Tourism in Lebanon: An Intelligent Geographical Information System Interactive Hypermap Tool

Diana A. Nahle
B.Sc., American University of Beirut

Project
submitted in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science
at the Lebanese American University
Beirut, Lebanon
February 1998

Signatures Redacted

Dr. Issam Moghrabi (Advisor)
Assistant Professor of Computer Science
Lebanese American University

Signatures Redacted

Dr. Ramzi Haraty
Assistant Professor of Computer Science
Lebanese American University

Acknowledgments

I would like to thank Dr. Issam Moghrabi, my advisor, for his encouragement and constructive comments. I am also very grateful to Dr. Ramzi Haraty, committee member in my MS project defense.

A special thank you to the Academic Computer Center staff.

I also thank all my friends. A special thank you to Mr. Ahmad Solh, Mr. Abbas Tarhini, Mr. Raed Ayyoub, and Mr. Sami Souki for their help.

Special thanks to my family, especially my mother, for their continuous support.

Abstract

The work presented herein focuses on the development of a novel spatial tourist-servicing system. It provides its users with a Geographical Information System hypermap navigation information that provides visitors, given the start time and places of interest, with a navigation plan. This plan specifies routes to go through, suggested stops such as hotels in the cities/towns they wish to visit, and associated durations by taking into consideration the traffic along the roads traversed. The production of such a plan is based on the adoption of some shortest path algorithm and traveling salesperson problem approximation reformulated for efficiency of implementation.

Table of Contents

Chapter 1

Introduction

This chapter provides a general description GIS and the product developed using it. It concludes by giving a brief description of each chapter.

Since GIS is used to capture, store, edit, manipulate, analyze, synthesize and display geographically referenced information, it has been used in this work to produce the front end, the base map of Lebanon. The user starts by selecting arcs on the displayed map and based on these selections, the tool is geared to find the shortest path between any pair of selected arc nodes. The shortest path calculations are then used as input to get the tour formulation having visited all selected nodes. The tour starts from the first selected node (starting node) and visits all other selected nodes, returning finally to the starting node. The list of hotel names or classes (international, 4 stars, etc.) in the selected nodes (excluding the starting node) are displayed. If the list of hotel names is provided, then by clicking a hotel name, multimedia information about it is then displayed. If the list of hotel classes is displayed, the user clicks on a hotel class to get the list of hotel names in that class. He/she then clicks on the hotel name to view its pictures, video and information about it.

In chapter 2, the meaning of a Geographic Information System and its capabilities are discussed. The ARC/INFO GIS software is also discussed. In chapter 3, data collection of hotel information and the data export process are provided. In chapter 4, the touristic multimedia system is described. The data

structures and algorithms based on these data structures are described. Dijkstra's algorithm for finding the shortest path and Nearest Insertion TSP algorithm for finding the order of visiting cities are used. The adoptions done to these algorithms to suit our problem are also described. In chapter 5, experimental results to demonstrate the operation of the algorithm are listed. In chapter 6, the conclusion and suggestions for further work are provided.

Chapter 2

Geographic Information Systems

1. Definition

A GIS is a data system for management of urban, environmental, and other planning data suitable for data analysis, plan preparation, decision making, scientific investigation, resource management and global change understanding. A GIS is comprised of the following components: software, hardware, data, training and administration

2. Capabilities and Powers

A GIS system has the following capabilities:

- Functioning as a decision support tool that is easily integrated into the specific environment of an organization.
- Offering users the ability to manipulate, analyze and visualize spatial and aspatial data.
- Linking data to application-based models to try to find answers to questions like: "What effect will a certain plan or its possible alternatives have on the surrounding area?"
- Converting existing digital information (not in map form) into forms it can recognize and use (e.g., digital satellite images).

3. Geographical Information System and ARC/INFO

A GIS does not hold maps and pictures - it holds a database. To go beyond just making pictures, one should know three pieces of information about every feature stored in the computer. What it is? Where it is? And how it relates to other features? Database systems provide the means of storing a wide range of such information and updating it without the need to rewrite programs as new data is entered. In the ARC/INFO GIS software, ARC handles where the features are, while INFO handles the feature descriptions and how each feature is related to others.

ARC/INFO stores the descriptive information for a feature in a tabular data file in which a record stores all the information about one occurrence of a feature (in our case, point/arc/polygon) and an item stores one type of information (i.e., attribute information) for all features in the database. These data files are known as feature attribute tables.

A map of a particular feature can be thought of as a layer of data about an area. In ARC/INFO, these layers are called coverages. ARC/INFO uses a command language that functions similar to the way computer's operating system works. Commands are entered at a prompt to perform specific tasks. It can be customized to use menus to point and click on the task to be performed.

ARC/INFO consists of the following subsystems:

1. **ARCEDIT** which is used to correct errors such as missing arc/label point(s).

2. **ARCPLOT** which is used to query data or create maps.

3. **TABLES** which is used to display and manipulate coverage attributes.

4. ARC/INFO Commands Used

ARC/INFO uses a command language. At the basic level, an operation is performed by typing in a command, along with any command arguments. Command usage refers to a listing of a command name along with its available arguments and options. Before executing commands, its usage can be easily obtained by typing the command without any of its arguments and then pressing the <enter> key.

Following is the list of ESRI ARC/INFO's commands mostly used in the preparation of the final product and a description of their usage [6].

♦ Build and Clean

These commands are used to construct topology. A summary of these commands capabilities are listed in the following table:

CAPABILITIES	BUILD	CLEAN
Processes		
Polygons	Yes	Yes
Lines	Yes	Yes
Points	Yes	No
Number features	Yes	Yes
Calculates spatial measurements	Yes	Yes
Creates intersections	No	Yes
Processing speed	Faster	Slower

♦ Editing

Editing was done using the ARCEDIT subsystem to correct errors such as missing arc(s), missing label point(s), many labels, etc.

♦ Joinitem

This command is used to physically merge the data file (having an extension ***DAT***) to the feature attribute table (having an extension ***PAT*** or ***AAT***), based on a shared item. Both the item definitions and the values of the two files are merged to create the output file. A record in the data file is matched to a record in the feature attribute table when their relate item values are equal. Then the item values from the two records are copied to the output file.

For example, we used the command to join:

1. city1.dat with **CITIES12**.pat based on cities12_id (refer to **Appendix B**).

2. city.dat with **CITIES03**.pat based on cities03_id (refer to **Appendix B**).

♦ Projection and Transformation

Map projections allow areas on the surface of the Earth (a spheroid) to be represented on a map (a flat surface) - expressing a three-dimensional surface in two dimensions. A projection is used to more precisely equate locations on a map with their true locations on Earth. This can be done by using the PROJECT command. PROJECT only copies feature attribute tables from the input coverage to the output coverage.

Input and output projections and their parameters are defined by a series of subcommands. These are organized into two groups: one defines the input projection and the other defines the output projection. PROJECT subcommands are used to define both the input and output projections. The subcommands are:

- **Input:** The subcommands following input define the input projection. It is the first subcommand issued when projection parameters are entered from a text file.
- **Projection <projection name>** (required for all projections): Available options for <projection name> are geographic, UTM, etc.
- **Units <units>** (required for all projections): Specifies the units of the coordinates. Possible units for the geographic option are: radians, DMS (degrees, minutes, seconds), DD (decimal degrees),

DM (decimal minutes), DS (decimal seconds). Possible units for other projections are feet or meters.

- **Zone <zone_number>** (optional for UTM or STATEPLANE only): The zone number for the UTM or STATEPLANE coordinate system projections. The following table lists zones, central meridians and longitude ranges for UTM. All values are in full degrees east (E) and west (W) of Greenwich (0 degrees).

Zone	Central Meridian	Range in Longitude
1	177W	180W-174W
2	171W	174W-168W

- **Xshift <distance>** or **yshift <distance>** (optional for all projections): The constant to add to the input coordinates. The use of *xshift* or *yshift* will cause a value (specified by distance) to be added to all coordinates. They are often used to subtract a value from the projected coordinate whose values are in the 3- to 6-million range (especially *y*) so that coordinate precision can be maintained during execution of various analysis program.
- **Parameters** (required for all projections): The *parameters* subcommand specifies the set of projection-specific parameters which make up the parameters package. It is the last part of each input or output section. Always specify parameters, even if no special parameters are required for the projection.
- **Output** (required for all projections): The subcommands following *output* define the output projection.

- **End** (required for all projections): Specifies the end of subcommand input. It must always conclude the list of subcommands.

Transformation is used to change from machine coordinates into real world coordinates (refer to **Appendix C** for steps followed to conduct the transformation).

♦ Mapjoin

This command appends up to 500 adjacent coverages. Once MAPJOIN is invoked, the user will be prompted to enter the names of the coverages to be appended. In our project, it was used to join the coverages **INTBND06** and **INTBND15** into one coverage **INTBND**.

The MAPJOIN command creates a new coverage containing all of features and attributes for both input coverages. It takes as argument, in addition to the output coverage <out_cov>, the set of features to be joined (poly/net):

- ***Poly*** is used to join for polygon features (used in my case for international boundary coverage - **INTBND**).
- ***Net*** is used to join arc and polygon feature coordinates and attributes.

♦ Append

Since MAPJOIN should not be used to join point or line features, the APPEND command with point/line option can be used to join point or

line coverages. This command does not automatically rebuild topology. BUILD or CLEAN must be used after it.

For example, this command was used to join the **CITIES05** and **CITIES14** to form **CITYCOV**. It was also used to join all road coverages into one (as described in **Appendix B** Sections 2 and 3 of this report).

Chapter 3

Data Collection and Export

1. Data Collection

"Lebanon Hotel Guide 1996" (refer to **Appendix D**) lists five of the six mohafazat (Beirut, Mount Lebanon, North, South, Nabatieh and Bekaa). Nabatieh is not considered a separate mohafazat in the guide. Its hotels are included among the hotels in the South.

For each mohafazat in this guide, the list of hotels was collected. For each hotel, information such as name, category (international, 4 stars, 3 stars, 2 stars and 1 star), telephone/fax/telex and number of rooms is included.

Field investigation was necessary for updating the 1996 guide and hence new hotels were added to our list as well as noting that old ones closed or under renovation. For example, "King's Hotel" - Raouche is currently closed for renovation. "Cedarland" - Hamra is rent by the American University Hospital (AUH) for doctors' use. "Cottage Club" in Nabatieh is not a hotel but a club. "Globus" - Qlaiaat is closed, etc.... For some of the hotels, it was possible to record video scenes while in others, it was only possible to get its brochures and scan those using the Deskscan II software.

2. Data Export

The computations carried out by the algorithms require roads to be stored and read as arcs and hence to export the roads table in ARCVIEW to a text file. To relieve the user from repeating this process if the table is updated or the map changed, the avenue script which calls the Delphi software to execute the algorithms was modified to export the roads table before execution.

Chapter 4

A Touristic Multimedia System

1. General Description

The purpose of this project is to provide its users with a GIS hypermap navigation tool that provides its users, given the start time and places of interest, with a navigation plan. The plan specifies routes to go through, suggested stops such as hotels in the cities/towns they wish to visit and associated durations by taking into consideration the traffic along the roads traversed. Such plan is the by-product of intelligent search algorithms that aim at maximizing the number of visited places while minimizing the travel time, subject to certain time constraints prescribed by the user.

The user selects arcs on the displayed map of Lebanon. Based on these selections, the tool is built to find the shortest path between any pair of selected arc nodes. The shortest path calculations are then used as an input to get the tour formulation having visited all selected nodes. The tour starts from the first selected node (starting node) and visits all other selected nodes, returning finally to the starting node. The list of hotel names or classes (international, 4 stars, etc.) in the selected nodes (excluding starting) are displayed. If the list of hotel names is provided then by clicking a hotel name, its picture or video, in addition to information about it are displayed. If the list of hotel classes is displayed, the user clicks on a hotel class to get the list of hotel names in

that class. He/she then clicks on the hotel name to view its pictures, video and information about it.

The system may, given the user start time, provide him/her with a visit plan. It takes into account the traffic at specific hours (input randomly since no data was available) of the day along the path prescribed to estimate travel duration.

2. Implementation

The project was implemented on a GIS platform and ultimately presented using ARCVIEW, Borland Delphi Desktop 2.0 and the Director multimedia package. Using ARC/INFO, each map coverage was digitized as a point, arc or polygon coverage. After digitizing each coverage, the data in the coverage just digitized was examined to ascertain that it is free of spatial errors (features in the right place and shape), features that connect actually do, all polygons have one and only one label point, and all features are within the outer map boundary. Errors found were corrected using ARCEDIT (an ARC/INFO tool). Then, descriptive attributes (if any) were added. To do so, a new data file with ***DAT*** extension was created, using the TABLES subsystem in ARC/INFO, to hold attributes. The attributes were added to the data file which was later joined to the feature attribute table of the coverage using the JOINITEM command. The step to follow was to use the PROJECT command to change from 3D to 2D and the TRANSFORM command to change from machine coordinates into real world coordinates. Next, using the EDGEMATCH command, the arc nodes of the two adjacent coverages (same coverage in both tiles e.g., EDGEMATCH main roads in **Tile 1** with main roads in **Tile 2**) were matched along the boundary of

both coverages to form one joining node thus making the two arcs as one. Finally, MAPJOIN or APPEND was used to join the two coverage tiles into one tile.

The map is then refined and prepared to serve as the ultimate front-end in ARCVIEW. Different colors and symbols are used to distinguish the various features on the map. For example, cities are represented as purple circles. By clicking on the source and destination cities, the user is provided with the path to follow in the tour. This is coupled with a displayed list of hotels in the selected cities. The user can then select a specific hotel to visualize multimedia links. If a list of hotel categories appears, then choosing a hotel category displays the list of hotels belonging to that category.

3. Data Structures

The straightforward implementation of Dijkstra is preferable if the graph is dense; whereas it is preferable to use heaps if the graph is sparse (refer to Section 4.1 of this chapter) [2]. Since our work is based on the map of Lebanon with 2512 arcs each of which is connected to a maximum of four other arcs (i.e., the map's graph is sparse), a brief overview of heaps (complete binary tree) and binary trees is provided.

3.1 Binary Trees

A binary tree is a *search tree* if the value contained in every internal node is larger than or equal to the values in its left-hand descendants, and less than or equal to the values contained in its right-hand

descendants. A binary tree is essentially complete if each of its internal nodes possesses exactly two children, one on the left and one on the right, with the possible exception of a unique special node situated at level 1, which possesses only a left-hand child and no right-hand child. Moreover, all the leaves are either on level 0, or else they are on levels 0 and 1, and no leaf is found on level 1 to the left of an internal node at the same level. The unique special node, if it exists, is to the right of all the other level 1 internal nodes. This kind of tree can be represented using an array T putting the nodes of depth k, from left to right, in the positions $T[2^k]$, $T[2^k+1]$, ..., $T[2^{k-1}-1]$ (with the possible exception of level 0, which may be incomplete) [2].

3.2 Heaps

A heap is essentially a complete binary tree, each of whose nodes includes an element of information called the value of the node. The heap property is that the value of each internal node is greater than or equal to the values of its children.

The fundamental characteristic of this data structure is that the heap property can be restored efficiently after modification of the value of a node. If the value of the node increases, so that it becomes greater than the value of its parent, it suffices to exchange these values and then to continue the same process upwards in the tree until the heap property is restored. The modified value has been percolated up to its new position. If, on the contrary, the value of a node is decreased so that it becomes less than the value of at least one of its children, it suffices to exchange the modified value with the larger of the values in the children, and then

to continue this process downwards in the tree until the heap property is restored. The modified value has been sifted-down to its new position.

The heap is an ideal data structure for finding the largest element of a set, removing it, adding a new node, or modifying a node [2].

4. Suggested Algorithms For Navigation Plan Determination

The user plans to visit a number of cities. The user chooses them on the map and gets the path to follow in his/her tour to visit these cities having travelled the shortest distance possible. Thus, to do so, it is necessary to find the shortest path between any two cities and based on this, produce the order in which to visit the cities.

4.1 Shortest Path

In a weighted graph or network, it is frequently desired to find the shortest path between two nodes (cities), *s* and *t*. The shortest path is defined as a path from *s* to *t* such that the sum of the lengths of the arcs on the path is minimal. To represent the network, *L[i,j]* is used to represent the length of the arc from *i* to *j*. If there is no arc from *i* to *j*, then *L[i,j]* is set to an arbitrary large value to indicate the infinite cost of going directly from *i* to *j*.

If all lengths are positive, the following algorithm, due to Dijkstra(*s,t*), determines the shortest path from *s* to *t*. The variable *distance[i]* keeps the cost of the shortest path known thus far from *s* to *i*. Initially,

distance[s] is 0 and *distance[i]* equals ∞, for all *i* not equal to *s*. A *checked* set contains all nodes whose minimal distance from *s* is known: that is those nodes whose *distance* value is permanent and will not change. If a node *i* is in *checked*, then *distance[i]* is the minimal distance from *s* to *i*. Initially, *checked* equals *[s]*. Once *t* becomes a member of *checked*, *distance[t]* is known to be the shortest distance from *s* to *t*, and the algorithm terminates.

The algorithm maintains a variable, *current*, that is the node that has been added to *checked* most recently. Initially, *current* equals *s*. For every successor *i* of *current*, if *distance[current]* + *L[current,i]* is less than *distance[i]*, the distance from *s* to *i* through *current* is smaller than any other distance from *s* to *i* found thus far. Thus, *distance[i]* must be reset to this smaller value.

Once *distance* has been recomputed for every successor of *current*, then *distance[j]* (for any *j*) represents the shortest path from *s* to *j* that includes only members of *checked* (except for *j* itself). This means that for the node *k*, not in *checked*, for which *distance[k]* is smallest, there is no path from *s* to *k* whose length is shorter than *distance[k]* (*distance[k]* is already the shortest path to *k* that includes only nodes in *checked*, and any path to *k* that includes a node *nd* as its first node not in *checked* must be longer, since *distance[nd]* is greater than *distance[k]*). Thus, *k* can be added to *checked*. *Current* is then reset to *k* and the process is repeated. At the end of the algorithm, the variable *d* returns the shortest distance from *s* to *t*.

In addition to calculating distances, the algorithm finds the shortest path itself by maintaining an array p such that $p[i]$ is the node that precedes node i on the shortest path found thus far.

The time complexity of Dijkstra's algorithm is $O(n^2)$ for a graph of n vertices [1].

The pseudo code for the Dijkstra's algorithm is as follows:

```
checked ← s
for i ← 1 to n do distance[i] ← maximum integer
distance[s] ← 0
current ← s
while (current <> t)
{ smalldist ← maximum integer
  for i ← 1 to n
  { if i not in checked
        if (distance[current] + L[current,i]) < distance[i]
           { distance[i] ← distance[current] + L[current,i]
             p[i] ← current }
        if distance[i] < smalldist
           { smalldist ← distance[i]
             k ← i }
    }
    current ← k
    checked ← checked + [current]
}
d ← distance[t]
```

A more efficient straightforward algorithm is stated below.

The sets C and S are the set of available candidate nodes and the set of nodes already chosen respectively. At every moment, S contains those nodes whose minimal distance from the source is already known, whereas C contains all the others. At the outset, S contains only the source itself: when the algorithm ends, S contains all the nodes of the graph and our problem is solved. At each step, the node in C, whose distance to the source is least, is chosen and added to S.

A path from the source to some other node is special if all the intermediate nodes along the path belong to S. At each step of the algorithm, an array *distance* contains the length of the shortest special path to each node of the graph. At the moment, a new node v is added to S, the shortest special path to v is also the shortest of all the paths to v. When the algorithm ends, all the nodes of the graph are in S, and hence all the paths from the source to some other node are special. Consequently, the values in *distance* give the solution to the problem.

To make life simple, assume the nodes of the graph are numbered from 1 to n, $N = \{1, 2, \ldots, n\}$, that node 1 is the source, and that a matrix L gives the length of each directed edge: $L[i,j] \geq 0$ if the edge (i,j) exists and $L[i,j] = \infty$ otherwise. Here is the algorithm:

$C \leftarrow \{2, 3, \ldots, n\}$
{$S = N/C$ exists only by implication}
for $i \leftarrow 2$ to n do $distance[i] \leftarrow L[s,i]$
repeat $(n - 2)$ times

- $v \leftarrow$ some element of C minimizing *distance[v]*
- $C \leftarrow C \backslash \{v\}$ {remove v from C and implicitly add it to S }
- For each $w \in C$ do

  distance[w] $\leftarrow$ min (*distance[w]*, *distance[v]* + *L[v,w]*)

If, in addition to the length of the shortest path, it is desired to remember the path as well, then it suffices to add a second array $p[2..n]$, where *p[v]* contains the number of the node that precedes v in the shortest path. To find the complete path, simply follow the pointers p backwards from a destination to the source. The modifications to the algorithm are simple:

- initialize *p[i]* to 1 for $i = 2, 3, \ldots, n$
- replace the contents of the inner *for* loop by

  if *distance[w]* > *distance[v]* + *L[v,w]* then

  distance[w] $\leftarrow$ *distance[v]* + *L[v,w]*

  p[w] $\leftarrow v$

Suppose this algorithm (Dijkstra) is applied to a graph having n nodes and r edges. Initialization is in $O(n)$. In a straightforward implementation, choosing v in the *repeat* loop requires all the elements of C to be examined, so that we look at $(n - 1), (n - 2), \ldots, 2$ values of distance on successive iterations, giving a total time in $O(n^2)$. The time required by this version of the algorithm is therefore $O(n^2)$.

If the number of edges r is much smaller than the square of the number of nodes n^2, it seems preferable to represent the graph by an array of n lists, giving for each node its direct distance to adjacent nodes. This allows us to save time in the inner *for* loop, since we only have to

consider those nodes w adjacent to v, but how to avoid taking a time in $O(n^2)$ to determine in succession the $(n - 2)$ values taken by v? The answer is to use a heap containing one node for each element v of C that minimizes *distance[v]*. If the heap is inverted, the element v of C that minimizes *distance[v]* will always be found at the root. Initialization of the heap takes time in $O(n)$. The instruction "$C \leftarrow C\backslash\{v\}$" consists of eliminating the root from the heap, which takes time in O(log n). As for the inner *for* loop, it consists of looking, for each element w of C adjacent to v, to see whether *distance[v]* + *L[v,w]* < *distance[w]*. If so, *distance[w]* must be modified and w must be percolated up the heap, which again takes time in $O(\log_n)$. This does not happen more than once for each edge of the graph.

To sum up, the root of the heap has to be removed exactly $(n - 2)$ times and to percolate at most r nodes, giving a total time in $O((r + n)\log_n)$. If the graph is connected, $r \geq (n - 1)$ and this time is in $O(r * \log_n)$. The straightforward implementation is therefore preferable if the graph is dense, whereas it is preferable to use a heap if the graph is sparse [2].

4.2 Travelling Salesperson Problem

The shortest paths between cities calculated by Dijkstra's are used to fill the distance matrix needed for the Travelling Salesperson Problem (TSP) algorithm. The algorithm determines the tour (the order of visiting cities). The travelling salesperson leaves one of these cities (starting city), to visit each other city exactly once. At the end, having travelled the shortest total distance possible, the algorithm has to return to the starting point. All the known exact algorithms for this problem require

exponential time (NP-complete problem). Hence, it is impractical for large instances.

It is tempting to use dynamic programming to solve TSP. The methodology can be described as follows:

Let $G = <N,A>$ be a directed graph. Take $N = \{1,2,\ldots,n\}$ and the lengths of the edges are denoted by L_{ij}, with $L[i,j] = 0$, $L[i,j] \geq 0$ if i≠j, and $L[i,j] = \infty$ if the edge (i,j) does not exist.

Suppose that the circuit begins and ends at node x. It therefore consists of an edge (x,j), $j \neq x$, followed by a path from j to x that passes exactly once through each node in $N\backslash\{x,j\}$.

Consider a set of nodes $S \subseteq N\backslash\{x\}$ and a node $i \in N\backslash S$, with $i = x$ allowed only if $S = N\backslash\{x\}$. Define $g(i,S)$ as the length of the shortest path from node i to node x that passes exactly once through each node in S. Using this definition, $g(x,N\backslash\{x\})$ is the length of an optimal circuit. By the principle of optimality, we see that

$$g(x,N\backslash\{x\}) = \min_{2\leq j\leq n} (L_{xj} + g(j,N\backslash\{x,j\})). \quad (*)$$

More generally, if $i \neq x$, $S \neq \Phi$, $S \neq N\backslash\{x\}$, and $i \notin S$,

$$g(i,S) = \min_{j \in S} (L_{ij} + g(j,S\backslash\{j\})). \qquad (**)$$

Furthermore,

$$g(i,\phi) = L_{ix}, \quad i = 2, 3, \ldots, n$$

The values of $g(i, S)$ are therefore known when S is empty. Apply (**) to calculate the function g for all the sets S that contain exactly one node (other than x); then apply (**) again to calculate g for all the sets S that contain two nodes (other than x), and so on. Once that value of

$g(j,N\backslash\{x,j\})$ is known for all nodes j except node x, use (*) to calculate $g(x,N\backslash\{x\})$ and solve the problem.

The computation time is $O(n^2 * 2^n)$. The storage space required is $O(n * 2^n)$ where n is the number of cities [2]. This is too pessimistic for spatial applications where the number of involved nodes/cities tends to be large.

Another method uses a decision-tree search algorithm based on circuit elimination that depends upon finding solutions to the assignment problem with only minor modification of the cost matrix. Each of these problems can be solved easily by storing the solution to the previous problem from which it was derived. In particular, the modification involved the setting of some entry $c(x_i,x_j)$ to ∞ for an arc (x_i,x_j) which is in the current assignment problem solution [3].

If the Hungarian algorithm [3] is used to solve the above-mentioned assignment problem, the entry $c(x_i,x_j)$ in the final relative cost matrix would have had the value 0 and an associated marker indicating that the assignment was in the solution. The change of value $c(x_i,x_j)$ would obviously necessitate the reallocation of that assignment, but would leave the other $(n - 1)$ assignments still valid. Thus, starting from the solution (assignments) of the problem before the modifications to the $[c_{ij}]$ ($[c_{ij}]$ is the cost matrix such that $c_{ii} = \infty \ \forall i$) were made, and removing the affected assignment, the solution of the new modified problem can be derived by re-entering the Hungarian algorithm at the last step since a single "breakthrough" (i.e., a single increase in the number of zero-assignments from $(n - 1)$ to n would in fact produce the optimal solution to the new assignment problem).

The performance of the above tree-search algorithm, with circuit elimination done according to the following branching rule. Let X be the set of all vertices in the graph, S contains a subset of the vertices in X and $\underline{S} = X - S$ ($\underline{S}$ is the set of vertices in X and not in S). Since an arc from S to $\underline{S}$ must start from some vertex in S, a problem could be split up into k subproblems $P_1, P_2, \ldots, P_k$ where for subproblem P_i, the initial vertex of the arc is $x_i \in S$ and the final vertex is some vertex in $\underline{S}$. This can be done by setting $c(x_i,x_j) = \infty \; \forall \; x_j \in S$ and leaving all other costs unchanged. In the solution of the resulting assignment problem there should certainly be the arc from x_i leading into S since all other alternatives have had their costs set to ∞. TSP with random asymmetric cost matrices could be solved in T seconds where

$$T \approx 0.55 \times 10^{-4} \times n^{3.46}$$

and n being the number of vertices in the problem. The above algorithm, however, does not perform well at all in symmetrical problems because the solutions of assignment problems are almost found to consist of large numbers of circuits of cardinality 2 which require many branchings to be completely eliminated [3].

Many greedy algorithms have been designed to solve the TSP. The greedy method builds solutions to problems step by step, according to the following simple precept: whenever faced with a simple decision, make that choice which produces the largest gain. Even though the greedy method makes each decision optimally from a local perspective, it may fail to achieve the global optimum[8].

A large family of greedy methods, known as insertion methods, build tours vertex by vertex by taking advantage of the triangle inequality. These methods start with a trivial cycle and expand it into a tour by incorporating a new vertex at each step. Call the new vertex x; in order to maintain a cycle, the algorithm replaces a single edge cycle, say *(u,v)*, with the two edges *(u,x)* and *(x,v)*. In order to minimize the increase in the value of the objective function, the edge to be replaced is chosen so as to minimize $d(u,x) + d(x,v) - d(u,v)$, which is non-negative when the triangle of inequality holds. These replacements do not really undo previous work, an important characteristic of greedy methods; they neither remove vertices nor alter their relative order in the cycle. Selection of the next vertex to include can be random or based on additional considerations, in hope of obtaining a better solution. The latter strategies include Nearest Insertion, which chooses the vertex closest to the group of already included vertices; Farthest Insertion, which chooses the vertex farthest from the group of already included vertices (with the goal of obtaining quickly a good "outline" of the solution and refining it in later stages); and Cheapest Insertion, which chooses the vertex that minimizes $d(u,x) + d(x,v) - d(u,v)$ over all choices of edge and vertex. (If $G = (V,A)$ where V is the set of vertices or cities and A is the set of edges or roads connecting cities, then $d(v_i,v_j)$ = distance from v_i to v_j if $(v_i,v_j) \in A$ (set of edges) and infinity otherwise. This algorithm is adopted from information retrieval applications.

All methods in this family require a separate initialization step: selection of the initial cycle. The natural choice is an empty cycle consisting of a single vertex. All above methods have their algorithms run in quadratic

time in the number of vertices , except for Cheapest Insertion, which has an added logarithmic factor [8].

The TSP algorithm implemented in JAVA needs the following files to compile Tspapp (the application for TSP algorithm):

- TSP source code (general class for working with TSP and algorithms).
- TSP supplementary code (template for a class of insertion heuristics for TSP's. These heuristics iteratively choose a city to add to a tour under construction).
- TSP insertion algorithm class supplementary code (Farthest Insertion heuristic: the farthest unvisited city is added next into the tour).

The parameters needed are the number of cities in the tour, their locations as a list of integers (x and y coordinates) and a ***GIF*** file containing the background image for the tour. If the locations are omitted, random (uniform) locations are generated. If the ***GIF*** file is present, the user can only reposition city 1; else if there is no background, users can move any city.

With the above parameters provided or generated, the algorithm computes the distances between any two cities for all cities. The tour permutation is initially the default (first city in the tour is the one with the first location input). Calculate the tour distance. If a city is moved, then its distance to other cities is updated according to its new coordinates as well as the tour distance and the tour image redrawn.

A number of functions and procedures are provided to do the following[9]:

- Initialize variables.
- Find the next city to insert in the tour (i.e. farthest).
- Insert the city found in the tour.
- Form the tour permutation in the order in which the cities are visited.
- Calculate tour distance.

A number of variables are used to implement the above algorithm. The variable *n* represents number of nodes in the graph, *maxdist* a maximum distance initialized to negative value of maximum integer, *far* the farthest node to be inserted in the tour, *fscity* the starting node of the tour, *fdist* an array of distances storing the minimum distance to a city in the tour for each unvisited city, *fcycle* stores the tour such that fcycle[1] is the 1st city, the next city is fcycle[fcycle[1]], etc..., *perm* the tour permutation (stores actual solution), and *dist[i,j]* the distance from *i* to *j*. The variables *end1*, *end2*, *index*, *inscost* and *fiteration* are temporary variables. The first three vary to finally get the location of the node to be inserted; whereas *inscost* and *fiteration* hold the cost of inserting the new node and the number of iterations to repeat in finding the node location.

The Farthest Insertion code for the above TSP algorithm works in the following manner [9]:

fiteration ← 1
repeat (*n*-2) times
  maxdist ← (- max. integer)
  far ← 1
  for *j* ←1 to *n*

```
        if fcycle[j] = -1
          if fdist[j] > maxdist
             maxdist ← fdist[j]
             far ← j

      inscost ← max. integer
      index ← fscity
      end1 ← 0
      end2 ← 0
      for j ← 1 to fiteration
        newnd ← fcycle[index]
        if newnd <> index
          dst ← dist[index,far] + dist[far,newnd] - dist[index,newnd]
        else
          dst ← dist[index,far] + dist[far,newnd]
        if (dst < inscost)
          inscost ← dst
          end1 ← index
          end2 ← newnd
        fcycle[far] ← end2
        fcycle[end1] ← far
        for j ← 1 to n
          if fcycle[j] = -1     {j unvisited node}
             if dist[far,j] < fdist[j]
                fdist[j] ← dist[far,j]
    fiteration ← fiteration +1

  val ← fscity
```

for $j \leftarrow 1$ to n

perm[j] $\leftarrow$ *val*

val $\leftarrow$ *fcycle[val]*

4.3 Modifications Done To Algorithms Used

Since response time is of paramount importance in our work, algorithms for finding the shortest path and solving the TSP in polynomial time were used. The algorithms used for finding the shortest path between two cities is Dijkstra and the algorithm used to solve the TSP is Nearest Insertion.

The implementation of the above two algorithms was done in Borland Delphi Desktop 2.0.

a. Dijkstra:

In addition to finding the shortest path, the preceding nodes of each node along the shortest path are maintained. Since Dijkstra is called once for each node in the graph, thus p is modified from one call to another; that is why we keep record of the value on each call in *precede* where *precede[i,j]* = *p[j]* after the procedure call Dijkstra(i) for all j.

Since C and S are defined as sets, the maximum set size in most programming languages is 256 and the approximate number of vertices in our case is 2512, the algorithm was adopted to use an array (*checked*) initialized to zero. A node x in C (before adoption) has

checked[x] = 0 (adoption). Similarly, a node x is S (before adoption) has *checked[x]* = x.

b. Travelling Salesperson:

The algorithm adopted to solve TSP was modified to do Nearest Insertion instead of Farthest Insertion. Thus, instead of inserting the farthest node to those nodes found so far, we insert the nearest. In addition to this, we had to devise a special code for the problem at hand since the readily available code did not prove to completely fit as is.

In both algorithms, *fcycle* stores the tour to be converted to the cities permutation (order of visiting cities). In farthest, the first city is *fcycle[1]*, the next city is *fcycle[fcycle[1]]* and so on. This style facilitates inserting cities into partially formed tours.

In our work, the first city is the starting city *fstartingcity*, the last city is *fcycle[fstartingcity]*, the node preceding the last is *fcycle[perm[selcities]]* (or equivalently *fcycle[fcycle[fstartingcity]]*), *fcycle[perm[selcities-1]]* (*selcities* is the number of selected cities including starting city) and so on.

The *for* loop controlled by the value of variable *fiteration* is wholly removed. This does not affect the functionality of the program.

Also, the function used to find the next city to insert in the tour was modified to insert the nearest among the selected not all graph nodes that are not yet inserted in the tour. To determine the path followed

and cities visited going from one city to another, we use both *perm* (holding nodes in order they are visited, each once) and *precede* to determine nodes on the path. *precede[perm[i],perm[i+1]]* gives the node *x* preceding *perm[i+1]* on the path from *perm[i]* to *perm[i+1]*. If *precede[perm[i], perm[i+1]] = perm[i+1]* then we go directly from *perm[i]* to *perm[i+1]* else push the value of *precede[x, perm[i]]* and repeat the above process until *precede[x,perm[i]] = x*. When (and after repetition) *precede[x,perm[i]] = x*, pop the stack elements and write them to a text file.

The roads table is exported from ARCVIEW to a text file. This text file is then read to fill *L[i,j]* for all values of *i* and *j* from 1 to *NumCities* (*L[i,j]* is the length of the arc from *i* to *j*).

Another adoption to Nearest Insertion is using Dijkstra as a part of the code. Dijkstra was used to compute the shortest path between any two nodes *i* and *j*. In the algorithm provided, these distances were input in a separate procedure.

Another modification has to do with choosing from the highlighted nodes (in ARCVIEW) the next node to insert in the tour. This is repeated until all highlighted nodes have been chosen. To determine the path to follow between any two of the highlighted nodes such as *i* and *j*, use the array *precede* to trace all the nodes along the shortest path from *i* to *j*.

5. Linking Platforms

One of the difficulties encountered was linking the Delphi form to the ARCVIEW map. To overcome this difficulty, the selected features in the map are written, in the selection order, to a text file read by the Delphi form which is called to execute. The output (results) of execution are written to another text file, read in ARCVIEW to highlight the path of the tour.

6. Time Complexity

The first version of our work implemented Dijkstra as described on pages 5, 6 and 7 of this chapter. The time taken by the algorithm formed by the combination of and adoptions to Dijkstra and Nearest Insertion is in $O(n^3)$ since Dijkstra is in $O(n^2)$ and is called $(n\text{-}1)$ times from within the for loop in Nearest Insertion.

A more efficient straightforward algorithm (described on pages 7, 8 and 9 of this chapter) takes time in $O(n^2)$. A time improvement to this algorithm uses an inverse heap to implement Dijkstra's algorithm (refer to page 9 of this chapter). By using an inverse heap, the execution time for Dijkstra was reduced from $O(n^2)$ to $O(n*log_n)$. Thus, the time taken by the algorithm formed by the combination of and adoptions to Dijkstra and Nearest Insertion is in $O(n^2*log_n)$ for a graph of n vertices.

Chapter 5

Experimental Results

This chapter presents experimental results for 8 test cases to demonstrate the operation of the algorithm formed by combining and adopting Dijkstra and Nearest Insertion (2nd version). These test cases show the time taken by the algorithm to calculate the tour formulation. It does not include the time to query the arcs forming the tour in ARCVIEW.

Test Case	No. of Nodes (Total)	Node Ids of Selected Arcs	Time Taken
I	500	3 1 4 2 75 11	5 sec.
II	800	3 1 4 2 75 11	20 sec.
III	1500	3 1 4 2 75 11	1 min. 20 sec.
IV	1500	4 2 48 39 119 113 196 193	1 min. 30 sec.
V	1800	3 1 4 2 75 11	2 min. 30 sec.
VI	1800	4 2 48 39 119 113 196 193	2 min. 30 sec.
VII	2512	4 2 48 39 119 113 196 193	4 min. 30 sec.
VIII	2512	911 919 920 908 919 920 1018 1002 1026 1018	5 min.

The test cases for the algorithm (1st version) are the following:

Test Case	No. of Nodes (Total)	Node Ids of Selected Arcs	Time Taken
I	500	3 1 4 2 75 11	40 sec.
II	800	3 1 4 2 75 11	3 min.

Chapter 6

Conclusion and Further Work

1. Conclusion

The project will be used by tourists or any citizen interested to visit one or more Lebanese sites. It will provide them with the path to follow in their tour having travelled the shortest distance possible, taking into consideration the starting node and time and the traffic along the roads traversed. Finally, it will display pictures, videos and information about hotels in selected sites.

2. Further Work

The project can be enhanced in the following manner:

- Incorporating a faster algorithm can be done very easily.
- Suggesting one or more tour plans given only the time to spend in the tour (to be input by the user) and the starting city.
- Providing the path to follow in the tour to traverse such that the time taken to visit selected sites does not exceed the time the user can spend (user input). If the time the user can spend is less than or equal to the time taken to visit selected sites then provide him/her with the path to follow. Else inform the user of the time needed to visit the selected sites and maybe suggest one or more plans including most of the selected sites such that the tour formed by these sites takes less time than user input time.

- Categorize the places to be visited according to their touristic activity (i.e., ancient sites, ski resorts, beaches, etc.). Ask the user to input the type of places he wishes to visit. This suggestion may be implemented with any of the above two suggestions.
- Providing the user with hotels in cities close to selected cities if the selected cities do not contain any hotel.

Appendix A

Source Map

Lebanon (scale 1:200,000)

2nd Revised Edition

Published by:

GEOprojects (UK) Ltd.

9-10 Southern Court

South Street

Reading

RG1 4QS

England

Appendix B

Coverages and Data Dictionary

The map was divided into two tiles. Each tile's coverages were digitized, built and edited independently. Then each coverage in the first tile was joined with its corresponding coverage in the second tile. The map was divided at 34°00'N. Thus the two tiles are:

- From 34°40'N to 34°00'N (thereafter referred to as **Tile 1**)
- From 34°00'N to 33°00'N (thereafter referred to as **Tile 2**)

In tables 1 and 2 below, **DMS** stands for degrees, minutes and seconds whereas **DD** stands for decimal degrees. The values in **DMS** appear on the source map described in Appendix A. These values were converted to **DD** using the following formula:

Decimal Degrees = Degrees + Minutes/60 + Seconds/3600

The master tic file for **Tile 1** (depicted in table 1) is:

Tic Id	x (DMS)	x (DD)	y (DMS)	y (DD)
13	35° 30′	35.5	34° 40′	34.6667
14	35° 40′	35.6667	34° 40′	34.6667
15	35° 50′	35.8333	34° 40′	34.6667
16	36° 00′	36	34° 40′	34.6667
17	36° 10′	36.1667	34° 40′	34.6667
18	36° 20′	36.3333	34° 40′	34.6667
19	36° 30′	36.5	34° 40′	34.6667
20	36° 40′	36.6667	34° 40′	34.6667
23	35° 30′	35.5	34° 00′	34
24	35° 40′	35.6667	34° 00′	34
25	35° 50′	35.8333	34° 00′	34
26	36° 00′	36	34° 00′	34
27	36° 10′	36.1667	34° 00′	34
28	36° 20′	36.3333	34° 00′	34
29	36° 30′	36.5	34° 00′	34
30	36° 40′	36.6667	34° 00′	34

Table 1: **Tile 1** tic ids, their x and y coordinate locations in **DMS** and **DD**

The master tic file for **Tile 2** (depicted in table 2) is:

Tic Id	x (DMS)	x (DD)	y (DMS)	y (DD)
23	35° 30′	35.5	34° 00′	34
24	35° 40′	35.6667	34° 00′	34
25	35° 50′	35.8333	34° 00′	34
26	36° 00′	36	34° 00′	34
27	36° 10′	36.1667	34° 00′	34
28	36° 20′	36.3333	34° 00′	34
29	36° 30′	36.5	34° 00′	34
30	36° 40′	36.6667	34° 00′	34
31	35° 10′	35.1667	33° 10′	33.1667
30	35° 50′	35.8333	33° 10′	33.1667

Table 2: **Tile 2** tic ids, their x and y coordinate locations in **DMS** and **DD**

1. Polygon Coverages

International Boundary

INTBND (depicted in figure 1) coverage obtained by applying the MAPJOIN operation to the coverages **INTBND06** and **INTBND15** in **Tile 1** and **Tile 2** respectively.

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
INTBND01	Digitize
INTBND02	BUILD - CLEAN
INTBND03	Edit
INTBND04	PROJECT into UTM and TRANSFORM
INTBND05	Copy of INTBND04 - BUILD
INTBND06	CLEAN

Tile 2:

Coverage Name	Command(s)/Operation(s) Applied
INTBND10	Digitize
INTBND11	Copy of INTBND10 - BUILD
INTBND12	Edit
INTBND13	PROJECT into UTM and TRANSFORM
INTBND14	Copy of INTBND13 - BUILD
INTBND15	CLEAN

Figure 1: **INTBND** coverage

Lake/Reservoir

Tile 1:
None present

Sabkhah

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
SABKHA01	Digitize
SABKHA02	BUILD - CLEAN
SABKHA03	Edit
SABKHA04	PROJECT into UTM and TRANSFORM

2. Label/Point Coverages

Museum

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
MUSEUM01	Digitize
MUSEUM02	BUILD
MUSEUM03	Edit
MUSEUM04	PROJECT into UTM and TRANSFORM

Caves

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
CAVES01	Digitize
CAVES02	BUILD
CAVES03	Edit
CAVES04	PROJECT into UTM and TRANSFORM

Cedars

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
CEDARS01	Digitize

CEDARS02	BUILD
CEDARS03	Edit
CEDARS04	PROJECT into UTM and TRANSFORM

View Point

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
VIEWPT01	Digitize
VIEWPT02	BUILD
VIEWPT03	Edit
VIEWPT04	PROJECT into UTM and TRANSFORM

Interest Place

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
INTPL01	Digitize
INTPL02	BUILD
INTPL03	Edit
INTPL04	PROJECT into UTM and TRANSFORM

Ancient Site

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
ANCTST01	Digitize
ANCTST02	BUILD
ANCTST03	Edit
ANCTST04	PROJECT into UTM and TRANSFORM

Estivage Centre

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
ESTCNT01	Digitize
ESTCNT02	BUILD
ESTCNT03	Edit
ESTCNT04	PROJECT into UTM and TRANSFORM

Ski Resort

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
SKIRES01	Digitize
SKIRES02	BUILD
SKIRES03	Edit
SKIRES04	PROJECT into UTM and TRANSFORM

Cities

CITYCOV (depicted in figure 2) coverage obtained by applying the APPEND command to the coverages **CITIES05** and **CITIES14** in **Tile 1** and **Tile 2** respectively.

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
CITIES01	Digitize
CITIES02	BUILD
CITIES03	Edit - JOINITEM with 'city.dat'
CITIES04	PROJECT into UTM and TRANSFORM
CITIES05	Copy of CITIES04 - BUILD

Added items

City_name	name of city

A table 'city.dat' (shown in **Appendix E**) was created. This table has two fields, namely the city identifier (cities03_id) and the city name (city_name). This table was joined with 'cities03.pat' table using the JOINITEM command with cities03_id in common to both tables.

Tile 2:

Coverage Name	Command(s)/Operation(s) Applied
CITIES10	Digitize
CITIES11	Copy of CITIES10 - BUILD - Edit
CITIES12	Copy of CITIES11 - JOINITEM with 'city1.dat'
CITIES13	PROJECT into UTM and TRANSFORM
CITIES14	Copy of CITIES14 - BUILD

A table 'city1.dat' (shown in **Appendix F**) was created. This table has two fields, namely the city identifier (cities12_id) and the city name (city_name). This table was joined with 'cities12.pat' table using the JOINITEM command with cities12_id in common to both tables.

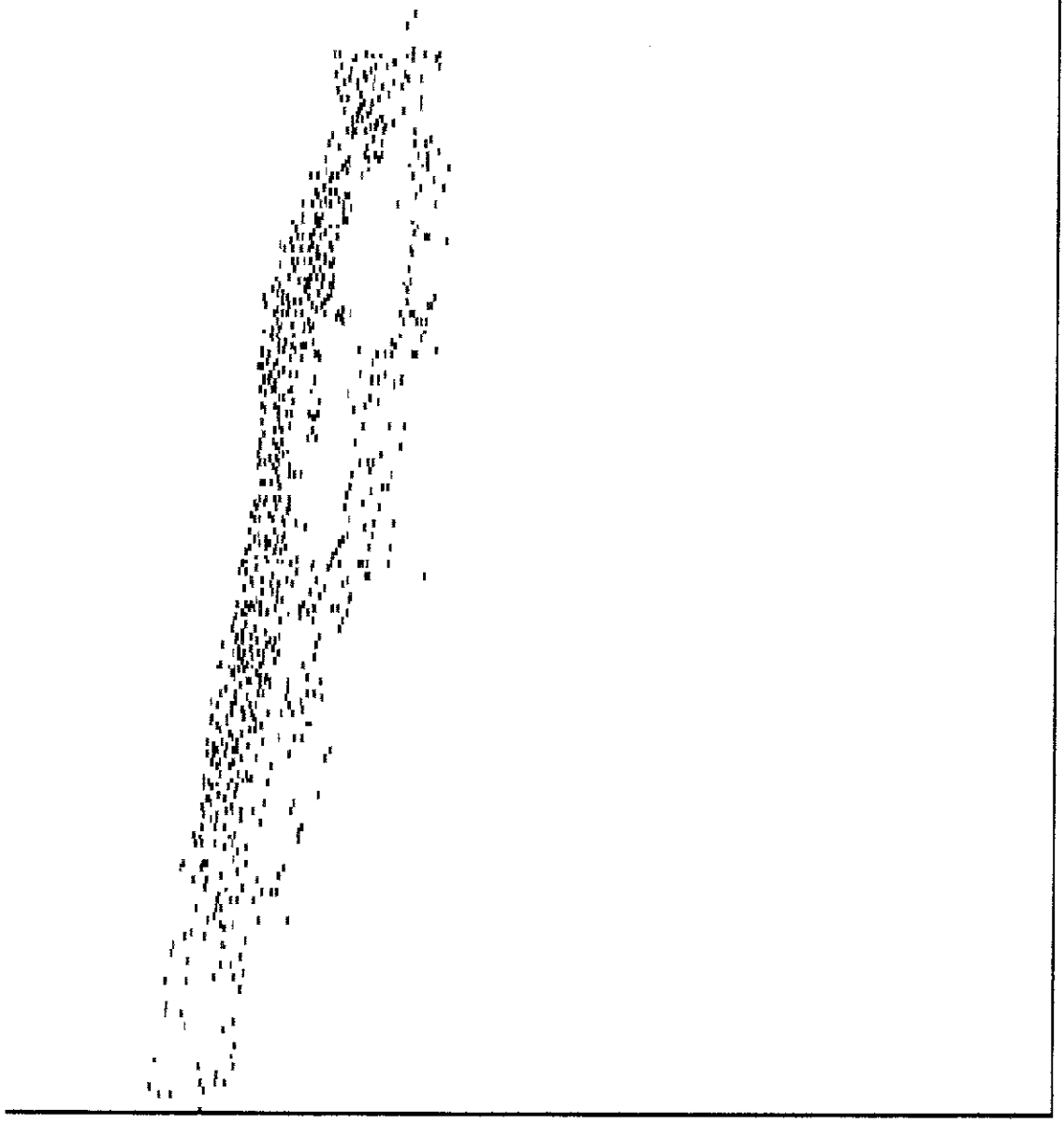

Figure 2: **CITYCOV** coverage

Beach Resort

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
BEACH01	Digitize
BEACH02	BUILD - CLEAN
BEACH03	Edit
BEACH04	PROJECT into UTM and TRANSFORM

3. Arc/Line Coverages

All Roads

ALLROAD (depicted in figure 3) obtained by appending all road type coverages

Coverage Name	Command(s)/Operation(s) Applied
ALLRD1	Copy of **ALLROAD** - BUILD
ALLRD2	CLEAN ALLRD1 - BUILD
ALLRD3	CLEAN ALLRD2 - BUILD
ALLRD4	CLEAN ALLRD3 - BUILD
ALLRD5	CLEAN ALLRD4 - BUILD
ALLRD6	CLEAN ALLRD5

Tiles of all road coverages (i.e., main, secondary, other, dual carriageway, dual carriageway under construction and main roads designed to be dual carriageway) were all combined to form one tile. The APPEND command was used to join the following coverages into the coverage, ***ALLROAD***:

- MAINRD05
- MAINRD14
- MNDCRD13
- SECRD05
- SECRD14
- OTHRRD05
- OTHRRD14
- DCRD05
- DCRD13
- DCUCRD05
- DCUCRD13

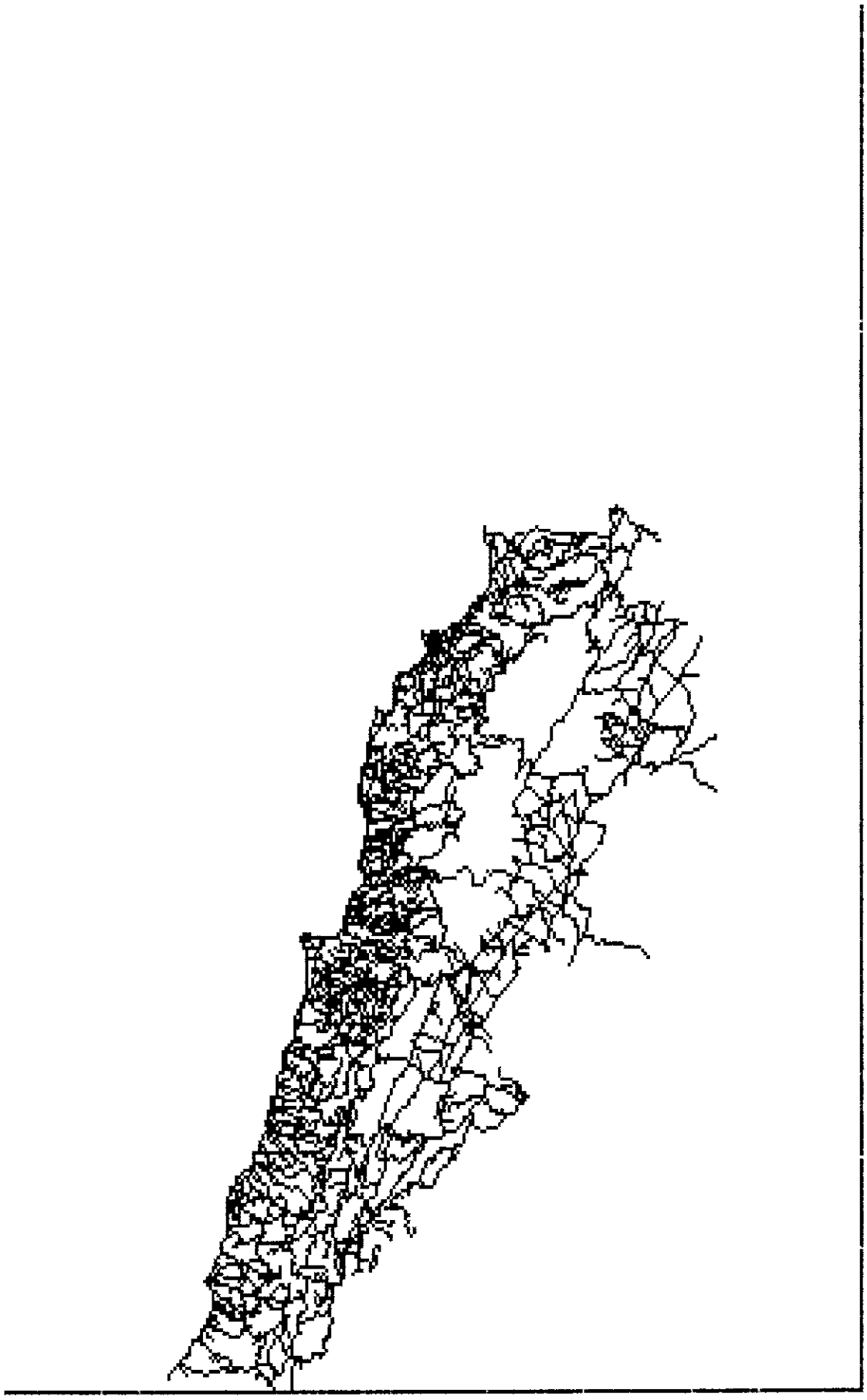

Figure 3: **ALLROAD** coverage

Main Roads

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
MAINRD01	Digitize
MAINRD02	BUILD - CLEAN
MAINRD03	Edit
MAINRD04	PROJECT into UTM and TRANSFORM
MAINRD05	Copy of MAINRD04 - BUILD

Tile 2:

Coverage Name	Command(s)/Operation(s) Applied
MAINRD10	Digitize
MAINRD11	Copy of MAINRD10 - BUILD
MAINRD12	CLEAN - Edit
MAINRD13	PROJECT into UTM and TRANSFORM
MAINRD14	Copy of MAINRD13 - BUILD

Main Roads Designed To Be Dual Carriageway

Tile 1:

None present

Tile 2:

Coverage Name	Command(s)/Operation(s) Applied
MNDCRD10	Digitize
MNDCRD11	Copy of MNDCRD10 - BUILD
MNDCRD12	PROJECT into UTM and TRANSFORM
MNDCRD13	Copy of MNDCRD13 - BUILD

Secondary Roads

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
SECRD01	Digitize
SECRD02	BUILD - CLEAN
SECRD03	Edit
SECRD04	PROJECT into UTM and TRANSFORM
SECRD05	Copy of SECRD04 - BUILD

Tile 2:

Coverage Name	Command(s)/Operation(s) Applied
SECRD10	Digitize

SECRD11	Copy of SECRD10 - BUILD
SECRD12	CLEAN - Edit
SECRD13	PROJECT into UTM and TRANSFORM
SECRD14	Copy of SECRD13 - BUILD

Other Road

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
OTHRRD01	Digitize
OTHRRD02	BUILD - CLEAN
OTHRRD03	Edit
OTHRRD04	PROJECT into UTM and TRANSFORM
OTHRRD05	Copy of OTHRRD04 - BUILD

Tile 2:

Coverage Name	Command(s)/Operation(s) Applied
OTHRRD10	Digitize
OTHRRD11	Copy of OTHRRD10 - BUILD
OTHRRD12	CLEAN - Edit
OTHRRD13	PROJECT into UTM and TRANSFORM
OTHRRD14	Copy of OTHRRD13 - BUILD

Dual Carriageway

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
DCRD01	Digitize
DCRD02	BUILD - CLEAN
DCRD03	Edit
DCRD04	PROJECT into UTM and TRANSFORM
DCRD05	Copy of DCRD04 - BUILD

Tile 2:

Coverage Name	Command(s)/Operation(s) Applied
DCRD10	Digitize
DCRD11	Copy of DCRD10 - BUILD
DCRD12	PROJECT into UTM and TRANSFORM
DCRD13	Copy of DCRD12 - BUILD

Dual Carriageway Under Construction

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
DCUCRD01	Digitize
DCUCRD02	BUILD - CLEAN
DCUCRD03	Edit
DCUCRD04	PROJECT into UTM and TRANSFORM
DCUCRD05	Copy of DCUCRD04 - BUILD

Tile 2 (only one arc):

Coverage Name	Command(s)/Operation(s) Applied
DCUCRD10	Digitize
DCUCRD11	Copy of DCUCRD10 - BUILD
DCUCRD12	PROJECT into UTM and TRANSFORM
DCUCRD13	Copy of DCRD12 - BUILD

Caza Boundary

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
CAZA01	Digitize
CAZA02	BUILD - CLEAN
CAZA03	Edit
CAZA04	PROJECT into UTM and TRANSFORM

Mohafazat Boundary

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
MOHFAZ01	Digitize
MOHFAZ02	BUILD - CLEAN
MOHFAZ03	Edit
MOHFAZ04	PROJECT into UTM and TRANSFORM

Permanent River

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
RIVER01	Digitize

RIVER02	BUILD - CLEAN
RIVER03	Edit
RIVER04	PROJECT into UTM and TRANSFORM

Wadi/Quadi

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
WADQAD01	Digitize
WADQAD02	BUILD - CLEAN
WADQAD03	Edit
WADQAD04	PROJECT into UTM and TRANSFORM

Track

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
TRACK01	Digitize
TRACK02	BUILD - CLEAN
TRACK 03	Edit
TRACK04	PROJECT into UTM and TRANSFORM

Railway

Tile 1:

Coverage Name	Command(s)/Operation(s) Applied
RAIL01	Digitize
RAIL02	BUILD - CLEAN
RAIL03	Edit
RAILWY04	PROJECT into UTM and TRANSFORM

Appendix C

Projection and Transformation

To project and transform each coverage, the following steps, stored in the file "geo.sml", were executed.

a. Type at the [ARC] prompt, "CREATE *georef* <coverage03>".

b. Type the command TABLES at the [ARC] prompt. Select the tic table of *georef* by typing *select georef.tic* at the <enter command:> prompt. Press <enter>. Then type UPDATE at the <enter command:> to change the values in *georef.tic* table to decimal degree values in the master tic file (master tic file listed on page 1 of this report under Description). The prompt <enter record number:> asks for the number of the record to update. This number can be obtained by typing the LIST command at the <enter command:> prompt. In addition to the record number, we get the tic id, xtic and ytic values. Type QUIT to exit Tables subsystem.

c. Execute "PROJECT cover *georef geoutm* geo.sml" at the [ARC] prompt. The content of geo.sml is:

input

projection geographic

units DD

parameters

output

projection utm

units meters

zone 36

yshift -3,000,000

parameters

end

d. At the [ARC] prompt, type CREATE <coverage04> *geoutm*

e. At the [ARC] prompt, TRANSFORM <coverage03> <coverage04>

Appendix D

Hotels

Hotel Name	Class/Cat.	Address	Telephone	Room/Suite
Achrafieh	3A		(01) 327768	57
Adonis	2B	Jouret El Tourmoss		37
Ahiram (video)	3B	1st Bif. to the right, El Mina, Jbeil	(09) 940440/ 941540	25
Ain Al Jirn	2A	Mairouba	(09) 951618	34
Ain Al Sawan	1B	Mairouba	(09) 951616	26
Ain El Sawan	2A	Jdita		
~~Akasia~~	~~2A~~	~~Beit Mery~~	~~(04) 970003~~	~~24~~
Akiki	2B	Rayfoon	(09) 954403/ 950114	20
Akl Hotel	1A	Berdawni, Zahleh	(08) 820701	15
Al Ahram	1B	Faraya	(09) 951812	15
Al Amrieh	3B	Bikfaya	(04) 980482/3	100
Al Bader	2A	Faraya	(09) 951601	23
Al Bustan	Int'l	Beit Mery	(04) 970400-2/ 972980-2 / (01) 425258-9	96
Al Mansour (video)	2A	Roundabout, Beit Mery	(04) 970660 /97069191	54
~~Al~~ Moukhtar	1A	Annaya	(09) 904219	10
Al Mushreck (video)	2A	Makdissi St+C36., Near Balaa Shoes,Hamra	(01) 345773	40
Al Naoura	1st Class	Tripoli	(06) 430533/ 431839	24
Al Naas	2B	Bikfaya	(04) 984734	43
Al Riyad Al Jadid	1B	Aley	(05) 550182	17
~~Alumni Association~~	~~2A~~	~~Hamra~~	~~(01) 340817/18~~	~~21~~
Aley Al Kabir (Jbeily)	3B	Aley	(05) 554760-1	44

America	1A	Brazil St., Zahleh	(08) 820536	21
~~Amwaj~~	~~1st Class~~	~~Jounieh~~	~~(04) 918700/1/2/3~~	~~40~~
Andalos	1A	Sir El Danniyeh	(06) 490065	*16*
Aquarium (video)	4A	Old Road, Jounieh	(09) 936858-62/ 911467/ 504525	60
Arabi (Casino)	1A	Zahleh	(08) 800144/ 826017	22
Arcada Marina	4B	Jounieh	(09) 915546/ 832250/ 832275	64
Astra	3B	Hamra	(01) 346600-1	85
Auberge Chateau Les Oliviers	1st Class	Tripoli	(06)423513/ 629271	21
Auberge Mont Liban	3rd Class	Ouyoune El Simane	(09) 950589/ 912051	45
Auberge Suisse	3rd Class	Ouyoune El Simane	(09) 953841	6
Azurama	2A	Harissa	(09) 780012/ 780019	63
Barakat	2B	Hadath Al Joubbe	(06) 677144	30
Bassil	2B	Hadath Al Joubbe	(06) 677003	29
Beau Rivage (video)	4A	Near Unesco, Ramlet El-Baida	(01) 864330/ 865390/ 866290	110
Beirut Commodore Hotel	4	Commodore St., Hamra	(01) 350400	235
Bellevue Palace	2B	Main Road, Broumana	(04) 960257, (03) 344220-1	64
Belmar	2A	Jounieh	(09) 917734	62
Belmont	3B	Ehden	(06) 673102/ 673166	42
Belverde Hnoud	2A	Broummana	(04) 961103	55
Berkeley	3A	Jeane D'arc St., Hamra	(01) 340600- 602250- 602281	**57**
Beverly Beach	3A	Main Road, Maameltein, Jounieh	(09) 900255/ 900485	70
Bhersaf	1A	Bhersaf	(04) 980013	20
Blue Beach	3A	Jounieh	(09) 910621-2/ 934001/ 934570	72
Bois De Boulogne Grand Hotel	4B	Bois De Boulogne	(04) 995100-1/ 995202/ 995301	102
Boule de Neige	2B	Ouyoune El Simane	(09) 953841	24
Braidi Al Kabir	2B	Ajaltoon	(09) 952124	25
Bristol Hotel	4A	Mme. Curie St., Hamra	(01) 351400/ 3514088/ 346390	145
Byblos Sur Mer	3A	Byblos Port, Jbeil	(09) 942983- 940356/ (03)303010	39
Bzoummar	3B	Bzoummar	(09) 902731	40
Cadmos	2A	Ein Al Mreisseh	(01) 602292-365995-366089-363196	34
Canary	2B	Dhour El Choueir	(04) 990501	22
Carlton	4B	Bhamdoun	(05) 562600-2	68
Carlton Hotel	4A	Main Rd., Raouche	(01) 868359 - 868360 - 869491	**140**

Casa D'or	3B	Jean D'arc St., Hamra	(01) 347850 - (03) 680687	68
Casade Assoun	1B	Assoun	(06) 663299	40
Casino Fayad	2B	Baabdate	(04) 975478	17
Cedar's	2A	Main Road, Broumana	(04) 960665/ 415015	26
~~Cedarland~~	~~3A~~	~~Hamra~~	~~(01) 340233~~	~~77~~
Central	1A	Tripoli	(06) 441544	13
Century Park Hotel	4B	Zouk Mikayel	(09) 938978/ 938197/ 832720/ 835245	80
~~Charles~~	~~3A~~	~~Rustom Bacha St.~~	~~(01) 808579 369248~~	~~65~~
Chateau Blanc	2A	Cedars	(06) 678005	57
Chateau D'eau	1A	Faraya	(09) 951602	44
Chbat	3A	Becharre	(06) 671237/ 671230	40
Chtaura Park Hotel	Int'l	Chtaura	(08) 540011	100
Coin Vert	1A	Faraya	(09) 950903/ 951844-5	24
Colibri	3A	Main Road, Baabdate	(04) 975402/ 975269/ 975153	62
Comfort	4B	Brazilia, Hazmieh	(01) 452613-5	84
Concorde	3A	Ras Beirut	(01) 740678 - 740664 - 740639	53
Coral Beach Hotel (video)	4A	Jnah	(01) 317200/ 317204/ 317175/ 317186	92
Cortina	2B	Cedars	(06) 671533	30
~~Cottage Club~~	~~2nd Class~~	~~Nabatieh~~	~~(07) 761539~~	~~20~~
Cottage Saint Jean	2A	Main Road, Ain Aar	(04) 925001	25
Dana Hamra	2nd Class	Ibl El Saqui		8
Dallas	4B	Jounieh	(09) 918519/ 937720-1	57
El Chedrawi	2B	Hadath Al Joubbe	(06) 677214	25
Embassy	3B	Makdissi St., Hamra	(01) 340814	48
Farabi	3B	Bhamdoun		42
Fatfat	1B	Sir El Danniyeh		24
Faytroun Al Kabir	2A	Faytroun	(09) 950009	34
Flotel	2A	Jal El Dib	(01) 404854/ 406316/ 417710/ 416458	54
Forest (video)	2B	Main Road, Mar Chaaya, Broumana	(04) 961877/ 960477	36
Four Seasons (video)	2B	Halat	(09) 957395/ 957516, (03) 312411	30
Fouad	1A	Mairouba		24
~~Fishing Club~~	~~2B~~	~~Jbeil~~	~~(09) 940213~~	~~12~~
Francis Hotel	2A	Ghineh	(09) 780789/ 908121	33

Garden (video)	3B	Broumana	(04) 960259/ 960579	42
Ghabet Al Khadra	2A	Shbanieh	(05) 530630, (01) 647864	36
Ghazi	1B	Sir El Danniyeh		15
~~Globus~~	~~2B~~	~~Qulaiaate~~	~~(09) 780011~~	~~38~~
Gran Mazar	4	Ouyoune Elsimane	(09) 710772-3	
Grand Hotel	2A	Mairouba	(09) 951614	30
Grand Hotel (Faraya)	2B	Faraya	(09) 951600	27
Grand Hotel Abshi	2B	Ehden	(06) 560001/ 56046262	60
Grand Hotel Douma	1A	Douma	(09) 520206/520106/520202	44
Grand Hotel Naas (video)	2B	Bikfaya	(04) 982622-3/ 982628	32
Grand Hotel Versailles	4A	Hamra St., Hamra	(01) 862561/ 865907-9/ 865830-70	**90**
Granada	2A	Bhersaf	(04) 980234	47
Green Hill	2A	Main Road, Fanar	(01) 880019/ 897779	40
Green Hotel	1A	Hasroun	(06) 675180	20
Hafroun	2A	Ehmej	(09) 904142/ 920621	30
Hawchar	1st Class	Tripoli	(06) 439978	14
High Hill	3A	Naas, Bikfaya	(04) 984200-1/ 984203/ 984310-2	55
High Land	2A	Aley	(03) 605020	24
Hitti	2B	Hadath Al Joubbe	(06) 677032	15
Holiday Beach	4B	Zouk Mosbeh	(09) 911168-9/ 911165	450
Hotel Al Salwa	1B	Dhour El Choueir	(04) 990013	21
Hotel Alexandre	4B	Adib Ishak St., Ashrafieh	(01) 200242-201132	**230**
Hotel Harissa	1A	Harissa	(09) 903918	24
Hrajel Hotel	1A	Hrajel	(09) 720263	18
Imperial Suites Hotel (video)	3A	Australia St., Raouche	(01) 862781/ 603598-9/ 603685/ 860986	72
Jaajaa	1A	Cedars	(06) 670057	19
Jammal Motels	2nd Class	Ghazir	(09) 950906/ 950904/ 950928	81
Kanaan Hotel	1A	Jezzine		19
Kanat Backiche	2A	Kanat Backich		10
Kassouf	3A	Dhour El Choueir	(04) 990501	62
Kfoury	1A	Dhour El Choueir		32
Khreizat	3B	Khreizat	(08) 960223/ 960093	45
~~Kings~~	~~4B~~	~~Raouche~~	~~(01) 813685-813689~~	~~75~~

L'Aiglon	Pension	Becharre	(06) 671529	8
L'Auberge de Faqra	1st Class	Faqra	(09) 895175/ 894721	28
L'Horizon	3A	Jounieh	(09) 832321/ 916619	60
La Cigale	3A	Zalka, Amaret Shalhoub, Jal El Dib	(01) 418324/ (03) 224450-2	24
La Mairie	4A	Ehden	(06) 560108	65
La Medina Hotel		Main Road, Maameltein	(09) 930875 / (03) 274011	40
La Mirage	3A	Ghazir	(09) 936127/ 831750/ 918489	40
Le Cavalier	4B	Abdel Baki St., Hamra	(01) 353001/ **602060**	64
Le Crillon (video)*	4B	Broumana	(04) 960221/ 960163	93
Le Gabriel		Aschrafieh	(01) 203700 - 203800 /(03) 208824	75
Le Marly	3A	Hamra	(01) 353970/1	50
Le Vendome	4A	Ein El-Mraysseh	(01) 369280/ (03) 212792	**70**
Lebanon Beach	4B	Khalde	**(01) 839932/ (03) 215949**	**65**
Legend	4A	Ain El Tineh-Verdun	(01) 801062-814838-860660-860170	60
Les Suites	3A	Beit Mery	(04) 972362-82/ (01) 887978	50
Liban	2B	Sir El Danniyeh	(06) 665047	60
Lord's Hotel	3A	Facing the Sea, Manara	**(01) 740382-3/85**	47
Luna House	1A	Bhamdoun	(05) 562162/ 560491	22
Lux	2B	Bikfaya	(04) 980185	15
Mace	2A	Near Cine Jean D'arc, Jean D'arc St., Hamra	(01) 344626-7	28
Mairouba	1B	Miarouba	(09) 951558	18
Marbella	3A	Sahel Alma, Jounieh	(09) 604611/ 934045/ 936498/ 918405	26
Marble Tower	3A	Makdissi St., Hamra	(01) 354586/ 346260-1	**60**
Mariott Hotel	4A	Adnan Hakim St., Khalde Blvd., Jnah	(01) 824494-5/ 824949	174
Masasad	2B	Faytroun	(09) 950286/ 951577	60
Mayflower	4B	Yafet St., Makdissi St., Hamra	(01) 340680 - 347080/ (03) 219789	85
Mediterranee	4B	Chouran St., Raouche	(01) 603015/ 603022-3	83
Merry Land (video)	2A	Bikfaya	(04) 980390/ 981590	51
Miami	1st Class	Main Road, Sahel Alma, Jounieh	(09) 933825-7/ 936673/ 935619/ 915950	45
Middle Beach	Pension	Jounieh	(09) 930727-831360	52
Mir Amine Palace (video)	1st Class	Beiteddine	(01) 861495, (05) 501315-8/ 500074	22
Miramar	2B	Ajaltoon	(09) 950227	30
Moderne	1A	Falougha		24

MoonLight	1A		(01) 352308	35
Mont Vert	2B	Broumana	(04) 961568/ 961622	40
Monte Alberto	2A	Al Wadi, Zahleh	(08) 820342/ 822365	28
Monte Bello	2B	Old St., Ajaltoun	(09) 950232/ 954507	80
Monte Carlo	1A	Qartaba	(09) 941847/940747/934570	16
Monte Verde Club	4B	Beit Mery	(04) 401806/ 400423	64
Montemar (video)	3A	Maameltein	(09) 918134-5	48
Motel Safra Marina	1st Class	Safra	(09) 880015	70
Motel Saint Francois (video)	3rd Class	Chekka	(06) 645524	24
Moueness Hotel		Khaizarane	(07) 724932/ 603663	216
Mrouje	1B	Mrouje	(04) 995814/5	22
Nader Hotel	2A	Fanar		40
Napoleon	4B	Makdissi St., Hamra	(01) 340013/ 354658/ 340207-9	75
Nassri	2B	Faytroun	(09) 950204	22
New Belvedere	2B	Broumana	(04) 960515-6	30
New Central	2B	Dhour El Choueir	(04) 990041	32
New Faraya	1A	Faraya		21
~~New Hamra~~	~~2A~~	~~Hamra~~	~~(01) 3466046~~	~~30~~
New Raoudah	2B	Dhour El Choueir	(04) 990400	44
Nirvana La Valade	3B	Laqlouq	(09) 904257	16
Old Bridge A.P.P.	2B	Faraya	(09) 952549	26
Orient Prince	3B	Hamra	(01) 340030-1	40
~~Pacific~~	~~3A~~	~~Rustom Bacha St.~~	~~(01) 362057 361197~~	~~65~~
Palace	2B	Hasroun	(06) 675115	22
Palace	2A	Saba St., Becharre	(06) 671460	24
Palace	2B	Tripoli	(06) 432256	14
Palmyra	3B	Baalbeck	(08) 870011/ 870230	35
~~Parklane~~	~~1A~~	~~Manara~~	~~(01) 804337~~	~~23~~
Pavillon Hotel	4B	Pavillon St., Hamra	(01) 350160/ 352300/ 352302	74
Pax	2A	Broumana	(04) 960027/ 960161	40
Peace Hotel	1A	Sir El Danniyeh	(06) 490360/ 460390	20
Pension Al Jawzeh	1A	Bteghrine	(04) 995222/111	5
Portemilio Suite Hotel	5	Jounieh, Kaslik	(09) 933300/ 604532-5/ 900831	200

Primotel (video)	3B	Broumana Valley, Broumana	(04) 963700/ 963142/ 963087	55
Printania Palace	Int'l	Broumana	(04) 960416-9/ 983910 /(01) 601125-601135	**130**
Queen's Land	4A	Queen's Land St., Haret Sakhr, Jounieh	(09) 936123/ 915945	62
Rabieh Marina	3B	Safra	(09) 880019/ 880023	48
Rahmeh	2B	Becharre	(06) 671146	30
Rancho Grandi	2A	Cedars	(06) 671501	55
Regency Palace	4A	Adma	(09) 934900/ 934909/ 491961	80
Regis	2A	Ain Mreisseh	(01) 361845	37
~~Residence Bel Azur~~	~~3~~	~~Jounieh~~	~~(09) 937753/ 915582-3~~	~~40~~
Residence De France	3A	Dbayeh	(01) 417872/ 404527/ 415582	26
Residence Edward V	2	Jounieh	(09) 935631/ 910436	42
Residence Half Moon	2	Jounieh	(09) 936988-90	46
Residence Hotel	3A	Ghazir		36
Residence Star Light	2A	Achrafieh	(01) 336905/ 334620	34
Richards	1A	Kfour	(09) 908713	21
Riviera	4B	Paris Ave., Manara	(01) 602273-5/ 617591-2	135
Rivoli	2A	Qartaba	(09) 905002	20
Royal Garden	4B	Emile Eddeh St., Hamra	(01) 350010-14/ 351801/ 352081	70
~~Russli~~	~~3A~~	~~Broummana~~	~~(04) 960015/16~~	~~54~~
Saba Hotel	2B	Bhamdoun	(05) 561063/ 560717	12
Saint Antoine	2nd Class	Jounieh	(09) 911880/ 831512	18
Saint Antonio	3B	Ouyoune El Simane	(09) 581651	34
Saint Bernard	3B	Cedars	(06) 671523	25
Saint Giorgio	3A	Faraya	(09) 720720/ 956355	54
Saint Joseph	Pension	Jounieh	(09) 931189	12
Saint Lorenzo	2A	Hamra	(01) 348604/5	37
Saint Michel	2A	Bikfaya	(04) 980223	50
Saint Paul	2B	Jouret El Tourmoss	(09) 908011	31
Saint Rock	4A	Rayfoon	(09) 950076-8	80
Salameh	1B	Kferzebian	(09) 710112	19
Sea Rock	4	Caracas St., Raouche Blvd., Beirut	(03) 269991-269993	50
Shangrilla	3A	Laqlouq	(09) 945521, (01) 200019, (03) 312411	57

Shouf Touristic Complex	1st Class	Baakline	(05) 501161/ 501273/ 503160	25
Sir Palace	2A	Sir El Danniyeh	(06) 490202/ 490407	25
Snow Land	3B	Kanat Bakishe	(09) 901456/ 917931	42
Sobh	1A	Aley	(05) 550325	33
Summerland	Int'l	Jnah	(01) 313030/ 304830/ 603220-4	153
Sunny Land	1A	Zahleh	(08) 826941	30
Tal Hotel	1A	Tripoli	(06) 628407	14
Tamer Land A.P.P.	3B	Faraya	(09) 951813	45
The Beach	1A	Jounieh	(09) 917667	32
Trabulsi	1A	Zahleh	(08) 820534	12
Vanda Hotel	3A	Jounieh	(09) 830107/ 936338/ 912038	48
Venus Palace	2A	Kfarnabrakh	(05) 501523/ 501525	24
Via Verde	3B	Chalet Suisse St., Fanar, Jdeidet El Metn	(01) 870368/ 870898/ 870256	63
Villa Backiche	3B	Kanat Backich		70
Villa L'Auberge	1A	Ghazir	(09) 931342/ 950428	10
Villa Mansour	1A	Bikfaya	(04) 980062/985471	15
Villa Sawaya	1A	Dhour El Choueir	(04) 990031	22
West House	4B	Jounieh	(09) 918287/ 936580	60
Wiener Haus	3A	Lyon St., Hamra	(01) 350050-2/352185/352237	60
Zouk Hotel	3B	Zouk Mikayel	(09) 917931/901456	42

Note:

"Lebanon Hotel Guide 1996" had no information about some of the hotels such as the telephone number(s) and the number of rooms/suites in Ain El Sawan Hotel in Jdita. Hotel names highlighted in marker pen were visited. Hotel names written in italics were not found in the guide. Other fields such as number of rooms/suites (written in bold) were updated by field investigation. If the hotel information is stroke out, then the hotel no longer exists, closed for renovation or is not a hotel (club). If "(video)" is found in the hotel name field, then video display will be provided for the hotel.

Appendix E

City.DAT

Identifier	Name	Identifier	Name
1	Kfardane	197	Zgharta
2	Haouch Barada	198	Kfar Habou
3	Es Saaldentifiere	199	Rachaain
4	Allaq	200	Aassoun
5	Haouch Tell Safiye	201	Izal
6	Laat	202	Beit Daoud
7	Nahle	203	Kfar Dlaqous
8	Younine	204	Deir Nbouh
9	Maqne	205	Karm el Mohr
10	Chaat	206	Aaimar
11	Rasm el Hadeth	207	Kaf el Malloul
12	Et Toufiqiye	208	Bhairet Toula
13	El Moqraq	209	Toula
14	El Laboue	210	Baslouqit
15	Aarsal	211	Mazraat et Teffah
16	En Nabi Osmane	212	Hmaiss
17	El Ain	213	Miziara
18	El Fakha	214	Bnechaai
19	Ras Baalbek	215	Sghab
20	Jdaldentifiere	216	Kfar Yachit
21	El Bejjaje	217	Laal
22	Tell Sougha	218	Kfar Hata
23	Zabboud	219	Kfar Fou
24	El Kharayeb	220	Sibaal
25	Mrah ech Chaab	221	Aitou
26	Mrah el Abed	222	Karm Sadde
27	Harbata	223	Ras Kifa
28	Qalile	224	Daraiya
29	Mrah Harfouch	225	Seraal
30	Nabha	226	Tourza
31	El Qeddam	227	Kousba
32	Mrah el Aouja	228	Bsarma
33	Barqa	229	Bachnine
34	Bechouat	230	Bishael
35	Es Safra	231	Kfar Chakhna
36	Riha	232	Kfar Zeina
37	Knaisse	233	Qarah Bach
38	Deir el Ahmar	234	Asnoun

39	Chlifa	235	El Qalamoun
40	Btedaai	236	Ras Masqa ej Jenoubiye
41	Boudai	237	Barsa
42	Dar el Quassaa	238	Dedde
43	El Yammoune	239	Qalhat
44	Mchaitiye	240	Zakroun
45	Aainata	241	Batroumine
46	Halbata	242	En Nakhle
47	Maaisra	243	Bdebba
48	Mazraat ouadi Faara	244	Fiaa
49	Quadi Faara	245	Aafsdiq
50	Mrah Housien Taan	246	Bterram
51	Quadi Bnit	247	Bechmizzine
52	El Qaa	248	Enfe
53	Ras el Assi	249	Chekka
54	Mazraat Ain ez Zarqa	250	Kfar Hazir
55	Mrah en Naouas	251	Amioun
56	Mrah Zouaitini	252	Kfar Aaqqa
57	Mrah Beit Allaou	253	Ehden
58	Ech Chouaghir	254	Aintourine
59	Mazraat Beit et Tachm	255	Kfar Sghab
60	Haouch Beit Ismail	256	Haouqa
61	El Qasr	257	Hadchit
62	Haouchariye	258	Bcharre
63	Mazraat el Talle	259	El Arz
64	Ez Zakbi	260	Bqaa Kafra
65	El Kouakh	261	Bqorqacha
66	El Breij	262	Harsroun
67	Charbine	263	Hadeth el Joubbe
68	Boueldentifiera	264	Qnaiouer
69	Zighrine et Tahta	265	Beit Menzer
70	El Baaoul	266	Qnat
71	Brissa	267	Berhalioun
72	Marjhine	268	Bnerhrane
73	Fissane	269	Ain Aakrine
74	Mrah es Souaisse	270	Rechdibbine
75	Quadi et Tourkmane	271	Bziza
76	Hmaire	272	Deir Billa
77	El Boustane	273	Hardine
78	Akroum	274	Niha
79	Kfar Toun	275	Kfour el Arabi
80	Qenia	276	Kfar Helda
81	Knaisse	277	Beit Chlala
82	Hnaldentifierar	278	El Heri
83	Qarha	279	Kefraiya
84	Aaouade	280	Kfar Hata
85	En Nasirye	281	Btaaboura
86	El Aarldentifiera	282	Kaftoun
87	Machta Hammoud	283	Dar Baechtar

88	Chadra	284	El Majdel
89	Aandqet	285	Hamat
90	Aaldentifiermoun	286	Ras Nhach
91	El Quienat	287	Ljd Aabrine
92	Martmoura	288	Boqsmaiya
93	El Qatlabe	289	Kfar Hai
94	Mounjez	290	RachkIdentifiera
95	El Fraldentifieris	291	Aabrine
96	El kouachara	292	Bijdarfil
97	Noura et Tahta	293	Koubba
98	Janine	294	Silaata
99	El Aabboudiye	295	Douma
100	El Bire	296	Tannourine el Tahta
101	El Majdel	297	Quata Houb
102	Aamaret el Bikat	298	Tannourine el Faouqa
103	Haitla	299	Chatine
104	Darine	300	Bcheale
105	Ghzaile	301	Hadtoun
106	Charbila	302	Racha
107	Msalla	303	Assia
108	Khirbet Shar	304	Helta
109	Deir Janine	305	Kour
110	Kfar Harra	306	Chabtine
111	Haizouq	307	Edde
112	Machha	308	Kfifane
113	Koucha	309	Jrane
114	Khraibet el Jindi	310	Fadaous
115	Tell Aabbas el Charqi	311	Smare Jbeil
116	Tell Aabbas el Gharbi	312	Ghouma
117	Kouaikhat	313	Aabdelli
118	Tell Hmaira	314	Toula
119	Tell Biri	315	Douq
120	Es Sammaqiye	316	Maifouq
121	El AarIdentifiera	317	Tartij
122	Cheikh Zennad	318	Jaj
123	Tell Bibi	319	Lehfed
124	El Massaoudiye	320	Haqel
125	Tell Kiri	321	Aabeydat
126	El Khirbe	322	Ghalboun
127	El Qlaiaat	323	Bejje
128	Khane Hayat	324	Maade
129	Qaabrine	325	Garzouz
130	Kfar Milki	326	El Mounsef
131	Ech Cheikh Mohmmed	327	El Heloue
132	Halbata	328	El Barbara
133	Identifierbil	329	Kfar KIdentifierde
134	Cheikh Taha	330	Hsarat
135	Llat	331	Hbaline
136	Jebrayel	332	Hosrayel

137	Rahbe	333	Aamchite
138	Tikrit	334	Mechmech
139	Beit Mellat	335	Khaabiya
140	Tachea	336	Kfoun
141	Beino	337	Behdayate
142	El Borj	338	Bentael
143	Ain Yaaquob	339	Edde
144	Bezbina	340	Hjoule
145	Akkar el Aatiqa	341	BeshtIldentifiera
146	Miniara	342	Ehmej
147	Hakour	343	Tourzaiya
148	El Qantara	344	Ras Qosta
149	El Houaich	345	El Breij
150	El Qraiyat	346	Aalmat
151	Chane	347	El Laqlouq
152	Khraibe	348	Aarab el Laqlouq
153	Zouq el Mqachrine	349	El Aaqoura
154	Bqerzla	350	El Majdel
155	Dinbou	351	Yanouh
156	Saissouq	352	El Mogheiri
157	Majdala	353	Mazraat es Siiyad
158	Habchit	354	Afqa
159	Quadi el Jamous	355	El Ghabat
160	Qarqaf	356	Lassa
161	Fnaldentifiereq	357	El Machnaqa
162	Mechmech	358	Bir el Hayt
163	Hrar	359	Bshille
164	Bzal	360	Blat
165	Berqayel	361	Halat
166	Bebnine	362	El FIdentifierar
167	El Aabde	363	Fatre
168	Bhannine	364	Nahr Ibrahim
169	El Minie	365	El Aaqaybe
170	Qabaait	366	El Bouar
171	Safinet el Qaitaa	367	Tabraya
172	Aayoune el Ghezlane	368	El Maameltayne
173	Jdaldentifieret el Qaitaa	369	Adonis
174	Btermaz	370	Yahchouch
175	Es Sfire	371	Jouret Bedrane
176	Sir ed Danniye	372	Ez Zaaytre
177	Haqel el Aazime	373	Ghable
178	Bakhaoun	374	El Ghine
179	Harf es Siyad	375	GhIdentifierrasse
180	Mrah es Sraij	376	Fatqa
181	Aadoui	377	Es Safra
182	Merkebta	378	El Kfour
183	En Nabi Youchaa	379	Shranaayn
184	Deir Aamar	380	Dlebta
185	Kefraiya	381	Baqaata

186	Aazqai	382	Raachine
187	Hailane	383	Hiyata
188	Boussit	384	Chahtoule
189	Aalma	385	Nahr ed Dahab
190	El Faouar	386	El Mchete
191	Meriata	387	Ain ed Delbe
192	Aachach	388	Mairouba
193	Rmaila	389	Hrajel
194	El Beddaoui	390	Faraiya
195	El Bahsass	391	Baalbek
196	Majedlaya	392	Jbail

Appendix F

City1.DAT

Identifier	Name	Identifier	Name
1	Haret Sakr	198	Chtaura
2	Ghadir	199	Taalabaya
3	Kaslik	200	Saadnayel
4	Ghosta	201	Ed Delhamiye
5	Harissa	202	Terbol
6	Aachqout	203	Taanayel
7	Sarba	204	Bar Elias
8	Aintoura	205	El Marj
9	Jeita	206	El Istabl
10	Zouq Mosbeh	207	Kfar Zabad
11	Nahr el Kalb	208	Haouch el Harime
12	Aajaltoun	209	El Khiara
13	Faitroun	210	Ed Dakoue
14	El Qlaiaat	211	Majdel Anjar
15	Daraiya	212	Es Souairi
16	Mazraat Kfar Debiane	213	Raite
17	Bqaatoua	214	Deir el Ghazal
18	Boqaatet Kanaan	215	Qoussaya
19	Kfar Aaqab	216	Ain Kfar Zabad
20	Zabbougha	217	El Jiye
21	Baskinta	218	En Nabi Younes
22	Sannine	219	Baassir
23	En Nabi Rchade	220	Ed Dibbiye
24	Taraya	221	El Borjein
25	Chmistar	222	El Marj
26	Kfar Dabach	223	Barja
27	Beit Chama	224	Dalhoun
28	Douris	225	Quadi ez Zeini
29	Ain Bourdai	226	Sibline
30	Majedloun	227	Ketermaya
31	Hizzine	228	Mazboud
32	Talia	229	Chime
33	Et Taibe	230	Er Rmaile
34	Britel	231	El Ourdaniye
35	En Nabi Sabt	232	El Jonailiye
36	Ham	233	Majdalouna
37	Maaraboun	234	El Mghairiye
38	Tfail	235	Joun

39	Hortaala	236	El Bramiye
40	El Khodr	237	Aabra
41	El Khraibe	238	Karkha
42	En Nabi Chit	239	Sfaray
43	Jenta	240	El Hara
44	Yahfoufa	241	Es Salihiye
45	Serraain el Faouqa	242	Lebaa
46	Serraain el Tahta	243	Kfar Falous
47	Bednayel	244	El Miye ou Miye
48	Qsarnaba	245	Ain ed Delb
49	Temnine el Faouqa	246	El Qraiye
50	Niha	247	Berti
51	En Nabi Ayla	248	El Mjeldentifieril
52	El Fourzol	249	Kfar Hatta
53	Temnine el Tahta	250	Kfar Melki
54	Ain en Nahri	251	Sarb es Sim
55	Rayak	252	Maghdouche
56	Haouch el Omara	253	Tanbourit
57	El Karak	254	Aanqoun
58	Quadi el Arayech	255	Qinnarit
59	Qaa er Rim	256	Daraiya
60	El Qommol	257	Aanout
61	Hazerta	258	Hasrout
62	Hemlaya	259	Gharife
63	Beit Chebab	260	Aatrine
64	Bikfaya	261	Ainbal
65	Zeghrine	262	Ez Zaarouriye
66	El Khenchara	263	Mazraat ed Dahr
67	Bteghrine	264	El Mtolle
68	El Mrouj	265	Bkifa
69	Bolonia	266	Bsaba
70	Dhour ech Choueir	267	Baiqoun
71	Ed Douar	268	Mazraat ech Chouf
72	Baabdat	269	Jdaldentifieret ech Chouf
73	Salima	270	El Moukhtara
74	Bzbdine	271	Aamatour
75	El Mtain	272	Bater
76	El Qaaqour	273	Anane
77	Aintoura	274	Bisri
78	Majdel Tarchich	275	Benouati
79	Tarchich	276	Bkassine
80	Kfar Selouane	277	Bteddine el Liqch
81	Jouar el Haouz	278	Aazour
82	Qornayel	279	Roum
83	Falougha	280	Aaray
84	Btekhnay	281	El Khraibe
85	Deir el Harf	282	Mrousti
86	Arsoun	283	Jbaa ech Chouf
87	Zakrit	284	Niha

88	Beirt ech Chaar	285	Rimat
89	Dbaiye	286	Saldentifieroun
90	Qornet Chehouane	287	Haitoura
91	Antelias	288	Zhalta
92	Jal ed Dib	289	Ain Majdalain
93	Bharsaf	290	Ain et Tine
94	Dahr es Souane	291	Machghara
95	Nabay	292	Kefraiya
96	Ez Zalqa	293	El Khraizat
97	El Jdaldentifiere	294	Tell Znoub
98	Ed Dekouane	295	Jobb Jannine
99	El Fanar	296	Kamed el Loz
100	Roumie	297	Soultane Yaqqoub el Faouqa
101	El Mansouriye	298	Soultane Yaqqoub el Tahta
102	Ain Saade	299	Bab Mareaa
103	Beit Meri	300	Aaitanit
104	Qsaibe	301	Lala
105	Ras el Matn	302	Kfar Mechki
106	Aabadiye	303	Kaoukaba bou Aarab
107	Baabda	304	Yanta
108	El Jamhour	305	Bakka
109	Aaraiya	306	Kfar Qouq
110	El Kahhale	307	Bakkifa
111	Chwite	308	Beit Lahia
112	Rouaisset el Ballout	309	Tannoura
113	Baalchmay	310	El Kfair
114	Quadi Chahrour	311	Mimes
115	Bsous	312	Hasbaiya
116	Kfar Chima	313	Chebaa
117	Ech Choueifete	314	AinQenia
118	El Qmatiye	315	Ebel es Saqi
119	Bkhechtay	316	Blat
120	Chanai	317	Kaoukaba
121	Deir Qoubil	318	Berghoz
122	Ain Anoub	319	El Aaichiye
123	Souq el Gharb	320	El Jarmaq
124	Ghaboun	321	Kfar Roummane
125	El Mansouriye	322	Habbouch
126	Bchamoun	323	Sejoud
127	Ainab	324	Er Rihane
128	Kaifoun	325	Aaramta
129	Baissour	326	El Louaize
130	Kfar Ammay	327	Jarjouaa
131	Bserrine	328	Ain Qana
132	Rouaisset en Naaman	329	Jbaa
133	El Mecherfe	330	Kfar Fila
134	Er Ramliye	331	Sarba
135	Kfar Niss	332	Houmine et Tahta
136	El Bire	333	Ez Zahrani

137	Brih	334	Mseileh
138	El Faouara	335	El Hajje
139	Batloun	336	El Maamariye
140	Kfar Nabrah	337	Deir ez Zahrani
141	Ain Ouzain	338	Toul
142	Majdel Meouh	339	El Kfour
143	Rechmaiya	340	Zebdine
144	Chartoun	341	Harouf
145	Aammiq	342	Jibchit
146	Kfar Qatra	343	Aabba
147	Bch et Fine	344	En Nmairiye
148	Kfar Faqoud	345	Zefta
149	Kfar Him	346	El Merouaniye
150	Sirjbal	347	Teffahta
151	El Jahliye	348	Kaoutariyet es Siyad
152	Baaqline	349	Insar
153	Baqaata	350	Khaizarane
154	Es Samqaniye	351	Es Sarafand
155	Beit ed Dine	352	El Babliye
156	Qabr Chmoun	353	Braiqaa
157	Dfoun	354	Ez Zrariye
158	Silfaya	355	El Kharayeb
159	Aaramoun	356	El Qasmiye
160	Khalde	357	El Bourgheliye
161	En Naame	358	Maarake
162	Haret en Naame	359	El Bass
163	Baaouerta	360	Er RachIdentifieriye
164	Daqqoun	361	Ras el Ain
165	Kfar Matta	362	Iskandarouna
166	El Binnay	363	En Naaqoura
167	Aabay	364	Aalma ech Chaab
168	El Mechref	365	Yarine
169	El Lahbiye	366	Rmaich
170	Es Saadiyat	367	Hanine
171	Dahr el Mghara	368	Debel
172	Bmariam	369	Rachaf
173	Ech Chebbaniye	370	Aainata
174	Ras el Harf	371	Bent Jbeil
175	Bouarej	372	Aaitaroun
176	Jdita	373	BIIdentifiera
177	El Mdaireij	374	Meis el Jabal
178	Dahr el Baldentifierar	375	Mhaibib
179	El Mraijat	376	Houla
180	Ain dara	377	Chaqra
181	El Aazzouniye	378	Adaisse
182	EghmIdentifier	379	Et Taibe
183	Charoun	380	Kfar Kila
184	Bedghane	381	El Qlaiaa
185	Bmahrain	382	Deir Mmass

186	Ain Zahlta	383	Seir Siriane
187	El Quarhaniye	384	El Qoussair
188	El Fraldentifieris	385	Zaoutar ech Charqiye
189	El Barouk	386	Zaoutar el Gharbiye
190	Maasser ech Chouf	387	El Ghandouriye
191	Tell el Akhdar	388	Deir Qanoun
192	Aammaq el Bekaa	389	Hanaouay
193	Deir Tahninch	390	Qana
194	Aana	391	Choukine
195	El Mansoura	392	Maifadoun
196	Tell Znoub ej Jdldentifiere	393	Kfar Tebnit
197	Ghazze		

REFERENCES

[1] Augenstein M.J. and Tenenbaum A. M. 1986. *Data Structures Using Pascal.* Prentice Hall.

[2] Brassard G. and Bratley P. 1988. *ALGORITHMICS Theory and Practice.* Prentice-Hall, Inc.

[3] Christofides N. 1975. *Graph Theory - An Algorithmic Approach.* Academic Press.

[4] ESRI 1996. *Avenue Customization and Application Development for ARCVIEW.* Environmental Systems Research Institute, Inc.

[5] ESRI 1995. *ARCVIEW The Geographic Information System for Everyone.* Environmental Systems Research Institute, Inc.

[6] ESRI 1990. *Understanding GIS - The ARC/INFO Method.* OnWord Press.

[7] Macromedia 1996. *Macromedia Director for Windows Version 5 Learning Director.*

[8] Moret B. M. E. and Shapiro H. D. 1991. *ALGORITHMS from P to NP.* The Benjamin/Cummings Publishing Company, Inc.

[9] ***Travelling Salesperson*** *Applet Page* in the yahoo engine (http://arachnid.cs.cf.ac.uk/Dave/JAVA/travel/travel.html)