M. Tech (Computer Science) Dissertation Series

[illegible] [illegible]

The Street Sweeping Problem

a dissertation submitted in partial fulfilment of the requirements for the M. Tech. (Computer Science) degree of the Indian Statistical Institute

By

Ramanendu Chattopadhayay

under the supervision of

Dr. Bimal Kumar Roy

INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road
Calcutta - 700 035

M. Tech. (Computer Science) Dissertation Series

The Street Sweeping Problem

a dissertation submitted in partial fulfilment of the requirements for the M. Tech. (Computer Science) degree of the Indian Statistical Institute

By

Ramanendu Chattopadhayay

under the supervision of

Dr. Bimal Kumar Roy

INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road
Calcutta - 700 035

CERTIFICATE

This is to certify that the work described in the dissertation entitled "The Street Sweeping Problem" has been undertaken by Ramanendu Chattopadhyay under my guidance and supervision. The dissertation is found worthy of acceptance for the award of the Degree of Master of Technology in Computer Science.

(Dr. Bimal Kumar Roy)

Indian Statistical Institute.
Calcutta.

Dated : July 22, 1994.

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my guide, Dr. Bimal Roy. It was really an enjoyable experience working under him.

My heartiest thanks to Mr. Arnab Bhattacharjee, Research Scholar, CSU, for having helped me out with the $\LaTeX$ typesetting.

Ramanendu Chattopadhyay

Indian Statistical Institute.
Calcutta.

Dated : July 22, 1994.

Contents

Chapter 1

Introduction

Combinatorial techniques find widespread application in the area of urban services. Some well-known problems are: design of rapid transit systems, location and staffing of police stations, assignment of shifts for municipal workers, routing of garbage trucks to pick up garbage, routing of street sweeping vehicles, and so on.

We have taken up the problem of designing the routing of street sweeping vehicles(refer Helly.W,[7] & Roberts.F.S, [1]) under the multifarious constraints that are involved in municipal street sweeping operations and the interesting graph-theoretic and combinatorial problems that are involved in the process of developing efficient solutions.

The problem has been studied in detail by L.Bodin and S.Kursh when they were developing a model for street-sweeping operations for the New York City Department of Sanitation in the Urban Science Program at the State University of New York at Stony Brook. The New York City Sanitation Department had a $200,000,000 annual budget of which $10,000,000 went to street-sweeping. The computerized sweeper routing based on this model saved about $1,000,000. The model was used in a part of the District of Columbia where it cut costs by over 20%.

As mentioned by Tucker and Bodin [2], the difficulties inherent in building good mathematical models for street-sweeping are indeed diverse in nature. The effort and money needed to collect proper data may be huge, changing any existing system to enhance efficiency may be next to impossible - the changeover may be economically prohibitive or there may be stiff resistance from the workers union. Moreover, it is known that more precisely one defines the problem taking care of all the underlying constraints, the more unwieldy the model becomes. On the other hand, certain idealized assumptions made to obtain elegant mathematical analysis may make the model far from being realistic.

Tucker and Bodin [2] have laid emphasis on the realization of many of the above-mentioned constraints, some of which are specific to the city under consideration. There is no analysis on the computational complexity or bounds of the algorithms presented by them. In this work, we have emphasised on the computational complexity of the problem, made a survey on the related optimization algorithms for networks that have been used, and have presented an ϵ-approximate algorithm for the routing in the case of a mixed graph, which happens to be the most generalized case.

Chapter 2

On What Has Been Done

2.1 Statement Of The Problem

The problem is to design an efficient set of routes for sweeping the streets in some city. By "sweeping a street", we mean that only the sides of a street along the curbs are to be swept. This task is to be performed by vehicles called "mechanical brooms".

At first sight the problem seems to be trivial; send a broom up and down the length of one street, north-south or east-west, and have it repeat this on the next parallel street; and partition the street into sets of north-south and east-west streets such that each street can be covered during a single broom's period on the streets. But let us consider the constraints involved. For a broom to sweep a curb, no parked cars must be present. Thus, parking regulations of the city must be taken into consideration. Secondly, major arterial routes cannot be swept during rush hours. In residential and manufacturing districts, where full parking capacity of the streets is needed much of the time, special regulations are needed. Thirdly, when among one-way streets, broom routes should try to avoid turns where the curb to be swept will switch from left to right. Finally, U-turns should also be minimized.

It has to be noted that the sweeper vehicle must frequently raise its broom and travel some distance along those streets that are not to be swept because of parking regulations or because they have been swept already or due to be swept by some other vehicle. It is natural that the vehicle will travel faster in these cases. The speed of the vehicle in this case is termed as the *deadheading speed*, and the time spent with the broom up is termed as the *deadheading time*. Since the time needed to sweep is predetermined, this deadheading time is what must be minimized.

2.2 The Mathematical Model

The general structure that can incorporate all this information is a *directed graph* $G = (N, E)$ where N is the set of *nodes* and a set E of *edges*, each directed from one node to another node. An edge represents *one side* of a street. The length of each edge is the time it takes for a mechanical broom to traverse the edge. A node represents a street-corner. An edge e_j from node x to node y can be written as $e_j = (x, y)$.

If the time constraints imposed by the parking regulations are considered, the large problem is broken into many short manageable problems. That is, for each period of the day or week, we form the subgraph of edges on which parking regulations (and rush-hour constraints) permit sweeping. For each of the subgraphs generated, we find a minimal set of routes that collectively cover the edges of the subgraph.

2.3 Analysis of the Mathematical Problem

Two different approaches have been used in seeking a near optimal number of routes to cover the edges of the directed graph. We can either divide the graph up into subgraphs of size small enough that each subgraph can be covered by a single route (of a given length), or we can find one extended route covering all the edges in the graph, and then break it up into appropriate sized routes. The first method has been called *"cluster first-route second"* and the second method has been called *"route first-cluster second"*. The latter method is more tractable because an exact mathematical solution exists to the minimal tour problem. The drawback of the "cluster first-route second" is that there may remain edges scattered about after subgraphs have been cut out.

2.3.1 The Route first-Cluster second Method

The first task in this approach is to obtain a minimal tour that covers all the edges of the given graph. The latter part of the job is concerned with breaking up the grand tour into subtours of appropriate size.

Existence of an *Euler circuit* simplifies the first part of the problem. Euler circuit exists if the difference between the *inner degree* and the *outer degree* for every node in the graph is equal to zero.

If, however, an euler circuit does not exist, then we would have to find a grand tour repeating edges in such a way that the extra cost is minimized.

Theorem 2.1 *Let G be a directed graph with a length assigned to each edge, and let H be a larger graph containing G. Let A be a minimal length collection of edges(a single edge may be counted several times in A) drawn from the graph H such that the addition of edges of A makes $d(x) = 0$ for each node x in the new graph. We assume that such a set A exists. Then A may be partitioned into paths from nodes of negative degree to nodes of positive degree. If $deg(x) = -k(or+k)$ in G, then k of the paths start(end) at x.*

The above theorem tells us how to minimize the deadheading time. We have to look at all

Turn	Weight
Straight ahead	0
Right Turn	1
Left Turn	4
U Turn	8
Switch sides of street	10
Raise or lower broom	5

Table 2.1: Weights assigned in each of the cases

ways of pairing off negative nodes with positive nodes with deadheading paths and then pick up the set of pairings that minimizes the total deadheading time.

To solve the minimum pairing problem, we need a matrix giving the lengths between the ith negative node x_i and the jth positive node y_j,for all i, j. The routes of the shortest paths between each of such x_i and y_j is to be stored. Even if the graph is disconnected it is required to solve the pairing problem for the whole graph at once since there is a possibility that the optimal pairing may link up nodes in the different components. In the matrix there is a row for each negative node and a column for each positive node. Entry a_{ij} of the matrix is the cost in going, via the shortest path, from the ith negative node x_i to the jth positive node y_j. This problem can now be solved as a *transportation problem*, the "supply" at x_i being $b_i = \mid d(x_i) \mid$, and the "demand" at y_j, $c_j = d(y_j)$. This method of solution is due to Edmonds J.[5] and a detailed discussion can be found in Edmonds J. and Johnson E.L.[4]. Note that in order to use this method, we always need $\sum b_i = \sum c_j$.

The solution of this problem gives a minimal set of pairings, i.e., it tells us which paths of deadheading edges to add. The next step is to make the pairing of the inward and outward edges in each node. Tucker and Bodin have done it in such a way that undesirable turns have been minimized. They have assigned weights to each possible type of pairing, as shown in Table 2.1.

At each node, the inward and the outward edges are paired up in such a way that the sum of the weights are minimized. By forming a matrix $W^{(k)}$ for the kth node v_k with a row for each inward edge and a column for each outward edge. Entry w_{ij} is the weight for pairing the ith inward edge with the jth outer edge. The problem is now posed as an *assignment problem*.After the assignment problem has been solved for each node, the grand circuit can be formed.

In the case of a disconnected graph, we would have found the euler circuit for each of the components. Each component is now considered to be a single node, and a *minimal spanning tree* is constructed using the nodes so that the total deadheading time needed to link together the individual components is minimized. To find the distance between the closest nodes in each pair of components approximate distance supplied by the *Manhattan distance* (difference of coordinates) have been used.

Now we arrive at the final stage; breaking up the grand tour into routes of feasible length. In the routing system developed for the New York City, this part was done by hand. We have developed approximate algorithms for this part, which is discussed in detail in Chapter 4 and 5.

Algorithm: Route first-Cluster second

1. Given the input graph G to obtain G'', append edges such that each of its components is eulerised.

   (a) Matrix of shortest distances H between negative nodes x_i and positive nodes y_j of G is calculated.

   (b) Transportation problem with the above matrix is solved, with row supplies $-d(x_i)$ and column demands $d(y_j)$.

   (c) Using the minimal path algorithm, the shortest paths in the larger graph H between the negative and positive nodes paired in step 1(b) is found.

   (d) The duplicated paths thus found are appended to G to obtain the new graph G''.

2. Euler circuit in each component of G'' is constructed.

   (a) Using the assignment problem approach inward and outward edges of G'' are matched. (b) Form the circuits arising from the match-ups in step 2(a) and paste them together to get a Euler Circuit in each component of G''.

3. Link together the components of G''-only if G'' is not connected.

   (a) Find the approximate shortest round-trip distances between the components of G''.

   (b) Using the distances in step 3(a) find a minimal spanning tree for the component graph.

   (c) Find the shortest path(in both directions) joining the closest pair of nodes in each pair of components linked in the minimal spanning tree.

   (d) Use these shortest paths to unite the tours of each component of G'' to get the desired grand tour which covers all the edges of G with minimal length.

4. Breaking up the grand tour.

   (a) Break up the grand tour into subtours of feasable length(this part was done manually for the model developed by Tucker and Bodin).

Chapter 3

Analysis and Computational Complexity of the Problem

3.1 Introduction

In this chapter we have carried out a mathematical formulation and and analysis of the street sweeping problem, and show that it belongs to the class of NP-Complete problems. In order to carry out the analysis, frequent reference to the *Chinese Postman Problem* will be made, since the problem bears a strong resemblance to it.

The first work on this problem appeared in a Chinese journal which called the problem the Postman Problem. This is from where the problem derived its name. The problem may be stated as follows :
Before starting his route, a postman must pick up his letters at the post-office, then he must deliver letters along each block in his route, and finally he must return to the post-office to return undelivered letters, covering his route with as little travelling as possible.

3.2 Mathematical Formulation

The problem can be viewed from two perspectives:
(a) The street sweeping time has been fixed. We are required to find out the minimum number of sweeper vehicles needed to do the job, and the paths of these vehicles.
(b) The number of sweeper vehicles is fixed. To find out the routes of the vehicles such that the time taken is minimum.

We assume that there are two types of vehicles.
Type A Here, the vehicle has the mechanical broom attached to only one side of it. So in one pass of a street, be it directed or undirected, only one side is cleaned.

Type B This is of improved quality in that it has its mechanical broom attached to both sides of it. So in one pass of a street, directed or undirected, the entire street is cleaned.

• A Counterintuitive Observation

Theorem 3.1 *If a single vehicle is given to sweep a road network, then the problem is polynomially solvable if the vehicle is of* Type A *and NP-Complete if it is of* **Type B**.

Proof:

The problem reduces to the case of Chinese Postman Problem for directed graphs if the vehicle is **Type A**. This is because this type of vehicle can clean only one side of a street in a single pass. So each street is to be traversed twice to clean it. As a result, each directed edge of the given graph G has to be duplicated and each undirected edge has to be duplicated and direction imposed - one opposite to the other. Let the resultant graph be called G'. Solving the Chinese Postman Problem on G' is equivalent to the street sweeping problem, which is known to be $O(n^3)$ (Refer Edmonds J.[5]).

If the vehicle is **Type B**, then the problem is equivalent to the Chinese Postman Problem for mixed graphs because the vehicle cleans both curbs in one pass for both directed as well as undirected cases. Chinese Postman Problem for mixed graph is polynomially transformable to the **3SAT** and hence is NP-Complete(Refer Papadimitriou C.[9]). □

We now give a proper formulation of the problem.

Given,

1. a network $N(V, E)$,
2. a mapping $t : E \rightarrow \mathcal{N}$,
3. a distinguished node v_0,
4. sweeping time T,
5. $\sum t(E) \leq m \cdot T$

we are required to find out if it is possible to sweep the entire network in time at most equal to T, using m trucks of **type A**.

In other words, the graph is to be divided into m connected subgraphs, $\{A_1, \ldots, A_m\}$, each subgraph having the following properties:

1. an edge starting from v_0 is present in every subgraph,
2. every edge appears in at least one subgraph
3. $\sum_{i \in A_k} f_i \cdot t(e_i) \leq T$, $\forall A_k$, $k = 1, \ldots, n$, f_i denotes the frequency of occurance of edge e_i in A_k.

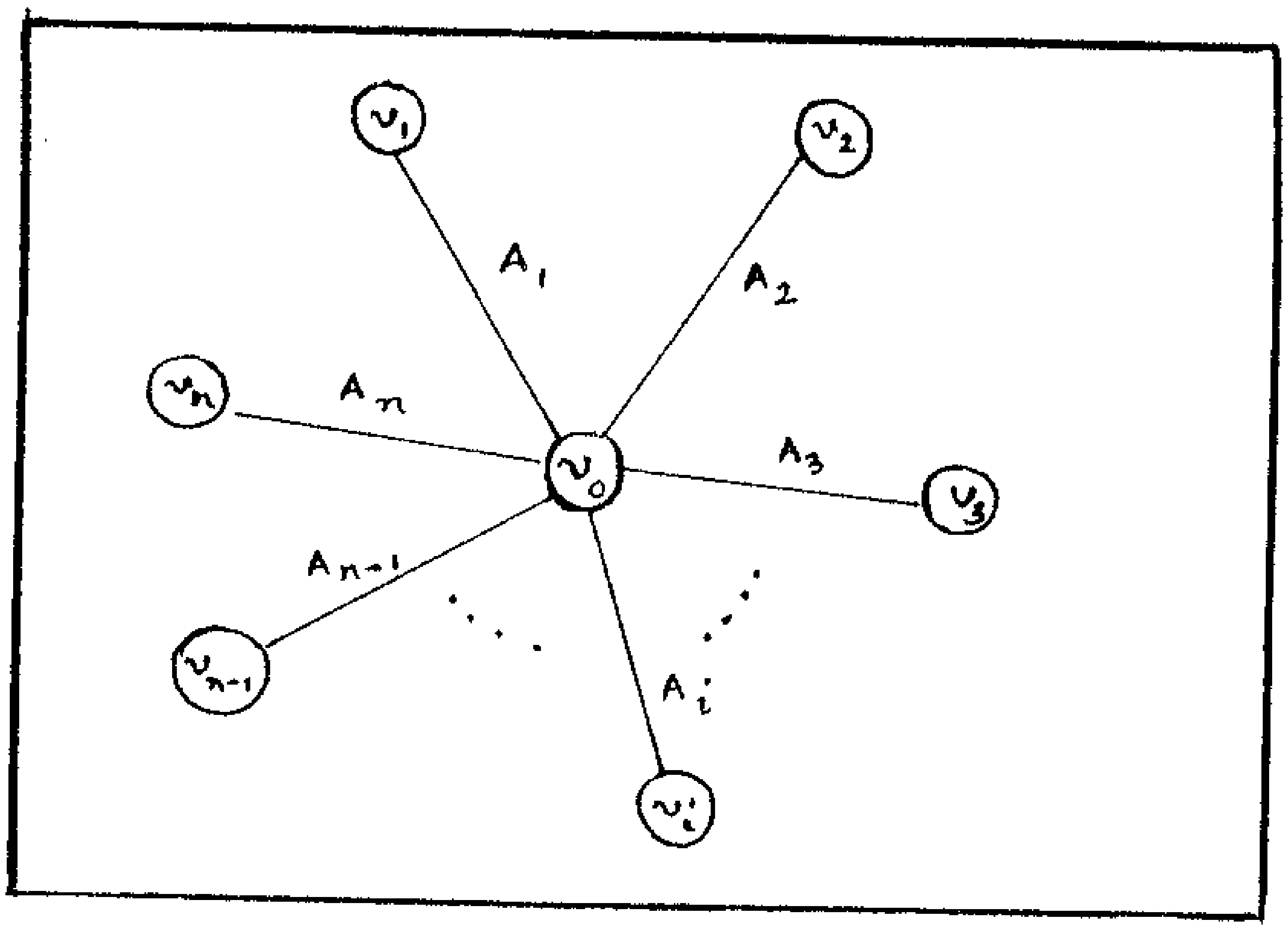

Figure 3.1: The street sweeping instance from **PARTITION** instance

3.3 The Computational Complexity of the Problem

Proving NP-Completeness for vehicles of Type **B** is trivial, as has been proved in theorem 3.1.

It will now be shown that the **PARTITION** problem, which has already been shown to be NP-complete, reduces to street-sweeping problem in polynomial time.

The **PARTITION** problem is defined as follows:

Given a set $S = \{a_1, a_2, \ldots, a_n\}$ of n integers such that $\sum a_i = 2T$ where T is an integer. Does there exist a set $R \subset S$, such that

$$\sum_{a_i \in R} a_i = \sum_{a_i \notin R} a_i = T\ ?$$

From an instance of **PARTITION** we can produce an instance of the street sweeping problem in linear time as follows:
Let there be $n+1$ nodes $v_0 \ldots v_n$, v_0 being the starting node. For each integer a_i construct a bi-directional edge between v_0 and v_i and name it A_i.
Define the mapping $t : E \to \mathcal{N}$ as $t(A_i) = a_i/2$

If **PARTITION** instance has a 'yes' answer, then

$$\sum_{a_i \in R} a_i = \sum_{a_i \notin R} a_i = T.$$

Without loss of generality we can assume that
$R = \{a_1, a_2, \ldots, a_k\}$ and $K = \{a_{k+1}, \ldots, a_n\}$, where $K = S - R$.

We claim that the street sweeping problem will also have an affirmative answer, as is evident below.
For the first vehicle, let the route traversed be
$\{\overrightarrow{A_1}, \overleftarrow{A_1}, \overrightarrow{A_2}, \overleftarrow{A_2}, \ldots, \overrightarrow{A_k}, \overleftarrow{A_k}\}$
For the second vehicle, let the route traversed be
$\{\overrightarrow{A_{k+1}}, \overleftarrow{A_{k+1}}, \overrightarrow{A_{k+2}}, \overleftarrow{A_{k+2}}, \ldots, \overrightarrow{A_n}, \overleftarrow{A_n}\}$
So the cost incurred by the first and the second vehicle is equal to T and the vehicles start and end at v_0..

On the other hand , if the street sweeping instance has an affirmative answer, then there must be two schedules
$R = \{a_1, a_2, \ldots, a_k\}$ and $K = \{a_{k+1}, \ldots, a_n\}$
each edge traversed twice in opposite directions by each vehicle. An edge occuring in R cannot appear in K as that would imply that the time taken would be more than T.
In this case,

$$\sum_{a_i \in R} a_i = \sum_{a_i \notin R} a_i = T,$$

thus proving that the **PARTITION** instance will also have affirmative answer.

Chapter 4

On the Formation of the Grand Tour with Type B vehicle

The formation of the grand tour happens to be the first phase in the "Tour-first Cluster-second" method. Tucker and Bodin[2] have used vehicles of **Type A** and thus dealt with only directed graphs. So the grand route obtained was optimal. But we know from that formation of the grand tour using vehicle of **Type B** is NP-complete(refer theorem 3.1). In this chapter we have studied the problem of formation of the grand tour with **Type B** vehicle. In order to do this, we consider three classes of graphs separately:

1. Undirected Graphs
2. Directed Graphs
3. Mixed Graphs

4.1 Type B traversal in undirected graphs

Here, the given graph is $G = (X, E)$ where the edges can be traversed in either direction. To analyse this, we consider two cases separately.

1. Graph G is *even*.

2. Graph G is not *even*

The total number of edges incident to a node in a graph is defined to be the *degree* of the node. If all the nodes of the graph are even, the *graph* is said to be even.

4.1.1 Even Graph

In this case, an optimal solution to the *Type B* vehicle is an Euler Tour, i.e., no edge is to be repeated. The Euler tour can be done in the following manner. Let the starting vertex be v_0. Any edge (v_0, x) is traversed first. The process of traversing unused edges is repeated till the starting vertex v_0 is reached. The process ensures that vertex v_0 is reached; for every vertex is even degree and every visit to a vertex leaves an even number of unused edges incident to that vertex. Hence, every time a vertex is entered, there exists an unused edge for leaving that vertex. The traversed edges constitute a cycle C_1. If all the edges present in the graph are traversed in C_1, then the process is stopped, otherwise another cycle C_2 is generated with the unused edges. The process of generating cycles $C_3, C_4, \ldots$ is continued till all the edges are used up. Next, all the cycles $C_1, C_2, \ldots$ are spliced together into one cycle C that contains all the edges of G. Cycle C contains all the edges exactly once and hence is an optimal solution. Splicing together two cycles is possible only if they share a common vertex. Let C_1 and C_2 be two cycles having a common vertex x and are to be spliced. Traversal along any edge of C_1 is commenced until vertex x is reached. Then a detour is made and all the edges of C_2 is traversed when node $x$is reached again. Finally, the remaining edges of C_1 are traversed and the starting point v_0 is reached. This procedure is extended to splice all the cycles.

4.1.2 Graph is not even

In any vehicle route, the number of times the vehicle enters a vertex equals the number of times the vehicle leaves that vertex. Consequently, if a vertex x is not of even degree, then at least one edge incident to the vertex has to be repeated.

Let $f(i,j)$ denote the number of times an edge (i,j) is repeated by the vehicle. Edge (i,j) is traversed $f(i,j)+1$ times by the vehicle. Of course, $f(i,j)$ must be a non-negative integer. Note that $f(i,j)$ contains no information about the direction of travel across edge (i,j).

A new graph $G^\star = (X, E^\star)$ is constructed, that contains $f(i,j)+1$ copies of each edge (i,j) in graph G. Now,$f(i,j)$ is to be selected so that
(a) Graph $G^\star$ is an even graph
(b) $\sum a(i,j)f(i,j)$, which is the total length of repeated edges, is minimized.

If vertex x is an odd degree vertex in graph G, then an odd number of edges incident to vertex x must be repeated by the vehicle, so that in graph $G^\star$ vertex x has even degree. Similarly if x is an even degree vertex in the graph, then an even number of edges incident to the vertex x must be repeated by the vehicle, so that in graph $G^\star$ vertex x has even degree.

If a chain of repeated edges is traced out in the graph, starting from an odd degree vertex, this chain must necessarily end at an odd degree vertex. Thus, the repeated edges form chains whose initial and terminal vertices are of odd degree. One or more of its intermediate vertices may, however be of even degree. So, the following decisions must be taken:

- which odd-degree vertices will be joined together by a chain,
- the precise composition of each such chain.

By performing either the Floyd[8] or Dantzig[6] algorithms, the shortest chain between each pair of odd-degree vertices can be calculated.

Which pairs of odd-degree vertices are to be joined by a chain of repeated edges, is decided as follows: Construct a graph $G'' = (X', E')$ whose vertex set consists of all odd-degree vertices in G and whose edge set contains an edge joining each pair of vertices. Let the weight of each such edge equal a very large number minus the length of the shortest path between the corresponding two vertices in the graph as has already been calculated. Next, a maximum weight matching for the graph G'' is done. Since graph G'' has an even number of vertices and each pair of vertices in G' is joined together by an edge, the maximum-weight matching will cover each edge exactly once. This matching matches together odd-degree in the graph G. The edges in a shortest chain joining a matched-pair of odd-degree vertices should be repeated by the vehicle. Since this matching has maximum total weight, the resulting route must have minimum total length.

4.2 Type B traversal for Directed Graphs

In the case of directed graphs, the procedure of finding the optimal solution is exactly identical to that of Type A vehicle. If $d(x) = 0$ for all vertices x in the given graph, the optimal route does not repeat edges. The process of finding an optimal route has been discussed in detail in Chapter 2, so it is not being repeated here.

4.3 Type B traversal for Mixed Graphs

It has been shown in Theorem 3.1 that traversal with **Type B** in this case is NP-Complete. For ease of analysis, two cases are being considered separately.

1. The mixed graph G is symmetric

2. G is not symmetric

4.3.1 Symmetric Mixed Graphs

Theorem 4.1 *If the given mixed graph is symmetric and even, then it is possible for the vehicle of* Type B *to perform its route without repeating any edges, directed or undirected.*

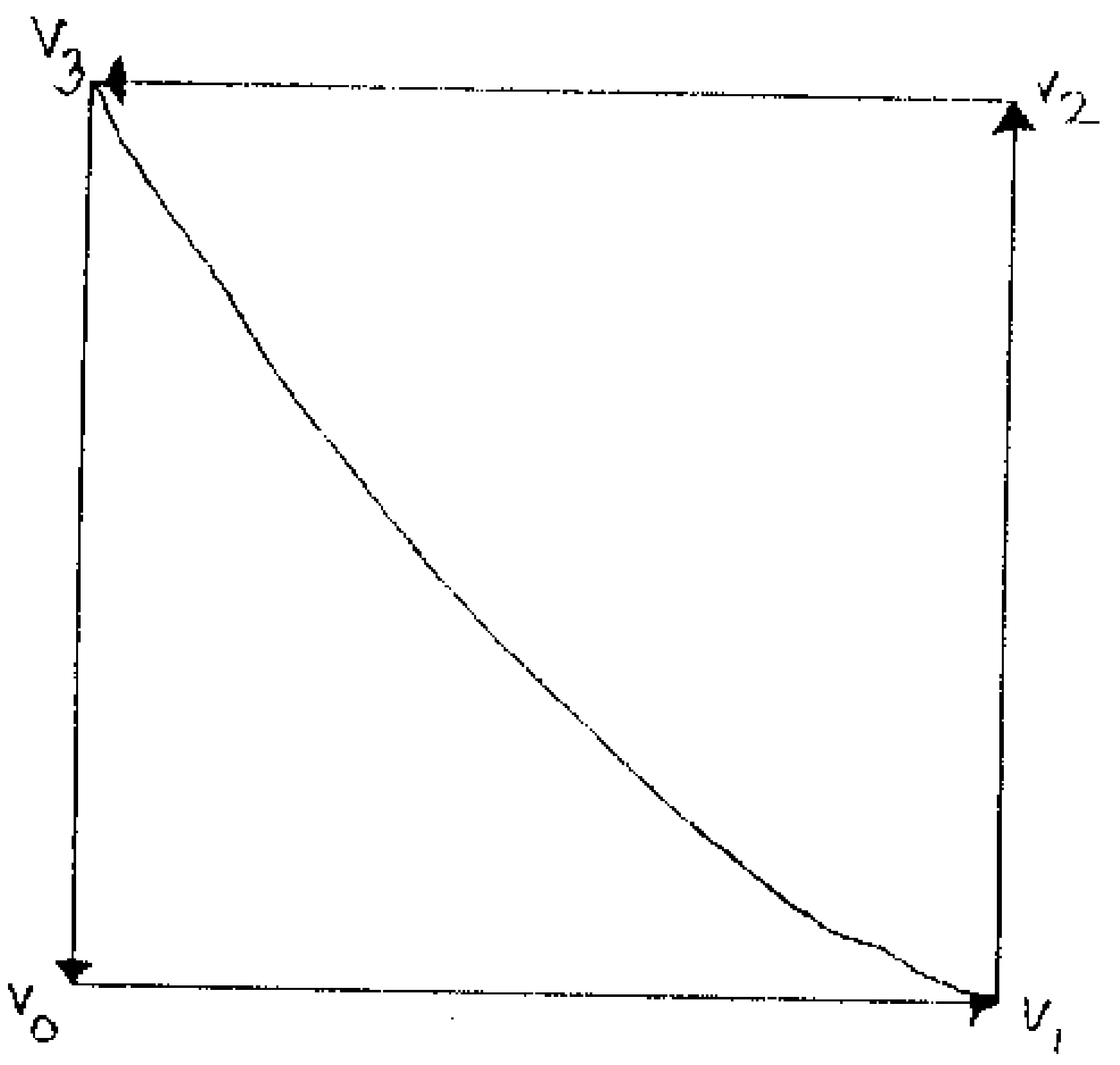

Figure 4.1: Example to show that for symmetric non-even graphs edges may have to be repeated

Proof:

Starting with any directed arc, a circuit of directed arcs is first done. This is possible because of the symmetric nature of the graph. This process is repeated till all the directed arcs have been used. This can be done because the unused arcs always form an even, symmetric graph.

Now, the undirected edges also form an even graph. So circuits of undirected edges are formed till all undirected edges have been used up.

Next, all the circuits are spliced together, thus forming a grand tour without repeating any edge.

If, however, the graph is symmetric but not even the above theorem does not hold. This is evident from the figure 4.1.

4.3.2 Non-Symmetrc Mixed Graphs

In this subsection, we deal with two cases separately:
- Graph G is even and non-symmetric
- Graph G is neither even nor symmetric

★ **Even and Non-symmetric**

Minieka E.[3] has presented an optimal algorithm to handle this case. We will now present the algorithm and its proof.

Algorithm: Optimal Route for Even Mixed Graph

Given any even, mixed graph $G = (X, A)$ the optimal route can be found as follows:
Let U denote the set of all undirected arcs in graph G. Let V denote the set of all directed arcs in$G$. Tentatively, select a direction for each arc in U. Call the resulting directed graph G_D. For each vertex i in G_D, calculate
$deg(i) = d^-(i) - d^+(i)$
If $deg(i) < 0$, then vertex i is a sink with demand equal to $-deg(i)$. If $deg(i) > 0$, then vertex i is a source with supply equal to $deg(i)$. If $deg(i) = 0$, then vertex i is an intermediate vertex.

If all the vertices in G_D are intermediate, then the graph is an even, symmetric directed graph, the method of its solution has already been presented.

Otherwise, construct a graph $G' = (X, A')$ as follows:

(a) For each arc $(i,j) \in V$, place an arc (i,j) in A' with infinite capacity and cost equal to the length of (i,j).
(b) For each arc $(i,j) \in U$, create two directed arcs (i,j) and (j,i) in A'. Let each of these arcs have infinite capacity and cost equal to the length of (i,j).
(c) For each arc $(i,j) \in U$, create a directed arc $(j,i)_1$ in A' whose direction is the reverse of the direction assigned to this arc in G_D. These arcs are called *artificial arcs*. Assign each artificial arc a zero cost and capacity equal to two.

Using the source supplies and sink demands defined above for graph G_D, apply the *minimum flow algorithm* to find a minimum cost flow in graph G' that satisfies all sink demands.

If no such flow exists, then no optimal route exists. Otherwise, let $f(i,j)$ denote the number of flow units sent through arc (i,j) in G' in the minimum cost flow produced by the minimum cost flow algorithm.

Next, create a graph $G^\star$ as follows:

(a) For each nonartificial arc (i,j) in G' place $f(i,j) + 1$ copies of arc (i,j) in graph $G^\star$
(b) If the flow in an artificial arc is two units, then place one copy of this arc in graph $G^\star$
(c) If the flow in an artificial arc is zero, then reverse the direction of this arc and place one copy of this arc in graph $G^\star$.

Graph $G^\star$ is an even symmetric directed graph. By the method described for such a case, the euler tour of graph $G^\star$ can now be found.

Theorem 4.2 (Minieka) *The Euler tour of $G^\star$ corresponds to an optimal route of the original graph G.*

Proof:

In the ideal case, we would obviously like to assign a direction to all the undirected arcs in such a way that the resulting directed graph is symmetric. Then the solution technique for even, symmetric graphs can be used to find the find the Euler tour of the graph.

However, such an orientation may not exist. In this case, some of the arcs(directed or undirected) must be repeated. The route is to be such that the total lengths of the repeated arcs is minimum.

This algorithm selects a tentative direction for each undirected arc to obtain a directed graph called G_D. The method for finding the grand tour for the directed graphs can then be used. The optimality of the resulting solution depends on the direction that has been imposed. It is quite possible that the tentative direction in G_D will result in a nonoptimal solution.

This algorithm generates a graph G' and using the same source supplies and sink demands as in graph G_D finds a minimum cost flow that satisfies all the sink demands.

There are three kinds of arcs in G_D:

(a) Arcs corresponding to directed arcs in G (these arcs have non-zero cost and infinite capacity).

(b) Nonartificial arcs corresponding to undirected arcs in G (these arcs also have non-zero cost and infinite capacity).

(c) Artificial arcs corresponding to undirected arcs in G (these arcs have zero cost and capacity equal to two).

The number $f(i,j)$ carried by an arc of type (a) or (b) in the minimum cost flow equals the number of times the corresponding arc is being repeated. Thus, the vehicle will traverse each arc $(i,j) \in V$ $f(i,j)+1$ times and traverse each arc $(i,j) \in U$ $f(i,j)+f(j,i)+1$ times. As shown later, each artificial arc carries either zero or two flow units. If an artificial arc $(j,i)_1$ carries two flow units in the minimum cost flow, this arc decreases the supply at j by two units and increases the supply at i by two units. This same effect could have been achieved by selecting the reverse tentative direction for this arc in graph G_D. Hence the algorithm reverses the tentative direction of this arc.

If an artificial arc $(j,i)_1$ carries no flow units in the minimum cost flow, then this arc has no effect on the supplies at vertices i and j. This is equivalent to retaining the tentative direction given this arc in graph G_D. Hence the algorithm retains the tentative direction given this arc.

From the minimum cost flow values $f(i,j)$ for the arcs in graph G', the algorithm generates a directed graph $G^\star$. Now, we are to show that

(a) Graph $G^\star$ is even and symmetric.

(b) An Euler tour of graph $G^\star$ corresponds to an optimal vehicle route of graph G

(c) If the minimum cost flow algorithm cannot find any flow that satisfied all sink demands in graph G', then no vehicle route exists for graph G.

Proof of (a):

Since graph G_D is an even graph, $d^+(i) + d^-(i)$ is an even number, for all vertices i in G_D. Thus $d^+(i)$ and $d^-(i)$ must either be both odd or both even. In either case, $deg(i)$ must be even. Consequently, all supplies and demands in graph G' are even numbers. Also, all arc capacities are even numbers. Hence, the minimum cost flow algorithm will produce an optimal flow in which all flow values are even numbers. Thus, flow units in a minimum cost flow will travel in pairs.

The value of $d(i)$ in $G^\star$ equals the value of $d(i)$ in G plus the number of flow units that enter or leave vertex i along non-artificial arcs. Since all flow units travel in pairs, it follows that $d(i)$ is even in graph $G^\star$. Thus $G^\star$ is even.

Now we are required to show that graph $G^\star$ is symmetric. A pair of flow units arriving at vertex i via a non-artificial arc increase $d^-(i)$ in $G^\star$ by two flow units and a pair of flow units leaving vertex i via a non-artificial arc increase $d^+(i)$ in $G^\star$ by two flow units. A pair of flow units arriving at vertex i via an artificial arc cause the tentative direction of this arc to be reversedwhich has the effect of increasing $d^+(i)$ by two units. Thus, each flow unit arriving at vertex i increases $d^-(i)$ by one unit, and each flow unit leaving vertex i increases $d^+(i)$ by one unit. Since the minimum cost flow satisfies
$\sum_j [f(i,j) - f(j,i)] = deg(i)$
for all vertices in G, it follows that the graph $G^\star$ is symmetric.

Proof of (b):

Suppose the resulting route is not optimal. Then there exists another route whose duplicated arcs have a smaller total length. This route must correspond to a flow in graph G' with even lower cost than the flow generated by the minimum cost flow algorithm, which is a contradiction.

Proof of (c):

Since $\sum_i d^-(i) = \sum_i d^+(i)$ in any graph G_D, the total of the source supplies in G_D must equal the total of the sink demands in G_D. Thus, all source supplies must be shipped out in order to satisfy all sink demands. If graph G contains an arc from i to j, then graph G' contains an arc with infinite capacity from i to j. Suppose that the minimum cost flow algorithm terminates without satisfying all the sink demands. Let S denote all the vertices that were colored after the last iteration of the minimum flow algorithm. From the flow augmenting algorithm we know that all arcs from a vertex in S to a vertex not in S must carry a full capacity flow. This is impossible since some of these arc capacities are infinite. Thus, no such arcs can exist, and all arcs with one colored endpoint are directed into set S. Consequently, once the vehicle reaches set S it cannot leave set S and no route is possible. This completes the proof. □

⋆ Non-even and Non-symmetric

No optimal solution technique is possible in this case as this problem has been shown to be NP-Complete by Papadimitriou C.H.[9].

Chapter 5

Partitioning of the Grand tour

In this chapter we take up the latter part of the "route first-cluster second" method; i.e., breaking up the grand tour obtained into subtours of feasable length, minimizing $f_i \forall i \in A_k$. This part had been done manually for the routing system used in the New York City. In this chapter we devise an algorithm which partitions the grand tour into subtours and which can give the total time that would be required in the worst case.

We shall be using the following notations:

V_i :	The ith vehicle.
M :	The total time required by single vehicle to clean up the grand tour.
c :	The ratio of the deadheading speed the cleaning speed.
v_0 :	The starting point of the vehicles.

5.1 Case of Two Vehicles

Consider two vehicles V_1 and V_2 of **Type B**. In an ideal case, the time required to clean up the entire network would have been $M/2$, i.e., if two edge disjoint equidistant paths P_1 and P_2 exist from v_0 to some point v_{mid} such that all the edges are covered and the vehicles V_1 and V_2 are made to traverse along P_1 and P_2 respectively.

The principle followed by our algorithm is to subdivide the grand tour into paths such that the vehicles commence at the same time instant and stop *cleaning* simultaneously. In other words, we are required to select an intermediate point v_{inter} such that V_1 cleans the eulerian path between $(v_0, \ldots, v_{inter})$ and V_2 cleans the latter part of the eulerian path between $(v_{inter}, \ldots, v_0)$ such that both vehicles start simultaneously at v_0, V_1 commencing cleaning instantaneously and V_2 travelling at deadheading speed to v_{inter} and then commencing cleaning and both of them terminating cleaning simultaneously, V_1 at v_{inter} and V_2 at v_0.

In order that both vehicles stop at the same time, V_1 has to clean some extra distance from v_{mid} to v_{inter}, the cleaning time (v_{mid}, v_{inter}) being equal to the time required by V_2 to arrive v_{inter} at a deadheading speed. This extra time is denoted by e.

Time traversed by V_1 is $M/2 + e$.

Time traversed by V_2 is $\frac{M/2+e}{c} + M/2 - e$.

Since their time of traversal is the same,

$$M/2 + e = \frac{M/2 + e}{c} + M/2 - e$$

or,

$$2e = \frac{M/2 + e}{c}$$

or,

$$e = \frac{M/2}{2c - 1} \tag{5.1}$$

Therefore, v_{inter} is located at the point on the Euler Circuit which would require a time $\frac{M \cdot c}{2c-1}$ at cleaning speed and $\frac{M}{2c-1}$ at deadheading speed.

5.2 Generalised Case

Let us now deal with the generalised case. Given k vehicles V_i, $i = 1, \ldots k$, we are required to find out the time required to clean up the given network and the amount of time each vehicle V_i deadheads and then cleans or vice versa.

Let the entire route be broken up into k equal intervals, each of the intervals requiring a cleaning time equal to M/k.

Let t_i be the extra time the ith-vehicle is required to clean in order that all the k vehicles stop cleaning at the same instant.

Time travelled by $V_1 = M/k + t_1$

Time travelled by $V_2 = \dfrac{M/k + t_1}{c} + M/k - t_1 + t_2$

$\vdots \quad \vdots \quad \vdots$

Time travelled by $V_i = \dfrac{(i-1)M/k + t_{i-1}}{c} + M/k - t_{i-1} + t_i$

Time travelled by $V_{i+1} = \dfrac{iM/k + t_i}{c} + M/k - t_i + t_{i+1}$

$\vdots \quad \vdots \quad \vdots$

Time travelled by $V_k = \dfrac{(k-1)M/k + t_(k-1)}{c} + M/k - t_{k-1}$

Since,

time travelled by V_i = time travelled by V_{i+1},

we get,

$$(i-1)M/k + t_{i-1} + cM/k - ct_{i-1} + ct_i = iM/k + t_i + cM/k - ct_i + ct_{i+1}$$

or,

$$(2c-1)t_i = M/k + t_{i-1}(c-1) + ct_{i+1} \tag{5.2}$$

Thus, we formulate the following recurrence relation:

$$\left\{ \begin{array}{ll} t_i = \dfrac{(i-1)M/k + ct_{i+1} + (c-1)t_{i-1}}{2c-1}, & for\ i = 1,2,\ldots,(k-1) \\ \text{and} & t_0 = t_k = 0 \end{array} \right\} \tag{5.3}$$

So, for $i = 1,2,\ldots,(k-1)$

$$ct_{i+1} - (2c-1)t_i + (c-1)t_{i-1} = -M/k \tag{5.4}$$

Multiplying throughout by x^i,

$$ct_{i+1}x^i - (2c-1)t_i x^i + (c-1)t_{i-1}x^i = -\frac{Mx^i}{k}$$

Therefore,

$$\frac{c}{x}\sum_{i=1}^{k-1} t_{i+1}x^{i+1} - (2c-1)\sum_{i=1}^{k-1} t_i x^i + x(c-1)\sum_{i=1}^{k-1} t_{i-1}x^{i-1} = -\frac{M}{k}\sum_{i=1}^{k-1} x^i$$

or,

$$\frac{c}{x}\left(G(x) - t_1 x\right) - (2c-1)G(x) + x(c-1)\left(G(x) - t_{k-1}x^{k-1}\right) = -\frac{Mx(1-x^{k-1})}{1-x}$$

or,

$$G(x)\left[\frac{c}{x} - (2c-1) + x(c-1)\right] = -\frac{Mx(1-x^{k-1})}{k(1-x)} + ct_1 + x(c-1)t_{k-1}x^{k-1}$$

or,

$$G(x) = \left[\frac{M}{k}(x^2 - x^{k+1})(1 + x + x_2 + \ldots) + xct_1 + x^{k+1}(c-1)t_{k-1}\right]$$
$$\times(1-x)^{-1}\frac{c-1}{c}\left(1 - \frac{x(c-1)}{c}\right)^{-1}$$

Comparing coefficients of x^2 on both sides of above,

$$t_2 = \frac{M(c-1)}{ck} - \frac{t_1(2c-1)(c-1)}{c} \tag{5.5}$$

Now, we know that

$$t_2 = \frac{(2c-1)t_1 - M/k}{c} \tag{5.6}$$

Substituting equation 5.6 in equation 5.5,

$$(2c-1)t_1 - M/k = (c-1)M/k - t_1(2c-1)(c-1)$$

or,

$$t_1 = \frac{M}{k(2c-1)} \tag{5.7}$$

So, the total time(T) traversed by each vehicle is

$$M/k + \frac{M}{k(2c-1)}$$

Therefore,

$$T = \frac{2Mc}{(2c-1)k} \tag{5.8}$$

Let us now formalise our algorithm to break up the grand tour up into subtours of feasable length.

Algorithm: Break up tour into k subtours

Given k trucks, we are required to find $(k-1)$ intermediate points $v_1, v_2, \ldots, v_k$ such that

$$\begin{array}{c}
\text{vehicle } V_1 \text{ cleans } (v_0, v_1) \\
\text{vehicle } V_2 \text{ cleans } (v_1, v_2) \\
\vdots \qquad \vdots \\
\text{vehicle } V_i \text{ cleans } (v_{i-1}, v_i) \\
\vdots \qquad \vdots \\
\text{vehicle } V_1 \text{ cleans } (v_{k-1}, v_k)
\end{array}$$

and all the vehicles start at the same instant and end their cleaning together.

- Find out v_1. v_1 is at the point which would take time T(as in equation 5.8) via the grand tour from v_0 at cleaning speed.
- For $i = 2, \ldots, (k-1)$ determine v_i. v_i would take time K_i via the grand tour path from v_{i-1} at cleaning speed, where

$$\begin{aligned}
K_i &= T - \sum_{k=1}^{i-1} K_k \\
K_1 &= \frac{2M}{(2c-1)k}
\end{aligned}$$

References

[1] Roberts,F.S *Graph Theory and its Applications to Problems of Society*, NSF-CBNF Monograph no.29, SIAM, Philadelphia.

[2] Tucker,A.C & Bodin,L *A model for municipal street sweeping operations* in Discrete and System Models, Vol.3 of Modules in Applied Mathematics,Springer-Verlag,N.Y.,1983.

[3] Minieka,E. *Optimization Algorithms for Networks and Graphs*, Dekker, NY, 1978.

[4] Edmonds,J. & Johnson,E. *Matching, Euler Tours and the Chinese Postman* in Math. Prog.1973,Vol.5.

[5] Edmonds,J. *The Chinese Postman Problem*, Oper. Res.,13,Suppl.1,1965.

[6] Dantzig,G.B. *All Shortest Routes in a Graph*, International Symposium, Rome,1966.

[7] Helly,W. *Urban System Models*, Academic Press, N.Y., 1973.

[8] Floyd,R.W. *Algm.97, Shortest Paths*, Comm. ACM,5,1962.

[9] Papadimitriou,C.H. *On the Complexity of Edge Traversal*, J. Assoc. Comp. Mach., 23,1976.