MODELLING OF AN OPTIMAL ROUTE

Algirdas Sokas
Vilnius Gediminas Technical University, Department of Engineering Graphics,
Saulėtekio al. 11, LT-10223 Vilnius, Lithuania
E-mail: algirdas.sokas@fm.vgtu.lt

Abstract. This article analyzes problems of determining the shortest way and optimal route among given towns. The model of the problem is presented as non-directional graph, where nodes are towns and crossings outside towns, and edges are roads among towns and crossings. Each node has some information attached to it: name and size of the town, maintenance organizations, and mark of the crossing. All towns are connected by roads. These roads are shown as graph edges. Each edge also has information sketched in: type of road, length of road, allowable speed on the road and other useful information. Retrieval of an optimal route is executed in two stages: finding shortest way between two towns and optimizing route to visit some towns. Algorithms to both exercises’ solutions are presented. Floyd-Warshall algorithm is selected for finding shortest way from one graph node to another selected node. Nearest neighbour algorithm is selected for route optimization. Graphical system for selected towns on Lithuanian map finds an optimal route. A solution is presented in graphical form and a list of route’s towns is written with length and time of the route. The program is written in Visual Basic language. It consists of main program’s dialog window and three class modules: points, roads and routes, which have some properties and methods. The program controls database with two tables: points and roads. Obtained results are discussed and conclusions are made.

Keywords: ActiveX Data Objects technology, graph model, object-oriented programming, optimal route.

1. Introduction

Various geographical information systems, which we can obtain, are different in their possibilities and price. Generally all systems can find the shortest way between two towns but cannot find an optimal route considering time and fuel expenditures. The goal of the task is to determine traveler’s shortest route that stops at given towns, but each town can be visited only one time.

Such task in the graph theory [1–2] is called traveling salesman problem. Problem is that a full graph with \( n \) nodes has \((n - 1)! \) cycles. In our case the graph with 50 nodes has \( 6 \cdot 10^{62} \) cycles. Obviously modern personal computer is incapable of analysis that number of cycles. So we refuse to find most optimal solution and try to search only for an optimal solution.

Literature analysis shows that different transport problems are solved by using graph theory. Example, examines a multi-period capacity expansion problem for rapid transit network design. The capacity expansion is realized through the location of train alignments and stations in an urban traffic context by selecting the time periods [3]. Second example, describes an exact algorithm for solving a problem where the same vehicle performs several routes to serve a set of customers with time windows [4]. Third example, introduces a class of cuts, called reachability cuts, for the vehicle routing problem with time windows [5].

Floyd-Warshall algorithm is often mentioned for finding shortest path. Example, make use of a different shortest path computation from classical approaches in computer science graph theory to propose a new variant of the pathfinder algorithm [6]. Example, compute the shortest time paths between all pairs of variables, using the Floyd-Warshall algorithm [7]. Third example, present a cache-oblivious implementation of the Floyd-Warshall algorithm for the fundamental graph problem of all pairs shortest paths by relaxing some dependencies in the iterative version [8]. Fourth example, where time windows are disregarded in these calculations. This step is performed by a single application of the Floyd-Warshall algorithm, using the direct travel times as input [5].

In literature Nearest neighbour algorithm is praised for route optimization. Example, study a rich vehicle routing problem incorporating various complexities found in real-life applications [9]. Second, example use variable neighborhood search to solve the multi-depot vehicle routing problem with time windows [10]. Third, problem is solved using a two-phase solution framework based upon a hybridized search, within a new reactive variable neighborhood search metaheuristic algorithm [11].
This article analyzes problems of determining the shortest way and optimal route among given towns.

2. Optimal route modelling with graph

The model of the problem is presented as non-directional graph \( G = (N, E) \), where nodes are towns and crossings outside towns \( N = (1, \ldots, n) \), and edges are roads among towns and crossings \( E = (1, \ldots, m) \). Each node has some information attached to it: name and size of the town, maintenance organizations, and mark of the crossing. All towns are connected by roads. These roads are shown as graph edges. Each edge also has information sketched in: type of road (main, country, and district), length of road, allowable speed on the road and other useful information.

Retrieval of an optimal route is executed in two stages: finding shortest way between two towns and optimizing route to visit some towns. Algorithms to both exercises’ solutions are presented.

2.1. Algorithms of shortest way between two towns

Literature presents several algorithms which find shortest way between two points from concrete graph node to all the other ones. They are Dijkstra, Bellman-Ford, Johnson, Floyd-Warshall algorithms. Floyd-Warshall algorithm is the simplest and fastest [2].

Floyd-Warshall algorithm is selected for finding shortest way and optimal route among given towns. Algorithms to both exercises’ solutions are presented.

2.2. Optimization of optimal route

Start and return towns are selected. These towns can be different. Towns to be visited on the way are indicated. Several algorithms that find the shortest route are found in literature: Traveling salesman problem, Nearest neighbour algorithm, Economy connection algorithm.

Nearest neighbour algorithm is selected for route optimization. From start town search nearest town from selected to visit. Then select nearest town, then from him again search nearest other town and in such way from all selected to visit towns.

There are start, end and transitional towns, all together \( i \) number of towns. At first, from matrix \([D]\) shortest distances are selected for these towns according to towns index \( ID \) (Fig 4) and travel distance \([TD]\) is written to the matrix. Then distance between specific towns is tested for shortest route.

The algorithm is realized in the article presented example third class Map Route (Fig 3) method Nearest Neighbour.

3. Object-oriented programming

Object-oriented programming as Visual Basic (VB) greatly facilitates a programmer’s work because task are divided into three parts, into three class objects.

3.1. Class objects

The first class Map Points (Fig. 1) have two methods. Method Create designs class object, which has following properties: town name, its coordinates and other field names as in the database Points. The second method Draw shows towns and crossings in the map model.

```
Public Sub Draw()
    If PTType = 1 Then
        If ID = Form1.Combo1.ListIndex + 1 Or ID = _
Form1.Combo2.ListIndex + 1 Or Form1.List1. _
Selected(ID - 1) = True Then
            Form1.Picture2.ForeColor = QBColor(12)
            Form1.Picture2.ForeColor = QBColor(12)
Else
            Form1.Picture2.ForeColor = QBColor(12)
End If
```

Fig. 1. Class Map Points with methods and properties

\[ d_{ij}^{(k)} = \begin{cases} w_{ij}, & k = 0, \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}), & k \geq 1. \end{cases} \]
The second class *Map Roads* (Fig. 2) have two methods. Method *Create* designs class object, which has following properties: start point ID, end point ID, road index, length and other field names as in the database *Roads*. The second method *Draw* shows roads in the map model.

![MapRoads](image)

**Fig. 2.** Class *Map Roads* with methods and properties

The third class *Map Route* (Fig. 3) executes matrix operations and presents graphical result. Method *Prepare Matrices* prepares array length matrix \([D]\), which keeps graph’s shortest distances among nodes, and path matrix \([P]\), which keeps the found shortest way intermediate node numbers.

![MapRoute](image)

**Fig. 3.** Class *Map Route* with methods and properties

Method *Floyd-Warshall* realizes following algorithm:

```vbnet
Public Sub FloydWarshall()
    Dim i, j, k As Integer
    For k = 1 To n
        For i = 1 To n
            For j = 1 To n
                If (D(i, k) + D(k, j) < D(i, j)) Then
                    D(i, j) = D(i, k) + D(k, j)
                    P(i, j) = k
                End If
            Next j
        Next i
    Next k
End Sub
```

Methods *Extract Route*, *Extract Route Part* finds optimal way between presented points pathway. Method *Nearest Neighbour* realizes nearest neighbour algorithm and finds optimal route for visiting several towns. There is \(t\) number of towns. Matrix travel distance \([TD]\) is formed. Then shortest distance among specified travel towns is verified.

```vbnet
For i = 1 To t
    For j = 1 To t
        TD(i, j) = D(PointIDs(i), PointIDs(j))
    Next j
Next i
```

Methods *Show Travel Route Point* presents a graphical view of route in the map model (Fig 6).

### 3.2. Connection with the database table

Modern programming database control technology is ActiveX Data Objects (ADO), created in 1996 [12]. An example of procedure with variable *DB_Connection* can read a concrete record *DB_Recordset1* from the database *Marsrutai.mdb* table *Points* (Fig 4). To prepare VB project, an instance of the application must be created, declaring a variable that will represent the connection with other application, first line. In the second line, database record set must be declared.

```vbnet
Dim DB_Connection As ADODB.Connection
Dim DB_Recordset1 As ADODB.Recordset
DB_PATH = "C:\VB\Marsrutai.mdb"
Set DB_Connection = New ADODB.Connection
    With DB_Connection
        .Provider = "microsoft.jet.oledb.4.0"
        .Open DB_PATH
    End With
Set DB_Recordset1 = New ADODB.Recordset
DB_Recordset1.Open "Points", DB_Connection, _
    dOpenKeyset, adLockBatchOptimistic, -1
```
Fig. 4. Database table Points

The new keyword in the third line starts a new session of database, adds provider and opens the database from a file named Marsrutai.mdb. In the next to last line, new records are set.

### 3.3. Map and information

Standard VB editor’s **PictureBox** object is used for presenting the map, which paints Lithuania’s outline map as background. Towns and roads are drawn on the background. Database table **Points** information is presented in the map and form lists in such way:

```vba
Do Until DB_Recordset1.EOF
    MapPoints(i).Create DB_Recordset1(0), DB_Recordset1(1), DB_Recordset1(2), DB_Recordset1(3), DB_Recordset1(4)
    If MapPoints(i).PType = 1 Then
        Combo1.List(i) = MapPoints(i).Name
        Combo2.List(i) = MapPoints(i).Name
        List1.List(i) = MapPoints(i).Name
        List1.Selected(i) = False
    End If
    i = i + 1
    DB_Recordset1.MoveNext
Loop
```

Class **Map Points** method **Create** reads in database fields and crossing all map (graph) nodes writes towns’ names. It also fills in form’s lists.

Database table **Roads** (Fig 5) information is also presented on the map:

```vba
Do Until DB_Recordset2.EOF
    MapRoads(i).Create DB_Recordset2(0), sp_x, sp_y, DB_Recordset2(1), DB_Recordset2(2), DB_Recordset2(3), DB_Recordset2(4), DB_Recordset2(5)
    i = i + 1
    DB_Recordset2.MoveNext
Loop
```

### 4. Example, optimal route in the Lithuania map

Graphical system for selected towns on Lithuanian map finds an optimal route. A solution is presented in graphical form (Fig 6) and a list of route’s towns is written with length and time of the route.

The program is written in Visual Basic language. It consists of main program’s dialog window and three class modules: points, roads and routes, which have some properties and methods. The program controls database with two tables: points and roads.

Fig. 5. Database table Roads

Fig. 6. Modeling system and optimal route Vilnius–Vilnius with visited towns Sirvintos, Anyksciai, Utena, Pabrade
In the selection lists of the program form we indicate travel start and end towns, select other visit towns. Travel start and end towns can be the same town, as presented in example (Fig 6), but can also be different towns (Fig 7). After pushing programs execute key, the form is presented with optimal route with list of towns, distance, time, and the graphical window shows path with accentuated line.

Fig 7. Optimal route Klaipeda–Mazeikiai with visited towns Silute, Silale, Telsiai, Siauliai, and Skuodas

The system is open and is possible to expand, append database with new towns, crossings and roads.

5. Conclusions

Solved problems of determining the shortest way and optimal route among given towns.

Floyd – Warshall algorithm is selected for finding shortest way from one graph node to another selected node. Nearest neighbour algorithm is selected for route optimization. These algorithms’ realization is presented programmatically.

Object-oriented programming language, classes with specific properties and methods allows writing a program with individual modules, which simplifies and clarifies programmer’s work.

Three classes are defined: points, roads and routes. The points and roads selected from the database and presented in the outline map. The routes designed according to the mathematical model with class methods and properties.

Designing systems’ connection with databases is necessary. Graphical objects selected from objects’ informational tables. Such tables can easily be written to the database tables and the program automatically finds the right parameters of element.

A programming language and graphical objects controlled by the language are required for design of such systems. For example, Visual Basic programming language works with the PictureBox objects.

The presented program is practical and clear for using, easy to select start, end and intermediate towns. Accessible result is clear and visual.

References