**CHAPTER 1**

**INTRODUCTION**

**OBJECTIVE:**

The main Objective of this project is find the shortest distance based on Road condition and Distance of all the possible routes.

**ABSTRACT:**

This project deals with the optimization of vehicle routing. Since the total traveling time is not always effective due to road condition and some other obstacles, the objective regarded in this project comprises not only total traveling distance, but also the total traveling time. We propose Graph Search Algorithm(GSA) to solve the problem. path query, k nearest neighbor (kNN) query, and keyword-based kNN query, are widely used in location-based systems. The basis for this framework is an assembly-based method to calculate the shortest-path distances between two vertices(Based on Road Condition and Distance). Based on the assembly-based method, efficient search algorithms(Fuzzy Logic and Graph Search) to answer kNN queries and keyword-based kNN queries are developed.

* 1. **MOTIVATION OF THE PROJECT**

The multidepot vehicle routing problem is one of the common optimization problems in the logistics area. In a real-world environment, drivers choose the shortest route to reach a destination since they assume that it should take the shortest time to travel the shortest route. However, if some events such as traffic congestions, accidents happen in the shortest route, the traveling time spent on this route can be greater than that on the longer route. Thus, this Project considers not only the cost due to the total traveling distance, but also the cost due to the total traveling time, as two objectives. We propose a stochastic search technique called FLGA to solve the problem. Based on the promising computational results obtained in this Project, the proposed model and technique will be effective for industries to be applied in solving real-world problems.

* 1. **OVERVIEW OF THE PROJECT**

This Project deals with the optimization of vehicle routing problem in which multiple depots, multiple customers, and multiple products are considered. Since the total traveling time is not always restrictive as a time window constraint, the objective regarded in this Project comprises not only the cost due to the total traveling distance, but also the cost due to the total traveling time. We propose to use a stochastic search technique called fuzzy logic guided genetic algorithms (FLGA) to solve the problem. The role of fuzzy logic is to dynamically adjust the crossover rate and mutation rate after ten consecutive generations. In order to demonstrate the effectiveness of FLGA, a number of benchmark problems are used to examine its search performance. Also, several search methods, branch and bound, standard GA (i.e., without the guide of fuzzy logic), simulated annealing, and tabu search, are adopted to compare with FLGA in randomly generated data sets. Simulation results show that FLGA outperforms other search methods in all of three various scenarios.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 A New Spatial Object Search Framework For Road Networks:**

In this paper, we present a new system framework called ROAD for spatial object search on road networks. ROAD is extensible to diverse object types and efficient for processing various location-dependent spatial queries (LDSQs), as it maintains objects separately from a given network and adopts an effective search space pruning technique

Based on our analysis on the two essential operations for LDSQ processing, namely, network traversal and object lookup, ROAD organizes a large road network as a hierarchy of interconnected regional sub networks (called Rnets). Each Rnet is augmented with 1) shortcuts and 2) object abstracts to accelerate network traversals and provide quick object lookups, respectively.

* 1. **A System For Distributed Spatial Group Keyword Search On Road Networks.**

Query (e.g., shortest path) on road networks has been extensively studied. Although most of the existing query processing approaches is designed for centralized environments, there is a growing need to handle queries on road networks in distributed environments due to the increasing query workload and the challenge of querying large networks. In this demonstration, we showcase a distributed system called DISKs (DI stributed Spatial Keyword search) that is capable of efficiently supporting spatial group keyword search (SGKS) on road networks. Given a group of keywords X and a distance r, an SGKS returns locations on a road network, such that for each returned location p, there exists a set of nodes (on the road network), which are located within a network distance r from p and collectively contains X.

* 1. **G-Tree: An Efficient Index For Knn Search On Road Networks**

In this paper we study the problem of kNN search on road networks. Given a query location and a set of candidate objects in a road network, the kNN search finds the k nearest objects to the query location. To address this problem, we propose a balanced search tree index, called G-tree. The G-tree of a road network is constructed by recursively partitioning the road network into sub-networks and each G-tree node corresponds to a sub-network. Inspired by classical kNN search on metric space, we introduce a best-first search algorithm on road networks, and propose an elaborately designed assembly-based method to efficiently compute the minimum distance from a G-tree node to the query location

* 1. **Processing Of Continuous Location-Based Range Queries On Moving Objects In Road Networks.**

With the proliferation of mobile devices, an increasing number of urban users subscribe to location-based services. This trend has led to significant research interest in techniques that address two fundamental requirements: road network-based distance computation and the capability to process moving objects as points of interests. However, there exist few techniques that support both requirements simultaneously. To address these challenges, we propose a novel approach to process continuous range queries. We build on our previous work of an infrastructure that supports location-based snapshot queries on MOVing objects in road Networks (MOVNet).

* 1. **Top-K Spatial Keyword Queries On Road Networks**

With the popularization of GPS-enabled devices there is an increasing interest for location-based queries. In this context, one interesting problem is processing top-*k* spatial keyword queries. Given a set of objects with a textual description (e.g., menu of a restaurant), a query location (latitude and longitude), and a set of query keywords, a top-*k* spatial keyword query returns the *k* best objects ranked in terms of both distance to the query location and textual relevance to the query keywords. So far, the research on this problem has assumed Euclidean space.

* 1. **Processing Of Continuous Location based Range Queries On Moving Objects In Road Networks**

With the proliferation of mobile devices, an increasing number of urban users subscribe to location-based services. This trend has led to significant research interest in techniques that address two fundamental requirements: road network-based distance computation and the capability to process moving objects as points of interests. However, there exist few techniques that support both requirements simultaneously. To address these challenges, we propose a novel approach to process continuous range queries. We build on our previous work of an infrastructure that supports location-based snapshot queries on MOVing objects in road Networks (MOVNet). We introduce several significant features to enable continuous queries. The dual index structure that we proposed for MOVNet has been appropriately modified. We further appoint a number of connecting vertices in each cell and pre compute the distances among them to expedite query processing. Most importantly, to alleviate the effects of frequent object updates, we introduce a Shortest-Distance-based Tree (SD-Tree). We illustrate that the network connectivity and distance information can be preserved and reused by the SD-Tree when the query point location is updated; hence, reducing the continuous query update cost. Our experimental results demonstrate that our method yields excellent performance with a very large number of moving objects.

* 1. **An Efficient Path Computation Model For Hierarchically Structured Topographical Road Maps**

Computing the shortest path in real road networks is of great interest to us. In fact, we are dealing with such routing problems almost every day. We want to get to a desired destination through the fastest way which may be the shortest travel time/distance, or we may balance among several influencing factors such as time, security, and toll charges. In network theory, this corresponds to the shortest path problem, and different influencing factors will only affect the form of arc weights based on user’s preference. The most classical shortest path algorithm is the Dijkstra algorithm (Dijkstra, 1959) with a complexity of O(|E|+|V|log|V|), where |V| is the number of vertices and |E| is the number of arcs. Though Dijkstra algorithm computes the optimal solution in a theoretical sense, it is often far too slow for real-time route guidance applications.

* 1. **Voronoi-Based K Nearest Neighbor Search For Spatial Network Databases.**

A frequent type of query in spatial networks (e.g., road networks) is to find the K nearest neighbors (KNN) of a given query object. With these networks, the distances between objects depend on their network connectivity and it is computationally expensive to compute the distances (e.g., shortest paths) between objects. In this paper, we propose a novel approach to efficiently and accurately evaluate KNN queries in spatial network databases using first order Voronoi diagram. This approach is based on partitioning a large network to small Voronoi regions, and then pre-computing distances both within and across the regions. By localizing the pre computation within the regions, we save on both storage and computation and by performing across-the-network computation for only the border points of the neighboring regions, we avoid global pre-computation between every node-pair. Our empirical experiments with several real-world data sets show that our proposed solution outperforms approaches that are based on on-line distance computation by up to one order of magnitude, and provides a factor of four improvements in the selectivity of the filter step as compared to the index-based approaches

* 1. **Analysis Of Multilevel Graph Partitioning.**

Recently, a number of researchers have investigated a class of algorithms that are based on multilevel graph partitioning that have moderate computational complexity, and provide excellent graph partitions. However, there exists little theoretical analysis that could explain the ability of multilevel algorithms to produce good partitions. In this paper we present such an analysis. We show under certain reasonable assumptions that even if no refinement is used in the uncoarsening phase, a good bisection of the coarser graph is worse than a good bisection of the finer graph by at most a small factor. We also show that for planar graphs, the size of a good vertex-separator of the coarse graph projected to the finer graph (without performing refinement in the uncoarsening phase) is higher than the size of a good vertex-separator of the finer graph by at most a small factor.

* 1. **Hierarchical Encoded Path Views For Path Query Processing: An Optimal Model And Its Performance Evaluation.**

Efficient path query processing is a key requirement for advanced database applications including GIS (Geographic Information Systems) and ITS (Intelligent Transportation Systems). We study the problem in the context of automobile navigation systems where a large number of path requests can be submitted over the transportation network within a short period of time. To guarantee efficient response for path queries, we employ a path view materialization strategy for pre computing the best paths. We tackle the following three issues: memory-resident solutions quickly exceed current computer storage capacity for networks of thousands of nodes, disk based solutions have been found inefficient to meet the stringent performance requirements, and path views become too costly to update for large graphs. We propose the HEP V (Hierarchical Encoded Path View) approach that addresses these problems while guaranteeing the optimality of path retrieval. Our experimental results reveal that HEP V is more efficient than previously known path finding approaches.

**CHAPTER 3**

**SYSTEM ANALYSIS**

* 1. **EXISTING SYSTEM**

Dijikstra algorithm Search Algorithm Use to find the Shortest Path. There are only measurements of latitude and longitude for time calculation. There are many existing works to support these three types of queries studied the problem of shortest path queries between two vertices, addressed the kNN search problem, worked on the problem of keyword-based kNN query on road networks. Though they achieve good results for individual type of queries, there are still limitations to be addressed for them to be of practical use. First, some are not efficient enough, especially on kNN query. Second, some fail to scale up to very large datasets. Third, they cannot be adaptive to different types of queries

* + 1. **Drawbacks of Existing System**
* Existing Algorithm doesn’t give the exact distance; it’s give only approximate distance.
* Getting Only Simulation Result
* We can’t get proper response for different type of inputs Queries.
  1. **PROPOSED SYSTEM**

Graph Search algorithm can compute the exact shortest paths in a distance. Two optimization techniques on graph compression and graph online search are also proposed, with the goal of further reducing index size and improving query accuracy The performance of LSB remains very stable in all distance ranges Depend upon the query. Here we can find shortest path through Distance and Road Condition. Here we find the optimal value of all the possible routes through fuzzy logic algorithm using road conditions.

**3.2.1 Advantages of Proposed System**

* Existing Algorithm doesn’t give the exact distance; it’s give only approximate distance. But In Proposed We can get Exact Distance.
* We significantly reduce the distance estimation error, compared with global landmark embedding.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 METHODOLOGIES**

Methodologies are the process of analyzing the principles or procedure . The following are the four modules involved in the project.

**4.1.1 NODE INITIALIZATION:**

* The Node Initialization consists of two sub modules namely :-

Source Node Initialization

Destination Node Initialization

* Source Node Initialization :-

It initializes the main places of the node, otherwise the starting place of the node.

* Destination Node Initialization :-

It initializes the sub places of the main node, otherwise the ending place of the node.

* + 1. **FINDING POSSIBLE ROUTE:**

After getting information we find out whether these routes are available in database. Based on the input data, we find the possible routes between source node and destination node using Fuzzy Logic Algorithm. And also we find out the possible neighbor nodes between source and destination node. To give the input as source and destination. To get the neighbor node near to source and destination, and different path is obtained from these neighbours.To find the center node between source and destination. To get the distance from one node to another node. Add all the obtained distances and repeat for all the possible paths. Display the result.

* + 1. **COMPUTE THE SHORTEST ROUTE:**

After getting the possible routes we find out other possible routes in module 2 using Fuzzy Logic. Using Genetic Algorithm now we calculate the shortest route among the possible routes found

**GENETIC ALGORITHM:**

* The path routes are given as input.
* Split the network paths.
* Find all possible neighbor path along with distance, road and traffic condition.
* Optimal value is calculated.
  + 1. **BEST DRIVER PREFERRED ROUTE:**

It is easy to find out the shortest path using GA. It takes minimum time to give the output i.e., crossover rate and mutation rate. It gives the exact distance. It is suitable for controlling crossover rate and mutation rate during the run of GA.

**CHAPTER 5**

**SYSTEM REQUIREMENTS**

**5.1 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the systems do and not how it should be implemented.

Processor : Any Processor above 500 MHz.

Ram : 128Mb.

Hard Disk : 10 Gb.

Compact Disk : 650 Mb.

Input device : Standard Keyboard and Mouse.

Output device : VGA and High Resolution Monitor

**5.1 SOFTWARE REQUIREMENTS**

The software requirements are the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team’s and tracking the team’s progress throughout the development activity.

* Operating system : Windows XP/7
* IDE : Microsoft Visual Studio 2010
* Database : Sql server 2014
* Coding Language : C#.NET.

**5.3 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements, which impose constraints on the design or implementation. The plan for implementing functional requirements is detailed in the system design.

**5.4 NON-FUNCTIONAL REQUIREMENTS**

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing non*-*functional requirements is detailed in the system architecture. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements"

Non-functional requirements can be divided into two main categories:

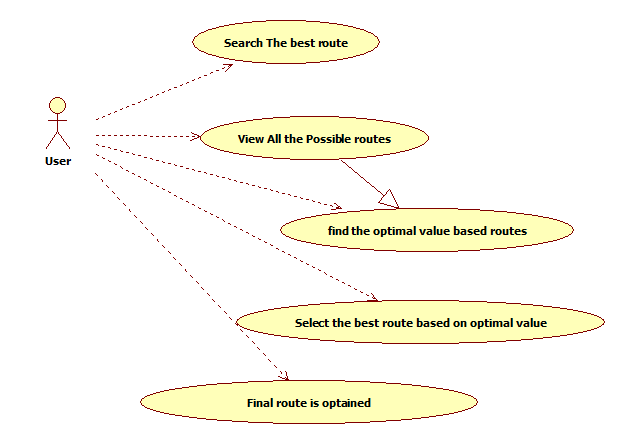
1. Execution qualities
2. Evolution qualities

**CHAPTER 6**

**DESIGN ENGINEERING**

**6.1 USE CASE DIAGRAM**

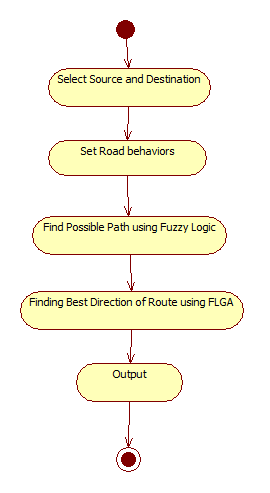
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**Fig 6.1 Use case diagram**

**6.2 ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflow of components in a system. An activity diagram shows the overall flow of control.

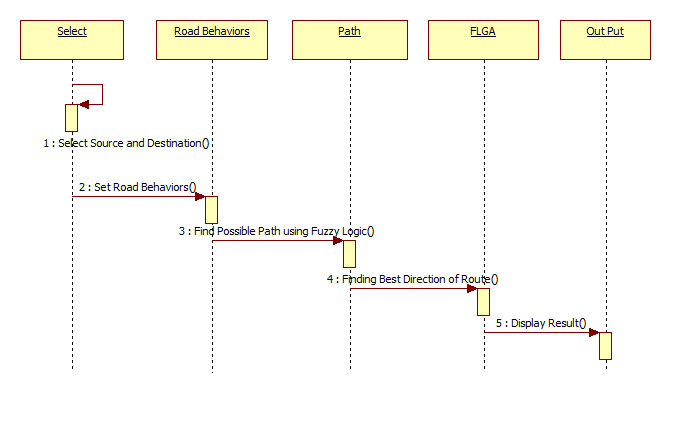


**Fig 6.2 Activity diagram**

**6.3 SEQUENCE DIAGRAM**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams.

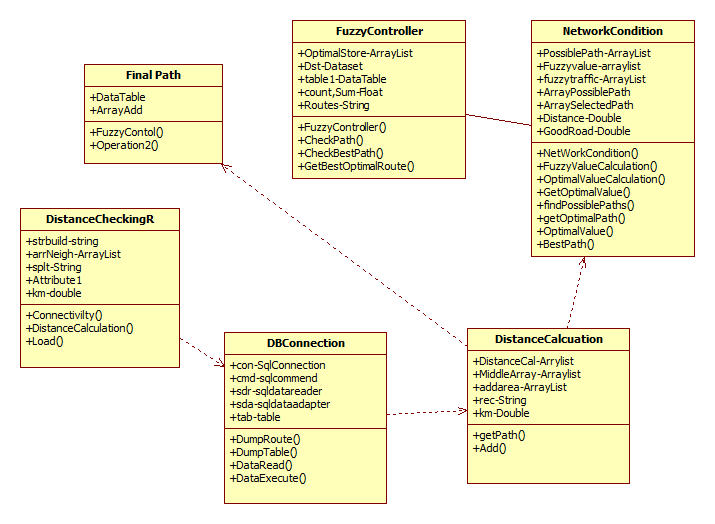
A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.



**Fig 6.3 Sequence diagram**

**6.4 CLASS DIAGRAM**

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, and the relationships between the classes. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.

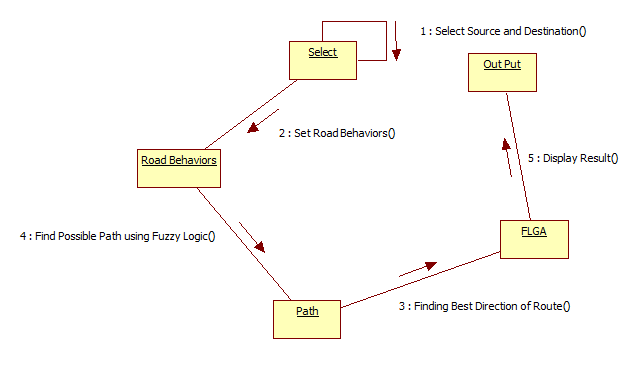


**Fig 6.4 Class diagram**

**6.5 COLLABORATION DIAGRAM**

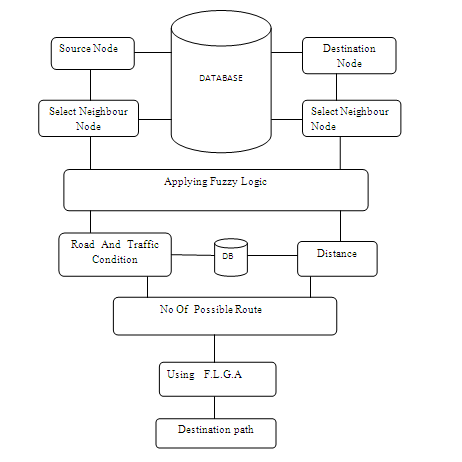
A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction. The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram. It can be the decomposition of a use case, use case diagram, or part of a use case diagram.

A collaboration diagram resembles a [flowchart](http://whatis.techtarget.com/definition/flowchart) that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in [real time](http://searchcio-midmarket.techtarget.com/definition/real-time). Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The [message](http://whatis.techtarget.com/definition/message)s between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.



**Fig 6.5 Collaboration diagram**

**Architeture Diagram:**

****

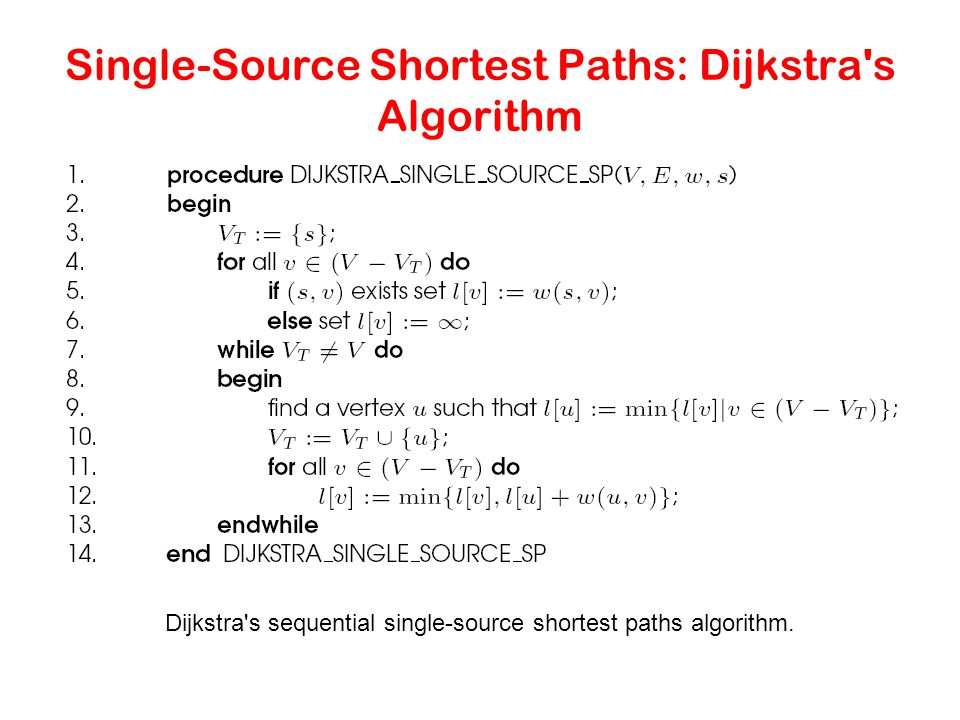
**ALGORITHM EXPLANATION**

**Single-Pair Shortest-Path Dijkstra's Algorithm:**

* The shortest path problem can be defined for graphs whether undirected, directed, or mixed. It is defined here for undirected graphs; for directed graphs the definition of path requires that consecutive vertices be connected by an appropriate directed edge.
* Two vertices are adjacent when they are both incident to a common edge. A path in an undirected graph is a sequence of vertices  such that  is adjacent to  for . Such a path  is called a path of length  from  to . (The  are variables; their numbering here relates to their position in the sequence and needs not to relate to any canonical labeling of the vertices.)
* The problem is also sometimes called the single-pair shortest path problem, to distinguish it from the following variations:
* The **single-source shortest path problem**, in which we have to find shortest paths from a source vertex *v* to all other vertices in the graph.
* The **single-destination shortest path problem**, in which we have to find shortest paths from all vertices in the directed graph to a single destination vertex *v*. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph.
* The **all-pairs shortest path problem**, in which we have to find shortest paths between every pair of vertices *v*, *v'* in the graph.
* These generalizations have significantly more efficient algorithms than the simplistic approach of running a single-pair shortest path algorithm on all relevant pairs of vertices.



* Consider Source node 1 and destination node 7
* A direct path between Nodes 1 and 7 will cost 14



**KNN ALGORITHM**:

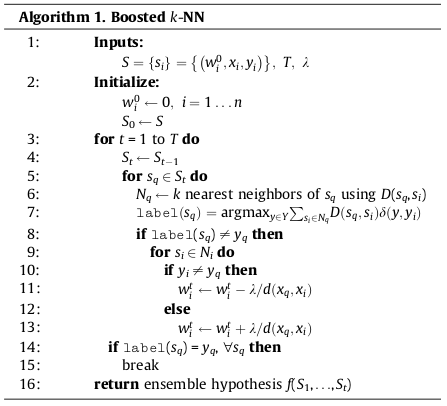
* The model for kNN is the entire training dataset. When a prediction is required for a unseen data instance, the kNN algorithm will search through the training dataset for the k-most similar instances. The prediction attribute of the most similar instances is summarized and returned as the prediction for the unseen instance.
* The similarity measure is dependent on the type of data. For real-valued data, the Euclidean distance can be used. Other types of data such as categorical or binary data, Hamming distance can be used.
* In the case of regression problems, the average of the predicted attribute may be returned. In the case of classification, the most prevalent class may be returned.
* We can then consider the similarity of two points to be the distance between them in this space under some appropriate metric. 2 The way in which the algorithm decides which of the points from the training set are similar enough to be considered when choosing the class to predict for a new observation is to pick the k closest data points to the new observation, and to take the most common class among these. This is why it is called the k Nearest Neighbours algorithm.
* The algorithm can be summarized as:

A positive integer k is specified, along with a new sample

We select the k entries in our database which are closest to the new sample

We find the most common classification of these entries

This is the classification we give to the new sample



**CHAPTER 7**

**DEVELOPMENT TOOLS**

**7.1 features of Dot NeT:**

.NET is a "**Software Platform**". It is a language-neutral environment for developing rich .NET experiences and building applications that can easily and securely operate within it. When developed applications are deployed, those applications will target .NET and will execute wherever .NET is implemented instead of targeting a particular Hardware/OS combination. The components that make up the .NET platform are collectively called the .NET Framework.

The .NET Framework is a **managed, type-safe environment** for developing and executing applications. The .NET Framework manages all aspects of program execution, like, allocation of memory for the storage of data and instructions, granting and denying permissions to the application, managing execution of the application and reallocation of memory for resources that are not needed.

The .NET Framework is designed for **cross-language compatibility**. Cross-language compatibility means, an application written in Visual Basic .NET may reference a DLL file written in C# (C-Sharp). A Visual Basic .NET class might be derived from a C# class or vice versa.

The .NET Framework consists of two main components:

* **Common Language Runtime (CLR)**
* **Class Libraries**

**Common Language Runtime (CLR)**

The CLR is described as the "**execution engine**" of .NET. It provides the environment within which the programs run. It's this CLR that manages the execution of programs and provides core services, such as code compilation, memory allocation, thread management, and garbage collection. Through the **Common Type** **System** (CTS), it enforces strict type safety, and it ensures that the code is executed in a safe environment by enforcing code access security. The software version of .NET is actually the CLR version.

**Working of the CLR**

When the .NET program is compiled, the output of the compiler is not an executable file but a file that contains a special type of code called  the Microsoft Intermediate Language (MSIL), which is a low-level set of instructions understood by the common language run time. This MSIL defines a set of portable instructions that are independent of any specific CPU. It's the job of the CLR to translate this Intermediate code into a executable code when the program is executed making the program to run in any environment for which the CLR is implemented. And that's how the .NET Framework achieves Portability. This MSIL is turned into executable code using a JIT (Just In Time) complier. The process goes like this, when .NET programs are executed, the CLR activates the JIT complier. The JIT complier converts MSIL into native code on a demand basis as each part of the program is needed. Thus the program executes as a native code even though it is compiled into MSIL making the program to run as fast as it would if it is compiled to native code but achieves the portability benefits of MSIL.



**Class Libraries**

Class library is the second major entity of the .NET Framework which is designed to integrate with the common language runtime. This library gives the program access to runtime environment. The class library consists of lots of prewritten code that all the applications created in VB .NET and Visual Studio .NET will use. The code for all the elements like forms, controls and the rest in VB .NET   
applications actually comes from the Class Library.

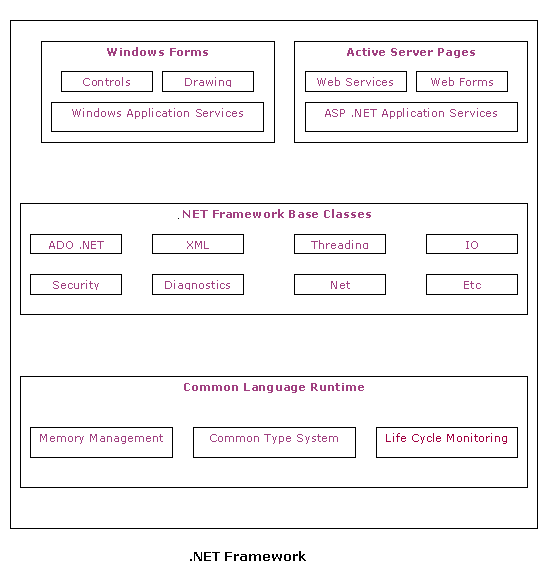
**Common Language Specification (CLS)**

If we want the code which we write in a language to be used by programs in other languages then it should adhere to the Common Language Specification (CLS). The CLS describes a set of features that different languages have in common. The CLS defines the minimum standards that .NET language compilers must conform to, and ensures that any source code compiled by a .NET compiler can interoperate with the .NET Framework.

Some reasons why developers are building applications using the .NET Framework

* Improved Reliability
* Increased Performance
* Developer Productivity
* Powerful Security
* Integration with existing Systems
* Ease of Deployment
* Mobility Support
* XML Web service Support
* Support for over 20 Programming Languages
* Flexible Data Access

**DOTNET FRAMEWORK**



**COMPILATION AND EXECUTION**

Source

Code

Compiler

IL Code

&

Meta Data

Linker

EXE

or

DLL

Base Class

Library

Class

Loader

Verifier

JIT

Compiler

Native

Code

**Execution**

**Compilation**

**OVERVIEW OF C#**

C# 2.0 introduces several language extensions, including Generics, Anonymous Methods, Iterators, Partial Types, and Null able Types.

* Generics permit classes, structs, interfaces, delegates, and methods to be parameterized by the types of data they store and manipulate. Generics are useful because they provide stronger compile-time type checking, require fewer explicit conversions between data types, and reduce the need for boxing operations and run-time type checks.
* Anonymous methods allow code blocks to be written “in-line” where delegate values are expected. Anonymous methods are similar to lambda functions in the Lisp programming language. C# 2.0 supports the creation of “closures” where anonymous methods access surrounding local variables and parameters.
* Iterators are methods that incrementally compute and yield a sequence of values. Iterators make it easy for a type to specify how the foreach statement will iterate over its elements.
* Partial types allow classes, structs, and interfaces to be broken into multiple pieces stored in different source files for easier development and maintenance. Additionally, partial types allow separation of machine-generated and user-written parts of types so that it is easier to augment code generated by a tool.
* Nullable types represent values that possibly are unknown. A nullable type supports all values of its underlying type plus an additional null state. Any value type can be the underlying type of a nullable type. A nullable type supports the same conversions and operators as its underlying type, but additionally provides null value propagation similar to SQL.

This chapter gives an introduction to these new features. Following the introduction are five chapters that provide a complete technical specification of the features. The final chapter describes a number of smaller extensions that are also included in C# 2.0.

The language extensions in C# 2.0 were designed to ensure maximum compatibility with existing code. For example, even though C# 2.0 gives special meaning to the words where, yield, and partial in certain contexts, these words can still be used as identifiers.

The C# foreach statement is used to iterate over the elements of an enumerablecollection.

In order to be enumerable, a collection must have a parameter less GetEnumerator method that returns an enumerator. Generally, enumerators are difficult to implement, but the task is significantly simplified with iterators.

An iterator is a statement block that yields an ordered sequence of values. An iterator is distinguished from a normal statement block by the presence of one or more yield statements:

* The yield return statement produces the next value of the iteration.
* The yield break statement indicates that the iteration is complete.

**7.8 SQL SERVER:**

SQL Server 2000 will be soon reaching its five-year mark, which in terms of software life-cycle translates into fairly advanced maturity. While this is still far from retirement age, the name of its successor, SQL Server 2005, suggests that it might be time for you to start looking into what the new generation has to offer. The release of SQL Server 2005, originally introduced as Yukon, has already been postponed, but its current Beta 2 implementation (with several incremental Community Technical Previews expected before Beta 3 becomes available early next year) brings promise of a timely RTM stage (planned for summer next year). In this series of articles, we will look into functional highlights of the new incarnation of the Microsoft database management system, focusing on those that are likely to remain unchanged in the final product.

Improvements to the database engine, the details of which are not published by Microsoft, and the corresponding changes to the main infrastructure components are reflected by a substantial number of new features as well as enhancements to existing ones. The most relevant ones can be grouped into several categories, such as high availability and scalability, security, data management, administration and maintenance, and development.

The demand for high availability is becoming increasingly common and is no longer limited to major corporate and governmental clients. This results not only from a growing level of customer expectations, but also from the new political climate associated with more stringent legislative and regulatory requirements, in which disaster recovery and business continuity are more relevant than ever. However, businesses are also, at the same time, extremely interested in keeping their costs to a minimum. Microsoft tries to address these expectations by implementing scalability enhancements, which ensure that SQL Server can perform equally well in environments of any size, and by the introduction of several versions of SQL Server 2005 (geared towards more specialized needs) such as:

* SQL Server Standard Edition - offering the most diverse set of features and intended for the majority of clients.
* SQL Server 2005 Express Edition - serving as the replacement for Microsoft Data Engine (MSDE) and available for download from t. Like its predecessor, it was designed with developers in mind, however, unlike the previous version, it also includes a Web based management interface.
* SQL Server 2005 Mobile Edition - as a successor to SQL Server 2000 Windows CE Edition, it is intended for Windows mobile-based devices, such as Tablet PCs, Pocket PCs, and Smart phones

**7.8.1 features of SQL SERVER**

**Microsoft SQL Server 2005**

The following is a list of the new features provided in SQL Server 2005:

* Database mirroring
* Database snapshots
* CLR integration
* Service Broker
* Database Mail
* User-defined functions
* Indexed views
* Distributed partitioned views
* INSTEAD OF and AFTER triggers
* New data types
* Cascading RI constraints
* Multiple SQL Server instances
* XML support
* Log shipping

**Database Mirroring**

Database mirroring is a new high-availability feature in SQL Server 2000. It's similar to server clustering in that failover is achieved by the use of a stand-by server; the difference is that the failover is at the database level rather than the server level. The primary database continuously sends transaction logs to the backup database on a separate SQL Server instance. A third SQL Server instance is then used as a witness database to monitor the interaction between the primary and the mirror databases.

**Database Snapshots**

A database snapshot is essentially an instant read-only copy of a database, and it is a great candidate for any type of reporting solution for your company. In addition to being a great reporting tool, you can revert control from your primary database to your snapshot database in the event of an error. The only data loss would be from the point of creation of the database snapshot to the event of failure.

**CLR Integration**

With SQL Server 2005, you now have the ability to create custom .NET objects with the database engine. For example, stored procedures, triggers, and functions can now be created using familiar .NET languages such as VB and C#. Exposing this functionality gives you tools that you never had access to before such as regular expressions.

**Service Broker**

This feature gives you the ability to create asynchronous, message-based applications in the database entirely through TSQL. The database engine guarantees message delivery, message order consistency, and handles message grouping. In addition, Service Broker gives you the ability to send messages between different SQL Server instances. Server Broker is also used in several other features in SQL Server 2000. For example, you can define Event Nonfictions in the database to send a message to a Queue in the database when someone attempts to alter a table structure, of if there is a string of login failures.

**Database Mail**

Database Mail, the eventual successor to SQL Mail, is greatly enhanced e-mail solution available in the database engine. With Database Mail, there is no longer a dependency on Microsoft Outlook or MAPI e-mail clients. Database Mail uses standard SMTP to send e-mail messages. These messages may contain query results, attachments (which can be governed by the DBA), and is fully cluster aware. In addition, the e-mail process runs outside of the database engine space, which means that messages can continue to be queued even when the database engine has stopped.

**CHAPTER 8**

**IMPLEMENTATION**

**IMPLEMENTATION:**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* AdminControl \*/**

**/\* \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**DBCONNECTION.CS:**

**using System;**

**using System.Data;**

**using System.Configuration;**

**using System.Web;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**using System.Data.SqlClient;**

**using System.Data.Sql;**

**/// <summary>**

**/// Summary description for DBConnection**

**/// </summary>**

**public class DBConnection**

**{**

**SqlConnection con;**

**SqlCommand cmd;**

**SqlDataAdapter da;**

**public SqlDataReader dr;**

**DataSet ds;**

**string sql,i;**

**DataTable tab;**

**public DBConnection()**

**{**

**//con = new SqlConnection(@"Data Source=.\SQLExpress;Integrated Security=True;User Instance=True;AttachDBFilename=|DataDirectory|Fuzzy.mdf");**

**con = new SqlConnection("server=UNIQF3;Database=FLGA;Integrated Security=true");**

**con.Open();**

**}**

**public DataTable DumpRoute(String sql)**

**{**

**ds = new DataSet();**

**da = new SqlDataAdapter(sql, con);**

**da.Fill(ds, "Temp");**

**DataTable dtable = ds.Tables["Temp"];**

**return dtable;**

**}**

**public void Dumptable(int Count)**

**{**

**if (NetworkCondition.ds.Tables.Count == 0)**

**{**

**for (int i = 0; i < Count; i++)**

**{**

**tab = new DataTable();**

**sql = "select \* from Distance";**

**da = new SqlDataAdapter(sql, con);**

**tab.TableName = i.ToString();**

**da.Fill(tab);**

**NetworkCondition.ds.Tables.Add(tab);**

**}**

**}**

**}**

**public string DataRead(String sql)**

**{**

**cmd = new SqlCommand(sql, con);**

**dr = cmd.ExecuteReader();**

**dr.Read();**

**try**

**{**

**i = dr.GetValue(0).ToString();**

**}**

**catch { i = null; }**

**return i;**

**}**

**public void DataExcute(String sql)**

**{**

**cmd = new SqlCommand(sql, con);**

**cmd.ExecuteNonQuery();**

**}**

**}**

**DISTANCECALCULATION.CS:**

**using System;**

**using System.Data;**

**using System.Configuration;**

**using System.Web;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**using System.Collections;**

**using System.Collections.Generic;**

**/// <summary>**

**/// Summary description for DistanceCalculation**

**/// </summary>**

**public class DistanceCalculation**

**{**

**//Variable Declaration**

**//Creating The Referrence And Object For The Classes To Access**

**public static ArrayList arryConnection = new ArrayList();**

**public static ArrayList PossiblePath = new ArrayList();**

**public static ArrayList PossiblePathDistance = new ArrayList();**

**ArrayList arraylink = new ArrayList();**

**ArrayList middlearray = new ArrayList();**

**ArrayList result = new ArrayList();**

**ArrayList Distancecalc = new ArrayList();**

**ArrayList addarea = new ArrayList();**

**ArrayList nooflink = new ArrayList();**

**public static ArrayList distance = new ArrayList();**

**Double km;**

**String rec, middle;**

**String[] splt;**

**string path;**

**//To Find All PossiblePaths Between The Source And The Destination Till The End In A Recursive Manner**

**public void getpath(string source, string destination, ArrayList mid, string midarea)**

**{**

**arraylink = arryConnection;**

**for (int i = 0; i < arraylink.Count; i++)**

**{**

**rec = arraylink[i].ToString();**

**splt = rec.Split('-');**

**if (midarea.Equals(splt[0]))**

**{**

**middle = splt[1];**

**}**

**else if (midarea.Equals(splt[1]))**

**{**

**middle = splt[0];**

**}**

**else { continue; }**

**if (middle.Equals(destination))**

**{**

**middlearray = new ArrayList();**

**for (int j = 0; j < mid.Count; j++)**

**{**

**middlearray.Add(mid[j]);**

**}**

**middlearray.Add(middle);**

**addarea.Clear();**

**km = 0;**

**for (int a = 0; a < middlearray.Count; a++)**

**{**

**addarea.Add(middlearray[a]);**

**}**

**for (int k = 0; k < addarea.Count; k++)**

**{**

**if ((k + 1) != addarea.Count)**

**{**

**path = addarea[k].ToString() + "-" + addarea[k + 1].ToString();**

**for (int x = 0; x < PossiblePathDistance.Count; x++)**

**{**

**rec = PossiblePathDistance[x].ToString();**

**splt = rec.Split('-');**

**if ((path == (splt[0] + "-" + splt[1])) || (path == (splt[1] + "-" + splt[0])))**

**{**

**km = km + Convert.ToDouble(splt[2]);**

**}**

**}**

**}**

**}**

**//TO ADD DISTANCE**

**distance.Add(km);**

**//TO ADD ROUTERS POSSIBLE DISTANCE**

**PossiblePath.Add(middlearray);**

**}**

**else if (!(source.Equals(middle)) && (!mid.Contains(middle)))**

**{**

**// RouteCalculation rc = new RouteCalculation();**

**middlearray = new ArrayList();**

**for (int j = 0; j < mid.Count; j++)**

**{**

**middlearray.Add(mid[j]);**

**}**

**middlearray.Add(middle);**

**//TO CALL GETPATH METHOD AS A RECURSIVE METHOD**

**getpath(source, destination, middlearray, middle);**

**}**

**}**

**}**

**}**

**DISTANCE CHECKING.CS:**

**using System;**

**using System.Data;**

**using System.Configuration;**

**using System.Web;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**using System.Collections;**

**using System.Collections.Generic;**

**using System.Text;**

**public class DistanceCheckingR**

**{**

**//VARIABLE DECLARATION**

**StringBuilder strbuild = new StringBuilder();**

**ArrayList arryNeighR1;**

**ArrayList arryNeighR2 = new ArrayList();**

**ArrayList arryNeigh = new ArrayList();**

**ArrayList aR = new ArrayList();**

**ArrayList aR1 = new ArrayList();**

**ArrayList arryCon = new ArrayList();**

**string[] splt, splt1;**

**string Machine,rec,strngB, middle,source, destination;**

**double km;**

**//TO FIND ALL ROUTER CONNECTION AND ROUTER CINNECTED LINK**

**public void Connectivity(string SR,string DR,ArrayList arryConnection)**

**{**

**DistanceCalculation.arryConnection.Clear();**

**DistanceCalculation.arryConnection = arryConnection;**

**arryCon = arryConnection;//LINK BETWEEN SOURCE AND DESTINATION**

**destination = DR;//DESSTINATION**

**source = SR;//SOURCE**

**for (int i = 0; i < arryConnection.Count; i++)**

**{**

**rec = arryConnection[i].ToString();**

**splt = rec.Split('-');**

**if (source.Equals(splt[0]))**

**{**

**middle = splt[1];**

**}**

**else if (source.Equals(splt[1]))**

**{**

**middle = splt[0];**

**}**

**else**

**{**

**continue;**

**}**

**if (middle.Equals(destination))**

**{**

**arryNeighR1 = new ArrayList();**

**arryNeighR1.Add(source);**

**arryNeighR1.Add(middle);**

**//ADD NEIGHBOUR NODE FOR EACH ROUTER**

**DistanceCalculation.PossiblePath.Add(arryNeighR1);**

**for (int k = 0; k < DistanceCalculation.PossiblePathDistance.Count; k++)**

**{**

**strngB = DistanceCalculation.PossiblePathDistance[k].ToString();**

**splt1 = strngB.Split('-');**

**if (((splt1[1] == source) && splt1[2] == destination) || ((splt1[2] == source) && (splt1[1] == destination)))**

**{**

**km = Convert.ToDouble(splt1[0]);**

**break;**

**}**

**}**

**// ADD ROUTER DISTANCE**

**DistanceCalculation.distance.Add(km);**

**}**

**else**

**{**

**arryNeighR1 = new ArrayList();**

**arryNeighR1.Add(source);**

**arryNeighR1.Add(middle);**

**DistanceCalculation dc = new DistanceCalculation();**

**//TO CALL GETPATH METHOD FOR CHECK ROUTER LINK**

**dc.getpath(source, destination, arryNeighR1, middle);**

**}**

**}**

**}**

**}**

**FUZZY CONTROLLER.CS:**

**using System;**

**using System.Data;**

**using System.Configuration;**

**using System.Collections;**

**using System.Web;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**/// <summary>**

**/// Summary description for FuzzyController**

**/// </summary>**

**public class FuzzyController**

**{**

**ArrayList OptimalStore = new ArrayList();**

**DataSet dst = new DataSet();**

**DataTable table1 = new DataTable();**

**float SumOptimal;**

**float count;**

**NetworkCondition objNetwork = new NetworkCondition();**

**public static ArrayList arryBest = new ArrayList();**

**public static ArrayList arryBestPath = new ArrayList();**

**String Sno, Routes, Distance; public static string OPtimal;**

**public FuzzyController()**

**{**

**table1.Columns.Add("SINo");**

**table1.Columns.Add("Route");**

**table1.Columns.Add("Distance");**

**table1.Columns.Add("Optimal Value");**

**//**

**// TODO: Add constructor logic here**

**//**

**}**

**public void CheckPath()**

**{**

**arryBest.Clear();**

**arryBestPath.Clear();**

**for (int i = 0; i < NetworkCondition.arryOptimzedPath.Count; i++)**

**{**

**string[] Route = NetworkCondition.arryOptimzedPath[i].ToString().Split('\_');**

**string[] route = Route[0].Split('-');**

**string R = " ";**

**for (int j = 1; j < route.Length - 1; j++)**

**{**

**R = R + "-" + route[j];**

**}**

**for (int j = 0; j < NetworkCondition.arrySelectedPath.Count; j++)**

**{**

**if (NetworkCondition.arrySelectedPath[j].ToString() == Route[0])**

**{**

**arryBest.Add(NetworkCondition.arryOptimzedPath[i].ToString());**

**if ((Route[1] == OPtimal) || ( float.Parse(Route[1]) - float.Parse(OPtimal)) < 0.15)**

**{**

**//if (count == 0)**

**//{**

**arryBestPath.Add(NetworkCondition.arryOptimzedPath[i].ToString() + "+" +( float.Parse(Route[1]) - float.Parse(OPtimal)));**

**count++;**

**//}**

**}**

**}**

**}**

**}**

**if(arryBestPath.Count ==0)**

**{**

**NetworkCondition.arryOptimzedBestPath.Clear();**

**NetworkCondition.arryOptimzedPath.Clear();**

**NetworkCondition.arryOptimzedSelectedPath.Clear();**

**objNetwork.OptimalValueCalculation();**

**CheckPath();**

**}**

**}**

**public void CheckBestpath(DataTable dr1)**

**{**

**Sno = dr1.Rows[0].ItemArray[0].ToString();**

**Routes = dr1.Rows[0].ItemArray[1].ToString();**

**Distance = dr1.Rows[0].ItemArray[2].ToString();**

**OPtimal = dr1.Rows[0].ItemArray[3].ToString();**

**CheckPath();**

**}**

**public DataTable GetBestOptimalRoute(ArrayList Routes, DataSet ds,ArrayList OptimalSort)**

**{**

**DataTable table = ds.Tables[0];**

**OptimalSort.Reverse();**

**for (int i = 0; i < OptimalSort.Count; i++)**

**{**

**for (int j = 0; j < table.Rows.Count; j++)**

**{**

**if (OptimalSort[i].ToString() == table.Rows[j].ItemArray[3].ToString())**

**{**

**DataRow row = table1.NewRow();**

**row[0] = table.Rows[j].ItemArray[0].ToString();**

**row[1] = table.Rows[j].ItemArray[1].ToString();**

**row[2] = table.Rows[j].ItemArray[2].ToString();**

**row[3] = table.Rows[j].ItemArray[3].ToString();**

**table1.Rows.Add(row);**

**}**

**}**

**}**

**dst.Tables.Add(table1);**

**return dst.Tables[0];**

**}**

**}**

**NETWORK CONDITION.CS:**

**using System;**

**using System.Data.SqlClient;**

**using System.Data;**

**using System.Collections;**

**using System.Collections.Generic;**

**using System.Configuration;**

**using System.Web;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**/// <summary>**

**/// Summary description for NetworkCondition**

**/// </summary>**

**public class NetworkCondition**

**{**

**//Variable Declaration**

**DBConnection objDBCON = new DBConnection();**

**ArrayList possiblepath = new ArrayList();**

**ArrayList fuzzyvalue = new ArrayList();**

**ArrayList fuzzytraffic = new ArrayList();**

**public static ArrayList fuzz = new ArrayList();**

**public static ArrayList arryOptimzedSelectedPath = new ArrayList();**

**public static ArrayList arryOptimzedPath = new ArrayList();**

**public static ArrayList arryOptimzedRecursive = new ArrayList();**

**public static ArrayList arryOptimzedBestPath = new ArrayList();**

**public static ArrayList getestimatedroutes = new ArrayList();**

**public static ArrayList routes = new ArrayList();**

**Random roadcondition = new Random();**

**Random traffic = new Random();**

**public static ArrayList arryPossiblePath = new ArrayList();**

**public static ArrayList arrySelectedPath = new ArrayList();**

**string sql;**

**double Getroad, Gettraffic;float Optmal\_return;**

**double Distance, goodroad, lowtraffic, lowdistance, calr, calt, caldist, values;**

**float count;**

**public DataTable table = new DataTable();**

**public static DataTable table2 = new DataTable();**

**public static DataTable dstab = new DataTable();**

**float optimal = 1;**

**float SumOptimal;**

**public static DataSet ds = new DataSet();**

**public NetworkCondition()**

**{**

**objDBCON.Dumptable(arryPossiblePath.Count);**

**}**

**public void FuzzyvalueCalculation(string Location, DataTable table\_Opt)**

**{**

**for (int k = 0; k < table\_Opt.Rows.Count; k++)**

**{**

**string Destn = table\_Opt.Rows[k].ItemArray[0].ToString();**

**for (int i = 0; i < table2.Rows.Count; i++)**

**{**

**if (((Location == table2.Rows[i].ItemArray[1].ToString()) && (Destn == table2.Rows[i].ItemArray[2].ToString())) || ((Destn == table2.Rows[i].ItemArray[1].ToString()) && (Location == table2.Rows[i].ItemArray[2].ToString())))**

**{**

**Distance = Convert.ToDouble(table2.Rows[i].ItemArray[3].ToString());**

**Getroad = Convert.ToDouble(table2.Rows[i].ItemArray[4].ToString());**

**Gettraffic = Convert.ToDouble(table2.Rows[i].ItemArray[5].ToString());**

**//Road Condition**

**if (Getroad == 0)**

**{**

**calr = 0;**

**}**

**else if ((Getroad == 5))**

**{**

**calr = 0.5;**

**}**

**else**

**{**

**calr = 1;**

**}**

**goodroad = (1 - calr);**

**//Traffic Condition**

**if (Gettraffic == 0)**

**{**

**calt = 0;**

**}**

**else if (Gettraffic == 5)**

**{**

**calt = 0.5;**

**}**

**else**

**{**

**calt = 1;**

**}**

**lowtraffic = (1 - calt);**

**//Distance**

**if (Distance <= 60)**

**{**

**caldist = 0;**

**}**

**else if ((Distance > 60) && (Distance <= 100))**

**{**

**caldist = 0.5;**

**}**

**else**

**{**

**caldist = 1;**

**}**

**lowdistance = (1 - caldist);**

**values = ((goodroad + lowtraffic + lowdistance) / 3);**

**//double Optimalvalue = OptimalValueCalculation(goodroad,lowtraffic,lowdistance,values);**

**sql = "update Distance set FuzzyValue='" + values + "'where Source='" + Location + "' and Dest='" + Destn + "'";**

**objDBCON.DataExcute(sql);**

**}**

**}**

**}**

**}**

**public void OptimalValueCalculation()**

**{**

**for (int i = 0; i < arryPossiblePath.Count; i++)**

**{**

**string[] Route = arryPossiblePath[i].ToString().Split('-');**

**SumOptimal = 0;**

**count = 0;**

**for (int j = 1; j < Route.Length -1; j++)**

**{**

**for(int k=0;k<ds.Tables[i].Rows.Count;k++)**

**{**

**if (((Route[j] == ds.Tables[i].Rows[k].ItemArray[1].ToString()) && (Route[j + 1] == ds.Tables[i].Rows[k].ItemArray[2].ToString())))**

**{**

**float FuzzyValue = float.Parse(ds.Tables[i].Rows[k].ItemArray[6].ToString());**

**float TrafficVal = float.Parse(ds.Tables[i].Rows[k].ItemArray[5].ToString());**

**float OptVa = GetOptimalValue(ref FuzzyValue, TrafficVal);**

**ds.Tables[i].Rows[k]["FuzzyValue"] = FuzzyValue;**

**ds.Tables[i].Rows[k]["OptimalValue"] = OptVa;**

**SumOptimal = SumOptimal + OptVa;**

**count++;**

**}**

**else if ((Route[j + 1] == ds.Tables[i].Rows[k].ItemArray[1].ToString()) && (Route[j] == ds.Tables[i].Rows[k].ItemArray[2].ToString()))**

**{**

**float FuzzyValue = float.Parse(ds.Tables[i].Rows[k].ItemArray[6].ToString());**

**float TrafficVal = float.Parse(ds.Tables[i].Rows[k].ItemArray[5].ToString());**

**float OptVa = GetOptimalValue(ref FuzzyValue, TrafficVal);**

**ds.Tables[i].Rows[k]["FuzzyValue"] = FuzzyValue;**

**ds.Tables[i].Rows[k]["OptimalValue"] = OptVa;**

**//sql = "Update Route" + i + " SET FuzzyValue ='" + FuzzyValue + "' , OptimalValue ='" + OptVa + "'where Source='" + Route[j].ToString() + "' and Dest='" + Route[j + 1].ToString() + "'";**

**//objDBCON.DataExcute(sql);**

**SumOptimal = SumOptimal + OptVa;**

**count++;**

**}**

**}**

**}**

**float FinalOptimal = SumOptimal / count;**

**if (FinalOptimal > optimal)**

**{**

**arryOptimzedBestPath.Add(arryPossiblePath[i].ToString() + "\_" + FinalOptimal);**

**arryOptimzedPath.Add(arryPossiblePath[i].ToString() + "\_" + FinalOptimal);**

**for (int x = 0; x < arrySelectedPath.Count; x++)**

**{**

**if (arryPossiblePath[i].ToString() == arrySelectedPath[x].ToString())**

**{**

**arryOptimzedSelectedPath.Add(arrySelectedPath[x].ToString() + "\_" + FinalOptimal);**

**}**

**}**

**}**

**else**

**{**

**arryOptimzedPath.Add(arryPossiblePath[i].ToString() + "\_" + FinalOptimal);**

**for (int x = 0; x < arrySelectedPath.Count; x++)**

**{**

**if (arryPossiblePath[i].ToString() == arrySelectedPath[x].ToString())**

**{**

**arryOptimzedSelectedPath.Add(arrySelectedPath[x].ToString() + "\_" + FinalOptimal);**

**}**

**}**

**}**

**}**

**if (arryOptimzedBestPath.Count == 0)**

**{**

**//new NetworkCondition();**

**arryOptimzedBestPath.Clear();**

**arryOptimzedPath.Clear();**

**arryOptimzedSelectedPath.Clear();**

**OptimalValueCalculation();**

**}**

**}**

**public float GetOptimalValue(ref float FuzzVal, float Traff)**

**{**

**if (Traff == 0)**

**{**

**FuzzVal = FuzzVal + (float) 0.125;**

**//if (FuzzVal > optimal)**

**//{**

**Optmal\_return = FuzzVal;**

**//}**

**}**

**else**

**{**

**FuzzVal = FuzzVal + (float)0.225;**

**//if (FuzzVal > optimal)**

**//{**

**Optmal\_return = FuzzVal;**

**//}**

**}**

**return Optmal\_return;**

**}**

**}**

**CONTROLLER CENTER.ASPX.CS:**

**using System;**

**using System.Data;**

**using System.Data.SqlClient;**

**using System.Configuration;**

**using System.Collections;**

**using System.Web;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**public partial class Default2 : System.Web.UI.Page**

**{**

**string sql;**

**DBConnection objdbcon = new DBConnection();**

**public static DataTable table;**

**string Location;**

**bool flag,flag1;**

**NetworkCondition objNetwok = new NetworkCondition();**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**DropDownList4.Items.Clear();**

**Location = Request.QueryString["id"];**

**Page.Title = "Welcome To Control Center of " + Location;**

**if (!Page.IsPostBack)**

**{**

**sql = "select dest from Route where source='" + Location + "'";**

**table = objdbcon.DumpRoute(sql);**

**if (table.Rows.Count == 0)**

**{**

**sql = "select source from Route where Dest='" + Location + "'";**

**table = objdbcon.DumpRoute(sql);**

**}**

**for (int i = 0; i < table.Rows.Count; i++)**

**{**

**DropDownList3.Items.Add(table.Rows[i].ItemArray[0].ToString());**

**}**

**sql = "select dist from Distance where source='" + Location + "' and Dest ='" + DropDownList3.SelectedItem.Text + "'";**

**string dist = objdbcon.DataRead(sql);**

**objdbcon.dr.Close();**

**if (dist == null)**

**{**

**sql = "select dist from Distance where Dest='" + Location + "' and source ='" + DropDownList3.SelectedItem.Text + "'";**

**dist = objdbcon.DataRead(sql);**

**}**

**DropDownList4.Items.Add(dist.ToString());**

**}**

**}**

**protected void DropDownList3\_SelectedIndexChanged(object sender, EventArgs e)**

**{**

**string Dis = DropDownList3.SelectedItem.Text;**

**sql = "select dist from Distance where source='" + Location + "' and Dest ='"+Dis+"'";**

**string dist =objdbcon.DataRead(sql);**

**objdbcon.dr.Close();**

**if (dist == null)**

**{**

**sql = "select dist from Distance where Dest='" + Location + "' and source ='" + Dis + "'";**

**dist = objdbcon.DataRead(sql);**

**}**

**DropDownList4.Items.Add(dist.ToString());**

**}**

**protected void Button1\_Click(object sender, EventArgs e)**

**{**

**sql = "update Distance set Road ='" + DropDownList1.SelectedItem.Value +"',Traffic='"+ DropDownList2.SelectedItem.Value +"'where source='" + Location + "' and Dest ='" + DropDownList3.SelectedItem.Text + "'";**

**objdbcon.DataExcute(sql);**

**sql = "select dist from Distance where source='" + Location + "' and Dest ='" + DropDownList3.SelectedItem.Text + "'";**

**string dist = objdbcon.DataRead(sql);**

**objdbcon.dr.Close();**

**if (dist == null)**

**{**

**sql = "select dist from Distance where Dest='" + Location + "' and source ='" + DropDownList3.SelectedItem.Text + "'";**

**dist = objdbcon.DataRead(sql);**

**}**

**DropDownList4.Items.Add(dist.ToString());**

**}**

**protected void Button2\_Click(object sender, EventArgs e)**

**{**

**try**

**{**

**objNetwok.FuzzyvalueCalculation(Location, table);**

**SqlConnection con = new SqlConnection("server=SHAHUL;Database=FLGA;Integrated Security =true");**

**SqlCommand cmd = new SqlCommand();**

**cmd.Connection = con;**

**string str;**

**str = "insert into fuzzyvalues values(@Route,@Distance,@RoadCondition,@TrafficCondition)";**

**con.Open();**

**{**

**cmd.Parameters.Add(new SqlParameter("@Route", DropDownList3.SelectedValue));**

**cmd.Parameters.Add(new SqlParameter("@Distance", DropDownList4.SelectedValue));**

**cmd.Parameters.Add(new SqlParameter("@RoadCondition", DropDownList1.SelectedValue));**

**cmd.Parameters.Add(new SqlParameter("@TrafficCondition", DropDownList2.SelectedValue));**

**cmd.CommandText = str;**

**cmd.ExecuteNonQuery();**

**con.Close();**

**}**

**}**

**catch(Exception ex)**

**{**

**Response.Write("USER CAN'T ACCESS THIS PROCESS");**

**}**

**}**

**}**

**MAP.ASPX.CS:**

**using System;**

**using System.Data;**

**using System.Configuration;**

**using System.Collections;**

**using System.Web;**

**using System.Drawing;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**public partial class \_Default : System.Web.UI.Page**

**{**

**DBConnection objdbcon = new DBConnection();**

**//SqlDataAdapter sqlda = new SqlDataAdapter();**

**//SqlCommand sqlcmd;**

**DataTable table, table2;**

**string sql, Area;**

**Pen pen = new Pen(Brushes.Red, 3);**

**CircleHotSpot c;**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**sql = "select Area from Location";**

**table = objdbcon.DumpRoute(sql);**

**sql = "select a.Point,b.Area from Table2 a inner join Location b on a.ANO=b.ANO";**

**table2 = objdbcon.DumpRoute(sql);**

**if (!IsPostBack)**

**{**

**for (int j = 0; j < table2.Rows.Count; j++)**

**{**

**c = new CircleHotSpot();**

**Area = table2.Rows[j].ItemArray[1].ToString();**

**string[] rec = table2.Rows[j].ItemArray[0].ToString().Split('-');**

**c.AlternateText = Area;**

**c.Radius = 15;**

**c.X = int.Parse(rec[0]);**

**c.Y = int.Parse(rec[1]);**

**c.NavigateUrl = "ControlCenter\_Login.aspx?id=" + Area;**

**ImageMap1.HotSpots.Add(c);**

**}**

**}**

**}**

**protected void ImageMap1\_Click1(object sender, ImageMapEventArgs e)**

**{**

**}**

**}**

**PATH.ASPX.CS:**

**using System;**

**using System.Data;**

**using System.Configuration;**

**using System.Collections;**

**using System.Web;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**public partial class \_Default : System.Web.UI.Page**

**{**

**DataSet ds = new DataSet();**

**DataSet ds1 = new DataSet();**

**DataTable table = new DataTable();**

**ArrayList arryOptimalAdd = new ArrayList();**

**FuzzyController objFuzzyCont = new FuzzyController();**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**if (!Page.IsPostBack)**

**{**

**table.Columns.Add("SINo");**

**table.Columns.Add("Route");**

**table.Columns.Add("Distance");**

**table.Columns.Add("Optimal Value");**

**Best(NetworkCondition.arryOptimzedBestPath);**

**arryOptimalAdd.Sort();**

**ds.Tables.Add(table);**

**DataSet dst = new DataSet();**

**DataRow row1, row2;**

**NetworkCondition.dstab = objFuzzyCont.GetBestOptimalRoute(NetworkCondition.arryOptimzedPath, ds, arryOptimalAdd);**

**objFuzzyCont.CheckBestpath(NetworkCondition.dstab);**

**GridView1.DataSource = ds;**

**GridView1.DataBind();**

**Best(NetworkCondition.arryOptimzedSelectedPath);**

**GridView2.DataSource = table;**

**GridView2.DataBind();**

**}**

**}**

**public void Best(ArrayList ListV)**

**{**

**ArrayList arryList = new ArrayList();**

**arryList = ListV;**

**table.Clear();**

**for (int i = 0; i < arryList.Count; i++)**

**{**

**DataRow row = table.NewRow();**

**string[] Route = arryList[i].ToString().Split('\_');**

**string[] route = Route[0].Split('-');**

**string R = " ";**

**for (int j = 1; j < route.Length - 1; j++)**

**{**

**R = R + "-" + route[j];**

**}**

**row[0] = i;**

**row[1] = R;**

**row[2] = route[route.Length - 1];**

**row[3] = Route[1];**

**table.Rows.Add(row);**

**arryOptimalAdd.Add(float.Parse(Route[1]));**

**//CheckBoxList1.Items.Add(NetworkCondition.arryOptimzedPath[i].ToString());**

**}**

**}**

**protected void Menu1\_MenuItemClick(object sender, MenuEventArgs e)**

**{**

**}**

**protected void Button1\_Click(object sender, EventArgs e)**

**{**

**Response.Redirect("FinalPath.aspx");**

**}**

**protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)**

**{**

**}**

**protected void GridView2\_SelectedIndexChanged(object sender, EventArgs e)**

**{**

**}**

**}**

**TOURISUM.ASPX.CS:**

**using System;**

**using System.Data;**

**using System.Configuration;**

**using System.Collections;**

**using System.Web;**

**using System.Web.Security;**

**using System.Web.UI;**

**using System.Web.UI.WebControls;**

**using System.Web.UI.WebControls.WebParts;**

**using System.Web.UI.HtmlControls;**

**public partial class \_Default : System.Web.UI.Page**

**{**

**DBConnection objdbcon = new DBConnection();**

**//SqlDataAdapter sqlda = new SqlDataAdapter();**

**//SqlCommand sqlcmd;**

**DataTable table, table2;**

**string sql, Area;**

**CircleHotSpot c;**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**sql = "select Nation from World";**

**table = objdbcon.DumpRoute(sql);**

**sql = "select a.Point,b.Nation from Table1 a inner join World b on a.ID=b.ID";**

**table2 = objdbcon.DumpRoute(sql);**

**if (!IsPostBack)**

**{**

**for (int j = 0; j < table2.Rows.Count; j++)**

**{**

**c = new CircleHotSpot();**

**Area = table2.Rows[j].ItemArray[1].ToString();**

**string[] rec = table2.Rows[j].ItemArray[0].ToString().Split('-');**

**c.AlternateText = Area;**

**c.Radius = 20;**

**c.X = int.Parse(rec[0]);**

**c.Y = int.Parse(rec[1]);**

**c.NavigateUrl = "World.aspx?id=" + Area;**

**ImageMap1.HotSpots.Add(c);**

**}**

**}**

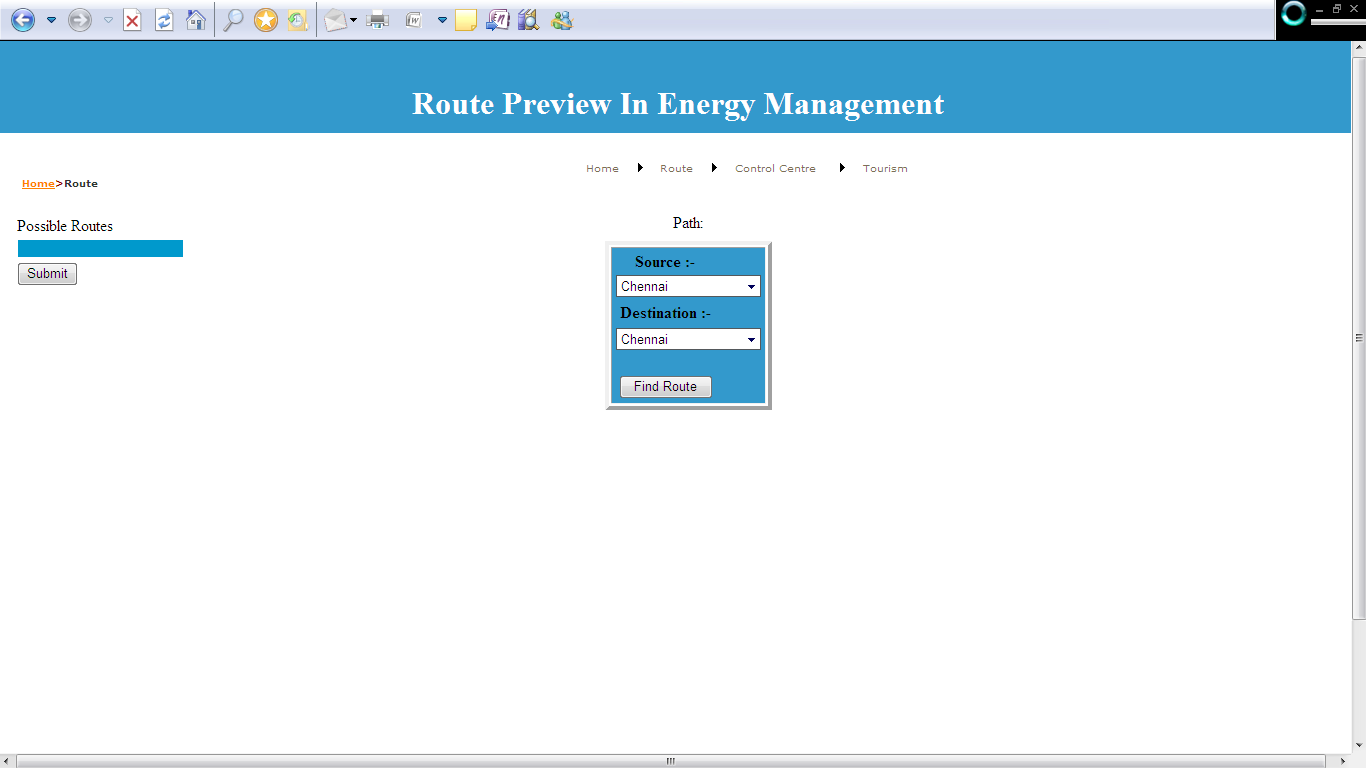
**}**

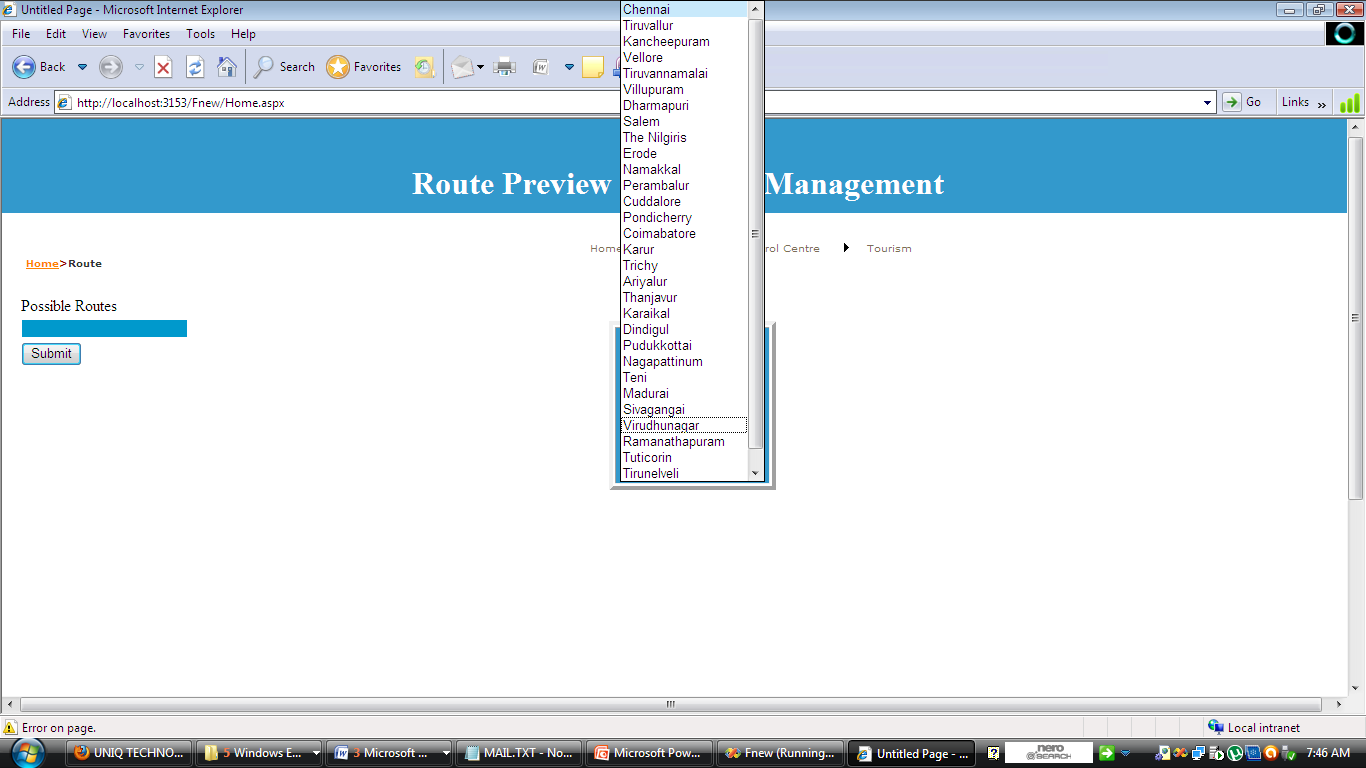
**CHAPTER 9**

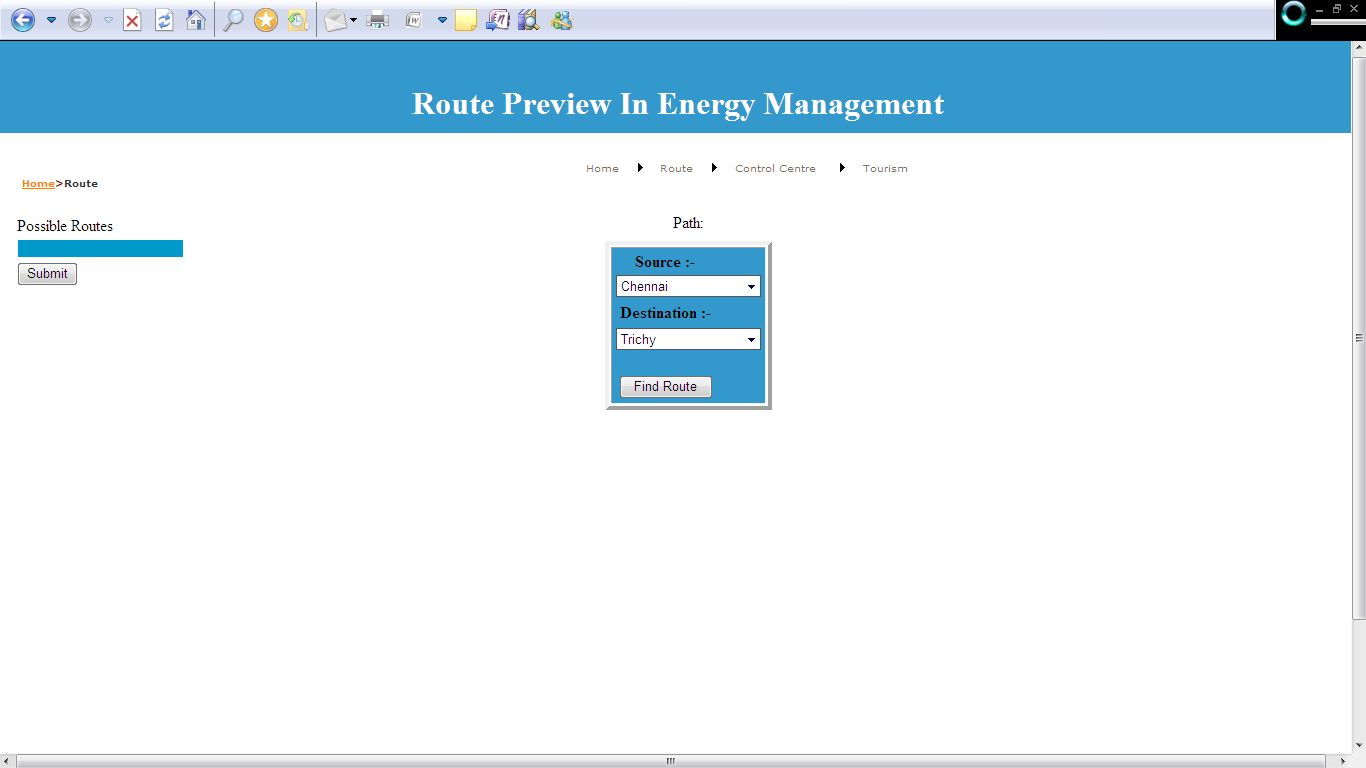
**SNAPSHOTS**

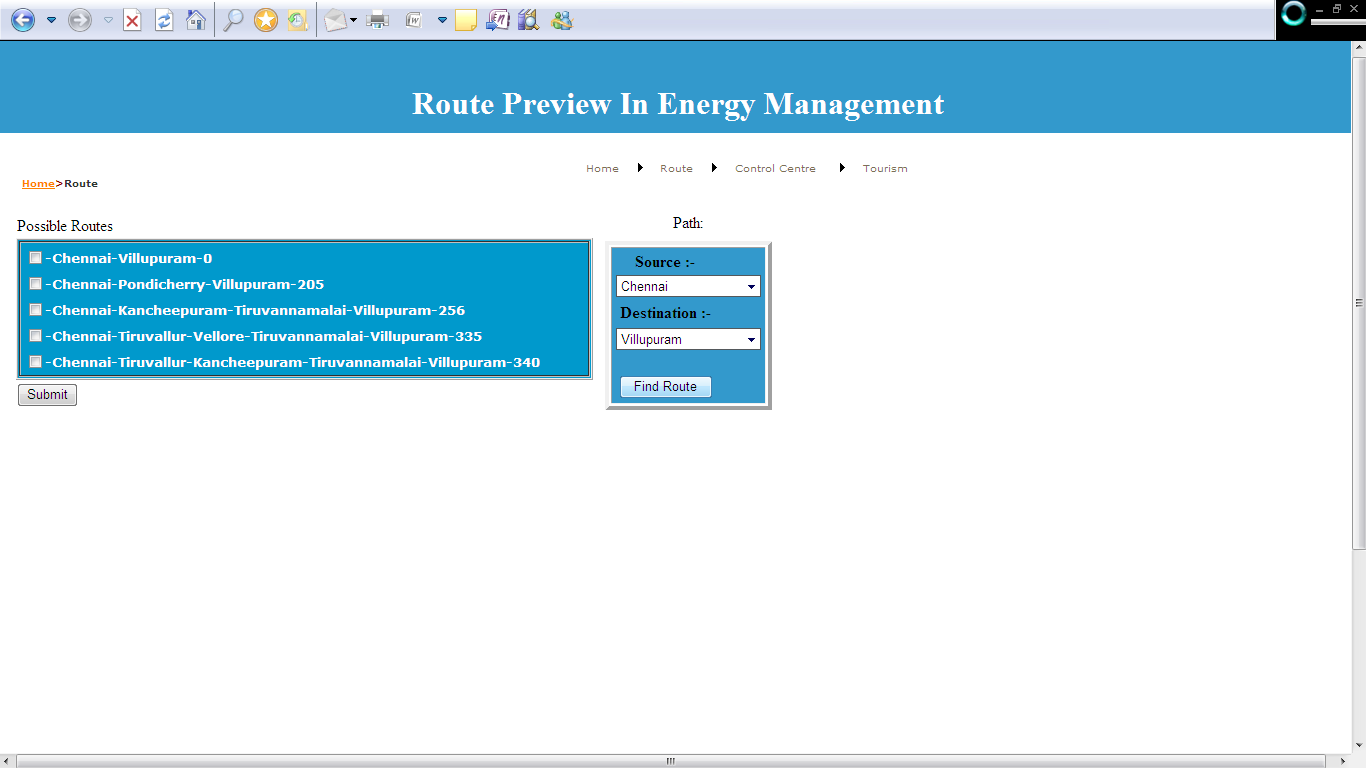
Snapshot is nothing but every moment of the application while running. It gives the clear elaborated of application. It will be useful for the new user to understand the progress of the testing.

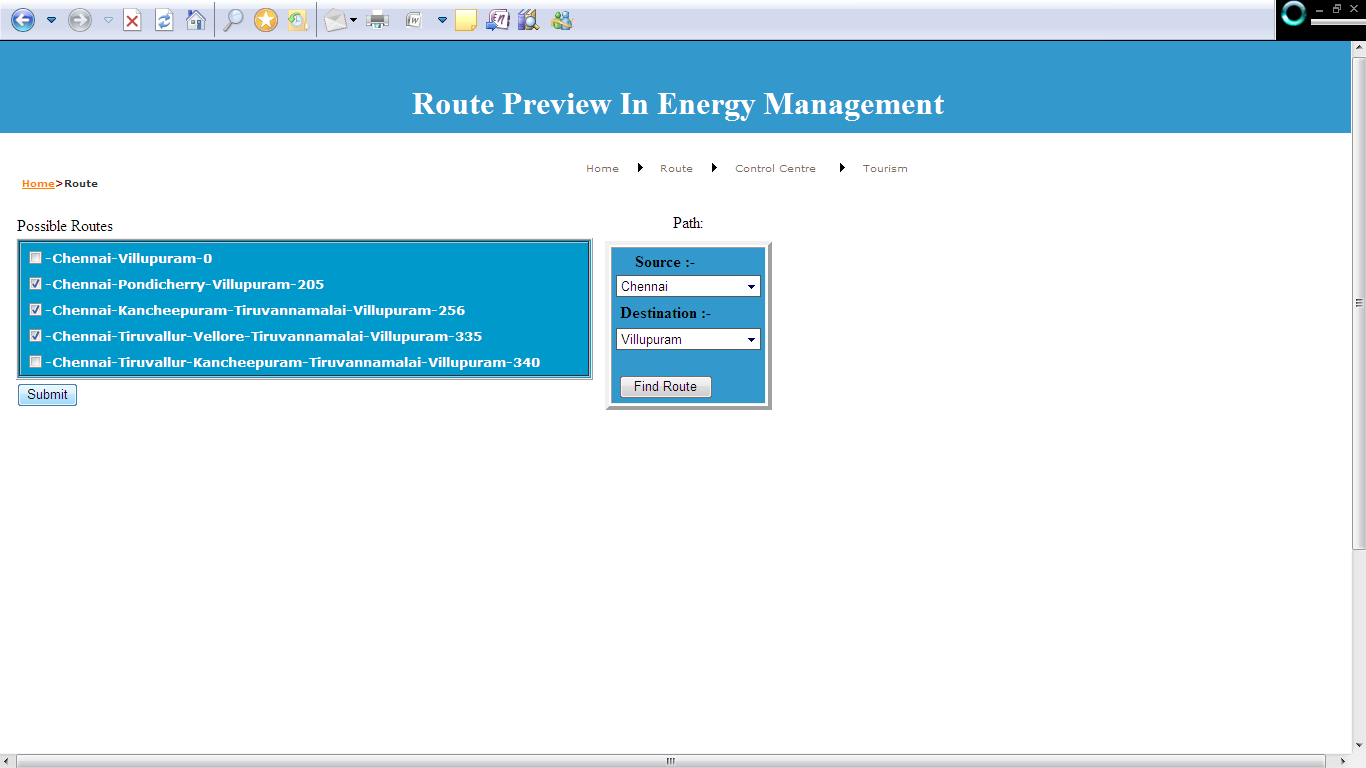
**9.1 VARIOUS SNAPSHOTS**

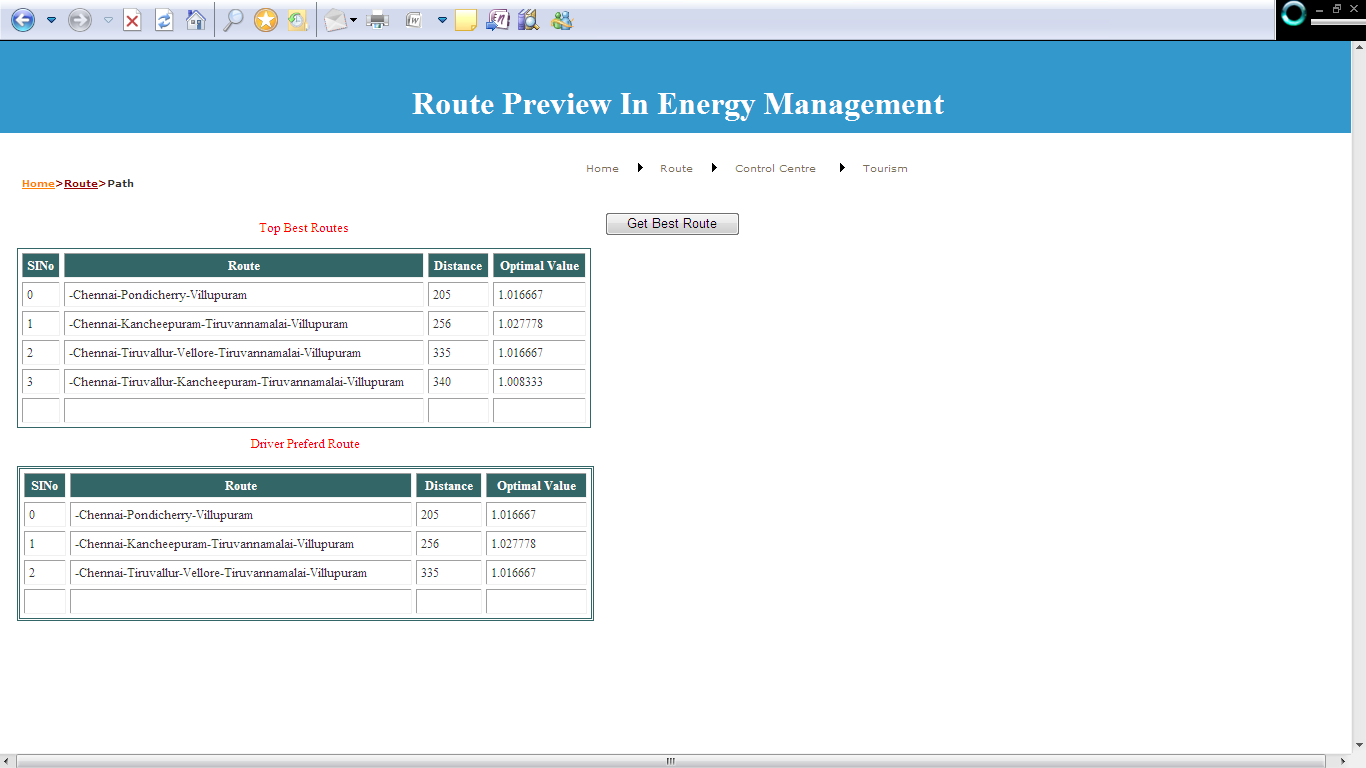


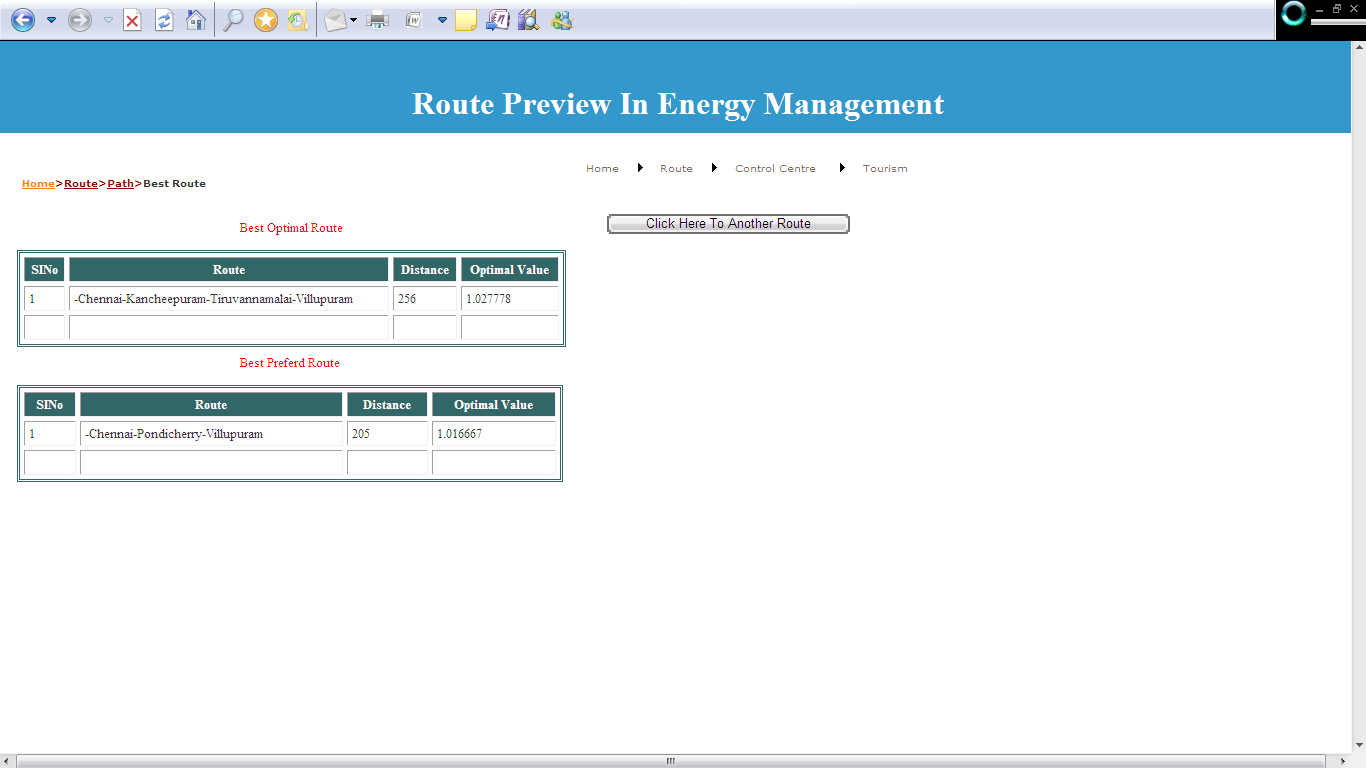












**CHAPTER 10**

**CONCLUSION**

FL was conceived as a better method for sorting and handling data but has proven to be a excellent choice for many control system applications since it mimics human control logic. It can be built into anything from small, hand-held products to large computerized process control systems. It uses an imprecise but very descriptive language to deal with input data more like a human operator. It is very robust and forgiving of operator and data input and often works when first implemented with little or no tuning.

**CHAPTER 11**

**FUTURE ENHANCEMENTS**

In this paper we considers not only the cost due to the total traveling distance, but also the cost due to the total traveling time, as two objectives. We propose a stochastic search technique called FLGA to solve the problem. Based on the promising computational results obtained in this Project, the proposed model and technique will be effective for industries to be applied in solving real-world problems.

We can calculate the optimal value using fuzzy logic and Genetic algorithm. And the value is also calculated based on control center feedback. In future we can also get the current road condition from the internet and find the optimal value to get the shortest route.

**REFERENCES**

[1] A. Paz and S. Peeta, “Information-based traffic control strategies consistent

with estimated driver behavior,” in *Proc. 86th Annu. Meeting Transp.*

*Res. Board*, Washington, DC, Jan. 2007a. CD-ROM.

[2] A. Paz and S. Peeta, “Network-level evaluation of real-time behaviorconsistent

traffic routing strategies,” in *Proc. 86th Annu. Meeting Transp.*

*Res. Board*, Washington, DC, Jan. 2007b. CD-ROM.

[3] D. Simon, “H-infinity estimation for fuzzy membership function optimization,”

*Int. J. Approx. Reason.*, vol. 40, no. 3, pp. 224–242, Nov. 2005.

[4] D. Simon and H. El-Sherief, “Hybrid Kalman/minimax filtering in phaselocked

loops,” *Control Eng. Pract.*, vol. 4, no. 5, pp. 615–623, 1996.

[5] D. Simon, “From here to infinity,” *Embedded Syst. Program.*, vol. 14,

no. 11, pp. 20–32, 2001.