# A DECOMPOSITION PROCEDURE FOR FINDING THE MINIMAL HAMILTONIAN CHAIN OF A SPARSE GRAPH

by

Ira Ray Levinton

Thesis submitted to the Graduate Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Industrial Engineering and Operations Research

Approved:

James M. Moore, Chairman

Vinod Chachra

Allen L. Soyster

August, 1978

Blacksburg, Virginia

## ACKNOWLEDGEMENTS

The author wishes to express his appreciation to all those who have contributed to the completion of this thesis.

, originally major advisor, provided guidance and suggestions in developing this research, before leaving the United States. His ideas have been appreciated. , as major advisor, provided guidance in organizing this thesis. His helpful advice and criticism is greatly appreciated. , as a committee member, provided suggestions in formalizing the theory and the algorithms. His ideas were most helpful in finalizing this research. The author would also like to recognize committee member for his interest in the research and his suggestions for improvement. Appreciation is also extended to who provided input to the literature search.

Much gratitude is offered to for typing the final draft of this thesis against a difficult schedule of meeting deadlines.

TABLE OF CONTENTS

LIST OF FIGURES

# Chapter 1

# INTRODUCTION AND BACKGROUND

## Statement of Problem

The problem encountered here is one of finding the minimal Hamiltonian chain of a graph. All n vertices in an undirected graph are traversed in such a manner that each vertex is visited once and only once. Beginning from one prespecified endpoint vertex a chain passes through all other vertices once, ending at the other prespecified endpoint. The direction of traversing the graph is reversible. The objective is to find the minimal distance Hamiltonian chain, where the endpoints are fixed. The measurement of "distance" associated with each edge could be cost, time, or any other measurable quantity. It is assumed that all edges in the graph considered to be in the Hamiltonian chain cannot exceed some distance value $\theta$.

It is assumed that the reader is familiar with the basic concepts of graph theory. The common graph theory terms are used in accordance with those used in the text by Chachra, Ghare, and Moore [1].

## Related Literature

Only a limited amount of previous work has been devoted to the problem of finding the minimal Hamiltonian chain of a graph. The same holds true for its directed graph equivalent, the Hamiltonian path. However, there has been extensive work done on a related problem, the traveling salesman problem. With respect to the existence of a Hamiltonian chain, literature has been published on sufficient conditions.

A Hamiltonian chain or path covers all vertices of a graph with a single route that has two distinct endpoints. A traveling salesman problem differs in that it consists of a circuit that covers all vertices of a graph. Boffey [2] states that if the minimal solution of one problem can be found, then the minimal solution of the other can be found.

Hueristic methods have been used to find an approximation solution to the traveling salesman problem. They include edge construction methods and edge swap methods which are often used in conjunction with one another. Such methods have been discussed by Bellmore [3], and Turner [4]. Turner's work deals with single vehicle nodal routing problems which include a traveling salesman problem. One of the methodologies discussed involves eliminating and replacing pairs of arcs or edges to improve a traveling salesman tour. The results are dependent on an initial starting solution. Lin [5] has since developed an effective hueristic which has a high probability of finding the optimal solution. It has been reported that 70 vertex problems have been solved with the optimal solution resulting the majority of the time. While hueristics tend to give quick results, their disadvantage is that they cannot guarantee an optimal solution.

Exact solutions for the traveling salesman problem can be obtained by using dynamic programming, integer linear programming, and branch and bound methods. These methods have been discussed briefly by Turner [6] and Bellmore [3].

Dynamic programming requires excessive storage space [3]. It has been able to handle up to only a 15 vertex problem. At each step a set of recursive equations are stored. At the $k^{th}$ step in an n vertex problem, there are $g(n,k)$ values to be stored, where

$$g(n,k) = (n-1)!/[(k-1)!(n-k-1)!] \qquad (1.1)$$

For $n=15$ the bottleneck comes at $k=8$, where there are 24024 values.

Integer linear programming is impractical to use due to the excessive number of variables and constraints [6]. It requires $(n^2-n)/2$ variables and $2^n-1$ loop constraints for an n vertex problem in an undirected graph [3]. Depending on the structure of the graph, convergence can take a long time.

Branch and bound methods have had greater attention in the literature than the other exact methods. Bellmore [3] discusses two basic types, tour building algorithms, and subtour elimination algorithms. Tour building algorithms have dealt with directed graphs. These are based on theory having to do with arcs. This type of algorithm has been shown to have a high variation in time needed, and is generally time consuming. A random problem of 40 vertices in a directed graph needed over 8 minutes on an IBM 7090. Subtour elimination algorithms have been based on the fact that a traveling salesman problem is a solution to an assignment problem. Every vertex in the tour through the graph, can be paired with its subsequent vertex in the tour. An assignment matrix is branched upon until a traveling salesman solution results. An algorithm of this type has been given by Shapiro and reported by Bellmore [3]. It has been reported to take considerably longer time to solve an undirected graph than a directed graph. This is due to the excessive number of subtours of length two.

More recent branch and bound algorithms have been based on the relationship between the traveling salesman problem and minimal spanning trees. The work in this area was initiated by Held and Karp [8], improved upon by Held and Karp [9], and then by Hansen and Krarup [10]. The most recent version of the algorithm, by Hansen and Krarup, has been shown to be the most time efficient of any of the branch and bounds. It has solved problems of 60 vertices in an average time of 39 seconds on an IBM 360/75, and would be quicker on an IBM 360/91.

The work deals exclusively with undirected graphs. A 1-tree is created which is a spanning tree with an added edge. From a 1-tree a traveling salesman tour is branched to. A tour is a 1-tree with all vertices of degree two. If the minimal 1-tree is a tour it is the minimal tour. By eliminating edges in a 1-tree from the graph, a minimal 1-tree with all vertices of degree two will eventually be found. An approximation algorithm is used in the algorithm before applying the branch and bound. In Hansen and Krarup's version the branch and bound is performed by removing a single edge at a time in each branching.

Decomposition and partitioning methods have been used to reduce large traveling salesman problems into smaller ones. Bellmore [3] gives several methods used to include edges or arcs that appear likely to be in the optimal solution. Karp [11] devised a probabilistic method to partition a graph.

Christofides [12] has developed two basic algorithms to find the minimal Hamiltonian chain by using minimal spanning trees. The relationship between Held and Karp's 1-tree and a traveling salesman tour is analogous to the relationship between the spanning tree and a Hamiltonian chain. A spanning tree with no vertex of degree above two is a Hamiltonian chain. Furthermore, if the minimal spanning tree is a Hamiltonian chain it is the minimal Hamiltonian chain. By eliminating certain edges from a minimal spanning tree, eventually a Hamiltonian chain will be found. The first algorithm is a branch and bound procedure that eliminates edges from the spanning tree that are incident to a vertex of degree three or more in the tree. The second algorithm is hueristic in nature. One assigns a penalty value to any edge in the spanning tree that is incident to a vertex of degree three or more in the tree. One then recalculates the minimal spanning tree.

Christofides' algorithms have been shown only for complete graphs, without a full descritpion of the algorithm. No results have been given for large scale problems. His branch and bound procedure needs to be modified or clarified to handle all cases, when at least two edges must be removed from the graph before all vertices could have no more than degree two in the spanning tree. The penalty algorithm cannot detect feasibility conditions in a graph that is not complete as no edges are removed. Christofides' branch and bound prodedure is given in Chapter 3.

Taociniz [13] has developed an hueristic procedure for finding the minimal Hamiltonian path. The application is only for directed graphs that contain a path of arcs in both directions between any two vertices of the graph.

With respect to the existence of a Hamiltonian chain, Kaufmann [14] has given an algorithm to find all feasible Hamiltonian paths. By considering each edge as an arc in both directions, all Hamiltonian chains can be found. The algorithm begins with all edges and then finds all chains of length two. Successively it adds feasible edges to each chain to form a chain of length one greater until the chains are Hamiltonian.

Sufficient conditions for a Hamiltonian chain to exist have been investigated. Goodman [15] gives a specific subgraph structure that if imbedded in a graph will result in the existence of a Hamiltonian chain. Goodman also states that several other authors have identified other sufficient conditions. Most deal with given graphs having vertices with high number of incident edges. However, many graphs will have Hamiltonian chains, without having particular subgraph structures or the required number of edges these conditions require. Thus the theorems given are weak theorems. It would be desirable to have existence theorems for a Hamiltonian chain that cover more general graphs and graphs with less incident edges per vertex.

In summary, there exists a limited amount of literature on the minimal Hamiltonian chain problem. However, there has been extensive work on the traveling salesman problem, which is closely related. Exact solution methods for the traveling salesman problem such as dynamic programming, and integer linear programming are not practical to use. Branch and bound techniques are more useful. The most efficient branch and bound technique deals with branching on minimal spanning tree. Christofides has used similar techniques on a smaller scale for the Hamiltonian chain problem. His work will be expanded on in subsequent chapters. The only conditions for existence of Hamiltonian chains are sufficient conditions. Necessary conditions for existence do not appear in the literature but will be investigated in this work.

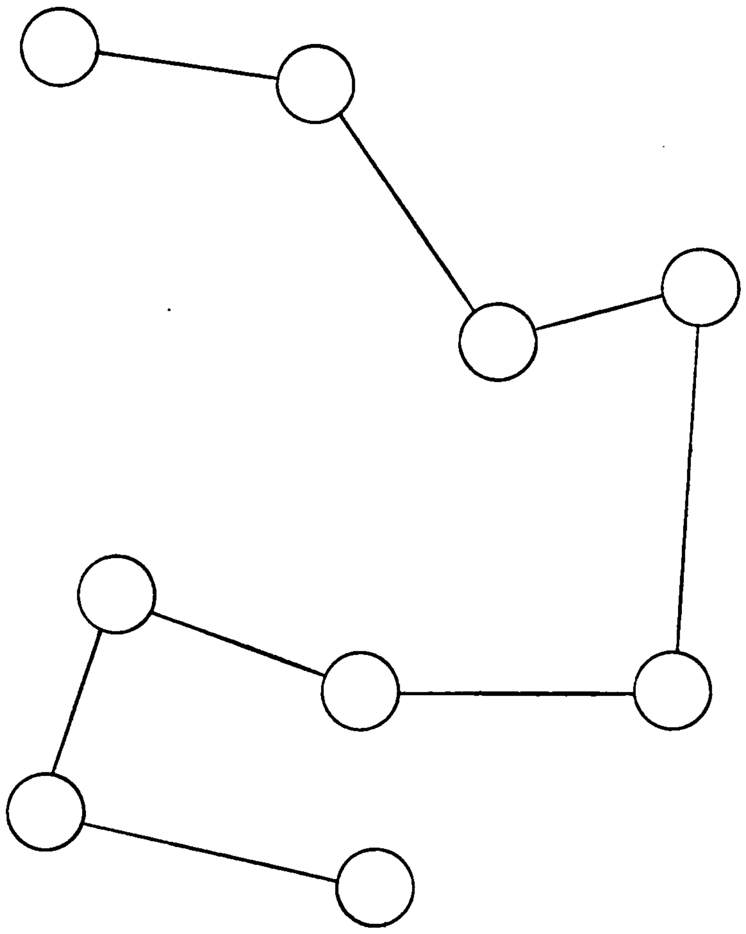


Figure 1.1. A Hamiltonian chain.

# Chapter 2

# CONCEPTUAL DEVELOPMENT OF THE MINIMAL HAMILTONIAN CHAIN PROBLEM

There are n vertices to be traversed in an undirected graph once and only once beginning and ending at fixed endpoints. The objective is to visit every vertex with the minimum total distance or time in covering the entire route through the graph. This problem is that of finding the minimal Hamiltonian chain of a graph. The mathematical model for this problem is presented in this chapter.

## Formulation of a General Hamiltonian Chain

Suppose a Hamiltonian chain is formed in a graph with n vertices. Let $a_{ij} = 1$ if an edge in the chain connects vertex $v_i$ with vertex $v_j$, and $a_{ij} = 0$ otherwise. As a chain is formed any vertex $v_i$ can have at most two edges connected to it in the chain. Therefore

$$\sum_{j=1}^{n} a_{ij} \leq 2 \quad \text{for every } v_i. \tag{2.1}$$

Every vertex must have an incident edge in the chain. Thus

$$\sum_{j=1}^{n} a_{ij} \geq 1 \quad \text{for every } v_i. \tag{2.2}$$

All n vertices in the graph are traversed in the chain and are not revisited. Any one edge in the chain encompasses two vertices, but each additional edge involves only one additional vertex. Therefore, for n vertices there are n-1 edges in a Hamiltonian chain. This gives

$$\sum_{1 \leq i < j \leq n} a_{ij} = n-1 \tag{2.3}$$

As the graph is undirected $a_{ji} = 1$ if and only if $a_{ij} = 1$. Therefore counting each edge twice, Equation 2.3 is equivalent to the following:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} = 2n-2 \tag{2.4}$$

Once a vertex is visited in a chain it cannot be revisited. Thus, no circuits exist in a Hamiltonian chain. This leads to the following equation:

$$\text{If } a_{bc} = 1,\ a_{cd} = 1,\ \ldots,\ a_{fg} = 1,\ a_{gh} = 1,$$

$$\text{then } a_{hb} \neq 1. \tag{2.5}$$

<u>Formulation of a Minimal Hamiltonian Chain with Fixed Endpoints</u>

Suppose a Hamiltonian chain is formed with endpoints $v_1$ and $v_2$. Let $c_{ij}$ be the distance measure of the edges between $v_i$ and $v_j$. Then

$$c_{ij}\ a_{ij} = c_{ij} \quad \text{if } a_{ij} = 1, \text{ and}$$

$$c_{ij}\ a_{ij} = 0 \quad \text{if } a_{ij} = 0. \tag{2.6}$$

Thus $c_{ij}\ a_{ij} > 0$ only if an edge in the Hamiltonian chain connects $v_i$ and $v_j$. As a result the total distance of the Hamiltonian chain is given by

$$\sum_{1 \le i < j \le n} c_{ij}\ a_{ij}. \tag{2.7}$$

Equation 2.7 is to be minimized.

The endpoints of a chain have one incident edge in the chain, while all other vertices have two incident edges. Therefore

$$\sum_{j=1}^{n} a_{ij} = 1 \quad \text{for } v_i = v_1, v_2, \text{ and} \tag{2.8}$$

$$\sum_{j=1}^{n} a_{ij} = 2 \quad \text{for } v_i \neq v_1, v_2 \tag{2.9}$$

To summarize, the model for a minimal Hamiltonian chain with fixed endpoints is given by the following:

$$\text{Minimize} \quad \sum_{1 \le i < j \le n} c_{ij}\ a_{ij}$$

$$\text{Subject to} \quad \sum_{1 \le i < j \le n} a_{ij} = n-1$$

$$\sum_{j=1}^{n} a_{ij} = 1 \quad \text{for } v_i = v_1, v_2$$

$$\sum_{j=1}^{n} a_{ij} = 2 \quad \text{for } v_i \neq v_1, v_2$$

If $a_{bc} = 1$, $a_{cd} = 1$, ..., $a_{fg} = 1$, $a_{gh} = 1$, then $a_{hb} \neq 1$.

where $a_{ij} = 1$ if an edge in the Hamiltonian chain connects $v_i$ and $v_j$

$a_{ij} = 0$ otherwise

$c_{ij}$ --distance from $v_i$ to $v_j$

$n$ --number of vertices

$v_1, v_2$--Hamiltonian chain endpoints

# Chapter 3

# HAMILTONIAN CHAINS FROM SPANNING TREES

Nicos Christofides [12] has developed two basic algorithms for finding the minimal Hamiltonian chain of a graph. Both develop a Hamiltonian chain from a minimal spanning tree. One is a penalty method which assigns penalty values to various edges. The other uses a branch and bound procedure to eliminate various spanning tree edges from the graph. The penalty method is discussed briefly. The other method is described in detail with a description of the algorithm and an example.

## Discussion of Christofides Methods

A spanning tree is a set of edges that connects all vertices in a graph but does not contain any circuits. The spanning tree with the least total distance is the minimal spanning tree. The minimal spanning tree can be calculated by the method of Kruskal.[1] A Hamiltonian chain is a spanning tree in which no vertex has a degree above two in the tree. Therefore, if the minimal spanning tree is a Hamiltonian chain it is the minimal Hamiltonian chain. If there were to exist a Hamiltonian chain of less total distance it would be less in total distance than the minimal spanning tree, but this is not possible since a Hamiltonian chain is a spanning tree. Also, only

[1]Appendix D describes Kruskal's minimal spanning tree algorithm.

two edges incident to any vertex can possibly be in a Hamiltonian chain. Therefore, if there are more than two edges in a spanning tree incident to a given vertex at most two could in a Hamiltonian chain. These properties are useful in defining algorithms to determine the minimal Hamiltonian chain from a minimal spanning tree.

A vertex with more incident spanning tree edges than can be in a Hamiltonian chain is defined to be an <u>oversaturated vertex</u>. As a vertex can have at most two incident edges in a Hamiltonian chain, any vertex with three or more incident edges in a spanning tree is an oversaturated vertex. Any endpoint vertex with more than one incident spanning tree edge is an oversaturated vertex. An edge in the spanning tree incident to such a vertex is defined to be an <u>oversaturated edge</u>.

In the penalty method defined by Christofides, one assesses a penalty to the distance value of every oversaturated edge in the minimal spanning tree; then one recalculates the minimal spanning tree. Convergence to a Hamiltonian chain occurs rapidly for a complete graph. For graphs that are not complete, there is no way of determine which combinations of edges can possibly be in a Hamiltonian chain. No edges are eliminated in the process. If no Hamiltonian chain exists this method would fail to detect this as it would iterate infinitely by calculating a new minimal spanning tree each iteration. Termination occurs only when a Hamiltonian chain is found. If only a few Hamiltonian chains exist, convergence may not result after many iterations.

Christofides Branch and Bound Algorithm [12]

The following algorithm has been shown by Christofides but has not been stated explicitly. It is stated here as implied. The algorithm performs a branch and bound operation on an undirected graph G. It eliminates different spanning tree edges incident to an oversaturated vertex in the minimal spanning tree. Christofides denotes $d_j^T$ as the number of edges incident to vertex $v_j$ in the minimal spanning tree. He then denotes $\varepsilon^T$ as the degree of difference of a minimal spanning tree to a Hamiltonian chain. It is given by

$$\varepsilon^T = \sum_{d_j^T > 2} (d_j^T - 2) \tag{3.1}$$

The algorithm is defined only when $\varepsilon^T \leq 1$. That is, only one vertex can have $d^T > 2$. Let this vertex be $v_i$.

When $\varepsilon^T = 1$, then $v_i$ is oversaturated by one degree. Then at least one of the three incident edges to $v_i$ in the spanning tree cannot appear in a Hamiltonian chain of graph G. For each case of eliminating one of these edges, a new graph $G^R$ is defined, containing all remaining edges. At each subsequent iteration the graph $G^R$ with the least valued minimal spanning tree is branched upon, until the given $G^R$ has a minimal spanning tree that is a Hamiltonian chain.

Let R be the set of all partial graphs $G^R$. Let $e_{ij}^R = 1$ if an edge exists in $G^R$ incident to $v_i$ and $v_j$. Let $s_{ij}^R = 1$ if the edge is included in the minimal spanning tree of $G^R$. Let $G^M$ be the graph currently being branched upon. For brevity, minimal spanning tree is denoted by MST, Hamiltonian chain by HC, and minimal Hamiltonian chain by MHC.

The algorithm is given as follows:

Step 1 Let $G^M = G$.

Step 2 Find the MST of $G^M$ by Kruskal's method.[2]

Step 3 Determine $\varepsilon^T = \sum_i [(d_i^T - 2) > 0]$.

Step 4 If $\varepsilon^T = 0$, then the MST is the MHC. Stop with solution.

Step 5 If $\varepsilon^T = 2$, then the algorithm is undefined. Stop.

Step 6 If $\varepsilon^T = 1$, find all edges incident to $v_i$, in the spanning tree, such that $d_i^T > 2$.

Step 7 For each edge, define $G^R \varepsilon R$, where $G^R$ excludes that edge from $G^M$.

Step 8 Find the MST of each $G^R$.

Step 9 Calculate $C^R = \Sigma c_{ij}\ s_{ij}^R$, for each $G^R$.

Step 10 Determine if $\varepsilon^T = 0$. That is the MST is an HC.

Step 11 If for any $G^R$, $\varepsilon^T = 0$, then find $C^* = \min C^R$ such that $\varepsilon^T = 0$, for the given $G^R$.

Step 12 Eliminate any $G^R$ such that $C^R > C^*$ if a $C^*$ exists.

Step 13 Sort all $G^R$ in ascending order of $C^R$.

Step 14 Let $G^M = G^R$ with min $C^R$.

Step 15 Exclude $G^M$ from R. Go to step 4.

[2]See Appendix D for Kruskal's minimal spanning tree algorithm.

Example of Christofides Branch and Bound Algorithm

See Figure 3.1 for graph G.

Step 1 $G^M = G$.

Step 2 Minimal spanning tree of $G^M$ given in Figure 3.1. (Total distance = 19.)

Step 3 $\varepsilon^T = 1$ as vertex b has $d^T = 3$.

Step 6 Edges in spanning tree incident to vertex b: (a,b), (b,c), (b,d).

Partial Graphs Formed, with Minimal Spanning Tree

| | Step 7 | Step 8 | Step 9 | Step 10 |
|---|---|---|---|---|
| Partial Graph ($G^R$) [case] | Excluded Edge from $G^M$ | New Edge in MST | Total Distance [$C^R$] | Is MST an HC? Does $\varepsilon^T = 0$? |
| 1 | (a,b) = 6 | (a,c) = 7 | 20 | Yes |
| 2 | (b,d) = 4 | (c,d) = 6 | 21 | Yes |
| 3 | (b,c) = 3 | (c,d) = 6 | 22 | No |

Step 11 $C^* = 20$, as $\varepsilon^T = 0$ for case 1.

Step 12 Eliminate cases 2 and 3. $C^R > 20$. (suboptimal)

Step 13 $G^R$ in order: case 1.

Step 14 Let $G^M = G^R$ for case 1.

Step 15 Exclude $G^R$ for case 1 from R.

Step 4 $\varepsilon^T = 0$ for $G^M$. Case 1 is a Hamiltonian chain, therefore the solution.

Minimal Hamiltonian Chain: a to c to b to d to e

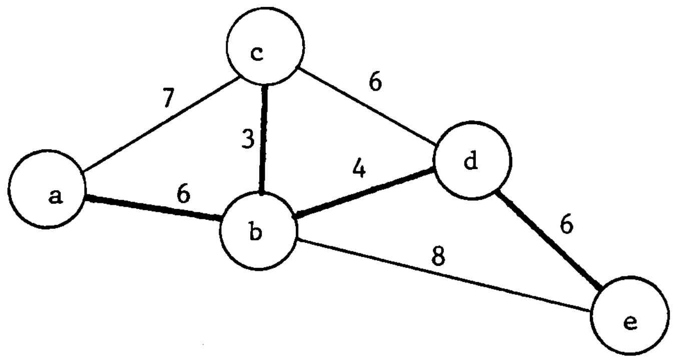


Figure 3.1. Graph G, with minimal spanning tree.

## Summary and Comments

Christofides uses basic principles relating a Hamiltonian chain to a spanning tree to find the minimal Hamiltonian chain. He develops a branch and bound algorithm to eliminate various edges in a graph. For each edge in the minimal spanning tree incident to an oversaturated vertex, he considers a branching case of eliminating that edge. If a single vertex is oversaturated by one degree, this is a simple matter of having three cases of eliminating one of three edges. If there are two or more oversaturated vertices, or a vertex is oversaturated by two or more degrees, the exact way of eliminating edges to form partial graphs has not been fully defined by Christofides. Thus the algorithm needs to be defined where $\varepsilon^T > 2$.

The algorithm has been considered by Christofides for complete graphs, but was demonstrated here for a graph that is not complete. If a graph is not complete it would be possible for given branching combinations to be infeasible, and perhaps no feasible solution to exist at all. Termination criteria for infeasibilities need to be developed.

# Chapter 4

# MODIFIED BRANCH AND BOUND ALGORITHM FOR FINDING THE MINIMAL HAMILTONIAN CHAIN

The algorithm discussed in the last chapter is a branch and bound procedure for finding the minimal Hamiltonian chain of a graph. In this chapter the algorithm is modified to include all situations where the previous algorithm is undefined.

## Development of the Algorithm

The algorithm developed by Christofides has only been applied for $\varepsilon^T \leq 1$. Therefore some modifications are required when $\varepsilon^T \geq 2$. One needs to treat the situation where more than one oversaturated vertex is in the minimal spanning tree, as well as the situation where any single vertex is oversaturated by two or more degrees. The simplest approach would be to branch on only one edge at a time, where all individual cases of eliminating a single oversaturated edge would be considered. However, when $\varepsilon^T$ is large many sequential branchings would be necessary before $\varepsilon^T$ may approach zero. In contrast, the other extreme would be to take all possible combinations of eliminating oversaturated edges from the graph in a single branching operation. This would result in an enormous number of branches in a single branching operation if $\varepsilon^T$ is large.

The algorithm given here will consider all combinations of eliminating incident spanning tree edges to a single oversaturated vertex, $v_i$, in one branching operation. All resulting partial graphs will have two edges incident to $v_i$, from the given spanning tree branched upon, one if $v_i$ is an endpoint. The oversaturated vertex with the fewest number of edge combinations is always the one chosen to branch upon. This should help to minimize the number of overall branchings. Endpoints are fixed in the algorithm.

If a non-endpoint vertex has $m \geq 3$ incident spanning tree edges, then at least m-2 of these edges must be eliminated to form a Hamiltonian chain. Thus for any oversaturated non-endpoint vertex, $v_i$, all combinations of m-2 out of m edges are considered for elimination. For endpoints each combination includes m-1 edges out of m, if $m \geq 2$. The $v_i$ with the least number of combinations is chosen. A partial graph is defined for each combination of edges eliminated from the previous graph.

Since the elimination of edges in a graph may result in a partial graph with no feasible Hamiltonian chain, termination criteria for given branching combinations are needed. If a spanning tree does not exist for a graph, then the graph is disconnected, hence no Hamiltonian chain exists. Also if a non-endpoint vertex has only one incident edge then the graph has no Hamiltonian chain. Thus any branching combination will terminate if either condition is detected. If no partial graph remains to branch upon and a Hamiltonian chain has not been found, then no Hamiltonian chains exists. This would terminate the algorithm.
The steps are formalized as follows:

## Modified Branch and Bound Algorithm

Let G be an undirected graph. Let $v_1$ and $v_2$ be the Hamiltonian chain endpoints. Let R be a set of graphs $G^R$. Let $e_{ij}^R = 1$ if an edge exists in $G^R$ incident to $v_i$ and $v_j$. Let $s_{ij}^R = 1$ if this edge is included in the minimal spanning tree of $G^R$. Let $G^M$ be the graph currently being branched upon. For brevity, again denote minimal spanning tree by MST, Hamiltonian chain by HC, and minimal Hamiltonian chain by MHC.

| | |
|---|---|
| Step 1 | Let $G^M = G$. |
| Step 2 | Find the MST of $G^M$ by Kruskal's method.[1] |
| Step 3 | If no MST exists, then no HC exists. Stop. |
| Step 4 | Determine $\varepsilon^T = \sum_i [(d_i^T - 2) > 0]$. |
| Step 5 | If $\varepsilon^T = 0$, then MST is the MHC. Stop with solution. |
| Step 6 | For each $v_i \neq v_1, v_2$, if $d_i^T > 2$, then define all sets, $S_i$, of m-2 out of m incident MST edges ($m = d_i^T$). Find the number of $S_i$ for that $v_i$. |
| Step 7 | For $v_i = v_1, v_2$, if $d_i^T > 1$, then define all sets, $S_i$ of m-1 out of m incident MST edges. ($m = d_i^T$). Find the number of $S_i$ for that $v_i$. |
| Step 8 | Find the $v_i$ with the fewest number of sets, $S_i$. |
| Step 9 | For that $v_i$, find all the edges in each set, $S_i$. |
| Step 10 | For each $S_i$, define $G^R \varepsilon R$, where $G^R$ excludes from $G^M$ all edges in that $S_i$. |

[1]See Appendix D for Kruskal's minimal spanning tree algorithm.

Step 11 If for any $G^R$, for $v_i \neq v_1, v_2$, if $\sum_j e_{ij}^R = 1$, then exclude that $G^R$ from R.

Step 12 Find the MST of each $G^R$.

Step 13 Calculate $C^R = \Sigma c_{ij} s_{ij}^R$, for each $G^R$.

Step 14 If no MST exists for some $G^R$, exclude that $G^R$ from R.

Step 15 Determine if $\varepsilon^T = 0$, for each $G^R$. If so the MST is an HC.

Step 16 If for any $G^R$, $\varepsilon^T = 0$, then find $C^* = \min C^R$ such that $\varepsilon^T = 0$ for the given $G^R$.

Step 17 Eliminate any $G^R$ such that $C^R > C^*$.

Step 18 Sort all $G^R$ in ascending order of $C^R$.

Step 19 Let $G^M = G^R$ with min $C^R$. If R is empty, then no HC exists. Stop.

Step 20 Exclude $G^M$ from R. Go to step 5.

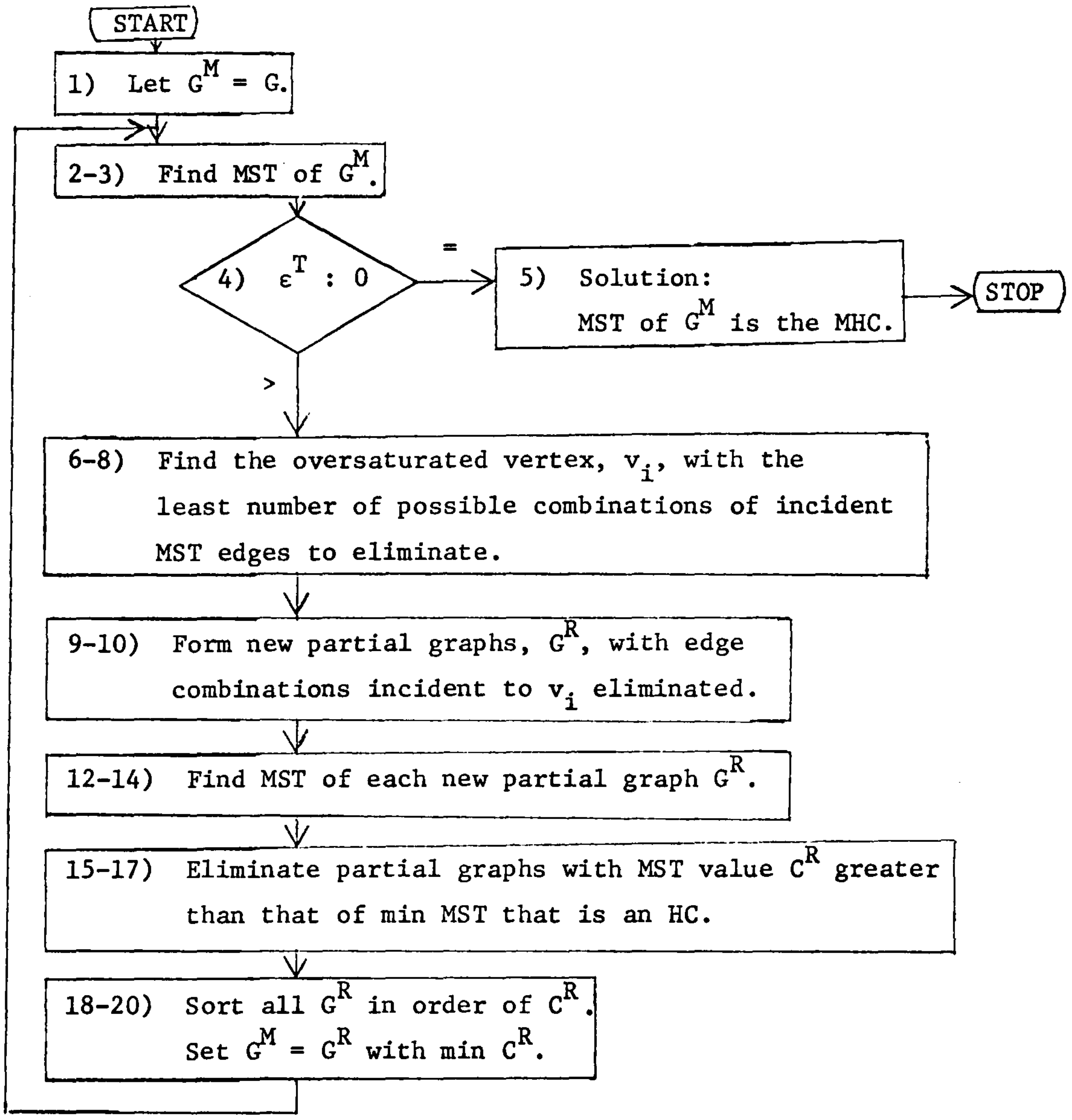

START
1) Let $G^M = G$.
2-3) Find MST of $G^M$.
4) $\varepsilon^T : 0$
=
5) Solution: MST of $G^M$ is the MHC.
STOP
>
6-8) Find the oversaturated vertex, $v_i$, with the least number of possible combinations of incident MST edges to eliminate.
9-10) Form new partial graphs, $G^R$, with edge combinations incident to $v_i$ eliminated.
12-14) Find MST of each new partial graph $G^R$.
15-17) Eliminate partial graphs with MST value $C^R$ greater than that of min MST that is an HC.
18-20) Sort all $G^R$ in order of $C^R$. Set $G^M = G^R$ with min $C^R$.

Figure 4.1 Flowchart of Modified Branch and Bound Algorithm

## Example of Modified Branch and Bound Algorithm

See Figure 4.2 for graph G.

Step 1 Let $G^M = G$ with vertices a and f as endpoints.

Step 2 Minimal spanning tree of $G^M$ given in Figure 4.2. (Total distance = 23.)

Step 4 $\varepsilon^T = 2$. Vertex b has $d^T = 3$. Endpoint vertex f has $d^T = 2$.

Step 6 For vertex b, m=3. Therefore $\binom{m}{m-2} = \binom{3}{1} = 3$ sets of $S_b$.

Step 7 For vertex f, m=2. Therefore $\binom{m}{m-1} = \binom{2}{1} = 2$ sets of $S_f$.

Step 8 Vertex f has least number of $S_i$, 2.

Step 9 $S_f$ for $v_f$: $\{(d,f)\}$, $\{(e,f)\}$ .

Partial Graphs Formed, with Minimal Spanning Tree

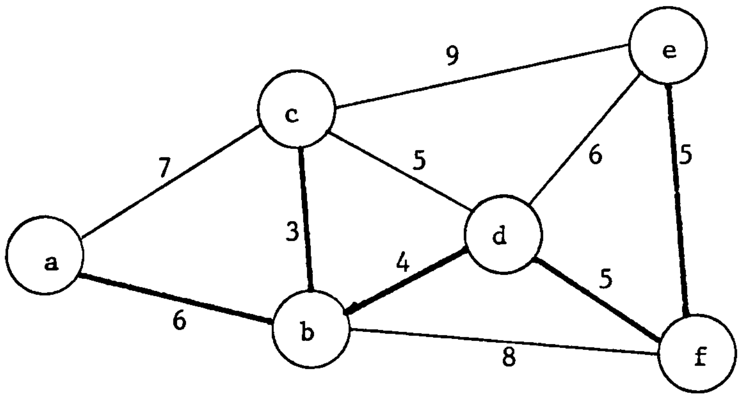
| | Step 10 | Step 12 | Step 13 | Step 15 |
|---|---|---|---|---|
| Partial Graph ($G^R$) (case) | Excluded Edge from $G^M$ | New Edge in MST | Total Distance ($C^R$) | Is MST an HC? Does $\varepsilon^T = 0$? |
| 1 | (d,f) = 5 | (d,e) = 6 | 24 | No |
| 2 | (e,f) = 5 | (d,e) = 6 | 24 | No |

Step 18 $G^R$ in order: case 1, case 2

Step 19 Let $G^M = G^R$ for case 1.

Step 20 Exclude $G^R$ for case 1 from R.

Note -- Steps which produce no changes are omitted.



a,f - endpoints

Figure 4.2. Graph G, with minimal spanning tree.

Partial Graph 1 Further Branched

See Figure 4.3 for partial graph 1.

Step 5 $\varepsilon^T = 1$. Vertex b has $d^T = 3$.

Step 6 For vertex b, m=3. Therefore $\binom{m}{m-2} = \binom{3}{1} = 3$ sets of $S_b$.

Step 8 Vertex b has least number of $S_i$, 3. (Only vertex with any $S_i$.)

Step 9 $S_b$ for vertex b: {(a,b)}, {(b,c)}, {(b,d)}.

Partial Graphs Formed, with Minimal Spanning Tree

| | Step 10 | Step 12 | Step 13 | Step 15 |
|---|---|---|---|---|
| Partial Graph ($G^R$) (case) | Excluded Edge from $G^M$ | New Edge is MST | Total Distance ($C^R$) | Is MST an HC? Does $\varepsilon^T = 0$? |
| 3 | (a,b) = 6 | (a,c) = 7 | 25 | Yes |
| 4 | (b,d) = 3 | (c,d) = 5 | 26 | No |
| 5 | (b,d) = 4 | (c,d) = 5 | 25 | Yes |

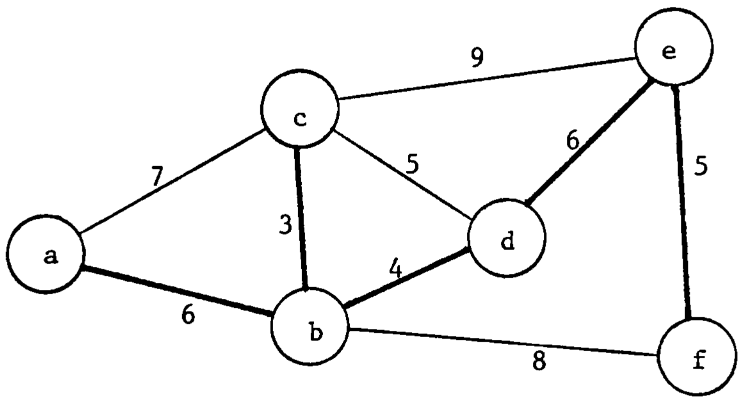
Step 16 $C^* = 25$, as $\varepsilon^T = 0$ for cases 3 and 5.

Step 17 Eliminate case 4. $C^R > 25$. (suboptimal)

Step 18 $G^R$ in order: case 2, case 3, case 5.

Step 19 Let $G^M = G^R$ for case 2.

Step 20 Exclude $G^R$ for case 2 from R.



a,f - endpoints

Total distance of MST = 24

Figure 4.3. Partial graph 1, with minimal spanning tree.

Partial Graph 2 Further Branched

See Figure 4.4 for partial graph 2.

It can be observed that edges (c,e) and (d,e) must be included in any Hamiltonian chain in partial graph 2 as any non-endpoint vertex must have a degree of 2 in the Hamiltonian chain. With these edges included in the spanning tree, the minimal spanning tree is as follows:

a - b - c - e - d - f
Total distance = (6)+(3)+(9)+(6)+(5) = 29

This spanning tree forms a Hamiltonian chain which is suboptimal. Therefore any partial graph resulting from case 2 being branched will be suboptimal, if feasible.

Steps 5-17 omitted

Step 18 $G^R$ in order: case 3, case 5

Step 19 Let $G^M = G^R$ for case 3.

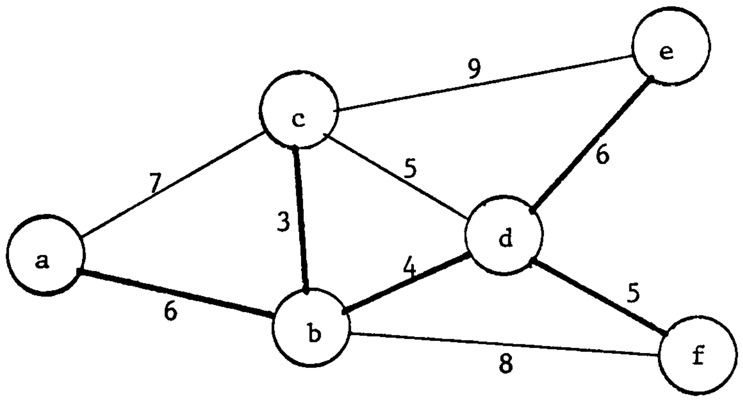
Step 20 Exclude $G^R$ for case 3 from R.

Step 5 $\varepsilon^T = 0$ for case 3. Case 3 is a Hamiltonian chain, therefore the solution.

Minimal Hamiltonian Chain: a to c to b to d to e to f.

Total distance = 25

Alternative solution - case 5: a to b to c to d to e to f.



a,f - endpoints

Total distance of MST = 24

Figure 4.4. Partial graph 2, with minimal spanning tree.

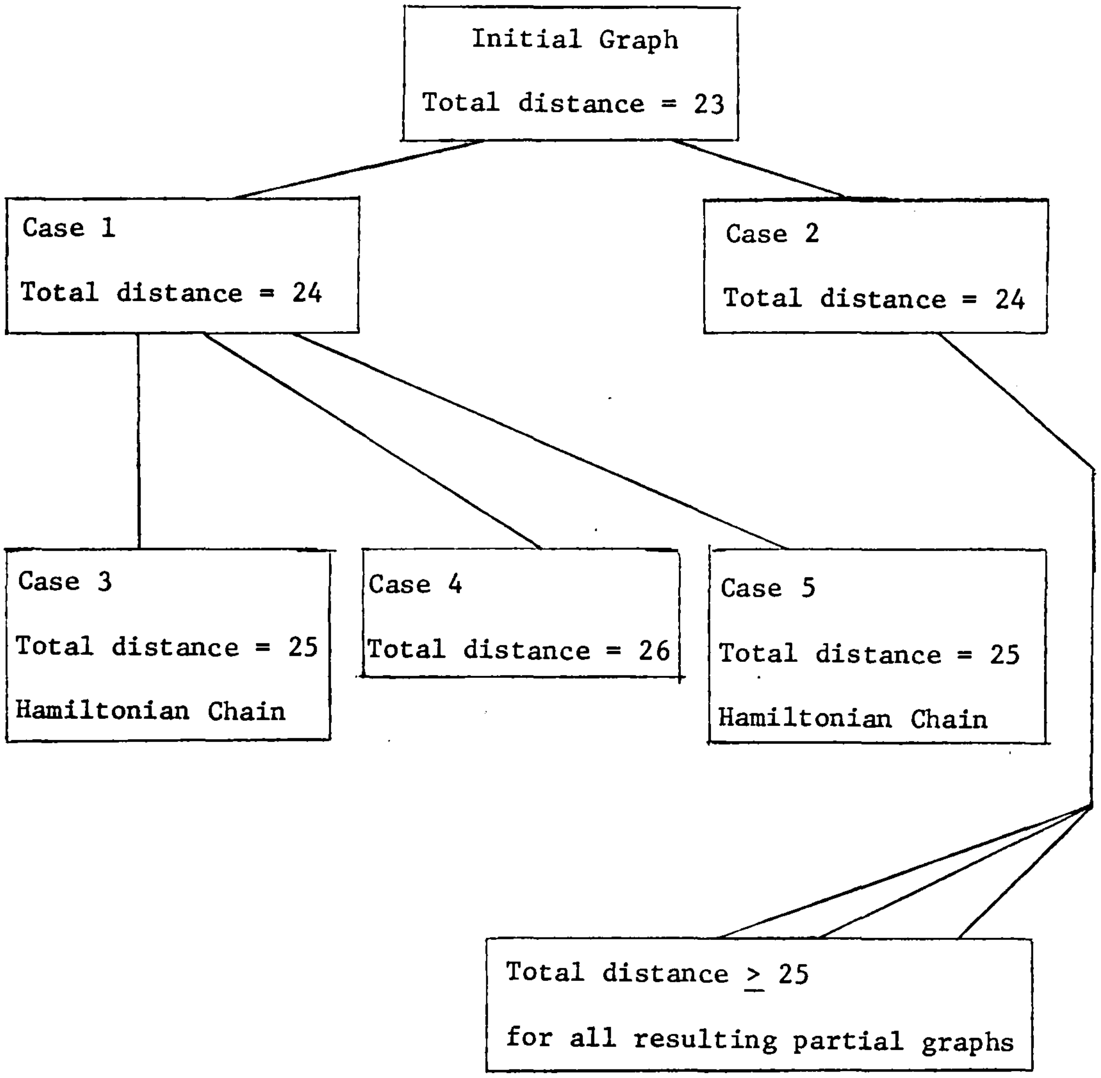


Figure 4.5 Summary of branching operations

## Summary and Comments

The algorithm in this chapter performs a branch and bound on the minimal spanning tree to find the minimal Hamiltonian chain. If the minimal spanning tree of the graph is a Hamiltonian chain it is the minimal Hamiltonian chain. Otherwise there exists at least one vertex $v_i$ such that $d_i^T>2$, or $d_i^T>1$ if $v_i$ is an endpoint. For a given such $v_i$, all combinations of eliminating all but two spanning tree edges incident to $v_i$ are taken at any branching. This includes all but one edge if $v_i$ is an endpoint. The branching process is successively repeated for the resulting partial graph with the minimal spanning tree of least total distance. Without a bound any partial graph can be branched until either its minimal spanning tree is a Hamiltonian chain or no spanning tree exists. Any Hamiltonian chain contained in a given graph is contained in at least one of the resulting graphs, when the given graph is branched upon. Thus, all Hamiltonian chains can be generated.

A bound results when a partial graph is found having a minimal spanning tree that is a Hamiltonian chain. The total distance value of the minimal spanning tree of the partial graph of least total value that is also Hamiltonian chain, serves as a bound. All partial graphs, having a minimal spanning tree of greater value, are eliminated. Any partial graph found to be infeasible is also eliminated. When a partial graph to be branched, has a minimal spanning tree that is a Hamiltonian chain, this spanning tree is the solution.

Chapter 5

# APPROACH TO PROBLEM SOLUTION

The proposed procedure will make use of Christofides method of eliminating edges in a spanning tree to eventually find the minimal Hamiltonian chain with fixed endpoints. The modified branch and bound algorithm in Chapter 4 will be used in conjunction with decomposition of the original graph. In order to reduce a large problem to a more manageable size the graph is partitioned into subproblems. Each resulting subproblem will be solved with much less branching operations than would be required for the graph as a whole.

The decomposition of the graph will be based on necessary conditions for a Hamiltonian chain to exist. Edges that must be in any feasible Hamiltonian chain will be determined as well as edges that cannot be in any Hamiltonian chain. A subgraph of the graph will be partitioned when it is determined that any chain of the entire graph can only pass through that subgraph once without revisiting vertices.

In order for the decomposition of the graph to be applicable, the graph should contain relatively few edges incident to each vertex. No decomposition is possible for a complete graph. In order for any decomposition to begin to take place either of the following conditions must occur: The graph contains

a) a non-endpoint vertex of degree two.

b) a subgraph that can be partitioned by a cutset about two vertices in the subgraph, as shown in Figure 5.1.

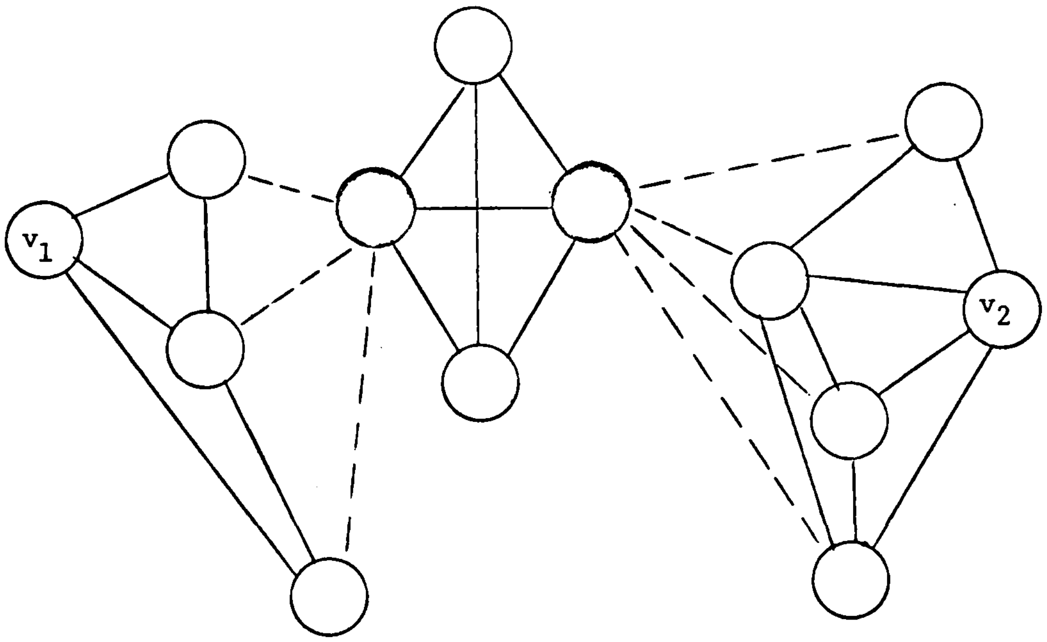


Figure 5.1. Graph G with a subgraph having a cutset about two vertices.

For a large problem with many incident edges per vertex, decomposition is not likely to occur. The resulting branch and bound problem will likely require extensive computations. The proposed procedure is not intended to handle the above situation. However, as an alternative the graph could be modified by eliminating numerous high valued edges and then the proposed procedure applied.

The procedure includes two phases. The first phase decomposes the original graph. The second phase performs a branch and bound operation on each partitioned subproblem from the first phase.

## Phase One

Initially all vertices of degree two are identified. Both incident edges to these vertices must be in any Hamiltonian chain. Any edge that must be in any Hamiltonian chain is defined to be a necessary edge. Any edge that forms a circuit with the necessary edges is eliminated. Any edge that is adjacent at the same vertex to two necessary edges is also eliminated. The endpoints are treated like all other vertices by adding two additional vertices to the graph, each of which is adjacent to one of the respective endpoints only.

Subgraphs are identified having the property that a cutset partitions the subgraph from the remainder of the graph about two vertices in the subgraph, as in Figure 5.1. These subgraphs are each replaced in the original graph by a single necessary edge connecting the vertices that separate the subgraph from the original graph. The subgraph becomes a subproblem where the separation vertices become endpoints of its Hamiltonian chain.

As decompositions occurs, edges in the original graph become necessary edges while other edges become infeasible. New situations may result where decomposition can occur. As necessary edges are included in any Hamiltonian chain, finding these edges has priority over finding subgraphs to partition. Therefore, whenever decomposition occurs in the original graph, the procedure reiterates back to the initial steps of searching for necessary edges.

The fewer incident edges per vertex the more the likelihood of decomposition occuring. This is partially due to the number of vertices of degree two that result in the process. For vertices of degree greater than two, the fewer incident edges the more likely the vertex will have degree two once other edges are removed from the graph.

In order that subgraphs can be partitioned they must first be constructed. A sequence of operations is employed to construct subgraphs. The analysis of these operations appears in Appendix E. There are certain instances where particular subgraphs that actually exist will not be identified.

## Phase Two

For every partitioned subgraph in Phase One or the whole graph in the case of no partitioning, its minimal Hamiltonian chain is found in Phase Two. Together with the necessary edges from Phase One, the minimal Hamiltonian chain of each subgraph will be included in the minimal Hamiltonian chain of the entire graph. The modified branch and bound algorithm given in Chapter 4 is used to find the minimal Hamiltonian chain of each subproblem.

A minimal spanning tree is initially found. Then all oversaturated vertices are found. For every oversaturated vertex $v_i$, all combinations of spanning tree edges are found such that if eliminated, $v_i$ would have two of its incident spanning tree edges remaining, one if $v_i$ is an endpoint. The $v_i$ with the fewest edge combinations is branched upon. For each edge combination a new graph is created, which excludes the given edges from the previous graph. The partial graph with the least valued minimal spanning tree is always the one further branched.

When a partial graph is found having a minimal spanning tree that is a Hamiltonian chain, the total distance of its minimal spanning tree becomes a bound. All partial graphs with minimal spanning trees of greater value are eliminated. When a new partial graph is found having a minimal spanning tree that is a Hamiltonian chain with less total distance than the bound, its distance value becomes the new bound. When the partial graph to be branched has a minimal spanning tree that is a Hamiltonian chain, this Hamiltonian chain is the solution.

If a partial graph is found to be infeasible with respect to the existence of a Hamiltonian chain, it is eliminated. That is if it contains a non-endpoint vertex of degree one or it contains no spanning tree then it is eliminated. If all partial graphs are terminated due to infeasibility then the partitioned subproblem is infeasible. As a result the entire problem is infeasible since a Hamiltonian chain must cover the entire graph.

In summary, the proposed procedure is a two phase procedure. The first phase attempts to decompose the graph. It finds edges necessary for the Hamiltonian chain, and partitions the graph into subproblems. The second phases uses a branch and bound on the minimal spanning tree to find the minimal Hamiltonian chain. Each branching operation eliminates edges in the spanning tree from the graph.

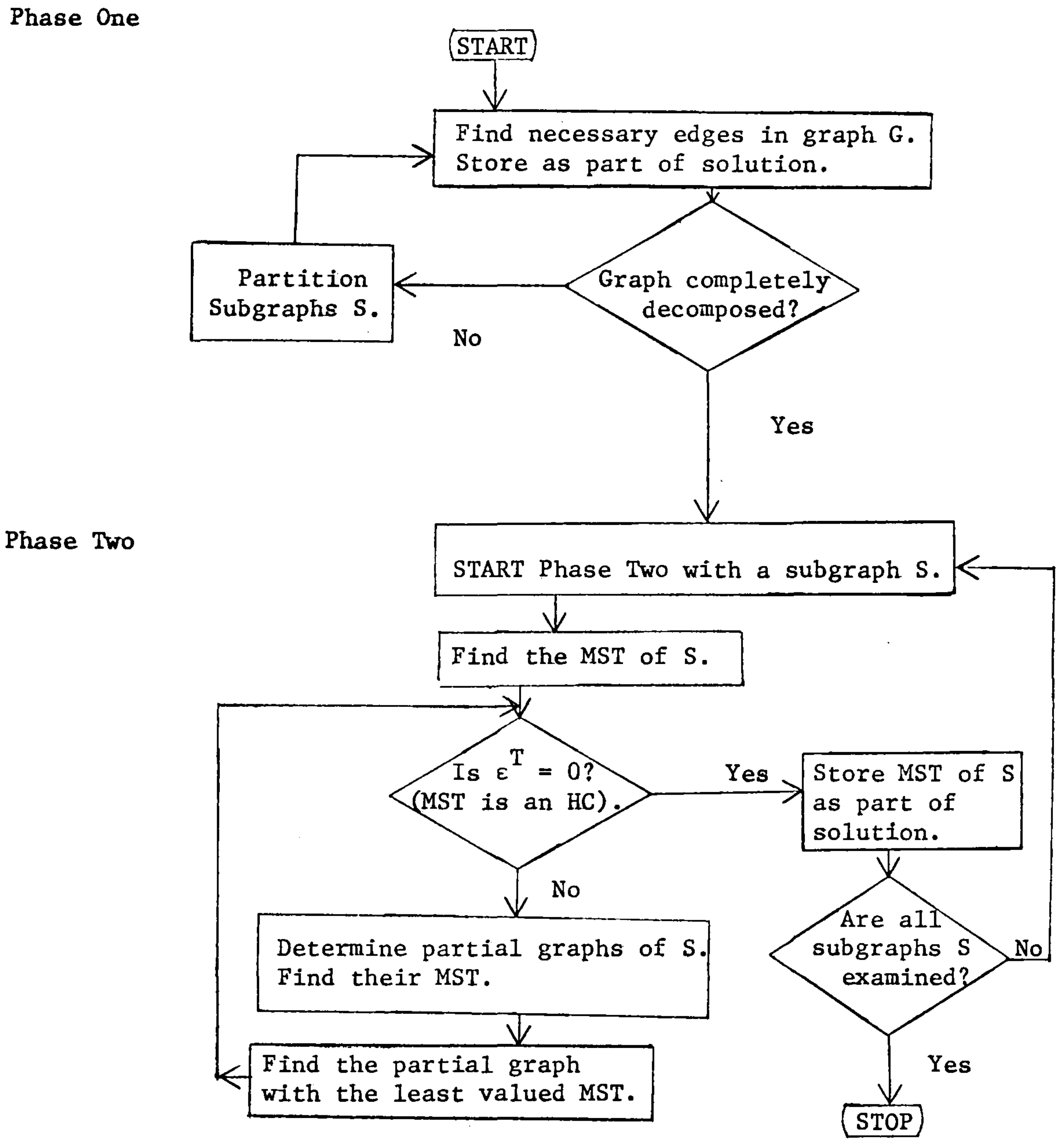

Phase One
START
Find necessary edges in graph G. Store as part of solution.
Graph completely decomposed?
No
Partition Subgraphs S.
Yes
Phase Two
START Phase Two with a subgraph S.
Find the MST of S.
Is $\varepsilon^T = 0$? (MST is an HC).
Yes
Store MST of S as part of solution.
No
Determine partial graphs of S. Find their MST.
Find the partial graph with the least valued MST.
Are all subgraphs S examined?
No
Yes
STOP

Figure 5.2 Flowchart of Proposed Procedure

# Chapter 6

# EXISTENCE CONDITIONS FOR HAMILTONIAN CHAINS

The analysis given in this chapter deals with the decomposition of a graph with respect to the feasibility of a Hamiltonian chain. Observations and theorems are given concerning edge feasibility, subgraph partitioning, and other conditions relating to Hamiltonian chains. To facilitate the analysis, for a given graph G, a graph $G^+$ is formed which contains an additional vertex adjacent to each endpoint in G.

## Algebraic Formulations for Analysis of Existence Conditions

For all possible pairs of vertices $v_i$ and $v_j$,

$$\begin{aligned} e_{ij} &= 1 \quad \text{if an edge exists, where } i \neq j; \\ e_{ij} &= 0 \quad \text{otherwise;} \end{aligned} \tag{6.1}$$

$$\begin{aligned} a_{ij} &= 1 \quad \text{if the edge is in a given Hamiltonian chain;} \\ a_{ij} &= 0 \quad \text{otherwise;} \end{aligned} \tag{6.2}$$

$$e_{ij} = e_{ij}; \text{ and} \tag{6.3}$$

$$a_{ij} = a_{ji}. \tag{6.4}$$

The end vertices, $v_1$ and $v_2$ in graph G, only need to have one incident edge in the Hamiltonian chain. In order that all vertices, $v_i$ in G, have the same properties with respect to the number of incident edges, a new graph $G^+$ is created from G.

This is done by adding two additional vertices, $v_*$ and $v_\#$, and two edges which make $v_*$ and $v_\#$ the end vertices in $G^+$. Thus, $v_*$ is adjacent to $v_1$, and $v_\#$ to $v_2$. As a result, for graph $G^+$, the following equations are true:

$$a_{*1} = 1 \text{ and } a_{\#2} = 1 \qquad (6.5)$$

$$e_{*j} = 0 \quad \text{for } v_j \neq v_1, \text{ where } v_j \varepsilon G^+$$

$$\text{and} \qquad e_{\#j} = 0 \quad \text{for } v_j \neq v_2, \text{ where } v_j \varepsilon G^+ \qquad (6.6)$$

where $v_1, v_2, v_*, v_\#$ are in $G^+$, but $v_*, v_\#$ are not in G.

Now as $\sum_{v_j \varepsilon G} a_{ij} = 1$ for $v_i = v_1, v_2$, then

$$\sum_{v_j \varepsilon G^+} a_{ij} = 2 \quad \text{for } v_i = v_1, v_2 \qquad (6.7)$$

It should be noted that any Hamiltonian chain in $G^+$ includes a Hamiltonian chain in G plus exactly two additional edges.

To begin the analysis of existence conditions, a Hamiltonian chain is a chain that connects all vertices in a graph. This leads to the following observation:

Existence Observation 1

If a graph is disconnected, then no Hamiltonian chain exists.

## Necessary Edges

Any Hamiltonian chain in $G^+$ must include exactly two edges incident to each vertex $v_i$ in G. Therefore if only two edges are incident to some $v_i$ in G, then both these edges must be included in any Hamiltonian chain in $G^+$. Any edge that must be included in a Hamiltonian chain is defined to be a necessary edge. If any vertex $v_i$ in G has only one incident edge in $G^+$, then no Hamiltonian chain exists. If three or more edges incident to some $v_i$ in G are necessary edges, then no Hamiltonian chain exists. Only two edges incident to a given vertex, can be included in a Hamiltonian chain. If two edges incident to some $v_i$ in G are determined to be necessary edges, then the remaining edges incident to $v_i$ must be excluded from the Hamiltonian chain.

If a chain of necessary edges is formed, then the edge or edge sequence connecting the endpoints of the chain cannot be in a Hamiltonian chain. Otherwise a circuit would be created. This leads to the following observation:

Existence Observation 2

If a set of necessary edges forms a circuit, then no Hamiltonian chain exists.

Figure 6.1 provides an illustration on this situation.

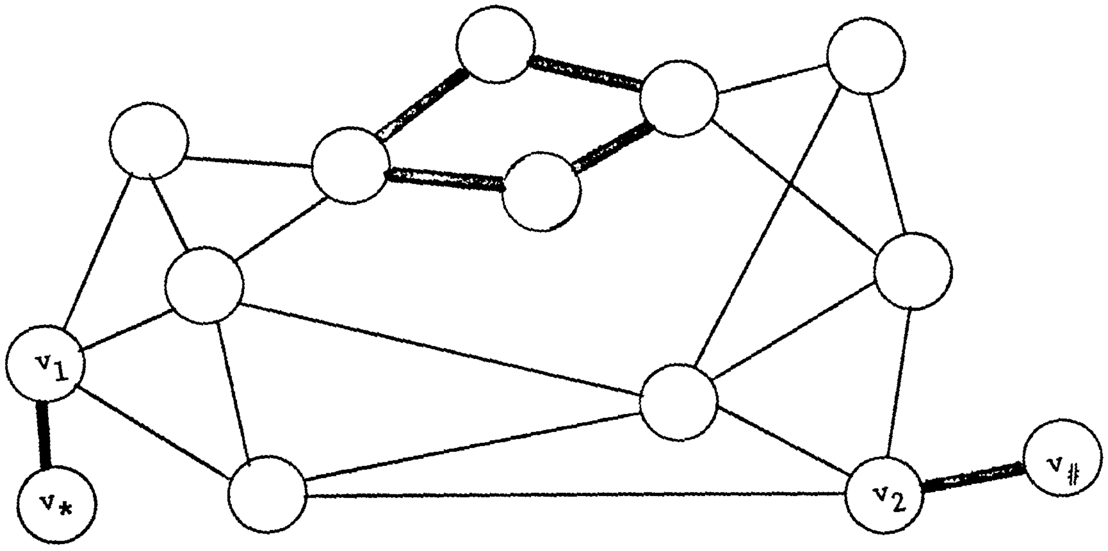


Figure 6.1. Graph $G^+$ with a circuit of necessary edges.

A Hamiltonian chain is a chain from endpoint to endpoint which covers all vertices in a graph. But it is possible for a chain of necessary edges to include both endpoints but not cover the graph. This leads to the following observation:

Existence Observation 3

If a set of necessary edges forms a chain between the endpoints but does not form a vertex covering, then no Hamiltonian chain exists.

Proof: Let A be a subgraph of G that includes $v_1$ and $v_2$ (but not all vertices in G), such that a chain of necessary edges connects all vertices in A, with endpoints $v_1$ and $v_2$. Then for each $v_i$ in A, $\sum_{v_j \varepsilon G^+} a_{ij} = 2$. Let $A^+$ be A plus vertices $v_*$ and $v_\#$ and the two edges incident to them.

For a Hamiltonian chain to exist in G, there exists some $v_k$ not in $A^+$, such that $a_{ki} = 1$ for some $v_i$ in $A^+$. Then for such a $v_i$ in $A^+$, $\sum_{v_j \varepsilon G^+} a_{ij} = 3$. But $\sum_{v_j \varepsilon G^+} a_{ij} = 2$ for all $v_i$ in G for a Hamiltonian chain to exist. Hence no Hamiltonian chain exists.

Figure 6.2 provides an illustration of this situation.

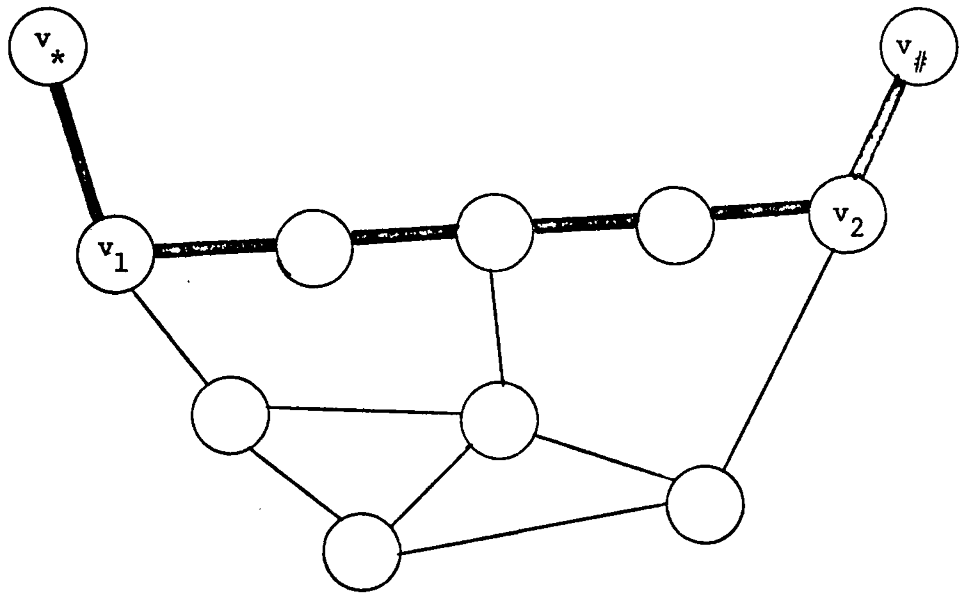


Figure 6.2. Graph $G^+$ with an endpoint to endpoint chain of necessary edges.

## Subgraph Partitioning

Certain graphs with subgraphs which are connected to the remainder of the graph via only one vertex in the subgraph can be determined as infeasible with respect to the existence of a Hamiltonian chain. Other graphs can be determined to contain subgraphs which can be partitioned as separate problems with respect to finding a Hamiltonian chain.

A given graph G may have a subgraph S such that only one vertex in S is adjacent to a vertex not in S, where neither endpoint in G is in S. This leads to the following observation:

Existence Observation 4

Let S be a subgraph of G ($v_*$ and $v_\#$ cannot be in S) such that a proper cutset partitions S from $G^+$ such that exactly one vertex in S, $v_h$, is incident to the edges in the cutset. (S has a least two vertices). Then no Hamiltonian chain exists in $G^+$.

Proof: Let $v_h$ be a vertex in a subgraph S in G, such that $e_{ij} = 1$ for $v_i \varepsilon S$, $v_j \varepsilon G^+$, $v_j \not\varepsilon S$, only if $i = h$. ($v_*$ and $v_\#$ are not in S.) For any vertex $v_i$ in S ($v_i \neq v_h$), any path from $v_i$ to $v_\#$ must include $v_h$. Also any path from $v_i$ to $v_\#$ must include $v_h$. Thus any chain between $v_*$ and $v_\#$ covering $v_i$ must include $v_h$ twice. Hence no Hamiltonian chain exists.

Figure 6.3 provides an illustration of this situation.

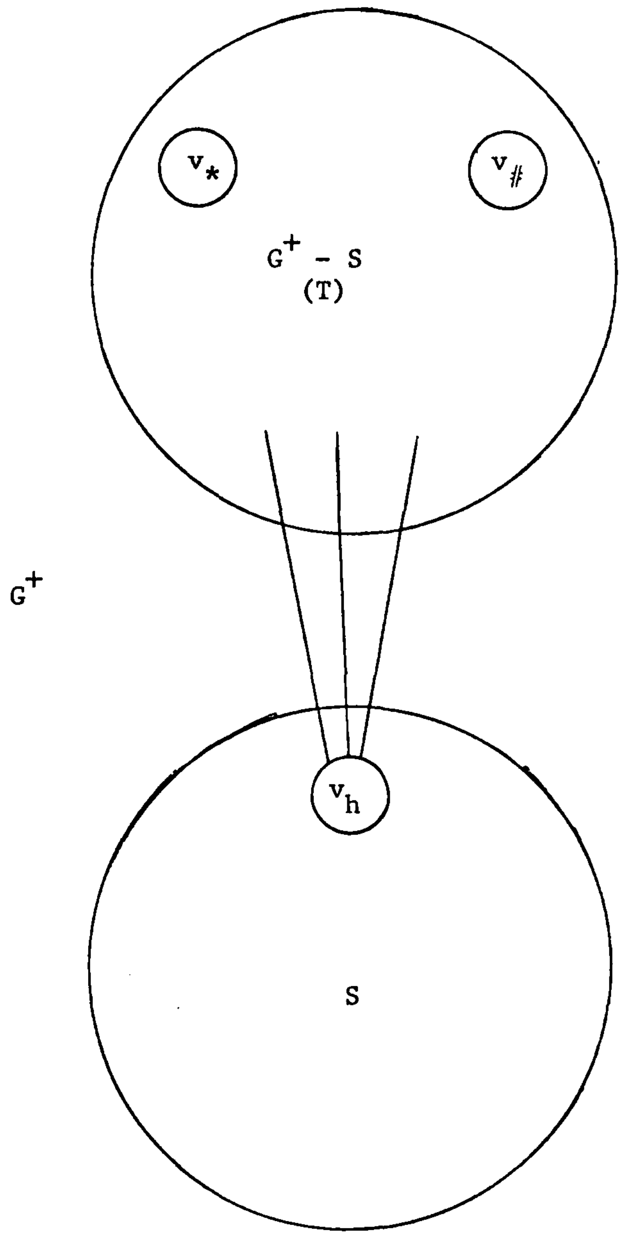


Figure 6.3. Graph $G^+$ with infeasibility due to subgraph S.

It is of interest to identify any subgraph S of G such that any Hamiltonian chain of G includes a Hamiltonian chain of S. Such subgraphs could be partitioned as a separate problem with respect to finding a Hamiltonian chain. A certain class of subgraphs are defined as separable subgraphs. It will be shown that they satisfy the above criteria.

A <u>separable subgraph</u> is a subgraph S of G such that

a) a cutset partitions $G^+$ into two subgraphs S and T such that every edge in the cutset is incident to one of two vertices, $v_h$ or $v_k$ in S. (Both $v_h$ and $v_k$ have at least one incident edge in the cutset.)

b) S is connected and has at least three vertices.

Specifically a separable subgraph S has at least three vertices and exactly two vertices $v_h$ and $v_k$ in S which are adjacent to some $v_j$ not in S. Figure 6.4 provides an illustration of a separable subgraph.

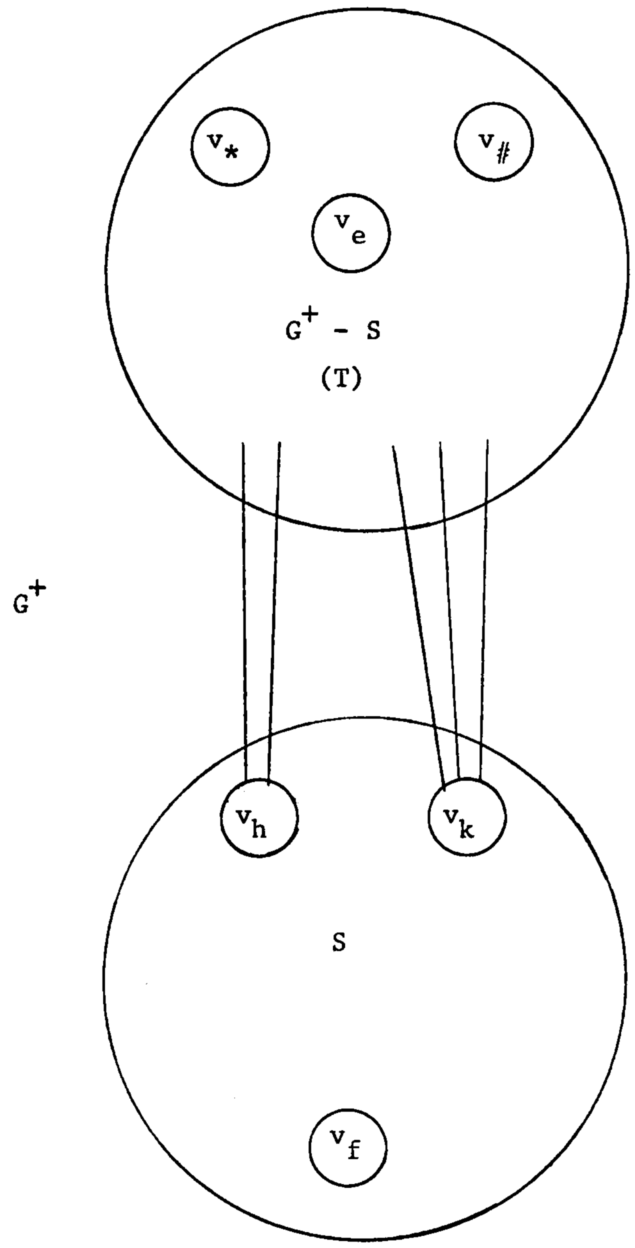


Figure 6.4. Graph $G^+$ with separable subgraph S.

This leads to the following theorem for separable subgraphs:

Theorem 1

Let S be a separable subgraph in $G^+$ with vertices $v_h$ and $v_k$ incident to the cutset, such that the following hold:

a) There exists some $v_e$ in G where $v_e$ is not in S.

b) Vertices $v_*$ and $v_\#$ are not included in S.

Then any Hamiltonian chain in $G^+$ includes a Hamiltonian chain in S with endpoints $v_h$ and $v_k$.

Proof: Let $v_h$ and $v_k$ be vertices in a subgraph S in G ($v_*$ and $v_\#$ are not in S) such that

$$\sum_{v_j \notin S} e_{ij} \geq 1 \quad \text{for } i = h,k \text{ where } v_i \varepsilon S \text{ and } v_j \varepsilon G^+ \qquad (6.8)$$

$$\text{Also } e_{ij} = 0 \text{ for } v_i \varepsilon S,\ i \neq h,k \text{ and } v_j \notin S,\ v_j \varepsilon G^+ \qquad (6.9)$$

There exists some $v_f$ in S, where $v_f \neq v_h, v_k$.

Also there exists some $v_e$ in G where $v_e$ is not in S.

Since S does not contain $v_*$ and $v_\#$, the endpoints in $G^+$, then any Hamiltonian chain must enter and leave S. As S contains only two vertices adjacent to a vertex not in S, $v_h$ and $v_k$, then any path from $v_*$ to $v_\#$, passing through S, must enter and leave through these two vertices, providing no vertex is traversed more than once. In order to have a vertex covering, any vertex in S must be reached within the path between $v_h$ and $v_k$. Any vertex not in S cannot be included in the path between $v_h$ and $v_k$, without revisiting vertices.

Therefore any Hamiltonian chain in G must include a Hamiltonian chain in S with endpoints $v_h$ and $v_k$.

Figure 6.4 provides an illustration of this situation.

As a result of Theorem 4, S can be partitioned from $G^+$ as a separate graph. The minimal Hamiltonian chain of S with endpoints $v_h$ and $v_k$ is included in the minimal Hamiltonian chain of G. This leads to the following obseration:

Existence Observation 5

The edge in a separable subgraph incident to both $v_h$ and $v_k$ cannot be included in a Hamiltonian chain, otherwise a circuit will result.

Now suppose the conditions of theorem 4 hold with the addition that $v_h$ and $v_k$ are the endpoints in G, $v_1$ and $v_2$ respectively. This leads to the following observation:

Existence Observation 6

If a separable subgraph S includes $v_1$ and $v_2$, the endpoints in G, but doesn't include all vertices in G, then no Hamiltonian chain exists.

In this situation the endpoints in S, $v_h$ and $v_k$, must be in $v_1$ and $v_2$. As a chain entirely within S is formed from $v_1$ to $v_2$, then all vertices in G but not in S cannot be included in this chain. Than any chain from $v_1$ to $v_2$ cannot form a vertex covering of G. Hence no Hamiltonian chain exists in this situation.

Now consider a special case where a cutset exists about three vertices in a subgraph S. The results of this theorem will allow the conditions of therem 4 to be applicable.

Theorem 2

If a cutset of three edges partitions $G^+$ into two subgraphs S and T, such that the following hold:

a) Vertices $v_h$, $v_k$, and $v_p$ in S, each have exactly one incident edge in the cutset. (None of these edges are adjacent to one another.)

b) The given edges incident to both $v_h$ and $v_k$ are necessary edges,

c) Vertices $v_*$ and $v_\#$ are not in S. (S is a subgraph of G.)

Then the edge in the cutset incident to $v_p$ cannot be included in any Hamiltonian chain in $G^+$.

Proof: Let $v_h$, $v_k$, and $v_p$ be vertices in a subgraph S in G ($v_*$ and $v_\#$ are not in S) such that

$e_{ij} = 1$ for $v_i \epsilon S$, $v_j \notin S$ only if $i = h,k,p$.

Also $\sum_{v_j \notin S} e_{ij} = 1$ for $i = h,k,p$ and $v_j \epsilon G^+$,

where $a_{ij} = 1$ if $e_{ij} = 1$ for $i = h,k$ and $v_j \notin S$.

None of these edges are adjacent to one another.

Let T be the set of vertices in $G^+$ not in S.

Vertices $v_*$ and $v_\#$ are in T. Then any Hamiltonian chain in $G^+$, starts and ends in T. From T, S is reached then T is returned to. Therefore at least two edges connecting S and T must be included the Hamiltonian chain. If S is returned to again, then T must be returned to again, involving two more edges connecting S and T. But there exist only three edges connecting S and T. Therefore only two of these edges can be included in a Hamiltonian chain. As the edges incident to $v_h$ and $v_k$ must be included, then the edge incident to $v_p$ must be excluded from the Hamiltonian chain. Hence $a_{ij} = 0$ for $i = p$ and $v_j \notin S$.

Figure 6.5 provides an illustration of this situation.

By the conditions of theorem 2, is can be observed that if the edge from $v_p$ to some $v_j$ not in S was a necessary edge, then no Hamiltonian chain exists. If $a_{ij} = 1$ for $i = p$ and $v_j \notin S$, this would contradict that $a_{ij} = 0$ for a Hamiltonian chain to exist.

The result of theorem 2 results in the conditions of theorem 1 being applicable. A cutset about two vertices in S now exists.

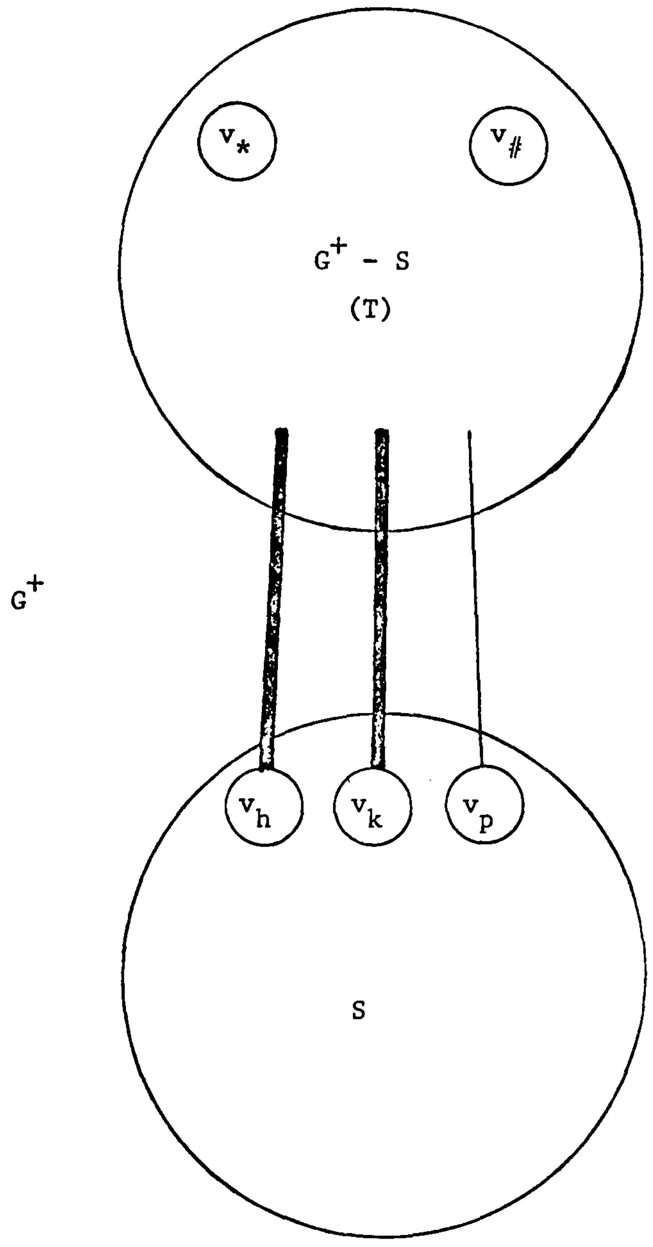


Figure 6.5. Graph $G^+$ with cutset of three edges partitioning subgraph S.

In summary, the two theorems, six observations, and other conditions for existence, determine the feasbility of various edges and subgraphs for a Hamiltonian chain to exist. Various edges can be determined as necessary edges for a Hamiltonian chain. Edges which cannot be in a Hamiltonian chain can also be determined. Two of the observations deal with sequences of necessary edges which lead to the non-existence of a Hamiltonian chain. Relationships between subgraphs which lead to the non-existence of a Hamiltonian chain can also be determined. Two of the observations deal directly with such situations. Finally, relationships between subgraphs which allow subgraphs to be partitioned can be determined. The two theorems given concern subgraph partitioning.

# Chapter 7

# PROPOSED PROCEDURE FOR FINDING THE MINIMAL HAMILTONIAN CHAIN

The proposed procedure for finding the minimal Hamiltonian chain of a graph is a two phase procedure. The first phase decomposes the graph. The second phase finds the minimal Hamiltonian chain of partitioned subproblems of Phase One.

In Phase One, the necessary edges for a Hamiltonian chain are found. The the graph is partitioned into subgraphs. A sequence of operations is included to generate potential subgraphs for partitioning. In Phase Two, the minimal Hamiltonian chain is found for each subgraph, using the branch and bound algorithm given in Chapter 4.

Phase One - Decomposition

Let G be an undirected graph with endpoints, $v_1$ and $v_2$. Without loss of generality, one can assume that G has no loops or parallel edges. If G had loops, one can eliminate them. If it had parallel edges, one retains only the smallest valued edge. Let $e_{ij} = 1$ if an edge exists with incident vertices $v_i$ and $v_j$. Let $a_{ij} = 1$ if this edge is in the Hamiltonian chain.

Let $G^+$ be G plus vertices $v_*$ and $v_\#$, such that $e_{*1} = 1$, $e_{\#2} = 1$, where $e_{*j} = 0$, and $e_{\#j} = 0$ for all other $v_j$ in $G^+$. Thus $a_{*1} = 1$, and $a_{\#2} = 1$.

Step 1 Set IC = 0. If there exists no $v_j \neq v_1, v_2, v_*, v_\#$ in $G^+$, then $G^+$ is decomposed. Stop.

Step 2 If for any $v_i$ in G, $\sum_{v_j \in G^+} e_{ij} \leq 1$, then no Hamiltonian chain exists. Stop.

Step 3 If for any $v_i$ in G, $\sum_{v_j \in G^+} e_{ij} = 2$, then if $e_{ij} = 1$ and $a_{ij} \neq 1$, set $a_{ij} = 1$. Record these edges. Set IC = 1.

Step 4 If for any $v_i$ in G, $\sum_{v_j \in G^+} a_{ij} \geq 3$, then no HC exists. Stop.

Step 5 If for any $v_i$ in G, $\sum_{v_j \in G^+} a_{ij} = 2$, then for $a_{ik} = 1$, $a_{ih} = 1$, set $a_{kh} = a_{kh}+1$. Set $e_{kh} = 1$. Delete $v_i$ and all edges incident to $v_i$ from $G^+$. Set IC = 1.

Step 6 If $a_{kh} = 2$, for any $v_h, v_k$ in $G^+$, then no Hamiltonian chain exists. Stop.

Step 7 If $a_{ij} = 1$ where $v_i = v_1$ or $v_*$ and $v_j = v_2$ or $v_\#$, and there exists some $v_k$ in $G^+$ such that $v_k \neq v_1, v_2, v_*, v_\#$, then no Hamiltonian chain exists. Stop.

Step 8 If IC = 1 go to Step 1.

Step 9 Subroutine: build vertex set to form subgraphs S. More details are given on page 60. If visual inspection is used this step is not needed.

Step 10 If there exists some subgraph S in G ($v_*$ and $v_\#$ are not in S) such that $\sum_{v_j \notin S} e_{hj} \geq 1$ for $v_h$ in S, and $e_{ij} = 0$ for all $v_i \neq v_h$ in S, $v_j$ not in S, then no Hamiltonian chain exists. Stop.

Step 11 If there exists some subgraph S in G ($v_*$ and $v_\#$ are not in S) such that

i) $\sum_{v_j \notin S} e_{ij} = 1$ for $v_i = v_h, v_k, v_p$ in S.

ii) $e_{ij} = 0$ for $v_i \neq v_h, v_k, v_p$ in S, $v_j$ not in S.

iii) $a_{ij} = 1$ for $e_{ij} = 1$ for $v_i = v_h, v_k$.

Then set $e_{pj} = 0$ for $e_{pj} = 1$, $v_j$ not in S. If $a_{pj} = 1$ then no Hamiltonian chain exists. Stop.

Step 12 Identify any subgraph S in G ($v_*$ and $v_\#$ are not in S) such that $\sum_{v_j \notin S} e_{ij} \geq 1$ for $v_i = v_h, v_k$ in S, and $e_{ij} = 0$ for all $v_i \neq v_h, v_k$ in S, $v_j$ not in S, where such $v_i$ exists.

Step 13 For each subgraph S in Step 12, set $a_{hk} = a_{hk}+1$. Let $G^P$ = S with endpoints $v_h$ and $v_k$. Delete all $v_i$ in S from $G^+$, $v_i \neq v_h, v_k$.

Step 14 If for any $v_h, v_k$, $a_{hk}>1$, then no Hamiltonian chain exists. Stop.

Step 15 If at least one subgraph S is identified by Step 12, then go to step 1. If visual inspection is used, go to step 1. (Steps 16-18 are not needed.)

Step 16 Combine any two vertex sets $S^1$ and $S^2$ where each has two vertices, adjacent to a vertex in the other set. More detail on this step can be found following this algorithm.

Step 17 When two sets $S^1$ and $S^2$ are found, go to step 10. If none are found continue here.

Step 18 Combine three or more vertex sets S that are connected by a circuit. Go to step 10. If step 18 is reached, a circuit exists connecting vertex sets. More detail on this step can be found on page 61.

Expansion of steps in vertex set construction are as follows:

Details of Step 9 - Sets of vertices are formed from adjacent vertices.

a) Identify any unsearched edge in G, whose incident vertices $v_i$ and $v_j$ are not included in the same vertex set. Initially no vertices are included in a vertex set. Determine all $v_h$ incident to $v_i$ and $v_j$. If no $v_h$ is found, search another edge. If all such edges have been searched, then go to step 9e.

b) For the edge in step 9a, form vertex set S with $v_i$, $v_j$, and all $v_h$ in G, adjacent to both $v_i$ and $v_j$.

c) Identify any $v_k$ not in S, adjacent to at least two $v_m$ in S. Add all such $v_k$ to S.

d) Repeat step 9c until no $v_k$ is found. Then go to step 9a.

e) If any $v_i$ is not in some S, then for each such $v_i$, define a set S with $v_i$ only.

f) Eliminate any vertex set S whose vertices are all contained in another vertex set.

Details of Step 16.

If for any sets $S^1$ and $S^2$ not identified by Step 12,

i) there exists at least two $v_i$ in $S^1$ such that for each $v_i$, $\sum_{v_j \varepsilon S^2} e_{ij} > 0$,

ii) there exists at least two $v_j$ in $S^2$ such that for each $v_j$, $\sum_{v_i \varepsilon S^1} e_{ij} > 0$, or $S^2$ contains only one vertex,

then combine $S^1$ and $S^2$ into one vertex set S.

Details of Step 18

a) Consider all sets S, not identified in Step 12. For all pairs of $S^b$ and $S^c$, define $A^{bc}$, such that $A^{bc}$ exists if $\sum_{v_i \varepsilon S^b} \sum_{v_j \varepsilon S^c} e_{ij} \geq 1$.

b) For some $S^1$, find an $S^2$, and $S^3$ such that $A^{12}$ exists, and $A^{23}$ exists. If $A^{13}$ exists combine $S^1$, $S^2$, and $S^3$ into S. Go to step 10.

c) Otherwise, given the last $S^k$ in the sequence begun in step 18b ($k = 3$ to begin with), identify $S^m$ where $m = k+1$, given $A^{km}$ exists. Find the first $i = 1,2,\ldots,k-1$, such that $A^{mi}$ exists. If such an $A^{mi}$ exists, combine $S^i, S^{i+1}, \ldots, S^k, S^m$ into a single set S. Then go to step 10.

d) If no $A^{mi}$ is found then set $k=m$. Repeat step 18c.

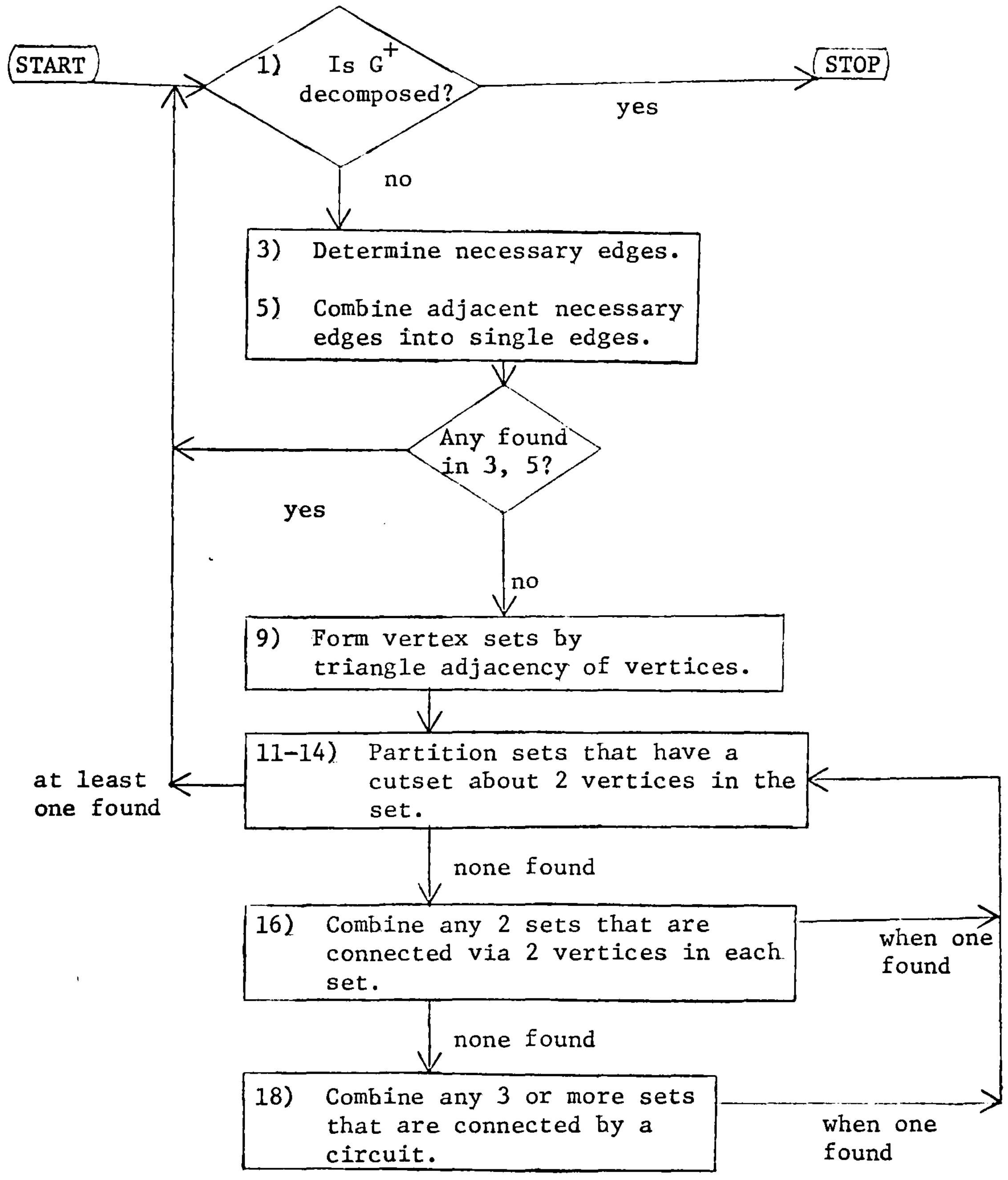

START
1) Is $G^+$ decomposed?
yes
STOP
no
3) Determine necessary edges.
5) Combine adjacent necessary edges into single edges.
Any found in 3, 5?
yes
no
9) Form vertex sets by triangle adjacency of vertices.
11-14) Partition sets that have a cutset about 2 vertices in the set.
at least one found
none found
16) Combine any 2 sets that are connected via 2 vertices in each set.
when one found
none found
18) Combine any 3 or more sets that are connected by a circuit.
when one found

Figure 7.1 Flowchart for Phase One - Decomposition

## Example of Phase One - Decomposition

Refer to Figure 7.2 for initial graph $G^+$.

| | | |
|---|---|---|
| Step 3 | Vertex of degree 2 | Necessary edges |
| | a | (a,c), (a,b) |

Step 5 Vertices with two necessary edges incident: c, a

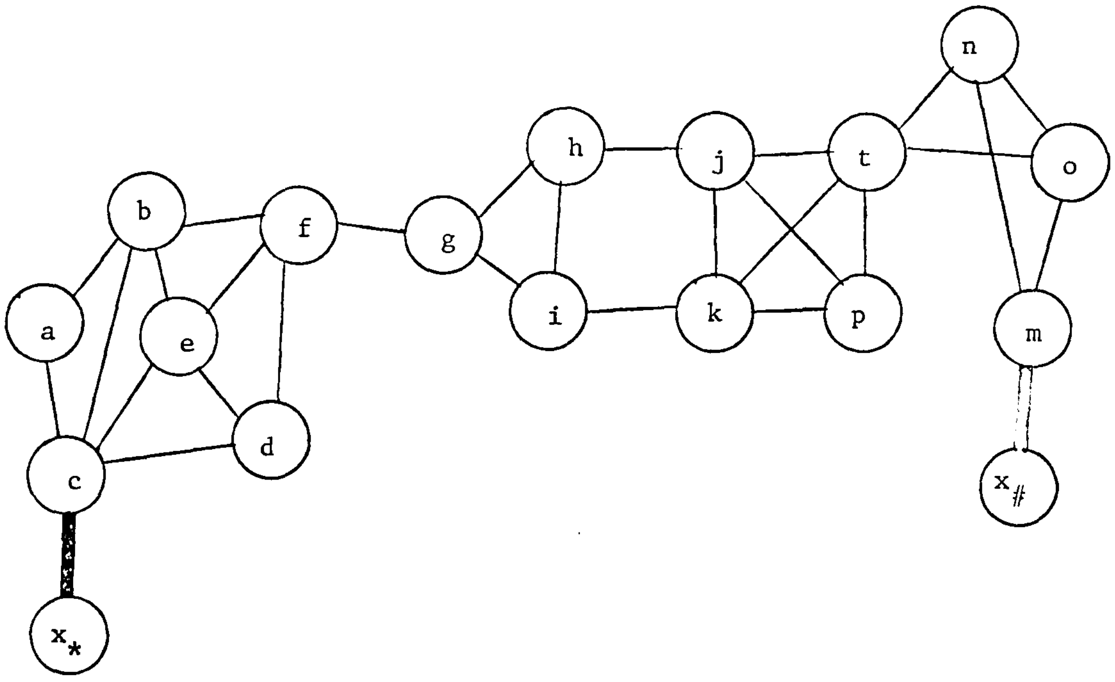
| Combine Edges | Form Edge | Delete Vertex | Delete Edges |
|---|---|---|---|
| $(x_*,c)$, (c,a) | $(x_*,a)$ | c | (c,b), (c,e), (c,d) |
| $(x_*,a)$, (a,b) | $(x_*,b)$ | a | |

| | | |
|---|---|---|
| Step 3 | Vertex of degree 2 | Necessary edges |
| | d | (e,d), (d,f) |

Step 5 Vertex with two necessary edges incident: d

| Combine Edges | Form Edge | Delete Vertex |
|---|---|---|
| (e,d), (d,f) | (e,f) | d |

Note - Steps where no changes occur are omitted.



c,m - endpoints

$x_*$,$x_\#$ - artificial vertices

Figure 7.2. Initial graph $G^+$.

Refer to Figure 7.3 for revised graph $G^+$.

| Step 3 | Vertex of degree 2 | Necessary Edges |
|---|---|---|
| | e | $\underline{(b,e)}$, (e,f) |

Step 5 Vertices with two necessary edge incident: b,e

| Combine Edges | Form Edge | Delete Vertex |
|---|---|---|
| $(x_*,b)$, (b,e) | $(x_*,e)$ | b |
| $(x_*,e)$, (e,f) | $(x_*,f)$ | e |

| Step 3 | Vertex of degree 2 | Necessary Edges |
|---|---|---|
| | f | $(x_*,f)$, $\underline{(f,g)}$ |

Step 5 Vertex with 2 necessary edges incident: f

| Combine Edges | Form Edge | Delete Vertex |
|---|---|---|
| $(x_*,f)$, (f,g) | $(x_*,g)$ | f |

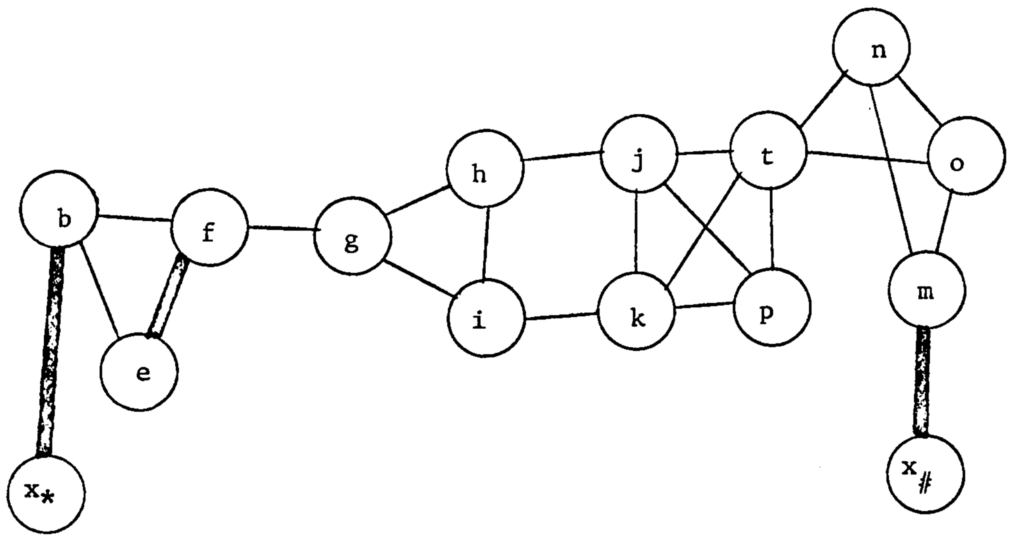


Figure 7.3. Graph $G^+$ revised from Figure 7.2.

Step 9 Refer to Figure 7.4 for revised graph $G^+$.

a) Edge (g,h).

b) Vertices g and h adjacent to i. Form set {g,h,i}.

c) Set {g,h,i}. No vertex adjacent to two in set.

a) Edge (h,j). No common adjacent vertex to h and j.

a) Edge (i,k). No common adjacent vertex to i and k.

a) Edge (j,k).

b) Vertices j and k adjacent to i and p. Form set {j,k,t,p}.

c) Set {j,k,t,p}. No vertex adjacent to two in set.

a) Edge (t,n).

b) Vertices t and n adjacent to o. Form set {t,n,o}.

c) Set {t,n,o}. Vertex m adjacent to n and o. Form set {t,n,o,m}.

c) Set {t,n,o,m}. No vertex adjacent to two in set.

a) All edges searched.

Step 12 Set {t,m,n,o} separated at t and m.

Step 13 Resulting separable subgraph 1: {t,m,n,o} with endpoints t and m. Replace by necessary edge (t,m).

Step 5 Vertex with 2 necessary edges incident: m

| Combine Edges | Form Edge | Delete Vertex |
|---|---|---|
| (t,m), (m,$x_{\#}$) | (t,$x_{\#}$) | m |

Step 16 Set {g,h,i} connected to {j,k,t,p} via h,i and j,k respectively. Combine to form set {g,h,i,j,k,t,p}.

Step 12 Set {g,h,i,j,k,t,p} separated at g and t.

Step 13 Resulting separable subgraph 2: {g,h,i,j,k,t,p} with endpoints g and t. Replace by necessary edge (g,t).

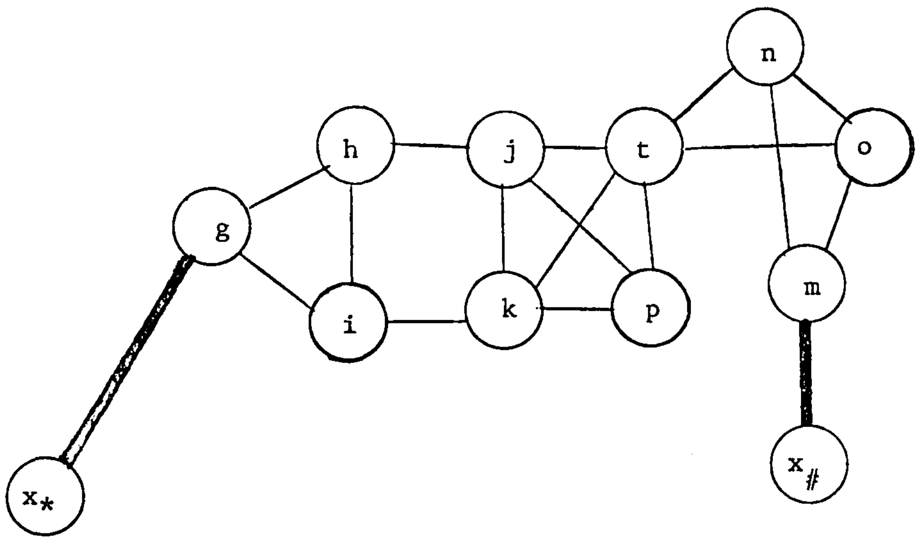


Figure 7.4. Graph $G^+$ revised from Figure 7.3.

See Figure 7.5 for resulting graphs.

Step 5 Vertices with two necessary edges incident: g,t

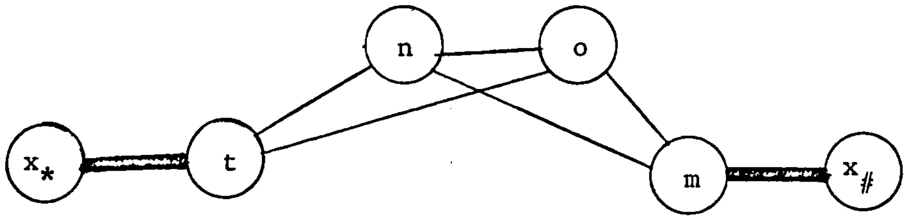
| Combine Edges | Form Edges | Delete Vertex |
|---|---|---|
| $(x_*,g)$, $(g,t)$ | $(x_*,t)$ | g |
| $(x_*,t)$, $(t,x_\#)$ | $(x_*,x_\#)$ | t |

Results

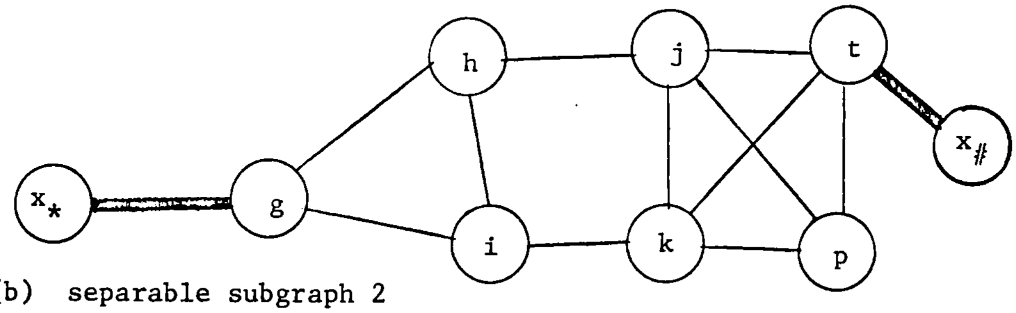
c to a to b to e to d to f to g

Separable subgraph {g,h,i,j,k,t,p} with endpoints g and t.

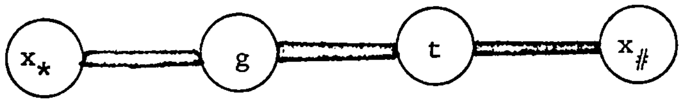
Separable subgraph {t,n,o,m} with endpoints t and m.



(a) separable subgraph 1



(b) separable subgraph 2



(c) original graph

Figure 7.5. Partitioning of graph $G^+$.

Phase Two - Branch and Bound

The modified branch and bound in Chapter 4 given on page 22 is used for each partitioned subgraph $G^P$. Additional considerations are needed where necessary edges are imbedded within partitioned subgraphs. G represents $G^P$ in the algorithm, while the endpoints in $G^P$, $v_h$ and $v_k$ are represented by $v_1$ and $v_2$.

In steps 2 and 12, first set any $s_{ij} = 1$, if $a_{ij} = 1$, then use Kruskal's method to complete the minimal spanning tree.

Apply steps 6 and 7 as is for given vertex $v_i$, if $\sum_j a_{ij} = 0$. Otherwise apply steps 6a or 7a.

Step 6a For each $v_i \neq v_1, v_2$, if $d_i^T > 2$, and $\sum_j a_{ij} = 1$, then define all sets, $S_i$, of m-2 out of m-1 incident MST edges excluding the edge where $a_{ij} = 1$. $(m = d_i^T)$. Find the number of $S_i$ for that $v_i$.

Step 7a For $v_i = v_1, v_2$, if $d_i^T > 1$, and $\sum_j a_{ij} = 1$, then define the set, $S_i$, of the m-1 incident MST edges excluding the edge where $a_{ij} = 1$. $(m = d_i^T)$. Only one $S_i$ exists.

Table 7.1 converts steps 6 and 7 from operating on G to operating on $G^+$.

| Number of necessary edges in $G^+$ | 0 | 1 | 2 |
|---|---|---|---|
| non-endpoints in G | $\binom{m}{m-2}$ Step 6 Page 22 | $\binom{m-1}{m-2}$ Step 6a Page 71 | will not result here when using procedure |
| endpoints in G | doesn't exist | $\binom{m}{m-1}$ no necessary edges in G Step 7 Page 22 | $\binom{m-1}{m-1} = 1$ 1 necessary edge in G Step 7a Page 71 |
| converted in $G^+$ (as in computer program) | $\binom{m}{m-2}$ | $\binom{m-1}{m-2}$ | $\binom{m-2}{m-2} = 1$ |

Table 7.1 Numbers of branching combinations to a vertex in Phase Two, converted from G to $G^+$

Example of Phase Two

Refer to figure 7.6 for graph $G^P$.

Edge (b,c) is a necessary edge. Distance is recorded in Phase One.

Step 1 Let $G^M = G = (G^P)$ with endpoints a and d.

Step 2 Find the minimal spanning tree subject to inclusion of the necessary edge given in Figure 7.6.

Include necessary edge (b,c).

| Edge | Distance | Included in MST |
|---|---|---|
| (c,d) | 3 | yes |
| (d,e) | 3 | yes |
| (c,e) | 4 | no |
| (b,d) | 5 | no |
| (a,c) | 6 | yes |

Step 4 $\varepsilon^T = 2$. Vertex c has $d^T = 3$. Endpoint vertex d has $d^T = 2$.

Step 6a For vertex c, m = 3. Edge (b,c) is a necessary edge.

Therefore $\binom{m-1}{m-2} = \binom{2}{1} = 2$ sets of $S_c$.

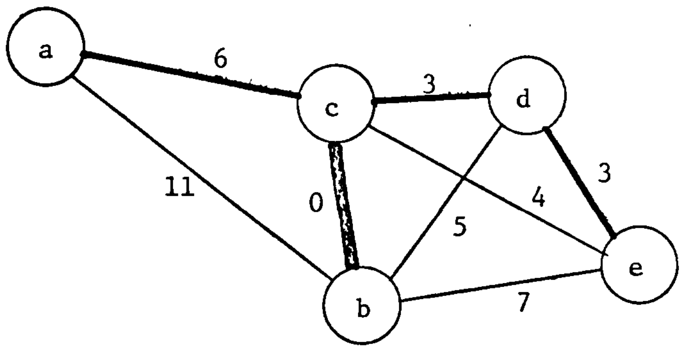
Step 7 For vertex d, m = 2. Therefore $\binom{m}{m-1} = \binom{2}{1} = 2$ sets of $S_d$.

Step 8 Vertex c and d both have two $S_i$.

Arbitrarily choose c as having least number.

Step 9 $S_c$ for $v_c$: $\{(a,c)\},\{(c,d)\}$.

The remainder of this example is not shown.



endpoints - a,d

necessary edge: (b,c)

Figure 7.6. Graph $G^P$, with minimal spanning tree.

## Summary of Proposed Procedure

In Phase One necessary edges are found. Then separable subgraphs are partitioned. The process reverts back to finding necessary edges until the entire graph is decomposed.

For each partitioned subgraph in Phase One, Phase Two finds its minimal Hamiltonian chain. A branch and bound procedure using a method of Christofides involving minimal spanning trees is used.

The minimal Hamiltonian chains of each partitioned graph are combined with the necessary edges found to form the minimal Hamiltonian chain of the entire graph.

# Chapter 8

# CONCLUSIONS AND RECOMMENDATIONS

The proposed procedure is designed for graphs that have few incident edges per vertex. A computerized version has been developed for Phase One (the decomposition process) and Phase Two (the branch and bound algorithm). Phase One can instead be implemented by a visual procedure. The overall procedure is efficient for finding the minimal Hamiltonian chain of large sparse graphs. The graph is partitioned into smaller graphs which results in fewer branching operations. The procedure could possibly be improved by revising or replacing the current vertex set construction routine. One way may be to search each vertex to find separation vertices.

Various necessary conditions for existence have been developed in this work. New sufficiency conditions for existence as well as conditions regarding optimal solutions could result from extending this research. Other procedures for finding the minimal Hamiltonian chain can be developed. The possible types of algorithms suggested for future research include:

1) Integer linear programming
2) Penalty assignments on minimal spanning tree edges
3) Adding edges onto the longest chain in the minimal spanning tree
4) Assignment into solution of given minimal spanning tree edges; then decomposition.

## Use of the Proposed Procedure

The proposed procedure is designed for graphs with few incident edges per vertex. Thus, it is likely that large sparse graphs will be efficiently decomposed by Phase One. In a graph with n vertices the procedure should work efficiently when there are at most k x n edges, where k is between 2 and 3. For graphs with more incident edges per vertéx, little or no decomposition is likely to occur. Unless there is some other evidence that decomposition will occur, Phase One should not be directly applied. If there are few vertices in the graph, 20 or less, then the modified branch and bound algorithm should be applied directly if the graph is dense in edges.

If the graph is large, over 30 vertices or over 100 edges, then the graph could be revised by eliminating various edges before applying the procedure. Edges with a distance value above some θ, would be eliminated. The value θ would be chosen in such a manner that at least two edges remain incident to each vertex, except the endpoints. The optimal solution and possibly all feasible solutions may be eliminated in the process. If all feasible solutions are eliminated then the value of θ could be raised, or the endpoints changed. Table 7.1 summarizes guidelines for using the procedure.

Phase One can be used in a procedure to find all Hamiltonian chains of a graph. All Hamiltonian chains of each separable subgraph would be generated after applying the decomposition process.

| If | Use |
|---|---|
| $\frac{\text{number of edges}}{n} < k$<br><br>k is some value to be determined between 2 and 3 | Both phases of proposed procedure |
| Otherwise, when<br><br>$n \leq 20$ | Modified branch and bound only |
| Otherwise, when<br><br>$n > 30$<br>or<br>number of edges $> 100$ | Eliminate edges before using proposed procedure |
| Otherwise | Has not been determined. Use ones judgement for a given situation. |

Table 8.1 Guidelines for finding the minimal Hamiltonian chain in a graph of n vertices

For the graph theory analyst the decomposition process can be more powerful as a visual process. By drawing the graph all edges and vertices can be examined. In using a visual procedure for Phase One, all separable subgraphs may be identified without a vertex set construction routine. The separable subgraphs can be checked for further decomposition.

## Efficiency of the Proposed Procedure

The procedure described in this research is considerably more efficient than an exhaustive enumeration procedure. When the given is quite sparse, the proposed procedure is more efficient than a direct branch and bound.

Compared to the modified branch and bound using Christofides' method, much smaller branch and bound operations will result. There will be less edges to remove and replace as well as less possible combinations of edges to remove and replace before a feasible solution can occur. Suppose there are j partitioned separable subgraphs in a graph where each one has $k_i$ possible branching combinations involving $p_i$ edge removals and replacement each, before a Hamiltonian chain may be found. As each separable subgraph is solved separately there are $k_1+k_2+...+k_j$ total possible branching combinations to eliminate the original oversaturated edges, where each involves $p_i$ edge removals and replacements. If the graph were not partitioned then there would be $k_1 x k_2 x ... x k_j$ total possible branching combinations to eliminate the original oversaturated edged, where each involves $p_1+p_2+...+p_j$ edge removals and replacements.

Of the possible branching combinations that can occur before a Hamiltonian chain is found, the actual number is minimized. For any branching operation the oversaturated vertex with the smallest number of possible combinations of edges removed is branched upon. This maximizes the number of possible subcases that each new partial graph can include. Whenever a partial graph is found to be infeasible or suboptimal, it is eliminated from further operations.

Infeasible graphs are determined when a vertex of degree one results, when a spanning tree cannot be found, or when necessary edges cannot be combined. Most infeasible edges are automatically accounted for when one of their incident vertices is deleted from the graph. All necessary edges are determined when a vertex of degree two results.

Many necessary edges will result when other edges are eliminated from the graph or separable subgraphs are partitioned. Suppose a vertex is an endpoint of a separable subgraph and has one other edge incident, not contained in the subgraph. When the separable subgraph is partitioned, it is replaced by a single necessary edge. The vertex now has degree two, therefore the other incident edge becomes a necessary edge. If an edge forms a single edge cutset, then it will be identified as a necessary edge in this manner. This will occur when one of the two separable subgraphs it connects is partitioned.

## Improvements in the Proposed Procedure

The proposed procedure includes a routine to identify sets of vertices that are then checked to see if they form separable subgraphs. In certain cases more than one separable subgraph is identified as one larger separable subgraph. The set routine could possibly be modified to identify these separable subgraphs as separate sets.

An alternative routine for identifying potential separable subgraphs could be developed. One such method would search all vertices to determine if they may be separation vertices of a separable subgraph. The pairs of potential separation vertices would be checked to see if removal would result in a disconnected graph. If this occurs, then a separable subgraph can be partitioned from the rest of the graph.

Additional conditions to identify necessary and infeasible edges could be employed in Phase One. This would result in further decomposition. Furthermore whenever separable subgraphs are partitioned they can be checked for further decomposition. An advantage to employing additional decomposition is to reduce the risk of not identifying an infeasible solution, in Phase One. A disadvantage may be increased computational time.

In Phase Two it is possible to employ the conditions used in Phase One to identify necessary and infeasible edges. As edges are eliminated to form new partial graphs in the branching process, vertices of degree two may result. Then additional necessary and infeasible edges may be found. If the partial graph is found to be infeasible, it would be eliminated. The partial graphs may be checked for decomposition as soon as they are branched to. Alternatively they can be checked for further decomposition only before they are further branched. In the latter approach suboptimal partial graphs can be eliminated without checking for decomposition. However more partial graphs will be stored as many infeasible partial graphs will not be eliminated beforehand.

The computerized procedure can be improved for programming efficiency in Fortran. Also it may be programmed more efficiently into another computer language that is designed for matrices.

## Future Research on Hamiltonian Chain Conditions

A great advancement would occur in theory concerning Hamiltonian chains if stronger sufficient conditions for existence were developed. The sufficient conditions that have been developed are limited in use. Many graphs with few incident edges per vertex will not satisfy these sufficient conditions, but will have Hamiltonian chains. It should be possible to develop sufficient conditions that incorporate various necessary conditions and require most vertices too have a degree of at least three.

With respect to minimal Hamiltonian chains, the use of the triangle inequality property should be investigated in further research. Given triangle inequality holds, it can easily be shown that pairs of edges that cross cannot be in the minimal Hamiltonian chain. Additional conditions based on this concept could be developed to determine edges that cannot be in the minimal Hamiltonian chain. Such conditions could use triangle inequality of imbedded complete subgraphs, along with planarity, and perhaps some existence conditions.

Assuming triangle inequality holds, the two edges that cross in a complete graph of four vertices cannot be in the minimal Hamiltonian chain. This concept relating to edges crossing should be applicable to situations where two edges cross but are not included in a complete subgraph of four vertices. The concept of planarity can be used to determine possible arrangements of visiting vertices between endpoints.

Such information can be determined since a planar graph has no edges that cross. The concepts of triangle inequality and planarity should be applicable to theory that involves planar subsets of edges within a graph.

More necessary conditions for the existence of a Hamiltonian chain could be developed in furthering this research. Two examples are given here of situations where such conditions can be devised.

One example involves a graph of exactly two subgraphs that are connected by a cutset of two edges. If one endpoint of a Hamiltonian chain is in each subgraph then only one of the two cutset edges can be in a Hamiltonian chain. If one of these edges is a necessary edge, the other edge is infeasible. See Figure 8.1 for an example.

The other example involves a graph that includes two subgraphs R and W, where $v_1$ is the only vertex not contained in either subgraph. Furthermore, $v_2$ is contained in R. Also there exists exactly one edge connecting R and W with incident vertices $v_h$ in R and $v_k$ in W.

In this situation the edge connecting $v_h$ and $v_k$ must be included in any Hamiltonian chain. Also $v_1$ can only be adjacent in a Hamiltonian chain to some $v_j$ in W, other than $v_k$, assuming W contains at least two vertices. Figure 8.2 gives an example.

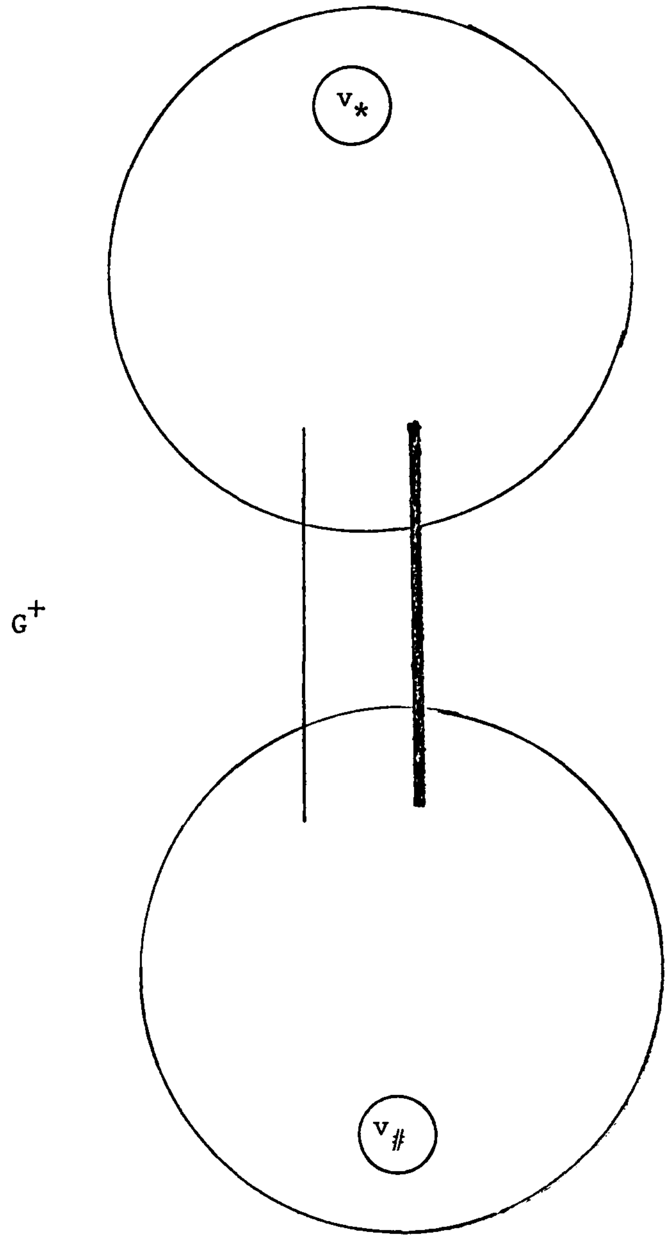


Figure 8.1. Graph $G^+$ with a cutset of two edges.

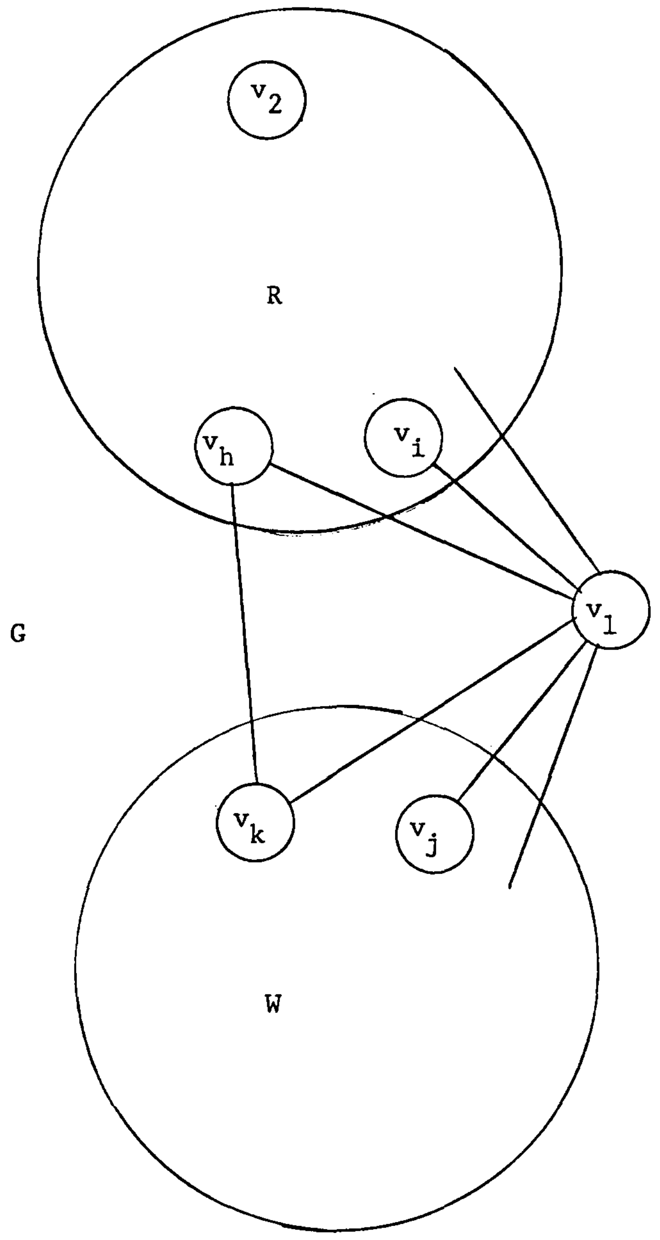


Figure 8.2. Graph G with two subgraphs connected by one edge.

## Future Research on Hamiltonian Chain Algorithms

The proposed procedure makes use of necessary conditions for a feasible Hamiltonian chain in searching for the optimal solution. It searches all possible alternatives that are not found to be infeasible or suboptimal. Therefore, it uses an indirect optimal searching algorithm, which is the case for any branch and bound. It indirectly uses a sufficient condition for a minimal Hamiltonian chain. The condition states that if a minimal spanning tree is a Hamiltonian chain it is the minimal Hamiltonian chain. If a direct optimal searching procedure is developed, it must directly apply some sufficient condition for an optimal solution.

It may be possible in the future to develop more direct optimal searching procedures to find the minimal Hamiltonian chain. Integer linear programming is applicable but not practical in its general form. Perhaps a special integer linear programming form could be developed which treats the no circuit constraints implicitly.

Two other types of algorithms have been considered in this research but have not worked. Future research could result in an efficient hueristic, a direct optimal searching procedure, or a branch and bound for either or both approaches. One method involves assignment of penalties to minimal spanning tree edges incident to oversaturated vertices. This type of algorithm, which iteratively finds a revised minimal spanning tree, has been shown by Christofides [16] to work for a complete graph.

A second method makes use of chains in the minimal spanning tree and successively adds in additional edges until a Hamiltonian is formed. A simple hueristic by this approach would be as follows where endpoints are not fixed:

1) Select the chain in the minimal spanning tree with the most edges.
2) Successively replace an edge in the chain by a chain of length two that covers a vertex not previously covered.
3) Stop when a Hamiltonian chain is formed.

An hueristic algorithm that assigns given minimal spanning tree edges can be employed. Edges in the minimal spanning tree not incident to an oversaturated vertex are assigned as solution edges in a Hamiltonian chain. By applying the necessary conditions for edge feasibility the graph is further decomposed. The minimal spanning tree is then recalculated. The overall process is repeated until a Hamiltonian chain results.

## Contributions of this Work

The theory developed in this work identifies numerous necessary conditions for a Hamiltonian chain to exist. The conditions are applicable in decomposing a graph with respect to finding Hamiltonian chains.

The proposed procedure is designed to find the optimal solution of a graph with many vertices that is sparse in edges. By partitioning the graph a branch and bound algorithm can be applied that is not unmanageable. The branch and bound algorithm that has been used is a modification of a basic algorithm developed by Christofides.

The decomposition process takes advantage of the structure of sparse graphs. As a result it finds edges necessary for a Hamiltonian chain and partitions a large problem into several subproblems. It uses the necessary conditions for existence that are developed in this work.

# APPENDIX A

## Notation Summary

| | |
|---|---|
| $a_{ij}$ | = 1 if the edge incident to $v_i$ and $v_j$ is in a Hamiltonian chain of graph G |
| $a_{ij}^R$ | = 1 if the edge incident to $v_i$ and $v_j$ is in a Hamiltonian chain of graph $G^R$ |
| $A^{bc}$ | Denotes that subgraphs $S^b$ and $S^c$ are connected by some edge or vertex |
| $c_{ij}$ | Costs of distance of the edge incident to $v_i$ and $v_j$ |
| $C^R$ | Total cost or distance of the minimal spanning tree of $G^R$ |
| $C^*$ | Total cost or distance of the least valued minimal spanning tree of all $G^R$ that form a Hamiltonian chain |
| $d_i^T$ | Number of incident spanning tree edges to $v_i$ |
| $e_{ij}$ | = 1 if the edge incident to $v_i$ and $v_j$ exists in graph G |
| $e_{ij}^R$ | = 1 if the edge incident to $v_i$ and $v_j$ exists in graph $G^R$ |
| $\varepsilon^T$ | Degree of difference of a spanning tree to a Hamiltonian chain |
| G | A graph |
| $G^+$ | Graph G with two additional vertices adjacent to the endpoints of G only |
| $G^M$ | A graph that is being branched upon |
| $G^P$ | A partitioned graph of G |
| $G^R$ | Graph G with given edges removed |

| | |
|---|---|
| $\Box^{+}$ | Vertex set or subgraph with two additional vertices adjacent to the Hamiltonian chain endpoints only |
| HC | Hamiltonian chain |
| IC | = 1 indicates that the graph was revised during the necessary edge subroutine of Phase One |
| MHC | Minimal Hamiltonian chain |
| MST | Minimal spanning tree |
| n | Number of vertices in a graph |
| $\theta$ | An upper bound on the distance value of edges considered in a graph |
| R | A set of graphs $G^R$ in which each $G^R$ is Graph G with given edges removed |
| $s_{ij}$ | = 1 if the edge incident to $v_i$ and $v_j$ is in a spanning tree of G |
| $s_{ij}^{R}$ | = 1 if the edge incident to $v_i$ and $v_j$ is in a spanning tree of $G^R$ |
| $S_i$ | A set of edges incident to $v_i$, that are in the minimal spanning tree |
| $S^i$, S | A set of vertices or a subgraph of G |
| T | The subgraph in $G^+$ that includes all vertices not in some subgraph S |
| $v_i$ | A vertex |
| $v_1, v_2$ | Endpoint vertices of a Hamiltonian chain in G |
| $v_*, v_\#$ | Endpoint vertices of a Hamiltonian chain in $G^+$ |

# APPENDIX B

## Definitions

| | |
|---|---|
| artificial edge | - an edge in graph $G^+$ but not in G which is extended from an end vertex in G |
| artificial vertex | - a vertex in graph $G^+$ but not in G which is adjacent to an end vertex in G only |
| chain | - an open sequence of distinct edges in which no vertex is covered more than once. |
| circuit | - a close sequence of distanct edges in which no vertex is covered more than once |
| cutset | - a set of edges in a graph that if eliminated the graph would be disconnected |
| degree (of a vertex) | - the number of edges incident to a vertex in a graph |
| dense graph | - a graph which includes many incident edges per vertex |
| endpoints or end vertices | - vertices with one incident edge in a Hamiltonian chain of a graph |
| Hamiltonian chain | - a chain which covers every vertex of a graph |
| infeasible edges | - edges which cannot be in any Hamiltonian chain of a graph |
| loop | - an edge which is twice incident to the same vertex |

| | |
|---|---|
| minimal spanning tree | - the spanning tree of a graph with the least total distance of its edges |
| necessary edges | - edges that are in all Hamiltonian chains of a graph |
| oversaturated edge | - an edge incident to an oversaturated vertex |
| oversaturated vertex | - a vertex with three or more spanning tree edges edges incident, two or more if an endpoint |
| parallel edges | - edges that are incident to both the same pairs of vertices |
| partial graph | - a graph which is formed by excluding edges from another graph |
| planar graph | - a graph with no edges that cross |
| proper cutset | - a cutset that includes only those edges necessary to partition the graph into two connected subgraphs |
| separable subgraph | - a subgraph in graph G that can be partitioned in $G^+$ by a cutset about two vertices in the subgraph |
| separation vertices | - vertices in a separable subgraph of G which are adjacent to vertices in $G^+$ not in the subgraph |
| spanning tree | - a set of edges in a graph that connects all vertices of the graph and has no circuits |
| sparse graph | - a graph which includes few incident edges per vertex |

| | |
|---|---|
| subgraph | - a set of vertices in a graph with all the edges in the graph having both incident vertices in the set |
| subtour | - a chain that does not cover all vertices in a graph but can be included in a Hamiltonian chain or circuit |
| tour | - a chain or circuit of edges that forms a vertex covering of a graph |
| 1-tree | - a spanning tree with the addition of one edge that results in a circuit |

## APPENDIX C

### References

[1] Chachra, V., Ghare, P., and Moore, J., Applications of Graph Theory Algorithms, Elsevier-North Holland Publishing Co., in press.

[2] Boffey, T., "A Note on Minimal Length Hamiltonian Paths and Circuit Algorithms," O. R. Quarterly, Vol. 24, No. 3, 1973, p. 437.

[3] Bellmore, M., and Nemhauser, G., "The Traveling Salesman Problem--A Survey," Operations Research, Vol. 16, 1968, p. 538.

[4] Turner, W., "Transportation Routing Problem--A Survey," AIIE Transactions, Vol. 6, No. 4, 1974, p. 288.

[5] Lin, S., "An Effective Heuristic Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 21, No. 2, 1973, p. 498.

[6] Turner, W., "Vehicle Routing--Classifications, Characteristics, and Solution Procedures," unpublished paper.

[7] Berge, C., Graphs and Hypergraphs, American Elsevier Publishing Company, Inc., N.Y., 1973.

[8] Held, M., and Karp, R., "The Traveling-Salesman Problem and Minimum Spanning Trees," Operations Research, Vol. 19, No. 6, 1970, p. 1138.

[9] Held, M., and Karp, R., "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II," Math. Programming, Vol. 1, 1971, p. 6.

[10] Hansen, K. H., and Krarup, J., "Improvements of the Held-Karp Algorithm for the Symmetric Traveling-Salesman Problem," Math. Programming, Vol. 7, 1974, p. 87.

[11] Karp R., "A Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane," Mathematics of Operations Research, Vol. 2, No. 3, August 1977, p. 209.

[12] Christofides, N., "The Shortest Hamiltonian Chain of a Graph," SIAM J. Appl. Math., Vol. 19, No. 4, December, 1970, p. 689.

[13] Taociniz, A., "Hamiltonian Paths to Circuits in Strongly Connected Graphs," Trabajados de Estadacia, Vol 17, No. 1, 1966, p. 17.

[14] Kaufmann, A., Graphs, Dynamic Programming, and Finite Games, Academic Press, N.Y., 1967.

[15] Goodman, S., and Hedetniemi, S., "Sufficient Conditions for a Graph to be Hamiltonian," Journal of Combinatorial Theory, (B), 16, No. 2, 1974, p. 175.

[16] Christofides, N., Graph Theory--An Algorithmic Approach, Academic Press, N.Y., 1975.

[17] Busacker, R., and Saaty, T., Finite Graphs and Networks, McGraw-Hill, N.Y., 1965.

## APPENDIX D

Kruskal Algorithm for Minimal Spanning Tree [16]

Step 1 Start with a completely disconnected graph T of n vertices.

Step 2 Order the edges of G in ascending order of distance.

Step 3 Starting from the top of the list add edges into T providing this addition does not close a circuit in T.

Step 4 Repeat Step 3 until (n-1) links have been added. T is then the MST of graph G.

Note -- Algorithm is taken from [16], p. 137.

# APPENDIX E

## Analysis of Vertex Set Construction

A separable subgraph of G is determined when a set of vertices is such that only two vertices in the set are adjacent to a vertex in $G^+$, not in the set. However the given sets must be constructed before they are tested to see if they form separable subgraphs. The sequence of operations for constructing vertex sets is analyzed in this section.

A vertex set is formed when two adjacent vertices each are adjacent to the same third vertex. The three vertices become a set. Any vertex adjacent to two vertices in the set is added into the set. Once no additional vertices can be added into the set, the set is checked to see if it forms a separable subgraph. Any separable subgraph formed is removed from the graph and replaced by a single edge.

The only problem with this method of identifying vertex sets is that certain pairs of separable subgraphs will be identified as a single separable subgraph. This occurs if both separation vertices of one separable subgraph have the following properties:

a) They are both included in the same other separable subgraph.

b) There exists a chain of two edges connecting them, within the other separable subgraph.

This can occur if one separable subgraph is imbedded in another. This can also occur if both separable subgraphs have the same separation vertices, which is not possible if a Hamiltonian chain exists.

If a vertex set does not form a separable subgraph then it must be included in a larger separable subgraph. The entire graph G can be considered a separable subgraph with the endpoints of the graph being the separation vertices.

If two sets each have two vertices incident to a vertex in the other set, then it is possible for a chain within a Hamiltonian chain to pass from one set to the other and back again. Therefore the two sets are combined into a single set, which may or may not be a separable subgraph. Any vertex that results in being adjacent to two vertices in some set is added into that set.

Any resulting vertex set that does not form a separable subgraph and cannot be combined with another set, must have at least three vertices adjacent to a vertex not in the set, if a Hamiltonian chain exists. No two vertices in one set can be adjacent to a vertex in the same other set. It is possible for a single vertex set to contain both endpoints in G, and therefore have a vertex adjacent to a vertex in one other vertex set. Therefore at least two vertex sets will remain if one remains. Hence, at least two vertex sets remain. As each vertex set must have at least three edges leading out of the set, then if there are k such sets, there are at least 3k incidences leading out of the sets. Two of these incidences are edges connecting to the artificial vertices in $G^+$.

The remaining 3k-2 must be incidences between the vertex sets. Therefore, as each edge from one set to another is counted from both sets, at least (3k/2)-1 edges must exist connecting the vertex sets. This includes one edge if the two vertex sets share a vertex. A circuit occurs if there are at least k edges connecting k sets. Now $[(3k/2)-1] \geq k$, for $k \geq 2$. Thus, there are at least k edges connecting the vertex sets. Therefore, a circuit must exist connecting some of the vertex sets. Once a circuit is found between some vertex sets, these vertex sets are combined into a single vertex set. This resulting vertex set may or may not form a separable subgraph. As the graph is finite, eventually all remaining vertex sets will be combined and/or extracted.

The ordering of the vertices is important when vertex sets must be combined by circuits. It is possible for a circuit to be found that connects a larger set, when a smaller set within can be combined by a circuit and result in a separable subgraph. Vertices should be numbered in an orderly sequence with respect to their adjacent vertices and the endpoints of the graph, when using the computer program.

## APPENDIX F

## Computer Program of the Proposed Procedure

```
C
C         THIS PROCEDURE DECOMPOSES A GRAPH FOR THE PURPOSE OF FINDING
C    HAMILTONIAN CHAINS.  EDGES THAT ARE NECESSARY FOR ANY HAMILTONIAN
C    CHAIN ARE DETERMINED.  EDGES WHICH CAN NOT BE IN ANY HAMILTONIAN
C    CHAIN ARE ELIMINATED.  SUBGRAPHS WHICH CAN BE SOLVED INDEPENENTLY
C    OF THE REMAINDER OF THE GRAPH WITH RESPECT TO THE EXISTENCE OF A
C    HAMILTONIAN CHAIN ARE PARTITIONED AS SEPARATE GRAPHS.
C
C      DATA CARDS (UNFORMATTED)
C     FIRST CARD    N -- NUMBER OF VERTICES
C         NOTE-- ALL VERTICES ARE ASSIGNED A NUMBER FROM 1 TO N.
C     SECOND CARD    IE1 -- END VERTEX 1,  IE2 -- END VERTEX 2
C     THEN A CARD FOR EACH EDGE
C    FOR EACH EDGE:  INCIDENT VERTEX I, INCIDENT VERTEX J, DISTANCE OR
C      COST OF EDGE
C
C       INPUT EACH EDGE ONCE
C     SUGGESTED ARRANGEMENT:  LET VERTEX I BE THE VERTEX WITH THE
C      SMALLER NUMBER IDENTIFYING IT, VERTEX J  THE LARGER.
C    ARRANGE DATA BY ASCENDING NUMERICAL ORDER OF VERTEX I, THEN BY
C     ASCENDING NUMERICAL ORDER OF VERTEX J.
C        NUMBER VERTICES IN AN ORDERLY PATTERN ACCORDING TO THEIR
C     LOCATION  AND/ OR ADJACENT VERTICES.  THIS WILL RESULT IN MORE
C     COMPUTATIONAL EFFICIENCY.
C
C       VARIABLES
C   ADJ -- ADJACENCY MATRIX
C   SPC -- NECESSARY EDGES UPDATED
C   MHC -- ACTUAL NECESSARY SOLUTION EDGES
C        =1  INDICATES INFORMATION IN ABOVE MATRICES
C   DIST -- DISTANCE MATRIX
C   IR  =0 IF VERTEX REMAINS IN GRAPH;  =1 IF VERTEX IS DELETED.
```

```
C  N -- NUMBER OF VERTICES
C  IE1,IE2 -- ENDPOINT VERTICES
```

```
C
      INTEGER ADJ,SPC,DIST
      INTEGER ADJS,SPCS
      DIMENSION ADJ(30,30),SPC(30,30),MHC(30,30),DIST(30,30),IR(30)
      DIMENSION ADJS(5,30,30),SPCS(5,30,30)
      COMMON ITERM,ADJ,SPC,MHC,N,N2,IR,DIST,IE1,IE2
      COMMON IORG,KJ
      COMMON ADJS,SPCS
      READ,N
      N2=N+2
      KJ=0
      IORG=0
      IF=0
C    SET INITIAL MATRICES TO ALL ZEROES FOR ORIGINAL GRAPH.
      DO 3 I=1,N2
      DO 4 J=1,N2
      ADJ(I,J)=0
      SPC(I,J)=0
      MHC(I,J)=0
      DIST(I,J)=1.0E10
    4 CONTINUE
    3 CONTINUE
      DO 41 I=1,N
      IR(I)=0
   41 CONTINUE
      READ,IE1,IE2
C   ADD ARTIFICIAL EDGE.
      ADJ(N+1,IE1)=1
      ADJ(IE1,N+1)=1
      ADJ(N+2,IE2)=1
      ADJ(IE2,N+2)=1
      SPC(N+1,IE1)=1
      SPC(IE1,N+1)=1
      SPC(N+2,IE2)=1
      SPC(IE2,N+2)=1
C  READ IN EDGES.
    1 READ,I,J,DIJ
      IF(I.EQ.0)GO TO 2
      ADJ(I,J)=1
      DIST(I,J)=DIJ
      ADJ(J,I)=ADJ(I,J)
      DIST(J,I)=DIST(I,J)
      GO TO 1
    2 CONTINUE
      IF(IF.EQ.1) GO TO 531
C  PRINT MATRICES OF ORIGINAL GRAPH.
      CALL WRITEA
      IF=1
  531 CONTINUE
C  DECOMPOSE BY FEASIBILITY OF EDGES.
      CALL EDGFES
C  PRINT REVISED MATRICES.
      GO TO 532
      CALL WRITEA
  532 CONTINUE
      IF(IORG.EQ.1) GO TO 8
      IF(ITERM.EQ.1) GO TO 6
C  DECOMPOSE BY PARTITIONING SEPARABLE SUBGRAPHS.
      CALL SETSUB
      IF(ITERM.EQ.1) GO TO 6
      GO TO 2
    6 WRITE(6,89)
   89 FORMAT(/1X,'INFEASIBLE')
      GO TO 7
      8 CONTINUE
        DISTF=0
        K=0
C     PRINTOUT FINAL RESULTS
        WRITE(6,86)
        WRITE(6,87)
     86 FORMAT(///2X,'SOLUTION EDGES',/)
     87 FORMAT(2X,'VI-VJ-DIST',/)
        DO 9 I=1,N
        DO 10 J=1,N
        IF(I.GE.J) GO TO 10
        IF(MHC(I,J).NE.1) GO TO 10
        K=K+1
        KD=DIST(I,J)
        PRINT,I,J,KD
     10 CONTINUE
      9 CONTINUE
        IF(KJ.EQ.0) GO TO 7
        DO 20 KS=1,KJ
        NE=0
        WRITE(6,83)KS
        WRITE(6,84)
     83 FORMAT(///5X,'SEPARABLE SUBGRAPH',2X,I2,/)
     84 FORMAT(2X,'VI-VJ-1-DIST-SPC',/)
        DO 11 I=1,N
        DO 12 J=1,N
        IF(I.GE.J) GO TO 12
        IF(ADJS(KS,I,J).NE.1) GO TO 12
        NE=NE+1
        IAJ=1
        SP=SPCS(KS,I,J)
        IF(SP.EQ.0) KD=DIST(I,J)
```

```
   IF(SP.EQ.1) KD=0
   PRINT,I,J,IAJ,KD,SP
12 CONTINUE
11 CONTINUE
   PRINT,NE
   WRITE(6,85)
85 FORMAT(/2X,'ENDPOINTS',/)
   DO 13 I=1,N
   IF(ADJS(KS,I,N+1).EQ.1) IES1=I
   IF(ADJS(KS,I,N+2).EQ.1) IES2=I
13 CONTINUE
   PRINT,IES1,IES2
20 CONTINUE
 7 STOP
   END
```

```
      SUBROUTINE WRITEA
      INTEGER ADJ,SPC,DIST
      DIMENSION ADJ(30,30),SPC(30,30),MHC(30,30),DIST(30,30),IR(30)
      COMMON ITERM,ADJ,SPC,MHC,N,N2,IR,DIST,IE1,IE2
      COMMON IORG,KJ
      WRITE(6,711)
C  PRINTOUT ORIGINAL GRAPH.
  711 FORMAT(///10X,'ADJACENCY MATRIX -- ORIGINAL GRAPH',//)
      DO 1 I=1,N2
      WRITE(6,90)(ADJ(I,J),J=1,N2)
   90 FORMAT(1X,30I2)
    1 CONTINUE
      WRITE(6,91)
   91 FORMAT(///2X)
      DO 3 I=1,N2
      WRITE(6,92)(DIST(I,J),J=1,N2)
   92 FORMAT(30(1X,I3))
    3 CONTINUE
      WRITE(6,91)
      GO TO 11
      DO 5 I=1,N2
      WRITE(6,90)(SPC(I,J),J=1,N2)
    5 CONTINUE
      WRITE(6,91)
      DO 7 I=1,N2
      WRITE(6,90)(MHC(I,J),J=1,N2)
    7 CONTINUE
   11 CONTINUE
      RETURN
      END
      SUBROUTINE EDGFES
      INTEGER ADJ,SPC,DIST
      DIMENSION ADJ(30,30),SPC(30,30),MHC(30,30),DIST(30,30),IR(30)
      DIMENSION II(3)
      COMMON ITERM,ADJ,SPC,MHC,N,N2,IR,DIST,IE1,IE2
      COMMON IORG,KJ
   19 CONTINUE
C     CHECK TO SEE IF GRAPH IS COMPLETELY DECOMPOSED.
      DO 41 I=1,N
      IF((I.EQ.IE1).OR.(I.EQ.IE2)) GO TO 41
      IF(IR(I).EQ.0) GO TO 18
   41 CONTINUE
      IORG=1
      GO TO 29
   18 CONTINUE
      ITERM=0
      ICHAN=0
      DO 11 I=1,N
      IF(IR(I).EQ.1) GO TO 11
      IADJ=0
C    FIND VERTICES OF DEGREE TWO.
      DO 12 J=1,N2
      IADJ=IADJ+ADJ(I,J)
   12 CONTINUE
      IF(IADJ-2)14,13,11
   13 ICHAN=1
      DO 15 J=1,N2
      IF(ADJ(I,J).NE.1) GO TO 15
      IF(SPC(I,J).EQ.1) GO TO 15
C    ASSIGN NECESSARY EDGES.
      SPC(I,J)=1
      SPC(J,I)=1
      MHC(I,J)=1
      MHC(J,I)=1
   15 CONTINUE
      WRITE(6,95)I
   95 FORMAT(//2X,I3)
      GO TO 43
      CALL WRITEA
   43 CONTINUE
   11 CONTINUE
      WRITE(6,96)
   96 FORMAT(//2X,'EDGES SPECIFIED')
      GO TO 16
   14 ITERM=1
      GO TO 29
   16 DO 21 I=1,N
      ISPC=0
      IF(IR(I).EQ.1) GO TO 21
C  FIND VERTICES WITH TWO INCIDENT NECESSARY EDGES.
      DO 22 J=1,N2
      ISPC=ISPC+SPC(I,J)
   22 CONTINUE
      IF(ISPC-2)21,23,24
   23 ICHAN=1
      IS=0
      DO 25 J=1,N2
      IF(SPC(I,J).NE.1) GO TO 25
      IS=IS+1
      II(IS)=J
      IF(IS.EQ.3) GO TO 24
   25 CONTINUE
      J1=II(1)
      J2=II(2)
```

```
C  COMBINE NECESSARY EDGES.
   32 SPC(J1,J2)=SPC(J1,J2)+1
      SPC(J2,J1)=SPC(J2,J1)+1
      ADJ(J1,J2)=1
      ADJ(J2,J1)=1
      IF(SPC(J1,J2).EQ.2) GO TO 24
      IR(I)=1
C  ELIMINATE EDGES INCIDENT TO A SOLVED VERTEX.
C  DELETE SOLVED VERTICES.
      DO 31 J=1,N2
      ADJ(I,J)=0
      ADJ(J,I)=0
      SPC(I,J)=0
      SPC(J,I)=0
   31 CONTINUE
      GO TO 42
      WRITE(6,95)I
      CALL WRITEA
   42 CONTINUE
C   CHECK IF CHAIN CONNECTS ENDPOINTS.  IF SO, ARE ALL VERTICES COVERED?
      IF((J1.EQ.N+1).OR.(J1.EQ.IE1)) GO TO 51
      IF((J1.EQ.N+2).OR.(J1.EQ.IE2)) GO TO 52
      GO TO 21
   51 IF((J2.EQ.N+2).OR.(J2.EQ.IE2)) GO TO 53
      GO TO 21
   52 IF((J2.EQ.N+1).OR.(J2.EQ.IE1)) GO TO 53
      GO TO 21
   53 CONTINUE
      DO 33 J=1,N
      IF((J.EQ.IE1).OR.(J.EQ.IE2)) GO TO 33
      IF(IR(J).NE.1) GO TO 24
   33 CONTINUE
   21 CONTINUE
      GO TO 36
C  INDICATES INFEASIBLE SOLUTION
   24 ITERM=1
      GO TO 29
   36 IF(ICHAN.EQ.1) GO TO 19
   29 RETURN
      END
      SUBROUTINE SETSUB
C
C  VARIABLES FOR VERTEX SET COMPOSITION
C  ISET -- VERTEX SETS WHERE  EACH ROW IS A VERTEX SET, AND =1 ON
C     A COLUMN INDICATES VERTEX IS IN THAT SET.
C  ISETA -- INDICATES ADJACENT VERTICES IN SAME VERTEX SET
C   ISEV -- PAIRS OF VERTICES IN SAME VERTEX SET
C   IS2N -- PAIRS OF ADJACENT VERTICES NOT IN SAME VERTEX SET
C    =1 IF CONDITION APPLIES IN ABOVE MATRICES
C
      INTEGER ADJ,SPC,DIST
      INTEGER ADJS,SPCS,DISTS
      INTEGER SPCT,ST
      DIMENSION ADJ(30,30),SPC(30,30),MHC(30,30),DIST(30,30),IR(30)
      DIMENSION ISET(20,30),ISETA(30,30),ISEV(30,30),IS2N(30,30)
      DIMENSION IX(20),KSET(20,30),DISTS( 5,30,30)
      DIMENSION ADJS( 5,30,30),SPCS( 5,30,30),MHCS( 5,30,30)
      DIMENSION IE1S( 5),IE2S( 5)
      DIMENSION KMAT(20,20),KTEM(20),KITEM(20)
      COMMON ITERM,ADJ,SPC,MHC,N,N2,IR,DIST,IE1,IE2
      COMMON IORG,KJ
      COMMON ADJS,SPCS
      K=1
      DO 28 IJ=1,N2
      ISET(1,IJ)=0
   28 CONTINUE
      DO 25 I=1,N2
      DO 26 J=1,N2
      ISEV(I,J)=0
   26 CONTINUE
   25 CONTINUE
      DO 1 I=1,N
      IF(IR(I).EQ.1) GO TO 1
      DO 2 J=1,N
      IF(IR(J).EQ.1) GO TO 2
      IF(ISEV(I,J).EQ.1) GO TO 2
      IF(I.GE.J) GO TO 2
      IF(ADJ(I,J).NE.1) GO TO 2
C  IDENTIFY EDGES WHOSE INCIDENT VERTICES ARE NOT IN A SET.
      ISET(K,I)=1
      ISET(K,J)=1
      L=0
C  FIND A COMMON ADJACENT VERTEX.
      DO 3 J1=1,N
      IF(IR(J1).EQ.1) GO TO 3
      IF(ISET(K,J1).EQ.1) GO TO 3
      IF(ADJ(I,J1).NE.1) GO TO 3
      IF(ADJ(J,J1).NE.1) GO TO 3
      L=1
      ISET(K,J1)=1
    3 CONTINUE
      IF(L.NE.0) GO TO 4
      ISET(K,I)=0
      ISET(K,J)=0
      GO TO 2
    4 DO 23 II=1,N
      DO 24 JJ=1,N
      ISETA(II,JJ)=0
   24 CONTINUE
   23 CONTINUE
      DO 5 II=1,N
      IF(ISET(K,II).NE.1) GO TO 5
      DO 6 JJ=1,N
      ISETA(II,JJ)=ADJ(II,JJ)
   6 CONTINUE
   5 CONTINUE
   9 ICHN=0
     DO 7 JJ=1,N
     IF(IR(JJ).EQ.1) GO TO 7
     IF(ISET(K,JJ).EQ.1) GO TO 7
     ISUM=0
C  FIND VERTICES ADJACENT TO TWO VERTICES IN VERTEX SET.
     DO 8 II=1,N
     ISUM=ISUM+ISETA(II,JJ)
   8 CONTINUE
     IF(ISUM.LT.2) GO TO 7
     ICHN=1
     ISET(K,JJ)=1
     DO 19 JJJ=1,N
     ISETA(JJ,JJJ)=ADJ(JJ,JJJ)
  19 CONTINUE
   7 CONTINUE
     IF(ICHN.EQ.1) GO TO 9
     DO 13 II=1,N
     IF(IR(II).EQ.1) GO TO 13
     DO 14 JJ=1,N
     IF(IR(JJ).EQ.1) GO TO 14
     IF((ISET(K,II).EQ.1).AND.(ISET(K,JJ).EQ.1)) GO TO 16
     GO TO 14
C  INDICATES THAT VERTICES ARE IN SAME VERTEX SET.
  16 ISEV(II,JJ)=1
  14 CONTINUE
  13 CONTINUE
     K=K+1
     DO 29 IJ=1,N2
     ISET(K,IJ)=0
   29 CONTINUE
    2 CONTINUE
    1 CONTINUE
      KO=K-1
C   IDENTIFY ADJACENT VERTICES NOT IN SAME VERTEX SET.
C    (EDGE IS A CONNECTOR BETWEEN VERTEX SETS.)
      DO 21 I=1,N2
      DO 22 J=1,N2
      IS2N(I,J)=ADJ(I,J)-ISEV(I,J)
      IF(IS2N(I,J).LT.0) IS2N(I,J)=0
   22 CONTINUE
   21 CONTINUE
      DO 31 I=1,N
      IF(IR(I).EQ.1) GO TO 31
      ISEVT=0
      DO 32 J=1,N
      ISEVT=ISEVT+ISEV(I,J)
   32 CONTINUE
C   INDICATES VERTEX SETS WITH A SINGLE VERTEX.
      IF(ISEVT.EQ.0) ISET(K,I)=1
      IF(ISEVT.EQ.0) K=K+1
      DO 27 IJ=1,N2
      ISET(K,IJ)=0
   27 CONTINUE
   31 CONTINUE
      K1=K-1
      DO 78 KK=1,K1
      IX(KK)=0
   78 CONTINUE
C    ELIMINATE VERTEX SETS TOTALLY A SUBSET OF ANOTHER VERTEX SET.
      DO 333 KA=1,KO
      IF(IX(KA).EQ.1) GO TO 333
      DO 334 KB=1,KO
      IF(IX(KB).EQ.1) GO TO 334
      IF(KA.EQ.KB) GO TO 334
      DO 335 I=1,N
      IF(ISET(KB,I).NE.1) GO TO 335
      IF(ISET(KA,I).NE.1) GO TO 334
  335 CONTINUE
      IX(KB)=1
      DO 337 I=1,N
      ISET(KB,I)=0
  337 CONTINUE
  334 CONTINUE
  333 CONTINUE
C  PRINTOUT DATA ON VERTEX SETS.
      WRITE(6,721)
  721 FORMAT(///20X,'SETS OF VERTICES',//)
      DO 33 KK=1,K1
      WRITE(6,97)(ISET(KK,I),I=1,N)
   97 FORMAT(/1X,30I3)
   33 CONTINUE
      WRITE(6,91)
      GO TO 501
      DO 34 I=1,N2
      WRITE(6,98)(ISEV(I,J),J=1,N2)
   34 CONTINUE
      WRITE(6,91)
  501 CONTINUE
      WRITE(6,722)
  722 FORMAT(///10X,'ADJACENT VERTICES NOT IN SETS',//)
      DO 35 I=1,N2
      WRITE(6,98)(IS2N(I,J),J=1,N2)
   35 CONTINUE
   98 FORMAT(1X,30I3)
C     CCCCC
C     CCCCCCCCCCCCCCCCCC
C     CCCCC
```

```
C
C    SEPARABLE SUBGRAPHS
C
C  MATRICES FOR SEPARABLE SUBGRAPHS
C  ADJS,SPCS,MHCS,DISTS REPRESENT EQUIVILENT INFORMATION AS ADJ,SPC,
C     MHC, DIST, FOR EACH SEPARABLE SUBGRAPH.
C  IX -- =0 IF VERTEX SET REMAINS; =1 IF VERTEX SET HAS BEEN SEPARATED
C     OR COMBINED WITH ANOTHER SET.
C
C  KSET -- INDICATES SEPARATION VERTICES OF SETS, TOTAL OF TWO IN ROW
C     INDICATES SET FORMS A SEPARABLE SUBGRAPH.
C
  201 CONTINUE
      ISEP=0
      DO 51 KK=1,KO
      IF(IX(KK).EQ.1) GO TO 51
      DO 79 I=1,N2
      KSET(KK,I)=0
   79 CONTINUE
      KSETT=0
C  DETERMINE IF VERTEX SET FORMS A SEPARABLE SUBGRAPH.
      DO 52 I=1,N
      IF(ISET(KK,I).NE.1) GO TO 52
      DO 53 J=1,N2
      IF(IS2N(I,J).EQ.1) GO TO 54
      GO TO 53
   54 KSET(KK,I)=1
      KSETT=KSETT+1
      GO TO 52
   53 CONTINUE
      KISTT=0
      DO 55 KKK=1,KO
      IF(ISET(KKK,I).EQ.1) KISTT=KISTT+1
   55 CONTINUE
      IF(KISTT.GE.2) KSET(KK,I)=1
      IF(KISTT.GE.2) KSETT=KSETT+1
   52 CONTINUE
      GO TO 533
      WRITE(6,999)KSETT,KJ,KK
  999 FORMAT(/2X,3I3,/)
  533 CONTINUE
      IF(KSETT-2)61,62,211
CCCCCCCCCCCCCCCCCC
C   IF VERTEX SET IS SEPARATED ABOUT THREE VERTICES, DETERMINE IF AN
C     EDGE ABOUT ONE VERTEX IS INFEASIBLE, RESULTING IN  A SEPARABLE
C     SUBGRAPH.
  211 IF(KSETT.NE.3) GO TO 51
      ST=0
      JJ=0
      IT=0
      DO 212 I=1,N
      IF(KSET(KK,I).NE.1) GO TO 212
      SPCT=0
      IS2NT=0
      DO 213 J=1,N2
      IF(ISET(KK,J).EQ.1) GO TO 213
      SPCT=SPCT+SPC(I,J)
      IS2NT=IS2NT+IS2N(I,J)
      GO TO 507
      WRITE(6,94) I,J,SPCT,IS2NT
  507 CONTINUE
  213 CONTINUE
      IF(SPCT.EQ.1) ST=ST+1
      IF(SPCT.EQ.0) JJ=I
      IF(IS2NT.EQ.1) IT=IT+1
      IF(IS2NT.GT.1) GO TO 51
  212 CONTINUE
      GO TO 534
      WRITE(6,94) ST,IT
   94 FORMAT(/5X,5I3)
  534 CONTINUE
      IF(ST.EQ.3) GO TO 61
      IF(ST.NE.2) GO TO 51
      IF(IT.NE.3) GO TO 51
      KSET(KK,JJ)=0
      DO 216 I=1,N
      IF(IS2N(JJ,I).NE.1) GO TO 216
      ADJ(I,JJ)=0
      ADJ(JJ,I)=0
      IS2N(I,JJ)=0
      IS2N(JJ,I)=0
  216 CONTINUE
      GO TO 62
CCCCCCCCCCCCCC
   61 ITERM=1
      GO TO 84
C  FORM SEPARABLE SUBGRAPHS.
   62 KJ=KJ+1
      DO 63 I=1,N2
      DO 64 J=1,N2
      ADJS(KJ,I,J)=0
      SPCS(KJ,I,J)=0
      MHCS(KJ,I,J)=0
      DISTS(KJ,I,J)=1.0E10
   64 CONTINUE
   63 CONTINUE
```

```
      IC=0
      WRITE(6,82)(KSET(KK,I),I=1,N)
      WRITE(6,91)
      DO 70 I=1,N
      IF(KSET(KK,I).EQ.1) GO TO 72
      GO TO 70
   72 IC=IC+1
      IF(IC.EQ.1) IES1=I
      IF(IC.EQ.2) IES2=I
   70 CONTINUE
C  FORMULATE ENDPOINT SETUP FOR SEPARABLE SUBGRAPHS.
      IE1S(KJ)=IES1
      IE2S(KJ)=IES2
      ADJS(KJ,N+1,IES1)=1
      ADJS(KJ,IES1,N+1)=1
      ADJS(KJ,N+2,IES2)=1
      ADJS(KJ,IES2,N+2)=1
      SPCS(KJ,N+1,IES1)=1
      SPCS(KJ,IES1,N+1)=1
      SPCS(KJ,N+2,IES2)=1
      SPCS(KJ,IES2,N+2)=1
      DO 75 I=1,N2
      IF((ISET(KK,I).EQ.1).AND.(KSET(KK,I).EQ.0)) IR(I)=1
      DO 76 J=1,N2
      IF(I.EQ.J) GO TO 76
      IF((ISET(KK,I).EQ.0).OR.(ISET(KK,J).EQ.0)) GO TO 76
      IF((I.EQ.IES1).AND.(J.EQ.IES2)) GO TO 77
      IF((I.EQ.IES2).AND.(J.EQ.IES1)) GO TO 77
      ADJS(KJ,I,J)=ADJ(I,J)
      ADJ(I,J)=0
      SPCS(KJ,I,J)=SPC(I,J)
      SPC(I,J)=0
      DISTS(KJ,I,J)=DIST(I,J)
      GO TO 76
C  REPLACE SUBGRAPH BY NECESSARY EDGE, IN ORIGINAL GRAPH.
   77 ADJ(I,J)=1
      IF(SPC(I,J).EQ.1) GO TO 61
      SPC(I,J)=1
   76 CONTINUE
   75 CONTINUE
      IX(KK)=1
      ISEP=1
C   WRITEA HERE CALL FOR GRAPH AND SUBGRAPH
C  PRINTOUT ORIGINAL GRAPH REVISED.
      WRITE(6,723)
  723 FORMAT(///10X,'ADJACENCY MATRIX -- ORIGINAL GRAPH',//)
      DO 121 I=1,N2
      WRITE(6,82)(ADJ(I,J),J=1,N2)
   82 FORMAT(1X,30I3)
  121 CONTINUE
      WRITE(6,91)
      GO TO 502
      DO 122 I=1,N2
      WRITE(6,82)(DIST(I,J),J=1,N2)
  122 CONTINUE
      WRITE(6,91)
      DO 123 I=1,N2
      WRITE(6,82)(SPC(I,J),J=1,N2)
  123 CONTINUE
      WRITE(6,91)
      DO 124 I=1,N2
      WRITE(6,82)(MHC(I,J),J=1,N2)
  124 CONTINUE
  502 CONTINUE
```

```
C  PRINTOUT SEPARABLE SUBGRAPH IMFORMATION.
      WRITE(6,83)KJ
   83 FORMAT(///10X,'SEPARABLE SUBGRAPH',2X,I2,//)
      WRITE(6,724)
  724 FORMAT(20X,'ADJACENCY MATRIX',//)
      DO 125 I=1,N2
      WRITE(6,82)(ADJS(KJ,I,J),J=1,N2)
  125 CONTINUE
      WRITE(6,91)
      GO TO 503
      DO 126 I=1,N2
      WRITE(6,82)(DISTS(KJ,I,J),J=1,N2)
  126 CONTINUE
      WRITE(6,91)
      DO 127 I=1,N2
      WRITE(6,82)(SPCS(KJ,I,J),J=1,N2)
  127 CONTINUE
      WRITE(6,91)
      DO 128 I=1,N2
      WRITE(6,82)(MHCS(KJ,I,J),J=1,N2)
  128 CONTINUE
      WRITE(6,91)
  503 CONTINUE
C    END WRITE CALL
   51 CONTINUE
      IF(ISEP.EQ.1) GO TO 196
      GO TO 85
   84 WRITE(6,89)KK
   89 FORMAT(/1X,'INFEASIBLE',5X,'SET',I3)
      GO TO 196
   85 CONTINUE
```

```
C
C   COMBINED SETS OF 2 ADJACENCY
C
      DO 141 KK=1,K1
      IF(IX(KK).EQ.1) GO TO 141
      DO 142 KKK=1,K1
      IF(KK.GE.KKK) GO TO 142
      IF(IX(KKK).EQ.1) GO TO 142
      ISA=0
C  DETERMINE IF VERTEX SET KK HAS TWO VERTICES ADJACENT TO VERTICES
C     IN VERTEX SET KKK.
      DO 143 I=1,N
      IF(ISET(KK,I).NE.1) GO TO 143
      DO 144 J=1,N
      IF(ISET(KKK,J).NE.1) GO TO 144
      IF(I.EQ.J) GO TO 145
      IF(IS2N(I,J).EQ.1) GO TO 145
      GO TO 144
  145 ISA=ISA+1
      GO TO 535
      WRITE(6,998)KK,KKK,I,J,ISA
  998 FORMAT(/1X,5I3,5X,'I',/)
  535 CONTINUE
      IF(ISA.EQ.2) GO TO 151
      GO TO 143
  144 CONTINUE
  143 CONTINUE
      GO TO 161
  151 JSA=0
      IF(KKK.GT.K0) GO TO 162
C  DETERMINE IF VERTEX SET KKK HAS TWO VERTICES ADJACENT TO VERTICES
C     IN VERTEX SET KK.
```

```
      DO 153 J=1,N
      IF(ISET(KKK,J).NE.1) GO TO 153
      DO 154 I=1,N
      IF(ISET(KK,I).NE.1) GO TO 154
      IF( I.EQ.J) GO TO 155
      IF(IS2N(I,J).EQ.1) GO TO 155
      GO TO 154
  155 JSA=JSA+1
      GO TO 536
      WRITE(6,997)KK,KKK,I,J,JSA
  997 FORMAT(/1X,5I3,5X,'J',/)
  536 CONTINUE
      IF(JSA.EQ.2) GO TO 162
      GO TO 153
  154 CONTINUE
  153 CONTINUE
      IF(JSA.LT.2) GO TO 161
      IF(JSA.EQ.2) GO TO 162
  161 CONTINUE
      GO TO 142
C  COMBINE VERTEX SETS KK AND KKK IF ABOVE CONDITIONS RESULT.
  162 IX(KKK)=1
      DO 163 I=1,N
      IF(ISET(KKK,I).NE.1) GO TO 163
      ISET(KK,I)=1
      ISET(KKK,I)=0
  163 CONTINUE
      DO 164 I=1,N
      IF(ISET(KK,I).NE.1) GO TO 164
      DO 165 J=1,N
      IF(ISET(KK,J).NE.1) GO TO 165
      IS2N(I,J)=0
  165 CONTINUE
  164 CONTINUE
C  PRINTOUT DATA ON VERTEX SETS.
      WRITE(6,88)
   88 FORMAT(//10X,'COMBINED SETS BY DOUBLE ADJACENCY',//)
      WRITE(6,723)
      DO 166 I=1,N2
      WRITE(6,87)(ADJ(I,J),J=1,N2)
  166 CONTINUE
      WRITE(6,721)
      DO 167 K5=1,K1
      WRITE(6,97)(ISET(K5,I),I=1,N2)
  167 CONTINUE
      WRITE(6,722)
      DO 168 I=1,N2
      WRITE(6,87)(IS2N(I,J),J=1,N2)
  168 CONTINUE
      WRITE(6,91)
   87 FORMAT(1X,30I3)
   91 FORMAT(//1X,/)
      GO TO 201
  142 CONTINUE
  141 CONTINUE
```

```
C
C   COMBINED SETS BY CYCLES OF SETS
C
C     VARIABLES USED FOR COMBINING SETS BY CYCLES
C  KMAT -- INDICATES VERTEX SETS THAT ARE CONNECTED VIA A VERTEX OR EDGE.
C  KTEM -- INDICATES POSITION OF VERTEX SET IN SEQUENCE SEARCHED FOR CYCLE.
C  KITEM -- INDICATES IF VERTEX SET HAS BEEN ADDED TO SEQUENCE OF SETS.
C        =1  INDICATES IMFORMATION IN ABOVE MATRICES
C
      DO 187 I=1,K1
      KTEM(I)=0
      KITEM(I)=0
  187 CONTINUE
      DO 169 I=1,K1
      DO 170 J=1,K1
      KMAT(I,J)=0
  170 CONTINUE
  169 CONTINUE
C  DETERMINE FOR ALL REMAINING SETS, THOSE THAT ARE CONNECTED BY A VERTEX
C     OR EDGE.
      DO 171 KA=1,K1
      IF(IX(KA).EQ.1) GO TO 171
      DO 172 KB=1,K1
      IF(KA.EQ.KB) GO TO 172
      IF(IX(KB).EQ.1) GO TO 172
      DO 173 I=1,N
      IF(ISET(KA,I).NE.1) GO TO 173
      DO 174 J=1,N
      IF(ISET(KB,J).NE.1) GO TO 174
      IF(I.EQ.J) GO TO 175
      IF(IS2N(I,J).EQ.1) GO TO 175
      GO TO 174
  175 KMAT(KA,KB)=1
      GO TO 172
  174 CONTINUE
  173 CONTINUE
  172 CONTINUE
  171 CONTINUE
      GO TO 537
      DO 193 KA=1,K1
      WRITE(6,199)(KMAT(KA,KB),KB=1,K1)
  193 CONTINUE
  199 FORMAT(/5X,20I3)
  537 CONTINUE
      DO 176 KA=1,K1
      IF(IX(KA).EQ.1) GO TO 176
      DO 177 KB=1,K1
      IF(IX(KB).EQ.1) GO TO 177
      IF(KA.GE.KB) GO TO 177
      IF(KMAT(KA,KB).NE.1) GO TO 177
C  INDICATE TWO CONNECTED VERTEX SETS TO BEGIN SEQUENCE.
      KTEM(1)=KA
      KITEM(KA)=1
      KTEM(2)=KB
      KITEM(KB)=1
      KT=2
      KC=KB
  182 CONTINUE
      DO 178 KD=1,K1
      IF(KITEM(KD).EQ.1) GO TO 178
      IF(KMAT(KC,KD).NE.1) GO TO 178
C  ADD VERTEX SET INTO SEQUENCE.
      KITEM(KD)=1
      KT1=KT+1
      KTEM(KT1)=KD
      KTM=KT-1
C  SEARCH FOR A CYCLE OF VERTEX SETS IN THE SEQUENCE.
      DO 179 L=1,KTM
      KE=KTEM(L)
      IF(KMAT(KD,KE).NE.1) GO TO 179
      L1=L+1
C  COMBINE CYCLED VERTEX SETS INT A SINGLE VERTEX SET.
      DO 180 LM=L1,KT1
      KF=KTEM(LM)
      IX(KF)=1
      DO 181 IK=1,N
      IF(ISET(KE,IK).EQ.1) GO TO 194
      ISET(KE,IK)=ISET(KF,IK)
  194 ISET(KF,IK)=0
  181 CONTINUE
      DO 185 I=1,N
      IF(ISET(KE,I).NE.1) GO TO 185
      DO 186 J=1,N
      IF(ISET(KE,J).NE.1) GO TO 186
      IS2N(I,J)=0
  186 CONTINUE
  185 CONTINUE
      WRITE(6,188)
  188 FORMAT(///10X,'COMBINED SETS BY CYCLES',/)
      GO TO 539
      WRITE(6,200)KA,KB,KC,KD,KE,KF,L,KT
  200 FORMAT(//2X,8I3)
  539 CONTINUE
      IF(LM.LT.KT1) GO TO 540
C  PRINTOUT DATA ON VERTEX SETS.
      WRITE(6,721)
      DO 191 K5=1,K1
      WRITE(6,97)(ISET(K5,I),I=1,N2)
  191 CONTINUE
      WRITE(6,722)
      DO 192 I=1,N2
      WRITE(6,87)(IS2N(I,J),J=1,N2)
  192 CONTINUE
      WRITE(6,91)
  540 CONTINUE
C  UPDATE MATRIX OF CONNECTED SETS.
      DO 183 IL=1,K1
      IF(IL.EQ.KE) GO TO 184
      IF(IL.EQ.KF) GO TO 183
      IF(KMAT(KE,IL).EQ.1) GO TO 184
      KMAT(KE,IL)=KMAT(KF,IL)
      KMAT(IL,KE)=KMAT(IL,KF)
  184 KMAT(KF,IL)=0
      KMAT(IL,KF)=0
  183 CONTINUE
  180 CONTINUE
      GO TO 201
  179 CONTINUE
      KT=KT+1
      KC=KD
      GO TO 182
  178 CONTINUE
  177 CONTINUE
  176 CONTINUE
  196 CONTINUE
      RETURN
      END
//DATA
```

```
C
C        THIS ALGORITHM PERFORMS A BRANCH AND BOUND TO FIND THE MINIMAL
C     HAMILTONIAN CHAIN OF A GRAPH.  MINIMAL SPANNING TREES ARE FOUND.
C     IN BRANCHING, EDGES INCIDENT TO VERTICES WITH MORE EDGES INCIDENT
C     IN THE SPANNING TREE THAN CAN BE IN A HAMILTONIAN CHAIN ARE ELIMINATED.
C     THE PROGRAM IS SETUP SO THAT BOTH  ENDPOINTS HAVE AN ARTIFICAL
C     INCIDENT EDGE ADDED.
C
C
C          VARIABLES
C     ADJI -- VERTEX I INCIDENT TO GIVEN EDGE
C     ADJJ -- VERTEX J INCIDENT TO GIVEN EDGE             I<J
C     DISTE -- DISTANCE OF EDGE FROM VI TO VJ
C     SPCE -- =1 IF EDGE IS A NECESSARY EDGE.
C     ADJE -- =1 IF EDGE IS IN PARTIAL GRAPH BEING BRANCHED;  =0 IF NOT
C     MST -- =1 IF EDGE IS IN MINIMAL SPANNING TREE OF PARTIAL GRAPH
C     BEING BRANCHED
C     ICONN -- =1 IF PAIR OF VERTICES CAN NOT BE CONNECTED IN MINIMAL
C      SPANNING TREE, UNLESS ASSOCIATED EDGE IS ALREADY IN THE SPANNING TREE
C     ITE -- NUMBER OF INCIDENT MINIMAL SPANNING TREE EDGES TO EACH VERTEX.
C     IADJ -- NUMBER OF INCIDENT EDGES TO VERTEX IN PARTIAL GRAPH BEING BRANCHED
C     IEDS -- NUMBER OF POSSIBLE EDGE COMBINATIONS TO BRANCH ON ABOUT EACH VERTEX
C     MIEDS -- MINIMAL VALUE OF IEDS FOR OVERSATURATED VERTICES
C     SPE -- NUMBER OF INCIDENT NECESSARY EDGES TO EACH VERTEX
C     MSE -- =1 INDICATES EDGE IS TO BE ELIMINATED IN SOME PARTIAL GRAPHS
C      TO BE BRANCHED TO
C     BA -- =1 INDICATES PARTIAL GRAPH HAS BEEN BRANCHED ON OR ELIMINATED
C     DISHC -- TOTAL DISTANCE OF MINIMAL SPANNING TREE THAT FORMS  THE
C      LEAST VALUED HAMILTONIAN CHAIN FOUND  (USED AS A BOUND)
C     IEDM -- NUMBER OF PARTIAL GRAPH BEING CREATED
C     DISTL -- VALUE OF LEAST VALUED MINIMAL SPANNING TREE OF PARTIAL GRAPHS
C      IN STORAGE
```

```
C    DISTC -- TOTAL DISTANCE OF MINIMAL SPANNING TREE BEING FOUND
C    DIST -- TOTAL DISTANCE OF MINIMAL SPANNINF TREE FOR EACH PARTIAL GRAPH
C
C     MATRICES WITH PARTIAL GRAPH RESULTS;   (I,B):  I - EDGE;  B - PARTIAL GRAPH
C    ADJEB -- =1 INDICATES EXISTENCE OF EDGE IN PARTIAL GRAPH
C    MSTB -- =1 INDICATES EXISTENCE OF EDGE IN MINIMAL SPANNING TREE
C     OF PARTIAL GRAPH
C
C
C    DATA (UNFORMATTED)
C    FIRST CARD:  N -- NUMBER OF VERTICES
C    SECOND CARD:  NE -- NUMBER OF EDGES
C    THIRD CARD:  IE1 -- POSITION OF END POINT VERTEX 1, IE2 -- POSITION
C                  OF ENDPOINT VERTEX 2
C    NEXT NE CARDS -- IMFORMATION FOR EACH EDGE
C         INCIDENT VERTEX I, INCIDENT VERTEX J, =1 IF EDGE EXISTS, DISTANCE
C             OF EDGE, =1 IF EDGE IS A NECESSARY EDGE (=0 OTHERWISE)
C
```

```
C
      IMPLICIT INTEGER(A-H,O-Z)
      DIMENSION KP(100),ADJI(100),ADJJ(100),DISTE(100),SPCE(100)
      DIMENSION MST(100),ICONN(40,40),ADJE(100),ITE(40),IEDS(40)
      DIMENSION SPE(40),MSE(40),BA(100),ADJEB(100,100),DIST(100)
      DIMENSION MSTB(100,100)
      DIMENSION IADJ(40)
      READ,N
      READ,NE
      READ,IE1,IE2
      K=0
      WRITE(6,89)
C  READ IN EDGES.
    3 READ,I,J,ADJ,DIJ,SPC
      WRITE(6,90)I,J,ADJ,DIJ,SPC
      K=K+1
      KP(K)=K
      II=KP(K)
      ADJI(II)=I
      ADJJ(II)=J
      ADJE(II)=ADJ
      DISTE(II)=DIJ
      SPCE(II)=SPC
      MST(II)=0
      IF(K.LT.NE) GO TO 3
      WRITE(6,89)
C  SORT EDGES IN ORDER OF DISTANCE VALUE.
      DO 11 I=1,K
      DO 12 J=1,K
      IF(I.GE.J) GO TO 12
      IF(DISTE(I).LE.DISTE(J)) GO TO 12
      DISTS=DISTE(I)
      ADJIS=ADJI(I)
      ADJJS=ADJJ(I)
      ADJES=ADJE(I)
      SPCES=SPCE(I)
      MSTS=MST(I)
      DISTE(I)=DISTE(J)
      ADJI(I)=ADJI(J)
      ADJJ(I)=ADJJ(J)
      ADJE(I)=ADJE(J)
      SPCE(I)=SPCE(J)
      MST(I)=MST(J)
      DISTE(J)=DISTS
      ADJI(J)=ADJIS
      ADJJ(J)=ADJJS
      ADJE(J)=ADJES
      SPCE(J)=SPCES
      MST(J)=MSTS
   12 CONTINUE
   11 CONTINUE
      DISHC=1000000
      BT=0
      INF=0
C   FIND INITIAL MINIMAL SPANNING TREE.
      WRITE(6,106)
  106 FORMAT(/5X,'    VI    VJ   ADJ   DIST   NES   MST')
      WRITE(6,89)
      CALL MIMSPT(ADJI,ADJJ,ADJE,MST,DISTE,SPCE,N,K,DISTC,INF)
      IF(INF.EQ.1) WRITE(6,198)
      IF(INF.EQ.1) GO TO 202
  201 CONTINUE
      DO 38 I=1,N
      ITE(I)=0
      SPE(I)=0
      IADJ(I)=0
   38 CONTINUE
      SPE(IE1)=1
      SPE(IE2)=1
      ITE(IE1)=1
      ITE(IE2)=1
      IADJ(IE1)=1
      IADJ(IE2)=1
C  FIND THE NUMBER OF INCIDENT SPANNING TREE EDGES TO EACH VERTEX.
      DO 41 II=1,K
      IF(ADJE(II).NE.1) GO TO 41
      AI=ADJI(II)
      AJ=ADJJ(II)
      IADJ(AI)=IADJ(AI)+1
      IADJ(AJ)=IADJ(AJ)+1
      IF(MST(II).NE.1) GO TO 41
      ITE(AI)=ITE(AI)+1
      ITE(AJ)=ITE(AJ)+1
      IF(SPCE(II).NE.1) GO TO 41
      SPE(AI)=SPE(AI)+1
      SPE(AJ)=SPE(AJ)+1
   41 CONTINUE
      WRITE(6,103)
  103 FORMAT(//5X,'NUMBER OF INCIDENT EDGES')
      WRITE(6,89)
      WRITE(6,91)(IADJ(I),I=1,N)
      DO 55 I=1,N
      IF(IADJ(I).LT.2) GO TO 46
   55 CONTINUE
      WRITE(6,104)
  104 FORMAT(//5X,'NUMBER OF INCIDENT MST EDGES')
```

```
      WRITE(6,89)
   90 FORMAT(5X,8I5)
      WRITE(6,91)(ITE(I),I=1,N)
   91 FORMAT(1X,40I3)
      IEPT=0
C  FIND THE NUMBER OF EDGE COMBINATIONS FOR BRANCHING AT EACH VERTEX.
      DO 51 I=1,N
      IEDS(I)=0
      IF(ITE(I).LE.2) GO TO 51
      IEPT=IEPT+ITE(I)-2
      IEDS(I)=ITE(I)*(ITE(I)-1)/2
      IF(SPE(I).EQ.1) IEDS(I)=ITE(I)-1
      IF(SPE(I).EQ.2) IEDS(I)=1
      IF(SPE(I).GT.2) GO TO 46
   51 CONTINUE
C  IF A HAMILTONIAN CHAIN IS BEING BRANCHED THEN IT IS THE SOLUTION.
      IF(IEPT.EQ.0) GO TO 59
      IB=0
      MIEDS=1000
C  FIND THE VERTEX WITH THE LEAST NUMBER OF EDGE COMBINATIONS TO ELIMINATE.
      DO 53 I=1,N
      IF(IEDS(I).EQ.0) GO TO 53
      IF(IEDS(I).GE.MIEDS) GO TO 53
      IB=I
      MIEDS=IEDS(I)
   53 CONTINUE
      WRITE(6,105)
  105 FORMAT(//5X,'VERTEX',3X,'# EDGE SETS')
      WRITE(6,89)
      WRITE(6,90)IB,MIEDS
      GO TO 57
   59 WRITE(6,92)
```

```
      DISHC=DISTC
   92 FORMAT(/1X,'HAMILTONIAN CHAIN')
      GO TO 202
   89 FORMAT(/2X)
   57 CONTINUE
C  INDICATE OVERSATURATED EDGES INCIDENT TO OVERSATURATED VERTEX BEING BRANCHED.
      DO 61 II=1,K
      MSE(II)=0
      IF(MST(II).EQ.0) GO TO 61
      IF(ADJI(II).EQ.IB) MSE(II)=1
      IF(ADJJ(II).EQ.IB) MSE(II)=1
   61 CONTINUE
      IEDM=IEDS(IB)
      DO 62 I=1,K
      DO 63 BR=1,IEDM
      ADJEB(I,BT+BR)=0
   63 CONTINUE
   62 CONTINUE
      BR=1
      IF(SPE(IB).EQ.1) GO TO 65
      IF(SPE(IB).EQ.2) GO TO 161
C  DETERMINE OVERSATURATED EDGES IN PARTIAL GRAPH BEING BRANCHE TO,
C   INCIDENT TO OVERSATURATED VERTEX BEING BRANCHED ON.
      DO 66 I=1,K
      DO 67 J=1,K
      IF(I.GE.J) GO TO 67
      IF(MSE(I).NE.1.OR.MSE(J).NE.1) GO TO 67
      ADJEB(I,BT+BR)=1
      ADJEB(J,BT+BR)=1
      BR=BR+1
   67 CONTINUE
   66 CONTINUE
```

```
      GO TO 72
C   IF VERTEX BEING BRANCHED HAS TWO NECESSARY EDGES INCIDENT,
C     RETAIN ONLY THOSE EDGES.
  161 DO 168 II=1,K
      IF(MSE(II).NE.1) GO TO 168
      IF(SPCE(II).EQ.1) ADJEB(II,BT+1)=1
      IF(SPCE(II).EQ.0) ADJEB(II,BT+1)=0
  168 CONTINUE
      BA(BT+1)=0
      GO TO 72
C   DETERMINE OVERSATURATED EDGES IN PARTIAL GRAPH BRANCHED TO, IF ONE
C   EDGE INCIDENT TO VERTEX BRANCHED ON IS A NECESSARY EDGE.
   65 ACB=0
      DO 68 II=1,K
      IF(MSE(II).EQ.1.AND.SPCE(II).EQ.1) ACB=II
   68 CONTINUE
      BR=1
      DO 69 II=1,K
      IF(ACB.EQ.0) GO TO 73
      ADJEB(ACB,BT+BR)=1
   73 CONTINUE
      IF(II.EQ.ACB) GO TO 69
      IF(MSE(II).NE.1) GO TO 69
      ADJEB(II,BT+BR)=1
      BR=BR+1
   69 CONTINUE
   72 DO 70 BR=1,IEDM
      BA(BT+BR)=0
C  DETERMINE ALL EDGES IN EACH NEW PARTIAL GRAPH.
      DO 71 II=1,K
      ADJEB(II,BT+BR)=ADJEB(II,BT+BR)+ADJE(II)-MSE(II)
   71 CONTINUE
```

```
      WRITE(6,98)(ADJEB(II,BT+BR),II=1,K)
   98 FORMAT(//1X,40I3)
   70 CONTINUE
      DO 81 I=1,IEDM
      DO 82 II=1,K
      ADJE(II)=ADJEB(II,BT+I)
   82 CONTINUE
      BI=BT+I
      WRITE(6,101)BI
  101 FORMAT(//5X,'PARTIAL GRAPH',2X,I3)
      WRITE(6,106)
      WRITE(6,89)
C  FIND THE MINIMAL SPANNING TREE FOR EACH NEW PARTIAL GRAPH.
      CALL MIMSPT(ADJI,ADJJ,ADJE,MST,DISTE,SPCE,N,K,DISTC,INF)
      IF(INF.EQ.1) BA(BT+I)=1
      IF(INF.EQ.1) GO TO 81
      DO 83 II=1,K
      MSTB(II,BT+I)=MST(II)
   83 CONTINUE
      DIST(BT+I)=DISTC
   81 CONTINUE
      IF(INF.EQ.1) WRITE(6,80)
      BT=BT+IEDM
      WRITE(6,217)(BA(J),J=1,BT)
  217 FORMAT(//1X,100I1)
C  ELIMINATE ANY PARTIAL GRAPH WITH MINIMAL SPANNING TREE HAVING
C   TOTAL DISTANCE GREATER THAN THAT OF SOME HAMILTONIAN CHAIN.
      DO 84 J=1,BT
      IF(BA(J).EQ.1) GO TO 84
      IF(DIST(J).GT.DISHC) BA(J)=1
   84 CONTINUE
  203 CONTINUE
```

```
      PB=0
C  FIND THE PARTIAL GRAPH WITH THE LEAST DISTANCE MINIMAL SPANNING TREE.
      DISTL=1000000
      DO 85 J=1,BT
      IF(BA(J).EQ.1) GO TO 85
      PB=1
      IF(DIST(J).GE.DISTL) GO TO 85
      DISTL=DIST(J)
      BJ=J
   85 CONTINUE
      IF(PB.EQ.0) GO TO 199
      BA(BJ)=1
      WRITE(6,102)BJ
  102 FORMAT(//5X,'BRANCH ON',2X,I3)
C  SET UP PARTIAL GRAPH TO BE FURTHER BRANCHED.
      DO 87 II=1,K
      ADJE(II)=ADJEB(II,BJ)
      MST(II)=MSTB(II,BJ)
   87 CONTINUE
      GO TO 200
   46 WRITE(6,80)
   80 FORMAT(/2X,'INFEASIBLE PARTIAL GRAPH')
      IF(BT.EQ.0) GO TO 199
      GO TO 203
  199 WRITE(6,198)
  198 FORMAT(///5X,'INFEASIBLE')
      GO TO 202
  200 CONTINUE
      GO TO 201
  202 CONTINUE
      WRITE(6,89)
      WRITE(6,110)
  110 FORMAT(5X,'SOLUTION')
      WRITE(6,111)
  111 FORMAT(/7X,'VI    VJ    DIST')
      WRITE(6,89)
      DO 120 I=1,K
      IF(MSTB(I,BJ).EQ.0) GO TO 120
      WRITE(6,190)ADJI(I),ADJJ(I),DISTE(I)
  120 CONTINUE
  190 FORMAT(5X,4I5)
      STOP
      END
CCCCCCCCCCCCCCCCCC   CCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE MIMSPT(ADJI,ADJJ,ADJE,MST,DISTE,SPCE,N,K,DISTC,INF)
      IMPLICIT INTEGER(A-H,O-Z)
      DIMENSION ADJI(100),ADJJ(100),DISTE(100),SPCE(100),MST(100)
      DIMENSION ICONN(40,40),ADJE(100),ITE(40)
      INF=0
      DO 16 I=1,N
      DO 17 J=1,N
      ICONN(I,J)=0
      IF(I.EQ.J) ICONN(I,J)=1
   17 CONTINUE
   16 CONTINUE
      N1=N-1
      DISTC=0
      JC=0
C   ADD NECESSARY EDGES INTO SPANNING TREE
      DO 45 II=1,K
      MST(II)=0
      IF(SPCE(II).EQ.0) GO TO 45
      MST(II)=1
      AI=ADJI(II)
      AJ=ADJJ(II)
      IF(ICONN(AI,AJ).EQ.1) GO TO 46
      JC=JC+1
      DISTC=DISTC+DISTE(II)
CCCCCCCCCCCCCCCCCCCCCCCCC
      EXECUTE CYCLE
   45 CONTINUE
C   SEARCH EDGES FOR FEASIBLITY IN MINIMAL SPANNING TREE
      DO 21 II=1,K
      IF(SPCE(II).EQ.1) GO TO 21
      IF(ADJE(II).EQ.0) GO TO 21
      AI=ADJI(II)
      AJ=ADJJ(II)
      IF(ICONN(AI,AJ).EQ.1) GO TO 21
      JC=JC+1
C  IF FEASIBLE ADD EDGE INTO SPANNING TREE.
      DISTC=DISTC+DISTE(II)
      MST(II)=1
      EXECUTE CYCLE
      IF(JC.EQ.N1) GO TO 30
   21 CONTINUE
      IF(JC.LT.N1) GO TO 46
C  PRINT OUT STATUS OF EACH EDGE AS TO WHETHER IT IS IN PARTIAL GRAPH
C   AND ITS MINIMAL SPANNING TREE.
   30 DO 31 II=1,K
      WRITE(6,90) ADJI(II),ADJJ(II),ADJE(II),DISTE(II),SPCE(II),MST(II)
   31 CONTINUE
      WRITE(6,101)DISTC
  101 FORMAT(/5X,'TOTAL DISTANCE',2X,I5)
   90 FORMAT(5X,8I5)
      GO TO 47
C  FIND ALL VERTEX PAIRINGS WHOSE EDGE WOULD FORM A CIRCUIT IN SPANNING TREE.
      REMOTE BLOCK CYCLE
      DO 22 I=1,N
      DO 23 J=1,N
      IF(I.GE.J) GO TO 23
      IF(ICONN(AI,I).EQ.1.AND.ICONN(AJ,J).EQ.1) GO TO 24
      IF(ICONN(AI,J).EQ.1.AND.ICONN(AJ,I).EQ.1) GO TO 24
      GO TO 23
   24 ICONN(I,J)=1
      ICONN(J,I)=1
   23 CONTINUE
   22 CONTINUE
      END BLOCK
   46 WRITE(6,80)
   80 FORMAT(/2X,'INFEASIBLE PARTIAL GRAPH')
      INF=1
   47 CONTINUE
      RETURN
      END
//DATA
```

A DECOMPOSITION PROCEDURE

FOR FINDING THE MINIMAL HAMILTONIAN CHAIN

OF A SPARSE GRAPH

by

Ira Ray Levinton

(ABSTRACT)

The problem considered here is one of finding the minimal Hamiltonian chain of a graph. A single chain must traverse all n vertices of a graph with the minimal distance. The proposed procedure reduces a large problem into several smaller problems and uses a branch and bound algorithm to find the minimal Hamiltonian chain of each partitioned subproblem.

The graph is decomposed and partitioned into subproblems with the use of necessary conditions for the existence of a Hamiltonian chain. This process is only applicable to graphs with relatively few incident edges per vertex.

The branch and bound algorithm makes use of concepts developed by Nicos Christofides. Hamiltonian chains are derived by using minimal spanning trees.