST OF FIGURES

# LIST OF FIGURES (CONTINUED)

Figure Page

# CHAPTER 1. INTRODUCTION

School bus routing is an extension of the vehicle routing problems with constrained total travel time, total distance traveled by the students and students reaching school on time. The heuristics for Multi School routing in Regional Education Associations or REAs (formerly known as Joint Powers Agreements or JPAs) routes a set of commonly shared school buses, to pick up students from the pickup locations and drop them at the school. The objective of the problem is to minimize the number of school buses used, the total travel time and the distance traveled by the students when they are in the school bus. The constraints of the problem are to service all the students without exceeding the maximum travel time limit and overloading the school bus. A school bus that picks up a student and travels more than 60 minutes fails to comply with the total travel time constraint. The school bus is considered as overloaded if the total number of students picked up by that bus exceeds the total capacity of the school bus.

The School districts in North Dakota are known for their participation in the Regional Education Associations or REAs to improve educational service to students and to enhance mutual aid in communities and geographic regions. The REA uses school buses owned or leased by different districts, usually housed at the bus driver's house or a location nearer to the driver's house and service students who are assigned to respective pick up points.

REAs now include 93% of all public school districts in the state, covering 92% of the land mass and improving services for over 98% of all public school students [1]. The

map in Figure 1.1 (courtesy of the North Dakota Department of Public Instruction) illustrates the above.

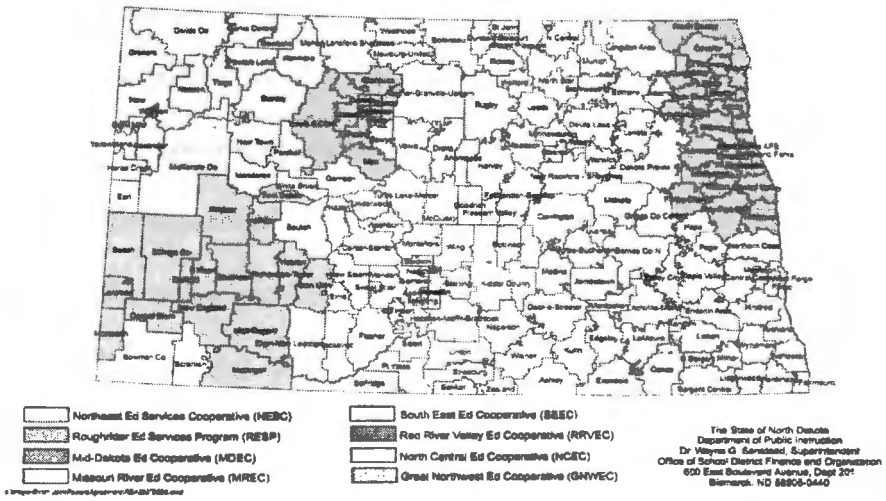



Figure 1.1 Regional Education Associations (2008 - 2009), 11/10/2008

Savelsbergh [2] has shown that finding a feasible solution to the traveling salesman problem with time windows (TSPTW) is a NP-complete problem [3]. Therefore, the heuristics for school bus routing in REAs is a highly combinatorial problem as it involves servicing students using multiple vehicles and multiple locations along with a time constraint.

### 1.1. The Problem

The Problem we address is the Multi School Routing problem with time windows. The Multi school routing problem involves routing a mixed fleet of vehicles, of known capacity and travel speed, from known garaged location to a set of geographically dispersed students with known drop off location (school they attend) within the specified time windows (school opening time). The mixed fleet of vehicles contains regular buses and special category buses. The students can be regular category students or special category students. This paper deals with finding the near optimal low cost routes for the

school buses using Genetic Algorithm (also known as GA) and building a graphical user interface given

The number of students and their category,

The pickup location of the students,

The school they attend and its location,

The number of buses,

The category of the buses,

The size and acceptable maximum travel speed of the buses,

The buses' originating point,

The opening times of the schools,

The acceptable total travel time of the student in the bus (in our problem it is always 60 minutes) and

The operating cost of the buses.

The remainder of the paper is organized as follows: Chapter 2 discusses the literature review of Vehicle Routing Problem. In Chapter 3, we explain the methodology used. Chapter 4 we present the database and graphical user interface. The various experiment results are shown in Chapter 5. Chapter 6 provides the observations and future work.

# CHAPTER 2. LITERATURE REVIEW OF VEHICLE ROUTING PROBLEMS

Vehicle routing problem belongs to the class of combinatorial problem in which customers are served by a fixed number of vehicles. Savelsbergh [2] has shown that finding a feasible solution for VRPTW using a fixed number of vehicles is NP-hard. Therefore, vehicle routing problem like the school bus routing involving fixed number of school buses servicing students with time constraints is a highly combinatorial problem. Though optimal or near optimal solutions to VRPTW can be obtained using exact methods, the computational time required to solve the problem is prohibitive [17]. Surveys about vehicle routing problem with time windows can be found in Desrosier [17], Solomon [14], Braysy and Gendreau [18], and so on. Conventional local searches and heuristic algorithms are commonly devised to find the optimal or near optimal solutions for VRPTW within a reasonable computation time [19]. However, these methods may be sensitive to the datasets given and some heuristic methods may even require a set of training data during the learning process, i.e., the accuracy of training data and the coverage of data distribution can significantly affect the performance [20].

In the past, several heuristics for vehicle routing were proposed by researchers and one such heuristics that interests most researchers is the genetic algorithm that mimics the effects of natural selection. Some of the previous works and the generic algorithm are being discussed briefly in this section.

Genetic algorithms are a class of heuristic search algorithms based on population genetics. It is a search heuristic that mimics the process of natural evolution. The basic concepts of genetic algorithms were primarily developed by Holland [15]. This heuristic is

routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms, which generate solutions to optimization problems using techniques inspired by natural evolution, such as reproduction [16], inheritance, mutation, selection and crossover. In genetic algorithm, a population of strings called as chromosomes that encode candidate solutions called genes, evolves toward better solutions. 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 stochastically selected from the current population based on their fitness and modified (using crossover or mutation) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced or a satisfactory fitness level has been reach for the population.

Sam R Thangiah developed a genetic algorithm based global search strategy for clustering customers in GIDEON [4]. As Genetic Algorithms are adaptive in nature, they can converge to near optimal solutions in many applications, a reason why they are chosen to solve many combinatorial problems. GIDEON had two methods within itself to solve the vehicle routing problem with time windows, a global customer clustering and a local post-optimization method. The first method helps in assignment of customers to the vehicles and the second method helps in optimization of the routes to produce better results. Sam performed the tests on a standard set of 56 VRPTW problems from the literature and proved that genetic algorithms can be competitive to other heuristic search methods. The flow of a genetic algorithm [4] is shown in Figure 2.1

```
Generation ← 0;
Initialize M (Generation);
Evaluate M (Generation);
LOOP (Until termination Condition)
    Generation ← Generation + 1;
    Select M (Generation) from M (Generation + 1);
    Crossover M (Generation);
    Mutate M (Generation);
    Evaluate M (Generation);
LOOP END
```

Figure 2.1 Flow of a Genetic Algorithm

Genetic algorithms prove to be stronger when proper genetic operators and fitness function are used. Genetic operators create a pool of candidates or chromosomes generated over a number of generations. The fitness value dictates the ability of a chromosome to move into the next generation. Chromosomes with higher fitness value are often chosen than a chromosome with less fitness value, resulting in a population of chromosomes with good fitness values. Crossover and Mutation are the primary operators used in a genetic algorithm.

Joe L. Blanton Jr. and Roger L. Wainwright proposed two new crossover operators in their Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms, Merge Cross #1 (MX1) and Merge Cross # 2 (MX2) [5]. These new operators actually produce a family of new crossover operators and they are based upon the notion of a global precedence among genes independent of any chromosome. They were able to

achieve superior results when compared to the traditional crossover operators. This gave us an insight of crossover operator that might be useful in our approach.

A Parallel approach was proposed by Jean Berger, Mohamed Barkaoui in their research. A parallel hybrid genetic algorithm to address the vehicle routing problem with time windows [6] was presented. Their approach involved parallel co-evolution of two populations. The total traveled distance is minimized in the first population while the second population minimizes the temporal constraint violation to generate a feasible solution. They also proposed new genetic operators at that time incorporating key concepts from promising techniques such as insertion heuristics, large neighborhood search and ant colony systems to intensify the search. They proved that their method matches or outperforms the best-known heuristic routing procedures.

Jean Francis Cordeau et al (April 2004) reviewed some of the best known meta heuristics proposed for the vehicle routing problem which are based on local search, on population search and on learning mechanisms. The population search includes a large number of variants known as genetic algorithms and memetic algorithms [7]. Their conclusion was out of all the Meta heuristics that have been put forward for the solution of VRP, population search and local search combinations provides better results. This helped us to choose genetic algorithms as a heuristic for our school bus routing problem in Joint Powers Regions.

In a GA Based Heuristics for the vehicle routing problem with Multiple Trips [8], S.Salhi and R.J.Petch worked on a classical vehicle routing problem. In their work, they dealt with vehicles that can be assigned to more than one route within a working period. They proposed a hybrid Genetic algorithm, which uses a new non-binary chromosome

representation. They chose to use the random initial population to avoid the problem of choosing an unfit solution that does not match the solution. For the fitness function, they implemented a two stage objective function, where a primary objective is used to find minimum overtime requirement and then a secondary objective is introduced which improves the scheduling cost. This approach enabled them to use the roulette wheel selection. With this approach, they were able to prove that genetic algorithm generated a significant amount of reasonable solution to the problem in a short period when compared to the benchmarks in the literature.

These researches helped us in many ways including the selection the Genetic algorithm as an effective heuristic approach, representation of chromosomes, identification of a better fitness function, crossover operators which performed better than traditional crossover and tournament selection of the best fit chromosomes.

# CHAPTER 3. THE METHODOLOGY

In this chapter, we discuss the proposed GA approach, the chromosome design, the Genetic algorithm and the steps, Fitness value and the genetic operators used in the algorithm.

## 3.1. The Approach

When the program is initiated, the data regarding the location of the student, school, buses, traveling speed of the bus, capacity of the bus, opening time of the school, the school which the student attends all are loaded into the program from the database. Then the chromosomes are generated with the structure described as in section 3.3. Once a random set of chromosomes are generated, the genetic algorithm in section 3.2 is applied and the results are written back to the database.

## 3.2. Genetic Algorithm

The main steps that are used in our Genetic algorithm are given below. Step 1 involves the generation of an initial population of chromosomes. In Step 3, new chromosomes are created using the genetic operators. Based on the fitness function, the chromosomes from the current population are selected and moved to the next generation. When the population limit is reached the step 2 is repeated again until we get the optimized results.

[Step 1]: A random population of chromosomes is created.

[Step 2]: Compute the fitness value as defined in section 3.4

[Step 3]: Generate a new population using the crossover (as defined in section 3.5.1), mutation (as defined in section 3.5.2) genetic operators and cloning.

[Step 4]: Using tournament selector (as defined in section 3.6), best chromosomes whose fitness is equal to or lesser than the computed fitness value are selected and placed in new population.

[Step 5]: Compute the new fitness value. If the current population's overall fitness value is greater than the new fitness value, repeat Step 2 through Step 5.

[Step 6]: When the current population's fitness value is equal to the new fitness value for more than 20 generations, stop the genetic optimization and save the results to the database.

### 3.3. Chromosome Design

The chromosome that we use in the genetic algorithm is a representation of a solution. Each chromosome has several genes in them. Each gene in turn represents a bus route with schools and students as allele. Figure 3.1 shows the chromosome representation.

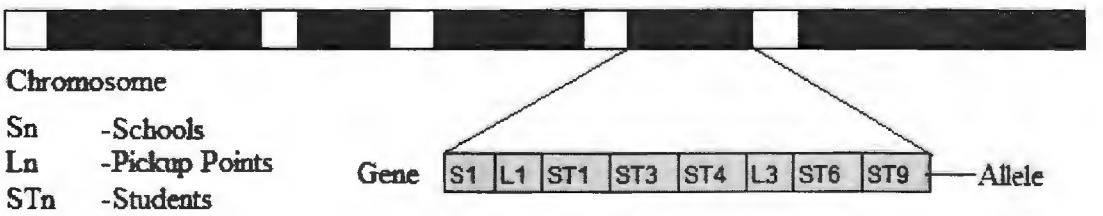



Figure 3.1 Chromosome Representation

### 3.4. Fitness Value

A Fitness value is a measure that is used to determine the feasibility of a chromosome to survive over to next generation. Every chromosome will have a fitness value assigned to it based on the constraints of the problem. Every student must reach the school on time, the number of students serviced by a given bus should not exceed its capacity, the time traveled by a student should not exceed 60 minutes and the bus should maintain its speed limit are the basic constraints that are used to calculate fitness errors. If

any one of these constraints are violated then the fitness value of that chromosome increases to 10^6 which will help us to identify that the particular chromosome. When there are no errors identified then the fitness value is calculated using the following formula:

Fixed Cost + (Variable Cost * Distance)

Chromosomes with less fitness value are the best chromosomes since the fitness value of the chromosome is the actual cost of the routes and will be selected to move on to the next generation.

### 3.5. Genetic Operators

The genetic operators that are used in our algorithm are crossover operator and mutation operators. These operators help in generating new offspring from the current population. The crossover operator generates a new offspring by exchanging substrings between two best parent chromosomes. The mutation operator generates a new chromosome by altering the alleles in the genes of a chromosome.

#### 3.5.1. Crossover

The crossover in this genetic algorithm happens at the default rate defined in the JGAP. A dynamic crossover rate is determined by JGAP on the fly since there is no provision of a fixed rate. The crossover rate is set to 98% or above by JGAP depending on the mutation rate chosen by the user. Using tournament selector two best parents from the current population are selected as the first step of crossover. Then it chooses two random crossover points. Genetic material from Parent A is copied before the first crossover point. Gene from Parent B is copied in the middle. The rest of the genetic material is again copied from Parent A thus forming a new offspring which now has genetic material from both Parent A and Parent B. Figure 3.2 depicts this scenario.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parent A: | S1 | L1 | ST1 | ST3 | | S2 | L2 | ST2 | ST4 | | S3 | L3 | ST5 | ST7 |
| Parent B: | S4 | L5 | ST6 | ST8 | | S5 | L6 | ST9 | ST11 | | S6 | L7 | ST12 | ST10 |
| Offspring: | S1 | L1 | ST1 | ST3 | | S5 | L6 | ST9 | ST11 | | S3 | L3 | ST5 | ST7 |

initial and final segment of Parent A
middle segment of Parent B

S represents Schools
L represents Locations
ST represents Students

Figure 3.2 Crossover Representation

### 3.5.2. Mutation Operators

The mutation operator helps in obtaining a better solution by altering the gene values in a chromosome and helps preventing the population from stagnating at any local optima. The mutation rate is provided as an input parameter. This variable is taken into consideration and the mutations are performed according to this user-defined mutation probability rate. This value is usually set to a low percentage. If it is set too high, the search will turn into a primitive random search [9]. The following mutation operators were used in our genetic algorithm.

#### 3.5.2.1. Pass Location

This mutation operator chooses a chromosome and within that chromosome, it picks two genes, selects a pickup point with students in one gene and passes it to another gene. By doing this mutation, it helps reducing the cost of the route for the first gene. In Figure 3.3, gene A and gene B are shown before pass location mutation.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Gene A | B2 | L2 | ST1 | ST3 | L9 | ST2 | ST4 | L3 | ST5 | ST7 |
| Gene B | B4 | L5 | ST6 | ST8 | L6 | ST9 | ST11 | L7 | ST12 | ST10 |

Figure 3.3 Genes before Pass Location Mutation

Figure 3.4 shows the mutated gene A and gene B.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Gene A | B2 | L2 | ST1 | ST3 | L2 | ST2 | ST4 | | | | | | |
| Gene B | B4 | L5 | ST6 | ST8 | L6 | ST9 | ST11 | L7 | ST12 | ST10 | L3 | ST5 | ST7 |

Figure 3.4 Genes after Pass Location Mutation

### 3.5.2.2. Swap Student

Swap student mutation operator also chooses a chromosome and within that chromosome, it picks two genes. It then picks a student from gene A and swaps that student for another student from gene B. Figure 3.5 and Figure 3.6 depicts this operation.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Gene A | B2 | L2 | ST1 | ST3 | L9 | ST2 | ST4 | L3 | ST5 | ST7 |
| Gene B | B4 | L5 | ST6 | ST8 | L6 | ST9 | ST11 | L7 | ST12 | ST10 |

Figure 3.5 Genes before Swap Student Mutation

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Gene A | B2 | L2 | ST12 | ST3 | L2 | ST2 | ST4 | L3 | ST5 | ST7 |
| Gene B | B4 | L5 | ST6 | ST8 | L6 | ST9 | ST11 | L7 | ST1 | ST10 |

Figure 3.6 Genes after Swap Student Mutation

### 3.5.2.3. Swap Location

Swap location mutation operator works exactly the same way as swap student mutation. Instead of swapping students, the swap location mutation swaps two locations. Figure 3.7 and Figure 3.8 depicts this operation.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Gene A | B2 | L2 | ST1 | ST3 | L9 | ST2 | ST4 | L3 | ST5 | ST7 |
| Gene B | B4 | L5 | ST6 | ST8 | L6 | ST9 | ST11 | L7 | ST12 | ST10 |

Figure 3.7 Genes before Swap Location Mutation

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Gene A | B2 | L2 | ST1 | ST3 | L2 | ST2 | ST4 | L6 | ST5 | ST7 |
| Gene B | B4 | L5 | ST6 | ST8 | L3 | ST9 | ST11 | L7 | ST12 | ST10 |

Figure 3.8 Genes after Swap Location Mutation

## 3.6. Tournament Selector

Tournament Selector is used in our genetic algorithm in order to select parents that will be used in the next generation for performing crossover and mutations. The tournament selection strategy is a fitness based selection process that selects a set of n individual randomly from a given population. In our approach, we select 20 individuals at

any given time and move them to the next generation. To ensure the quality of the chromosome this elite model is incorporated where the current best solution from the previous generation is moved on into the new generation.

# CHAPTER 4. THE GRAPHICAL USER INTERFACE

This chapter explains the graphical user interface, the underlying database tables, and some additional details of the software that were used to build the interface. The program was designed using JAVA, JGAP java genetic algorithm package and Microsoft Access as the underlying database for storing the initial input for the program and the results after optimization. The graphical user interface that was created for this problem allows a user to select an input Microsoft Access file and perform the genetic optimization. Once the genetic optimization completes, the program writes the results back to the same Microsoft Access database file for further analysis.

## 4.1. JAVA

Until today, the Java platform has been used in every major industry segment and has a presence in a wide range of devices, computers and networks. Java has the most versatility, efficiency and portability. It enables developers to write software on one platform and run it on virtually any other platform, create programs to run within a web browser and web services, combine applications or services using the Java language to create highly customized applications or services and more.

Some of the Technical advantages of Java are as follows [10]. Java is object oriented because programming in java is based on creating objects, manipulating objects and making objects work together. This enables us to create modular programs and reusable codes. Java is interpreted meaning an interpreter is needed to run Java programs. The programs are compiled into Java Virtual Machine code called byte code that can be run on any machine that has a java interpreter. Java is robust; its compilers are able to

detect many problems that would first show up during execution time in other languages. Its multithreaded capability allows us to perform several tasks simultaneously within a program. In java, multithreaded programming has been smoothly integrated into it while in other programming languages, operating system specific procedures have to be called in order to enable multithreading.

## 4.2. JGAP

The JGAP [11] is a genetic algorithms and genetic programming component provided as a Java framework. It provides basic genetic mechanisms that can be easily used to apply evolutionary principles to problem solutions. JGAP is designed to be very flexible and pluggable. It allows users to create own genetic operators, random number generators, natural selectors, fitness function and so on. To support all of this JGAP uses a configuration object that must be setup with all of the settings needed and it includes a default configuration class that comes already setup with most common settings. With this setup, it only needs the fitness function that needs to be used, the chromosome set up and the number of chromosomes in the population.

## 4.3. MS Access

MS Access [12] is a self-styled relational database management that can store data in specific formats for sorting, querying and reporting. A relational database management system allows users to create multiple tables that can be linked together via one or more shared field values between tables. Access will also allow the tables to be interrelated via forms, reports or queries. Access can use structured query language (SQL) to query the tables. MS Access operates the same way as other products such as Microsoft SQL server, Oracle, MySQL or IBM DB2 and many of the SQL statements that work properly in

Access could also be used in these other products without any modification. Since Access is a Microsoft product it is quite simple to import data from excel spreadsheet into access tables. It can also import data from a wide range of other sources.

## 4.4. The Tables

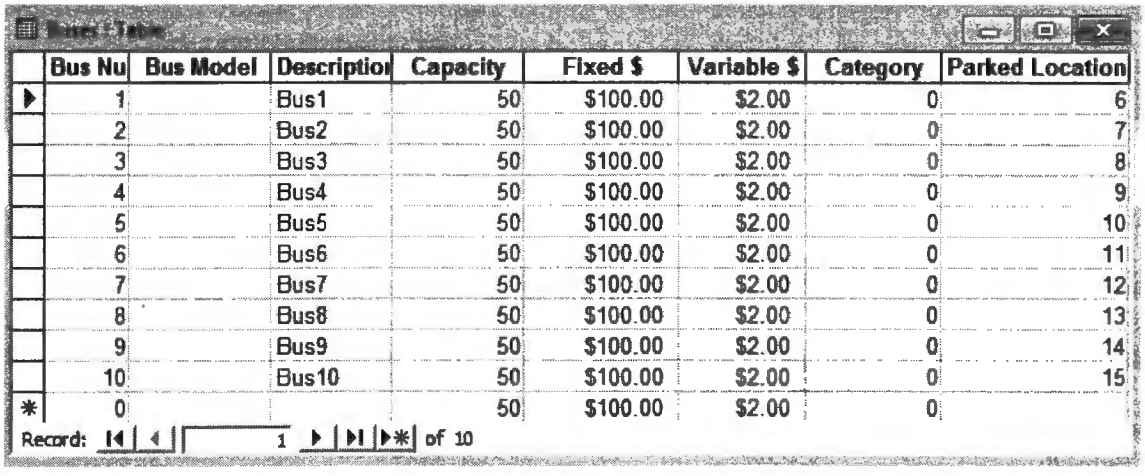
The input data for this genetic algorithm routing is stored in MS Access database. Various inter related tables are used to store information about the schools, students, buses, location of the schools, location of the students, location of the buses, bus traveling speed, number of chromosomes in the current generation, category of the bus either if it is a special category bus or a regular category bus and so on. In the sections to come, the following tables are explained in detail.

- Buses
- Categories
- Locations
- Parameters
- Schools
- Students

### 4.4.1. Buses

The Buses table is used to store the information related to all the buses that are available to serve the students. The Buses table has Bus Number, Bus Model, Description, Capacity, Fixed Cost, Variable Cost, Category and Parked Location. The Bus Number can be the vehicle registration number or any number that is used by the school to identify the bus. The Description and the Bus Model can be used to enter additional details about the bus in order to identify it more quickly by a user. The Capacity column stores the vehicle

total capacity including the driver. The Fixed cost stores the amount of cost it would need to use the bus. The Variable cost stores the cost of operation per mile. The Category has the category of the bus either if it serves regular students or special students. The Parked Location Column has the location id, where the bus is being parked or garaged during the night. A sample buses table is displayed in Figure 4.1



| Bus Nu | Bus Model | Description | Capacity | Fixed $ | Variable $ | Category | Parked Location |
|---|---|---|---|---|---|---|---|
| 1 | | Bus1 | 50 | $100.00 | $2.00 | 0 | 6 |
| 2 | | Bus2 | 50 | $100.00 | $2.00 | 0 | 7 |
| 3 | | Bus3 | 50 | $100.00 | $2.00 | 0 | 8 |
| 4 | | Bus4 | 50 | $100.00 | $2.00 | 0 | 9 |
| 5 | | Bus5 | 50 | $100.00 | $2.00 | 0 | 10 |
| 6 | | Bus6 | 50 | $100.00 | $2.00 | 0 | 11 |
| 7 | | Bus7 | 50 | $100.00 | $2.00 | 0 | 12 |
| 8 | | Bus8 | 50 | $100.00 | $2.00 | 0 | 13 |
| 9 | | Bus9 | 50 | $100.00 | $2.00 | 0 | 14 |
| 10 | | Bus10 | 50 | $100.00 | $2.00 | 0 | 15 |
| 0 | | | 50 | $100.00 | $2.00 | 0 | |

Record: 1 of 10

Figure 4.1 Buses Table

### 4.4.2. Categories

There are times when a special category bus is needed to serve special students. The Categories table in our database is used to store information about the categories that is used to classify the buses and students. There are two main categories – Regular and Wheel chair category buses. The Regular buses are used to serve normal students and the Wheel chair category buses are used to serve special students. Figure 4.2 is the snap shot of the categories table from the database.

The genetic algorithm defined in the program finds the near best solution for special category students by generating a separate population of chromosomes. This happens in a sequential pattern. The algorithm defined in section 3.2 is executed for special category students and then the same algorithm is executed afterwards for regular category students.

The Categories table defined in Figure 4.2 helps us to achieve the sequential execution of the genetic algorithm by assigning the category id to the special category students and the special category buses.

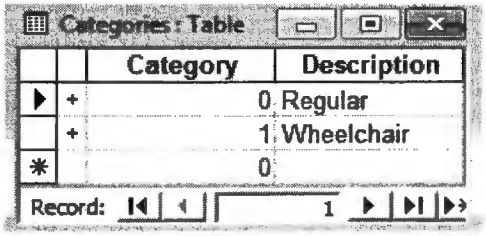



Figure 4.2 Categories Table

### 4.4.3. Locations

The Location table is used to store the Cartesian coordinates of the schools, buses and pick up points. Each entry is given a location id that is used to relate the row in other tables. The location id is stored in the Location Id Column. We represent the map as a two-dimensional plane that enables us to use the X coordinate and the Y coordinate to identify a location. In other words the X coordinate mimics the latitude representation of a location and the Y Coordinate mimics the longitude representation of a location. Hence, we have the X coordinate of the location stored in the X Coordinate Column and the Y coordinate of the location stored in the Y Coordinate Column. The Description column is used to store the description of the location in our case it is the Schools, Buses and Pickup Points. The location coordinate of a bus indicates where the bus is parked during the nighttime. It can be either a driver's house or a common depot. The REAs encourage drivers to park the buses in their houses for better performance and to save time and depot rental cost. A pick point is the meeting location for a small set of students close to their house. Buses pick and drop students at these locations and schools only. Figure 4.3 is the snap shot of the locations table from the database.

Locations : Table

| Location Id | X Co ordinate | Y Co ordinate | Description |
|---|---|---|---|
| 1 | 3.45 | 5.92 | School1 |
| 2 | 4.96 | 3.9 | School2 |
| 3 | 4.23 | 1.77 | School3 |
| 4 | 2.25 | 6.63 | School4 |
| 5 | 0.95 | 0.62 | School5 |
| 6 | 6.79 | 0.85 | Bus1 |
| 7 | 1.57 | 2.73 | Bus2 |
| 8 | 1.7 | 3.13 | Bus3 |
| 9 | 0.37 | 5.35 | Bus4 |
| 10 | 1.45 | 2.41 | Bus5 |
| 11 | 4.99 | 1.73 | Bus6 |
| 12 | 0.01 | 0.9 | Bus7 |
| 13 | 4.7 | 1.94 | Bus8 |
| 14 | 4.18 | 3.72 | Bus9 |
| 15 | 6.31 | 5.09 | Bus10 |
| 16 | 5.91 | 5.4 | Pickup Point16 |
| 17 | 4.82 | 1.64 | Pickup Point17 |

Record: 1 of 35

Figure 4.3 Locations Table

### 4.4.4. Parameters

The Parameters table is used to store the important constraints that are involved in the problem and the genetic parameters. The Bus Speed and the Maximum Travel Time columns contains the maximum travel speed (in miles per hour) that a bus is allowed to travel, in this case it is 35 miles per hour and the maximum time (in minutes) that a bus is allowed to travel, in this case it is 60 minutes. The maximum number of evolutions that should happen in the process of finding the best solution is evaluated by the program when the previous generation's fitness function remains the same for the next generations at least 20 times. The total number of chromosomes in each generation is stored in the Chromosomes column. In this case, it will be 300 chromosomes for each generation. Figure 4.4 shows the parameters table from the database.

Parameters : Table

| | Id | Bus Speed | Max Travel Time | Chromosomes |
|---|---|---|---|---|
| ▶ | 0 | 35 | 60 | 300 |

Record: 1 of 1

Figure 4.4 Parameters Table

### 4.4.5. Schools

The Schools table has the list of schools that the students attend. These are the only schools where the buses will drop the students. The School Id column identifies each school with a unique Id. The Location Id column has the Id column from the location table. In this case for School 1 the Location will be 3.45 X and 5.92 Y. The School 1 opens at 8 am in the morning. This information is stored in the Open Time column. The School Name Column is used to store the name of the school. Figure 4.5 is a snap shot of the schools table from the database.

Schools : Table

| | School Id | Location Id | Open Time | School Name |
|---|---|---|---|---|
| ▶ + | 1 | 1 | 8:00 | School1 |
| + | 2 | 2 | 8:00 | School2 |
| + | 3 | 3 | 8:00 | School3 |
| + | 4 | 4 | 8:00 | School4 |
| + | 5 | 5 | 8:00 | School5 |
| * | 0 | | 8:00 | |

Record: 1 of 5

Figure 4.5 Schools Table

### 4.4.6. Students

The Students table contains all the required information about the students. The Student Id column stores the identification number of a student. The Student Id also serves the purpose as a unique identifier. The First Name and Last Name columns store the first name or given name of a student and the last name or family name of a student

respectively. Figure 4.6 is a snap shot of the students table in the database.

Students's Table

| Student Id | First Name | Last Name | School Id | Loc Id | Category |
|---|---|---|---|---|---|
| 1 | Aidan | Sandler | 1 | 16 | 0 |
| 2 | Betty | Jackson | 5 | 23 | 0 |
| 3 | Caleb | Schaffer | 5 | 33 | 0 |
| 4 | Deborah | Everts | 5 | 24 | 0 |
| 5 | Elvis | Dangar | 2 | 25 | 0 |
| 6 | Bethany | Knowles | 4 | 19 | 0 |
| 7 | Griffin | Tilton | 5 | 19 | 0 |
| 8 | Hannah | Madison | 4 | 20 | 0 |
| 9 | Ike | Dexter | 3 | 34 | 0 |
| 10 | James | Hanley | 3 | 33 | 0 |
| 11 | Krista | Johnson | 4 | 23 | 0 |
| 12 | Lucas | Morrison | 5 | 32 | 0 |
| 13 | Mackenzie | Trevor | 1 | 34 | 0 |
| 14 | Nadia | Davis | 4 | 18 | 0 |
| 15 | Orlando | Tennyson | 3 | 24 | 1 |
| 16 | Pearl | Madison | 5 | 30 | 0 |
| 17 | Quinton | Joyce | 3 | 29 | 0 |
| 18 | Rhiannon | Sands | 1 | 30 | 1 |
| 19 | Shakira | Towers | 2 | 27 | 1 |
| 20 | Troy | Delves | 2 | 22 | 0 |
| 21 | Ulysses | Yager | 1 | 26 | 0 |

Record: 21 of 80

Figure 4.6 Students Table

The School Id holds the Id field from the schools table. This helps us to keep track of which school that a particular student is attending and where he or she has to be dropped by the school bus. The Location Id column stores the pick point location where the student will be waiting to be picked. The Category column is used to store whether the student is a regular student or a special child. This information will be used to assign the appropriate type of school bus to that student. Based on the data available in the table, the Student Aidan Sandler attends School 1 (as per Schools table), he will be waiting at the location 16 (X Coordinate 5.91, Y Coordinate 5.4 as per Locations table) and he needs a regular bus since his category says 0 which is Regular Student as per the Categories table.

## 4.5. The Graphical User Interface

The Graphical User Interface designed for our problem has three sections or panels.

- Control Panel
- Display Panel
- Log Panel

Figure 4.7 shows the snap shot of the graphical user interface for school bus routing.

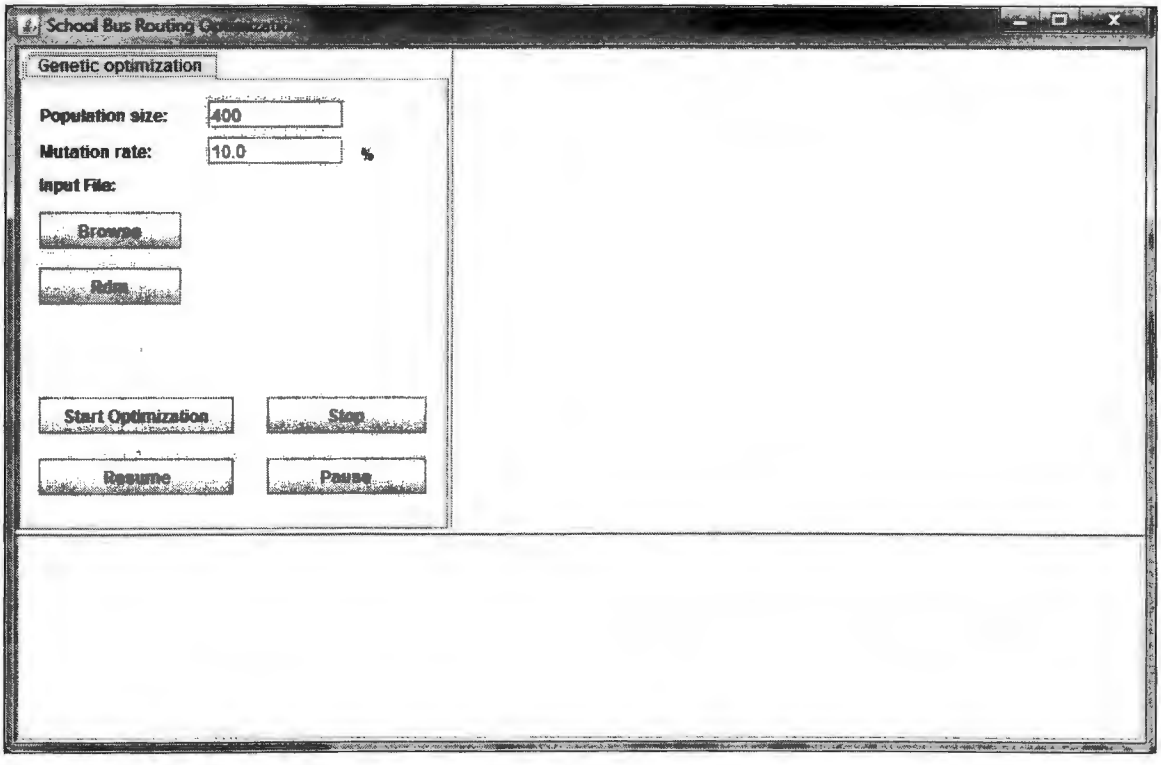



Figure 4.7 The Graphical User Interface for School Bus Routing

The left most section in Figure 4.7 is called the Control panel where the ability to control the program is given to a user. The right section in Figure 4.7 is called the Display Panel where the progress of the problem is displayed. The third section that is located below the left and right section is called the Log panel. The Log panel displays a detailed log of the routes with students, schools and the buses that serve them.

### 4.5.1. Control Panel

The Control panel allows the user to enter some genetic parameters for the problem, select an input file for the problem and allows the user to start, stop, pause or resume the program. The Population Size parameter in the control panel is set to 400 by default. This overrides the number of chromosomes parameter that is set in the Parameters table in the database. The mutation rate is set to 10 percent by default. This can be changed to any value as desired. It is recommended that the mutation level should be set to a lower percentage for better-optimized results. Figure 4.8 shows the default values that are assigned when the program is opened.

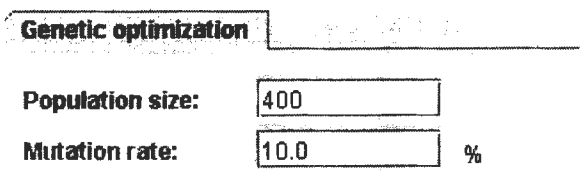



Figure 4.8 Genetic Parameters with Defaults

The user will then use the Browse button in the control panel to browse and select the input database file that will contain all the necessary data and parameters for the program to run. Once the input file is selected, the complete path of the database file will be displayed in the Control panel. The Rdm (Random) Button is used for testing purposes only. It helped us to create random data that was used to perform the tests that are discussed in detail in the following chapter.

Figure 4.9 shows the browsing window through which a user can input the database file. Once the input file is selected, the user should start the optimization by pressing the Start Optimization button. The user also has the ability to stop the program in the middle, pause or resume the program. Once the program is stopped the results that are optimized at that point of time is written into the database that will be shown in the Control panel as

'Saving Results'. The Total Cost and the Total Distance traveled will also be shown in the Control Panel. Figure 4.10 shows the snap shot of the control panel with the discussed status messages.

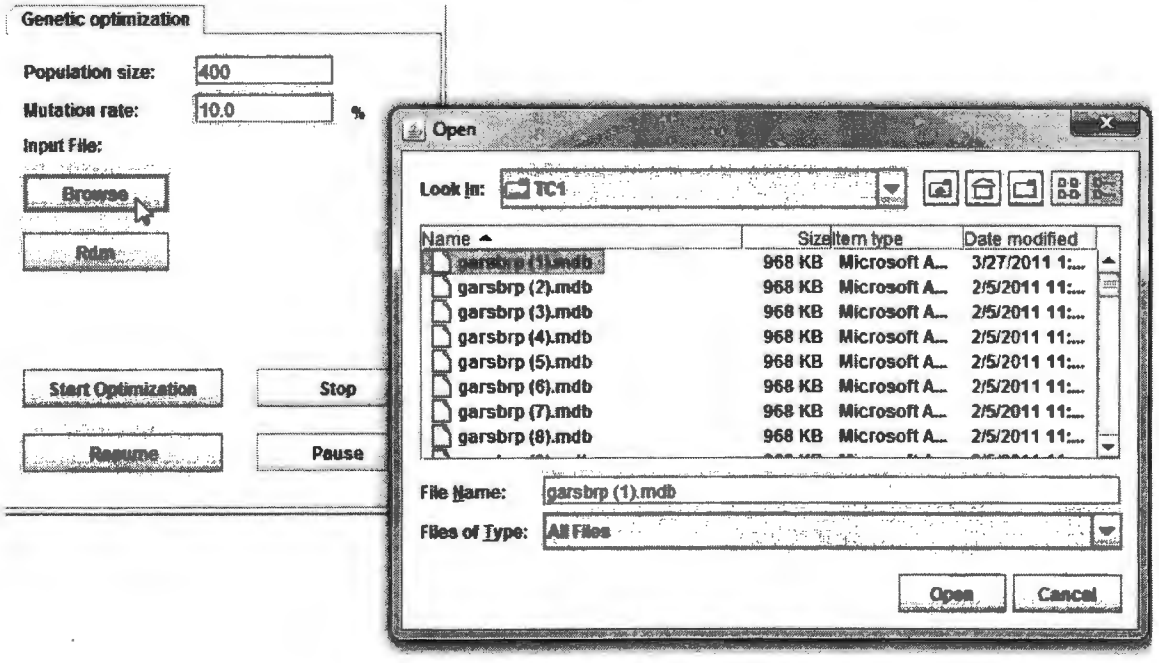



Figure 4.9 Browsing Window

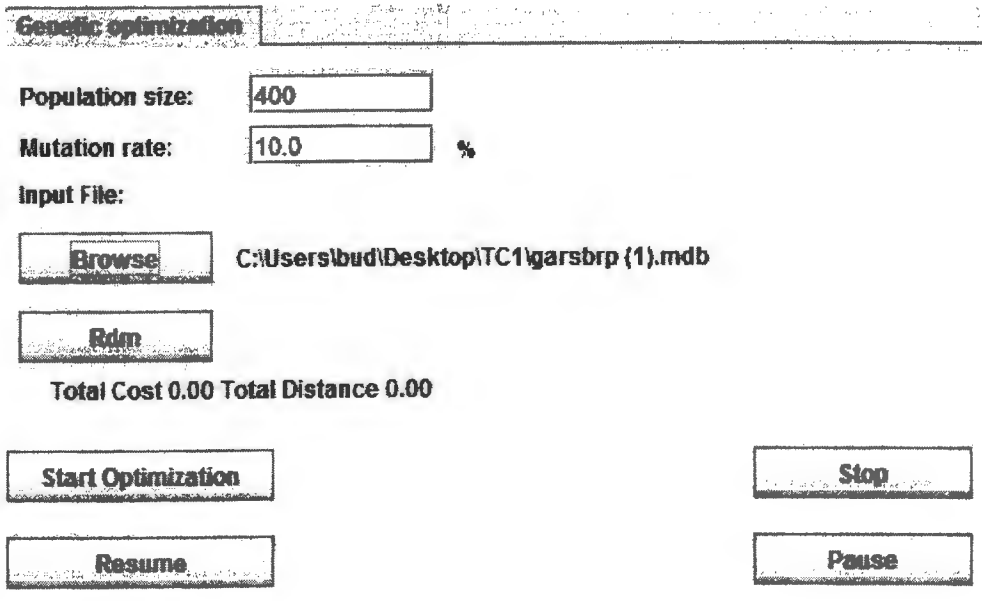



Figure 4.10 Control Panel with Message

### 4.5.2. Display Panel

The Display panel displays the progress of the problem. The Schools, pickup points and the buses' initial location are displayed when the data is loaded. School n represents schools where n ranges from 1 to Total Number of Schools and Bus n represents buses where n ranges from 1 to Total Number of Buses available. Once the program is initiated the initial solution of routes are generated and lines are drawn to show the routes to the user. Figure 4.11 is the icon glossary of the display panel and Figure 4.12 shows the display panel after loading the data into the Program.

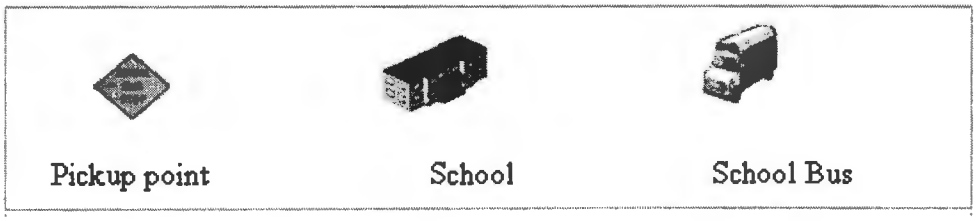



Figure 4.11 Images used in Display Panel

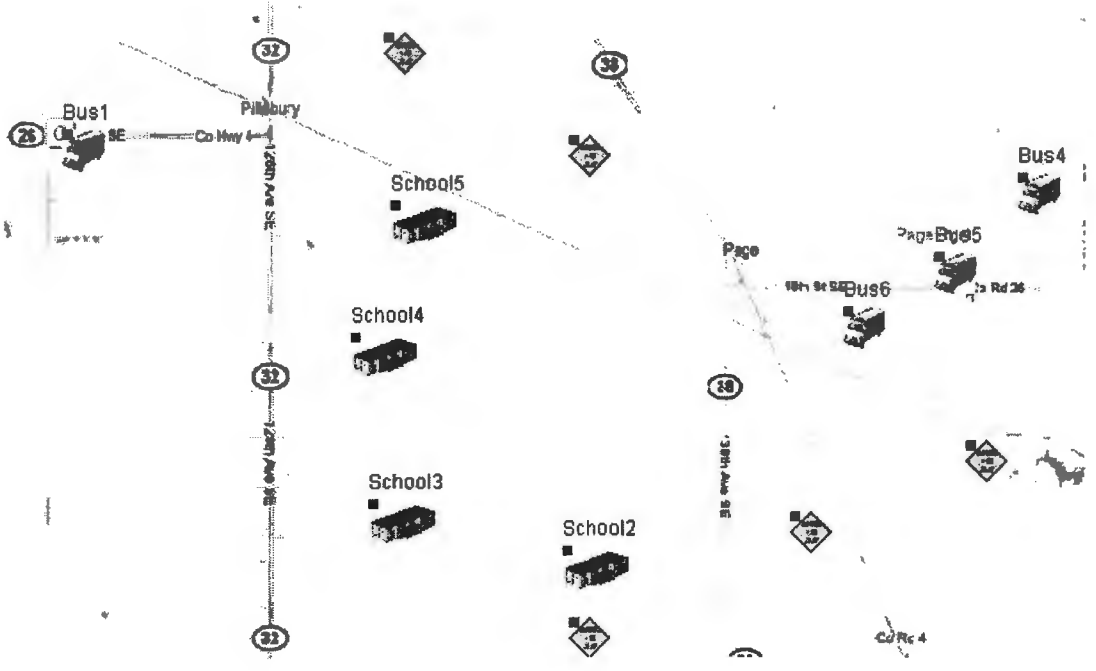



Figure 4.12 Display Panel after Loading

The line Starts from a Bus location and connects all the students that it will serve and then finally ends at a school. The lines keep shifting from one location to another as per the optimized route. In each generation, we will see the lines are drawn as per the new optimized routes. Once the program finds the best solution the lines will stop shifting locations, this is when the program writes the output to the database.

Figure 4.13 shows the snap shot of the program while performing the optimization.

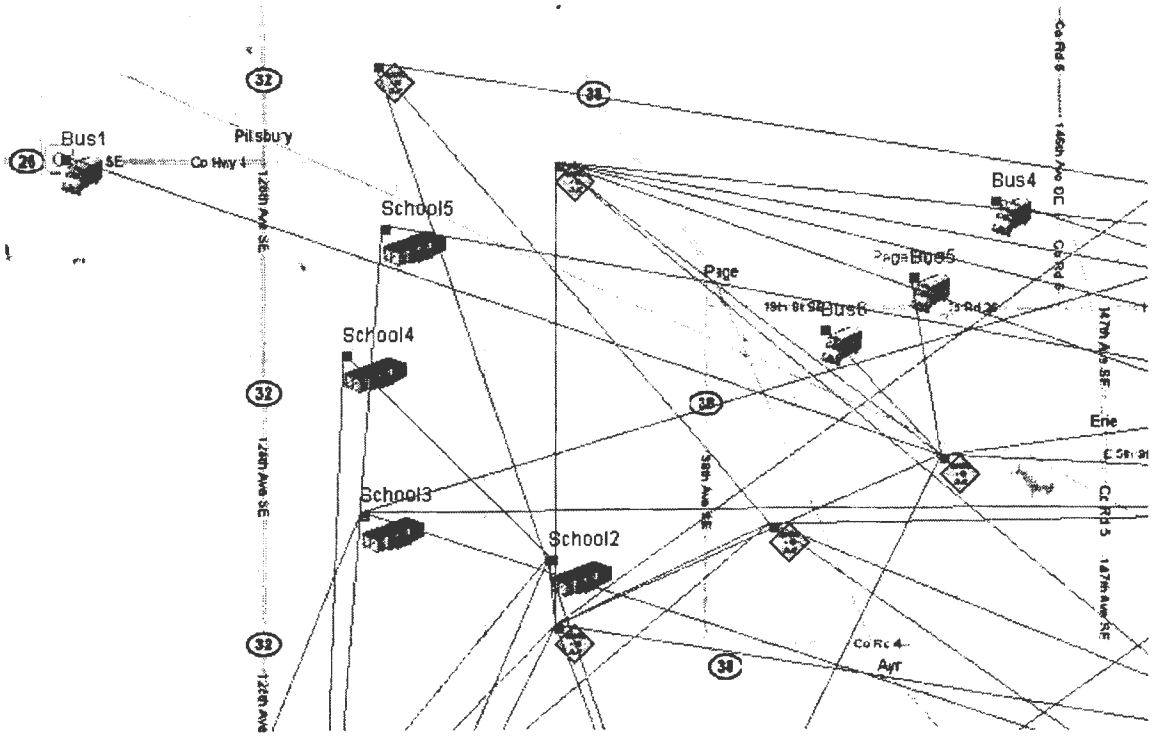



Figure 4.13 Display Panel while Performing the Optimization

## 4.5.3. Log Panel

The Log panel displays the detailed routes that are being generated during the optimization. It captures the total fitness, total distance of all the buses used in the routing, total cost of each bus, total distance traveled by each bus, the time when the bus picks the students, the location from where the bus picks the students and which students it picks.

The log panel also displays the change in routes as the program continues to perform the optimization. Figure 4.14 is a snap shot of the log panel while performing the genetic optimization.

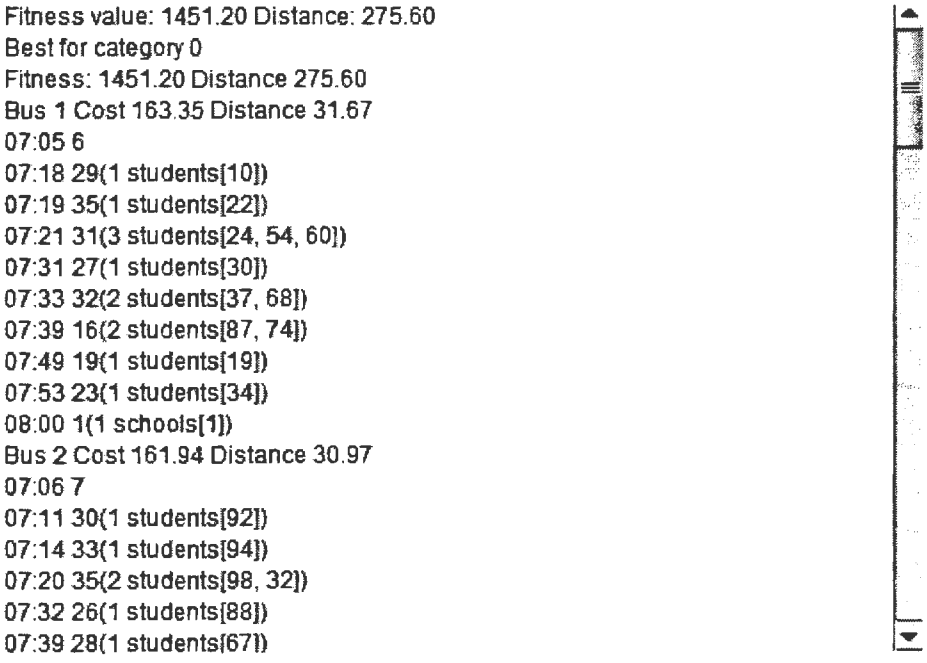



Figure 4.14 Log Panel while Performing the Optimization

## 4.6. The Output Tables

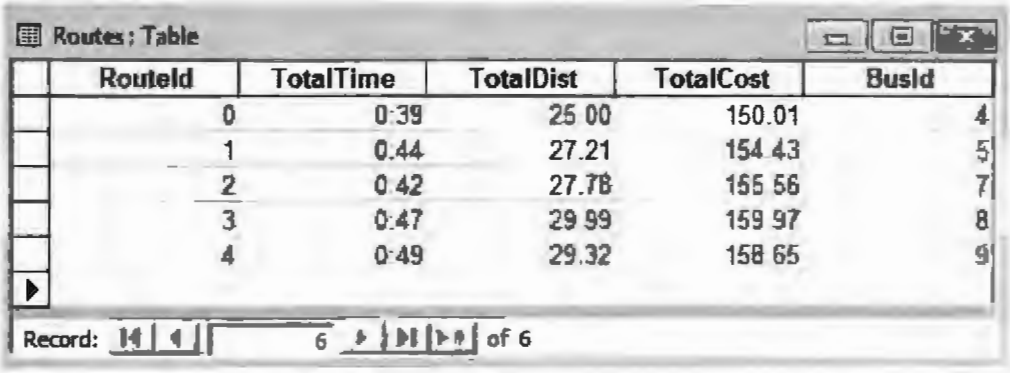
The output data after the genetic optimization is written back the Microsoft Access database file. The tables that are used to store the results have information like the bus id, location id, student id, school id, category id, time at the location, total distance of a particular route, total cost of the route and total travel time.

The tables that will be available in the database with the above-mentioned information are

- Routes
- Detail Routes

### 4.6.1. Routes

The Routes table that is shown in Figure 4.15 will have high-level information of the optimized routes.


Routes : Table

| RouteId | TotalTime | TotalDist | TotalCost | BusId |
|---|---|---|---|---|
| 0 | 0:39 | 25.00 | 150.01 | 4 |
| 1 | 0.44 | 27.21 | 154.43 | 5 |
| 2 | 0.42 | 27.78 | 155.56 | 7 |
| 3 | 0:47 | 29.99 | 159.97 | 8 |
| 4 | 0:49 | 29.32 | 158.65 | 9 |

Record: 6 of 6

Figure 4.15 Routes Table

A user can infer the total number buses used for serving the students, using this table. The number of records in this table is the total number of buses used for the routing. The total travel time for each bus is shown in minutes under Total Time column. The total distance traveled by each bus is shown under the Total Distance column and the total operating cost of the bus is shown in the Total Cost column. The bus that will be serving that particular route is stored in the Bus Id column.

### 4.6.2. Detail Routes

The Detail Routes table will have the routes information at the student level. For each student this table will have an entry and for each school this table will have an entry.

For example in Figure 4.16 the Bus with Bus Id 4 serves 16 students in total, picking up the first two students at 7.20 am, continues to pick the rest of the students and reaches the school on time at 7.59 am.

DetailRoutes : Table

| RecordId | RouteId | BusId | LocId | StudentId | School | CategoryId | Time |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 4 | 27 | 28 | | 0 | 7:20 |
| 1 | 0 | 4 | 27 | 17 | | 0 | 7:20 |
| 2 | 0 | 4 | 24 | 12 | | 0 | 7:25 |
| 3 | 0 | 4 | 35 | 11 | | 0 | 7:31 |
| 4 | 0 | 4 | 29 | 6 | | 0 | 7:33 |
| 5 | 0 | 4 | 28 | 18 | | 0 | 7:37 |
| 6 | 0 | 4 | 23 | 42 | | 0 | 7:38 |
| 7 | 0 | 4 | 19 | 72 | | 0 | 7:40 |
| 8 | 0 | 4 | 19 | 61 | | 0 | 7:40 |
| 9 | 0 | 4 | 19 | 96 | | 0 | 7:40 |
| 10 | 0 | 4 | 25 | 1 | | 0 | 7:45 |
| 11 | 0 | 4 | 25 | 20 | | 0 | 7:45 |
| 12 | 0 | 4 | 25 | 64 | | 0 | 7:45 |
| 13 | 0 | 4 | 26 | 29 | | 0 | 7:50 |
| 14 | 0 | 4 | 20 | 71 | | 0 | 7:56 |
| 15 | 0 | 4 | 20 | 50 | | 0 | 7:56 |
| 16 | 0 | 4 | 4 | | 4 | 0 | 7:59 |
| 17 | 1 | 5 | 33 | 82 | | 0 | 7:15 |
| 18 | 1 | 5 | 33 | 75 | | 0 | 7:15 |
| 19 | 1 | 5 | 19 | 70 | | 0 | 7:22 |
| 20 | 1 | 5 | 19 | 19 | | 0 | 7:22 |
| 21 | 1 | 5 | 18 | 44 | | 0 | 7:22 |
| 22 | 1 | 5 | 18 | 14 | | 0 | 7:22 |
| 23 | 1 | 5 | 29 | 10 | | 0 | 7:26 |
| 24 | 1 | 5 | 35 | 22 | | 0 | 7:29 |
| 25 | 1 | 5 | 35 | 48 | | 0 | 7:29 |
| 26 | 1 | 5 | 17 | 99 | | 0 | 7:30 |
| 27 | 1 | 5 | 31 | 54 | | 0 | 7:31 |
| 28 | 1 | 5 | 31 | 24 | | 0 | 7:31 |
| 29 | 1 | 5 | 31 | 60 | | 0 | 7:31 |
| 30 | 1 | 5 | 16 | 87 | | 0 | 7:34 |
| 31 | 1 | 5 | 16 | 74 | | 0 | 7:34 |
| 32 | 1 | 5 | 27 | 30 | | 0 | 7:40 |

Record: 1 of 105

Figure 4.16 Detail Routes Table

# CHAPTER 5. EXPERIMENTS AND RESULTS

The program that we created for solving the School bus routing problem was run on three different data scenarios denoted as Test Case 1, Test Case 2 and Test Case 3. In each Test case, the program's ability to generate optimized routes was tested, and the results were analyzed statistically. All the tests were performed on a 2.13 GHZ Intel Core 2 Duo machine with 4GB RAM and Windows 7 operating system.

## 5.1. Test Case 1

In Test Case 1, 53 problems were run in the program and the results were obtained. Each problem had 80 students, 5 schools, 10 buses and 20 pickup points. The buses' travel speed was set to a maximum of 35 miles per hour and the maximum travel time per student was set to 60 minutes. The number of chromosomes in a single generation was set to 400 and the mutation rate was set to 2%. In all of the 53 problems it was assumed that there are no special category students and hence no special category buses. The goal of this particular test case was to compare the programs output when we use Euclidean metrics and taxicab metrics for calculating the distance between two points. The Euclidean metric is the ordinary distance between two points. For example the Euclidean distance between point X1,Y1 and X2,Y2 is Square root of $(X1-X2)^2+(Y1-Y2)^2$. The Taxicab metric on the other hand is the city block distance or rectilinear distance between two points. For example, the Taxicab distance between two points X1, Y1 and X2, Y2 is $|X1-X2| + |Y1-Y2|$. The Results that we obtained proved significant importance of the taxicab metric for applications that involve real time travel. Table 5.1.1 contains the average computational results of the 53 random generated problems using the Euclidean Metrics and Table 5.1.2

contains the average computational results of the 53 random generated problems using Taxicab Metrics.

| Problems solved with Euclidean Metrics | B | S | ATD | ATT | MTT | ATC | ACPU | ANG |
|---|---|---|---|---|---|---|---|---|
| Average | 5 | 80 | 114.28 | 2.58 | 3.42 | 739.89 | 1.63 | 156 |
| Standard Deviation of Total Distance | 10.74 miles | | | | | | | |
| Standard Deviation of Total Travel Time | 17 minutes | | | | | | | |
| Standard Deviation of Total Cost | 51.36 dollars | | | | | | | |

Table 5.1.1 Average Results of 53 Problems using Euclidean Metrics

| Problems solved with Taxicab Metrics | B | S | ATD | ATT | MTT | ATC | ACPU | ANG |
|---|---|---|---|---|---|---|---|---|
| Average | 5 | 80 | 146.54 | 3.43 | 4.30 | 861.00 | 2.51 | 188 |
| Standard Deviation of Total Distance | 15.55 miles | | | | | | | |
| Standard Deviation of Total Travel Time | 23 minutes | | | | | | | |
| Standard Deviation of Total Cost | 93.87 dollars | | | | | | | |

Table 5.1.2 Average Results of 53 Problems using Taxicab Metrics

B: Buses,

S: Students,

ATD: Average Travel Distance of all students in miles,

ATT: Average Travel Time of all buses in hours,

MTT: Maximum Travel Time in the 53 problem in hours,

ATC: Average Total Cost of all buses in dollars,

ACPU: Average CPU time for solving the problems in minutes,

ANG: Average Number of Generations.

In order to visually see the distribution of the data, normal distribution graphs were created for the Total Distance Traveled, Total Cost and Total Travel Time.

In Figure 5.1.1, the normal distribution of Total Distance traveled by all the buses in the 53 problems is shown.

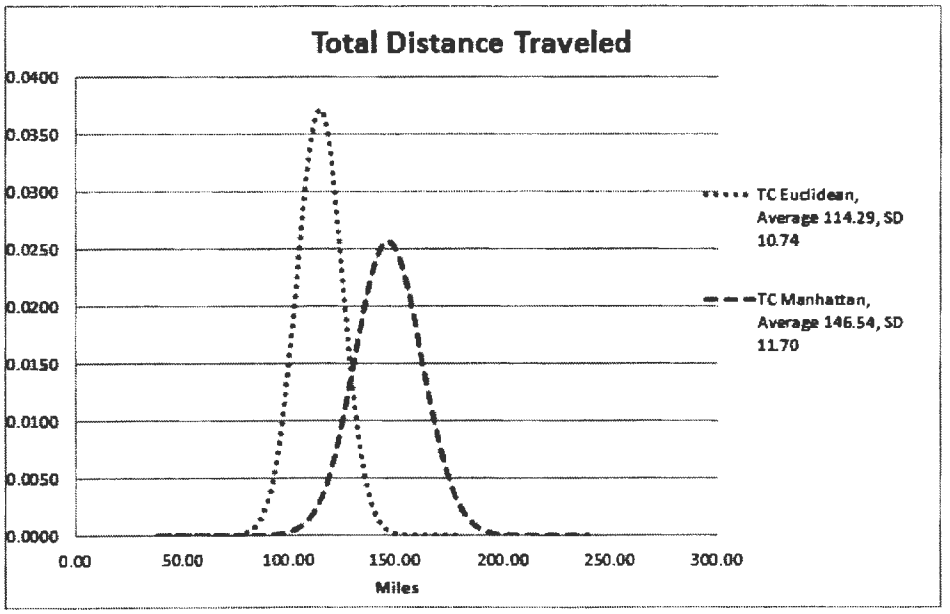



Figure 5.1.1 Normal Distribution of Total Distance Traveled using Euclidean and Taxicab or Manhattan Metrics

The blue dotted line represents the distribution of the results obtained from the program with the Euclidean metrics. The red separated line indicates the distribution of the results obtained from the program with Taxicab metrics.

In Figure 5.1.2, the normal distribution of the Total Cost of all the buses in the 53 problems is shown. The blue dotted line represents the distribution of the results obtained from the program with Euclidean metrics. The red separated line indicates the distribution of the results obtained from the program with Taxicab metrics.

Similar to the other two comparisons shown in Figure 5.1.1 and 5.1.2 in Figure 5.1.3 the normal distribution of the total travel time of all the buses is shown. When compared, the results obtained with the taxicab metric seem to be more realistic than the

results obtained through the Euclidean metrics.

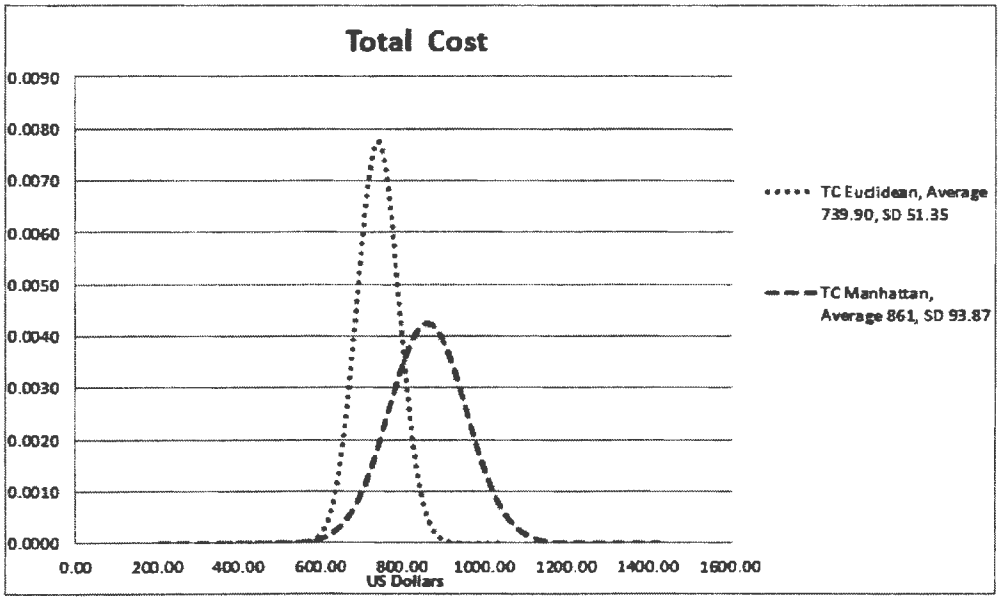



Figure 5.1.2 Normal Distribution of Total Cost using Euclidean and Taxicab or Manhattan Metrics

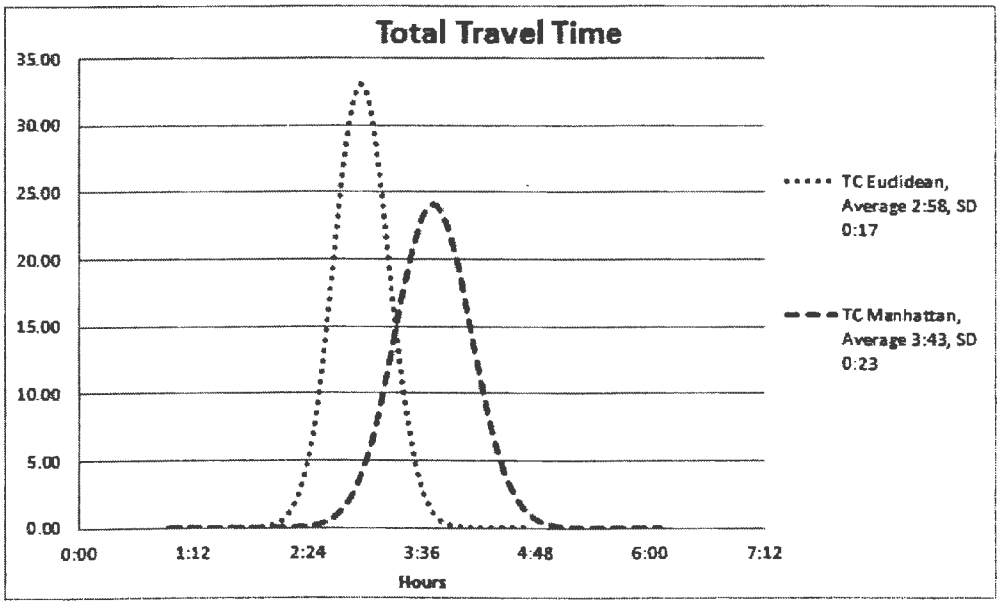



Figure 5.1.3 Normal Distribution of Total Travel Time using Euclidean and Taxicab or Manhattan Metrics

The bell curve formed by the Taxicab metric in all the three figures suggests that the results from the Taxicab metric are spread out evenly than the Euclidean metric for the same set of 53 test problems.

## 5.2. Test Case 2

In Test case 2, our interest was towards how the program handles problems with different complexities. In order to test that, three sets of 53 problems were run in the program and the results were obtained.

We have named the three sets as Test Set A, Test Set B and Test Set C. The Test Set A had 53 problems each with 100 students, 5 schools, 10 buses and 20 pickup points. The buses' travel speed was set to a maximum of 35 miles per hour and the maximum travel time per student was set to 60 minutes. The number of chromosomes in a single generation was set to 400 and the mutation rate was set to 2%.

The Test Set B had 53 problems each with 80 students (at least 5 special category students), 5 schools, 15 buses and 20 pickup points. The Test Set C had 53 problems each with 80 students, 5 schools, 10 buses and 20 pickup points. The buses' travel speed was set to a maximum of 35 miles per hour and the maximum travel time per student was set to 60 minutes. The number of chromosomes in a single generation was set to 400 and the mutation rate was set to 2%. We ran these three test sets on the program and obtained the results.

The Average total distance, average total time, maximum travel time, average total cost, average CPU time, average number of generations, standard deviation of total distance, standard deviation of total travel time and standard deviation of total cost are displayed in Table 5.2.1. In Figure 5.2.1, the normal distribution of Total Distance traveled

by all the buses in the three test sets is shown.

| | B | S | ATD | ATT | MTT | ATC | ACPU | ANG | SD TD | SD TT | SD TC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Test Set A** | 10 | 100 | 124.91 | 3.14 | 4.00 | 768.69 | 2.81 | 210 | 11.50 | 17 | 60.10 |
| **Test Set B** | 15 | 80 | 131.80 | 3.14 | 3.57 | 1069.25 | 1.56 | 143 | 11.70 | 17 | 114.50 |
| **Test Set C** | 10 | 80 | 114.29 | 2.58 | 3.42 | 739.89 | 1.63 | 156 | 10.74 | 17 | 51.36 |

Table 5.2.1 Average Results of Test Case 2

SDTD: Standard Deviation of Total Distance,

SDTT: Standard Deviation of Total Travel Time,

SDTC: Standard Deviation of Total Cost.

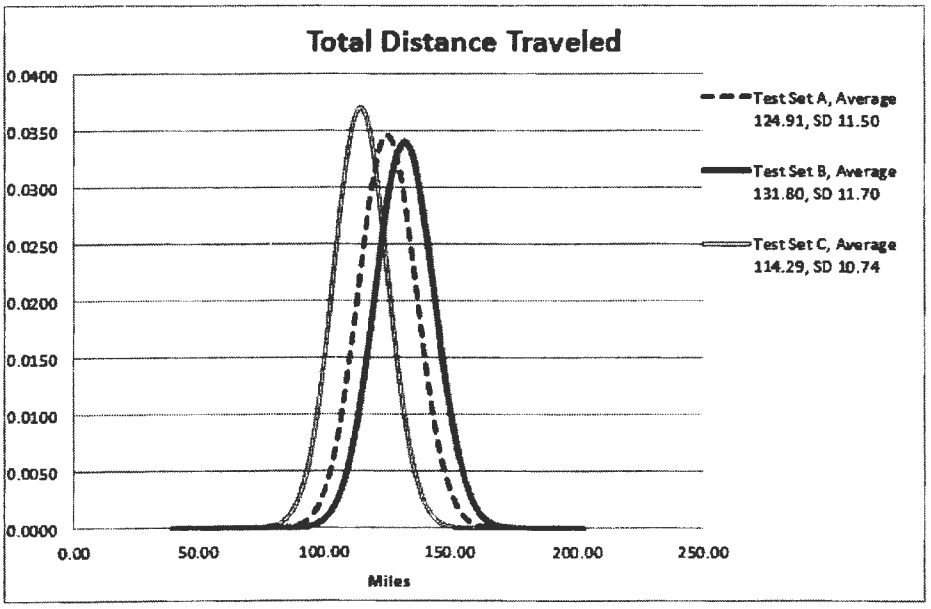



Figure 5.2.1 Normal Distribution of Total Distance Traveled for the Three Test Sets in Test Case 2

In Figure 5.2.1, the blue dotted line represents the distribution of the results obtained from the program with Test Set A. The red continuous line indicates the distribution of the results obtained from the program with Test Set B. Finally, the Green double line represents the distribution of the results obtained from the program with Test

Set C. After analyzing the distribution, it is clear that the program handles all types of complexities of different problems in a similar fashion. Although there is a shift in the distribution of Test Set A, Test Set B and Test Set C, it is an expected behavior when the amount of data it handles is different. Based on Complexity and the Size of the Problem, the three test cases can be arranged in the following order. Test Set C < Test Set A < Test Set B. Also based on the output results we can conclude that Test Set C's complexity and distribution is less when compared to Test Set A and Test Set B. Test Set A's complexity and distribution is less when compared to Test Set B and Test Set B is the most complex problem out of all three problems. In Figure 5.2.2, the normal distribution of Total Cost of all the buses in the three test sets is shown.

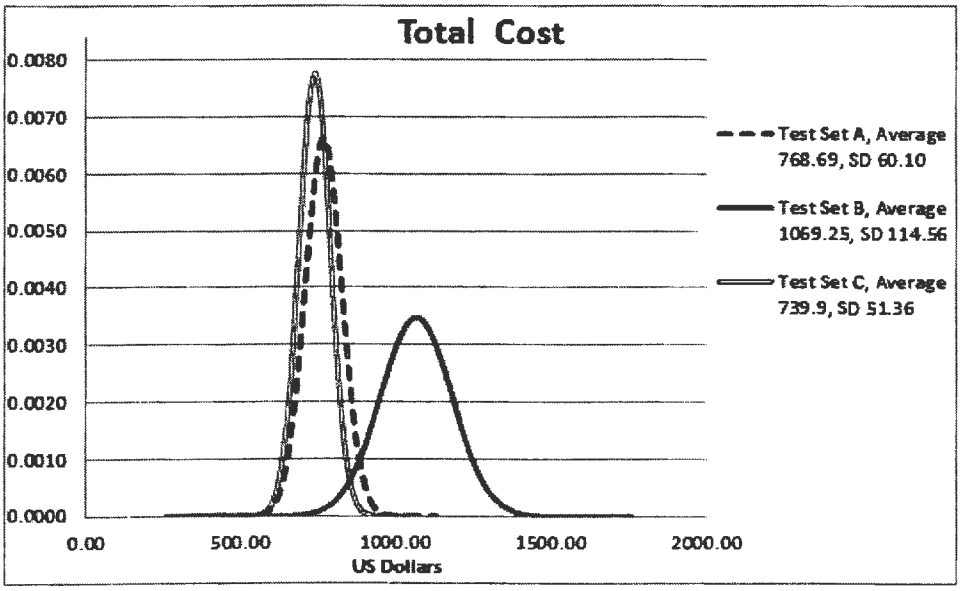



Figure 5.2.2 Normal Distribution of Total Cost for the Three Test Sets in Test Case 2

The average operating cost of all the buses in Test Set A obtained was 768.69 with a standard deviation of 60.10. The average operating cost of Test Set B obtained was 1069.25 with a standard deviation of 114.56. The reason why this came back with slightly bigger number is because this problem included special category students and they need

special category bus for transportation. The average operating cost for Test Set C obtained was 739.9 with a standard deviation of 51.36. In Figure 5.2.3, the total travel time obtained for Test Set A and Test Set B were distributed almost equally. The average travel time for Test Set A is 3 hours and 14 minutes with a standard deviation of 17 minutes and the average travel time for Test Set B is 3 hours and 14 minutes with a standard deviation of 17 minutes. The normal distribution of total travel time of all the buses in the three test sets is shown in Figure 5.2.3. The two Test Sets gave us almost equal results because they have same number of students. The Test Set C has a lesser number of students when compared to Test Set A and Test Set B.

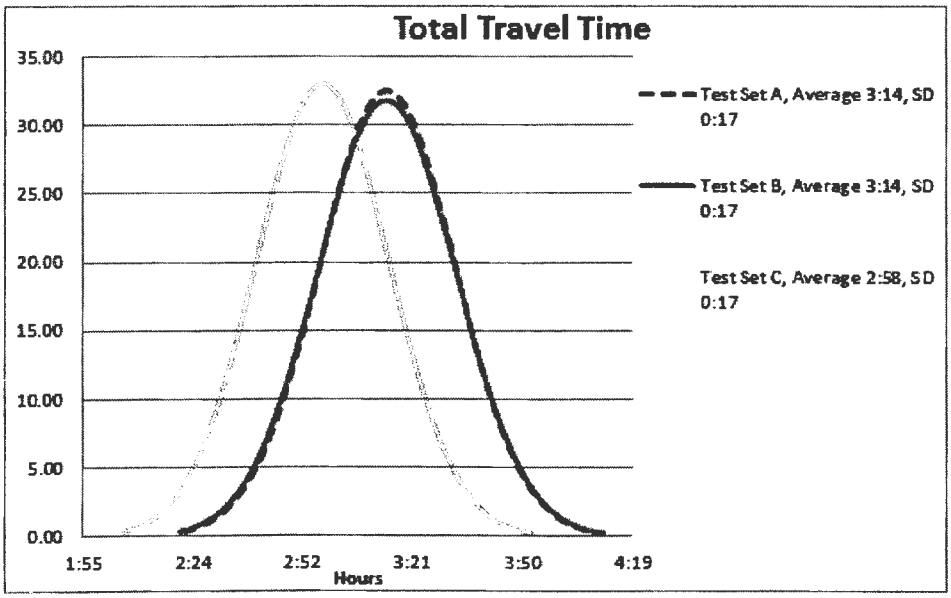



Figure 5.2.3 Normal Distribution of Total Travel Time for the Three Test Sets in Test Case 2

A random problem from Test Set A was chosen and the result from the problem was plotted in a graph for number of generations versus the fitness. The best fitness and the

mean fitness for the problem was computed and captured in the graph shown in Figure 5.2.4.

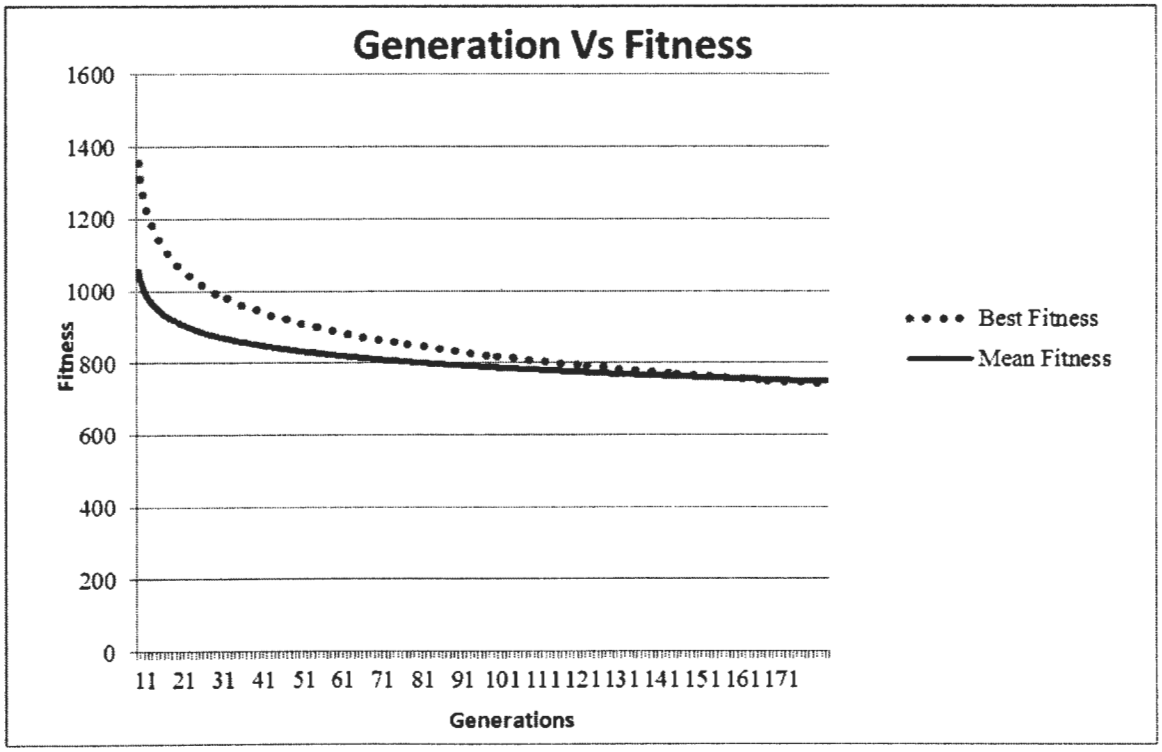



Figure 5.2.4 Fitness Graph of Best Fitness and Mean Fitness

## 5.3. Test Case 3

In order to weigh the quality of the program we performed a stress test with five different problems. We call the five problems as PB1, PB2, PB3, PB4 and PB5. In PB1, there were 100 students, 5 schools, 20 pickup points and 30 buses. In PB2 the number of students was increased to 500; there were 5 schools, 20 pickup points, 30 buses.

For PB3 we had 300 students, 5 schools, 20 pickup points, 30 buses. For PB4 we had 500 students, 5 schools, 20 pickup points, 30 buses. In PB5, there were 1000 students, 5 schools, 20 pickup points, 30 buses. The buses' travel speed was set to a maximum of 35 miles per hour and the maximum travel time per student was set to 60 minutes. The number

of chromosomes in a single generation was set to 400 and the mutation rate was set to 2%. The test results for these five problem sets are shown in Table 5.3.1.

| | S | B | SC | TD | TT | TC | BU | NG | CPU |
|---|---|---|---|---|---|---|---|---|---|
| **PB1** | 100 | 30 | 5 | 114.67 | 3:27 | 729.34 | 5 | 189 | 2.28 |
| **PB2** | 200 | 30 | 5 | 147.80 | 4:22 | 895.6 | 6 | 408 | 9.31 |
| **PB3** | 300 | 30 | 5 | 217.23 | 6:21 | 1134.46 | 7 | 532 | 21.47 |
| **PB4** | 500 | 30 | 5 | 498.24 | 14:23 | 2296.48 | 13 | 984 | 32.42 |
| **PB5** | 1000 | 30 | 5 | 957.59 | 27:36 | 4315.18 | 24 | 1918 | 68.38 |

Table 5.3.1 Results of Test Case 3

B: Buses,

S: Students,

SC: Schools,

TD: Total Distance traveled in miles,

TT: Total Travel Time in hours,

TC: Total Cost of all buses in dollars,

BU: Total number of buses used for routing,

NG: Number of Generations,

CPU: CPU time in minutes.

From Figures 5.3.1, 5.3.2 and 5.3.3 it is clear that the time traveled, distance traveled and the operating cost goes up with the increase in number of students. It is also evident from these graphs that our program is capable of handling any number of students.

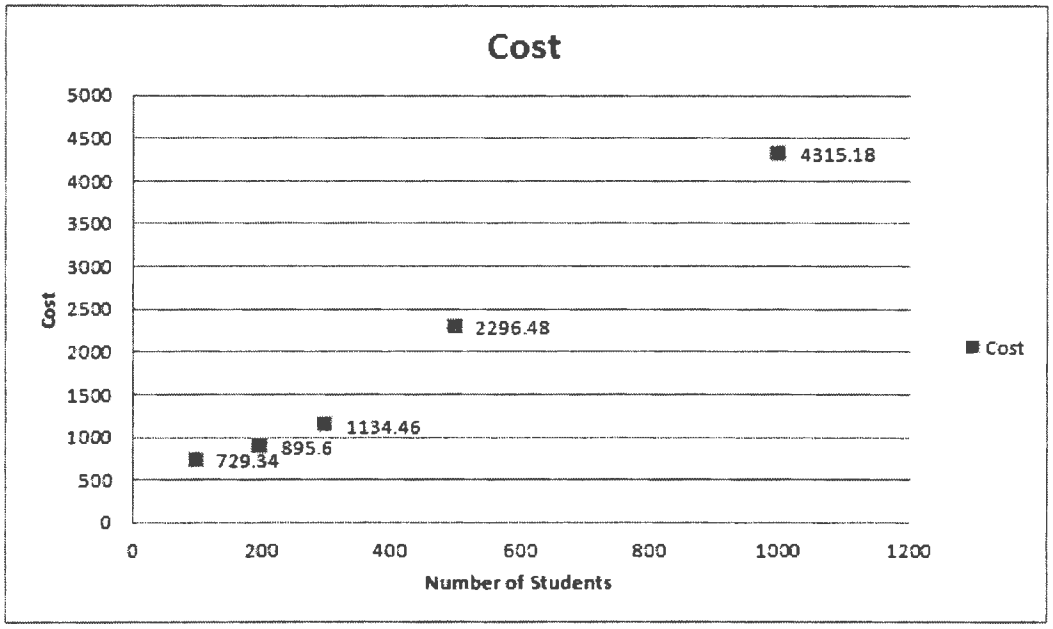



Figure 5.3.1 Total Cost for the 5 Problems in Test Case 3

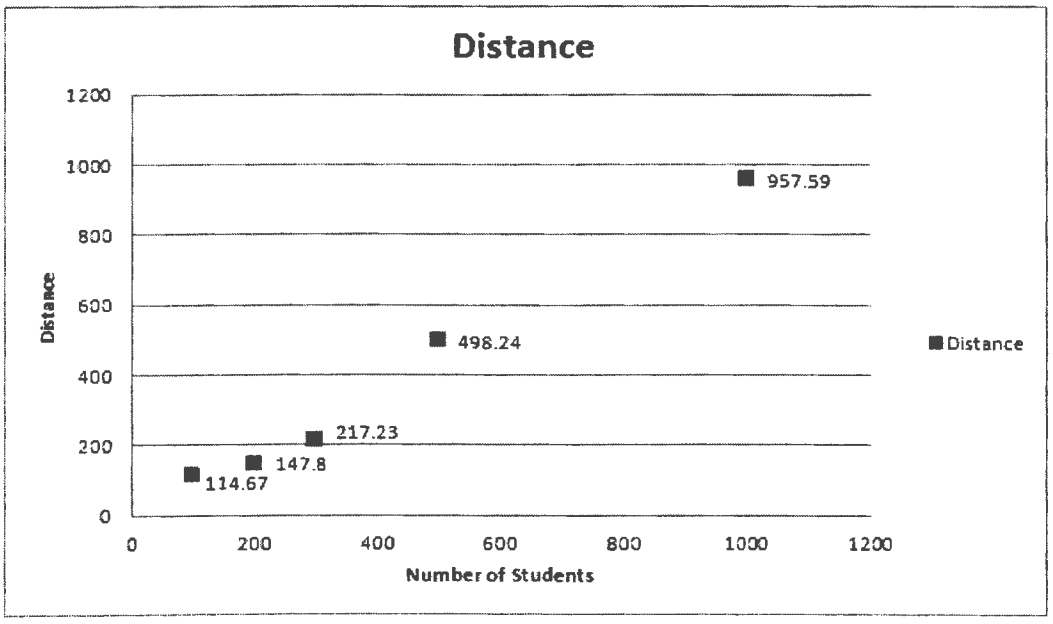



Figure 5.3.2 Total Distance for the 5 Problems in Test Case 3

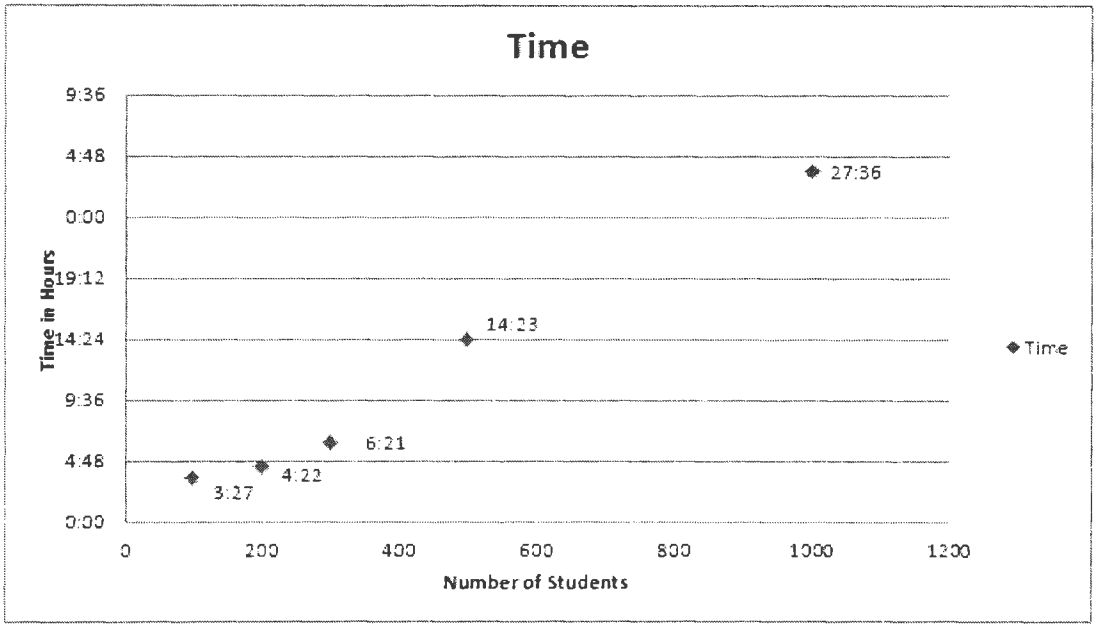



Figure 5.3.3 Total Time for the 5 Problems in Test Case 3

# CHAPTER 6. CONCLUSION

This paper presented a genetic algorithm and a graphical user interface for school bus routing in Joint Powers Regions. The genetic algorithm creates two populations sequentially, one for regular students and one for the special needs students and solves the problem based on the proposed algorithm.

## 6.1. Observations

Based on the results we got from various experiments while considering the goal of this paper "Genetic Search for Multi School Routing", we can come to a conclusion that Genetic algorithm can be used to design a program to route school buses in the complex situation of Joint Powers Regions as in North Dakota. Some of the complexities of this problem include multiple bus-garaged location, mixed vehicle fleet and mixed student types. Our testing indicates that the program performs well in response to the varying size of the problem.

The program scales relatively well with reasonable computational times as the problem sizes are increased. It also proved to be cost effective for the schools by reducing the number of buses used in the routing even though it is supplied with a higher number of available buses. With a user-friendly display of the routes and a back end database, that also stores the results; the program helps users to enter data about the students, schools, locations and buses information easily.

## 6.2. Future Work

The proposed heuristics method in this paper is mainly based on genetic algorithm and taxicab distance metrics. This model can be improved by implementing on digitized

road network that will enable us to use real time distances between locations to calculate distance instead of taxicab metric. This program can also be improved by integrating with programs that has functionalities like student registration that will send real time information of the student's availability at the pickup location. This information will help to reduce the travel time for students and operating cost for schools by avoiding that pickup point completely (based on the assumption that pick point has been assigned with only one student).

Another improvement that can be introduced or researched would be the genetic parameters that are used in the program such as the mutation rate and the population size. These parameters are capable of altering the results of the program and/or the performance of the program. It will be interesting to understand the practical meaning of each of these parameters and assign appropriate values accordingly. It would also be a wise improvement if the program uses optimized crossover operators [13] as that approach have shown quality results in terms of total distance traveled when compared to Solomon 1987 [14].

# REFERENCES

[1] North Dakota Regional Education Associations http://www.ndrea.org/index.html Nov 2008.

[2] Savelsbergh M. W. P, "Local Search for Routing Problems with Time Windows" December 01 1985.

[3] Sam R Thangiah, Adel Fergany, Bryan Wilson, Anthony Pitulga and William Mennell "School Bus Routing in Rural School Districts".

[4] Sam R. Thangiah, "Vehicle Routing with Time Windows using Genetic Algorithms" 1995.

[5] Joe L. Blanton Jr., Roger L. Wain Wright, "Multiple Vehicle Routing with time and Capacity Constraints using Genetic Algorithms" 1993.

[6] Jean Berger, Mohamed Barkaoui, "A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows" 1995.

[7] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, J.-S.Sormany "New Heuristics for the Vehicle Routing Problem" April 2005.

[8] S.Salhi.R.J.Petch, "A GA Based Heuristic for the Vehicle Routing Problem with Multiple Trips" April 2007.

[9] Zuhaimy Bin Ismail, "Development of Heuristics Methods based on Genetic Algorithm (GA) for solving Vehicle routing problem" 2008.

[10] Java Advantages and Disadvantages http://www.webdotdev.com/nvd/content/view/1042/204/ M.S, April 2007.

[11] JGAP Java Genetic Algorithm Packages http://jgap.sourceforge.net/, Klaus Meffert / Neil Rotstan 2001 – 2011.

[12] Microsoft Access, http://en.wikipedia.org/wiki/Microsoft_Access May 2010.

[13] H.Nazif and L.S.Lee, "Optimized Crossover Genetic Algorithm for Vehicle Routing Problem with Time Windows" 2010.

[14] Solomon, M.M, "Algorithms for the vehicle routing and scheduling problems with time window constraints" 1987.

[15] Holland, J.H, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor.

[16] Goldberg, David E, "Genetic Algorithms in Search Optimization and Machine Learning", Addison Wesley.

[17] Desrochers, M., Desroseiers, and M.M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows." 1992.

[18] Braysy, O., and M.Gendreau, "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms." 2005.

[19] J.-F. Cordeau, G.Laporte, and A.Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows." 2001.

[20] Bertsimas, Dimitris and Simchi – Levi, David "A New Generation of Vehicle Routing Research: Robust Algorithms, Addressing uncertainty." 1993.