A PARAMETRIC STUDY OF MATCHINGS AND COVERINGS IN WEIGHTED GRAPHS

Lee James White

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in the
University of Michigan
1967

ABSTRACT

A PARAMETRIC STUDY OF GRAPHICAL MATCHING AND COVERING PROBLEMS

by Lee J. White

Given a weighted graph, a matching is a subset of the edges such that no two edges of the subset are incident to the same vertex. A covering is a subset of the edges such that each vertex is incident to at least one edge of the subset. Two interesting problems are to find a maximum weight matching and minimum weight covering. In the special case of bipartite graphs, matching and covering problems are equivalent to the well-known assignment and transportation problems of linear programming theory.

Edmonds has developed an algorithm of algebraic growth to find a maximum matching. A review of this method is presented.

A new algorithm for the solution of the minimum covering problem is developed and presented in this dissertation.

Under certain conditions, Lagrange multiplier methods can be applied to discrete programming problems. This is called the parametric approach, and is used in this dissertation to find maximum matchings, subject to the condition that the number of edges in the solution is some fixed number, k. A study of methods for this "k-cardinality" problem indicates several improvements in Edmonds' matching algorithm.

The parametric technique is extended to solve the minimum

k-cardinality covering problem. This algorithm is shown to be a generalization of Kruskal's algorithm for a minimum spanning tree. This leads naturally to a discussion of matroid systems and "greedy" algorithms, and their relationship to the parametric approach.

Matchings and coverings in weighted graphs have applications in resource allocation, diagnostics, and large scale system design.

ACKNOWLEDGEMENTS

The author wishes to thank Professors B. W. Arden, H. L. Garner, L. F. Kazda, E. L. Lawler, and A. W. Naylor for serving on his doctoral committee, and for their many helpful suggestions and criticisms. He is especially indebted to the chairman of his committee, E. L. Lawler, for his continued interest and encouraging guidance during the preparation of this dissertation. The author would like to express his appreciation to Mr. Robert Urquhart for his advice and many hours of stimulating discussion.

The author's wife typed the complete rough draft of the manuscript and offered many valuable suggestions. For this and for her encouragement, the author is deeply grateful.

The research for this thesis was supported by the Systems Engineering Laboratory of the University of Michigan through Air Force Contract AF 30(602)-3546.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

LIST OF FIGURES

LIST OF FIGURES (CONT'D)

LIST OF FIGURES (CONT'D)

LIST OF FIGURES (CONT'D)

CHAPTER I

INTRODUCTION

1.1 Introduction

Many engineering problems encountered in the design of large scale systems can be formulated as optimization problems in graph theory. Examples of such engineering questions are found in the areas of communications, data storage, and system diagnostics.

These problems are primarily combinatorial in nature. For large systems, efficient algorithms must be developed in order to find solutions which can be programmed without involving inordinate amounts of computation time and data storage.

One successful technique for the efficient solution of such combinatorial problems is linear programming. Special algorithms have been developed to solve the class of transportation and assignment problems.

The purpose of this dissertation is to develop efficient algorithms to find optimum matchings and coverings, which are generalizations of the assignment problem. Before defining these problems, let us consider some necessary definitions from graph theory.

1.2 Definitions

A graph $G = (E,V)$ is a set of vertices V, together with a set of edges E. Each edge $e_{ij} \epsilon E$ may be considered an unordered pair of vertices, $e_{ij} = (v_i, v_j)$, where v_i and v_j are incident to the edge e_{ij}. Vertices v_i and v_j are called the endpoints of e_{ij}. Where it is clear from the context, a linear ordering on the edges may be used.

A subgraph $G_1 = (E_1, V_1)$ of a graph $G = (E, V)$ is a subset V_1 of the nodes V, together with a subset E_1 of the edges E, such that both endpoints of each edge in E_1 are in V_1.

A weighted graph $[G, c_{ij}]$ is a graph together with an integer-valued weight c_{ij} associated with each edge e_{ij}.

A path in a graph is a sequence of edges $P = (e_{12}, e_{23}, \ldots, e_{n-1,n})$ such that consecutive edges in the sequence have a common vertex, and each edge appears only once in the edge-sequence P. The vertex v_1 is called the initial vertex of path P. The vertex v_n is the terminal vertex of path P. Each vertex which appears in the path, and is neither the initial nor terminal vertex, is called an intermediate vertex.

The degree of a vertex relative to a graph is the number of edges incident to the vertex in the graph.

A simple path is a path for which

(1) The degree of the initial and terminal vertices is exactly one.

(2) The degree of intermediate vertices is exactly two.

A cycle is a path for which the initial and terminal vertices are coincident. A simple closed path is a cycle for which the degree of every vertex of the path is exactly two.

A connected graph is a graph for which a path exists between each pair of vertices. For simplicity of exposition, we will consider all graphs to be connected. However, certain subgraphs may not be connected; define a component as a maximal connected subgraph.

The incidence matrix **A** of a graph is defined as

$$a_{ij} = \begin{cases} 1, \text{ if edge } e_j \text{ is incident to vertex } v_i, \\ 0, \text{ otherwise.} \end{cases}$$

The ith row vector of matrix **A**, corresponding to vertex v_i, will be indicated $\mathbf{A}_i$.

Define a bipartite graph as a graph such that the vertices can be partitioned into two sets in such a way that each edge connects a vertex in one set to some vertex in the other set.

1.3 Graphical Matching and Covering Problem

Given a weighted graph, G, where c is a vector of edge weights, and **A** the incidence matrix of the graph, define a matching M in G as a subset of the edges such that no more than one edge of M is incident to any node of G. The maximum matching problem is to find a matching of maximum weight sum. This can be written as an integer program as:

$$\text{Max } cx \qquad \text{subject to } \mathbf{A}x \leq 1, \qquad x_{ij} = 0 \text{ or } 1,$$

where cx is a scalar inner product of vectors c and x.

Define a covering C in G as a subset of the edges such that each vertex is incident to at least one edge of the subset. The minimum covering problem is to find a covering of minimum weight sum; in integer program form:

$$\text{Min } cx \qquad \text{subject to } \mathbf{A}x \geq 1, \qquad x_{ij} = 0 \text{ or } 1.$$

Note that if the edge weights of G are all unity, the above problems correspond to finding the maximum cardinality matching and minimum cardinality covering respectively.

If a matching is also a covering for some graph G, it is called a perfect matching (or 1-factor).

1.4 Background

Graphical matching and covering problems have been studied for some time. In 1947, Tutte [16] gave necessary and sufficient conditions for the existence of a perfect matching in a given graph.

The maximum matching problem for the special case of a bipartite graph is known as the assignment problem. An efficient solution to this problem was obtained in 1955 by Kuhn [13]. An alternate characterization of bipartite graphs is that the graphs contain no odd cycles; it is precisely this property of odd cycles which is shown to cause difficulty in matching and covering problems in Chapters II and III respectively.

In 1957, Berge [3] gave necessary and sufficient conditions for a maximum cardinality matching; this result is stated as Theorem 2.1 in Chapter II. However, this result did not lead to an efficient algorithm.

In 1959, Norman and Rabin [15] gave necessary and sufficient conditions for a minimum cardinality covering, but again this result did not lead to any efficient algorithm. They also showed that the solution to the maximum cardinality matching problem implies the solution to the minimum cardinality covering problem, and conversely. This is demonstrated by Theorem 3.1 in Chapter III. In 1962, Edmonds [6] extended this work considerably, but without hint of an efficient algorithm.

In 1965, however, Edmonds [7] solved the maximum cardinality matching problem, and subsequently solved the more general maximum matching problem [8], giving an efficient algorithm for both.

1.5 Algorithm Growth

What is meant by an efficient algorithm? Since the number of edges of the graph is finite, one could always exhaustively list all possible matchings, and compare their weights. But this could not be done for any graph with a reasonably large number of nodes.

Let the number of nodes of a graph be N. Since we know that the number of edges of any graph cannot exceed $\frac{N(N-1)}{2}$, N seems to be a logical parameter by which to judge algorithm complexity.

The number of distinct matchings which exist for a graph of N nodes is on the order of $N!$. If Berge's necessary and sufficient condition [2] were implemented to find a maximum matching, the growth of such an implementation would be of exponential order, 2^N.

By growth of an algorithm we mean a good upper bound on the complexity of the algorithm, where complexity is measured by the time required to find a solution using a conventional computer with unlimited storage capacity.

An algorithm with exponential growth is probably not practical or useful, in that for reasonably large graphs, the computation time would be excessive. Rather, we should look for algorithms which possess algebraic growth with N, i.e., N^k, where k is a constant independent of the size of the graph.

For example, consider existing shortest path algorithms for weighted graphs; the best of such algorithms have growth on the order of N^2 or N^3. Less efficient algorithms exist for the solution of shortest path problems with exponential growth, but no one would ever use such an approach for a large graph.

Edmonds' algorithms [7,8] for the solution of the maximum matching problem possess algebraic growth, on the order of N^4. An analysis of this N^4 growth figure is presented in Appendix B, together with a computer program for Edmonds' algorithm.

Subsequent algorithms developed in this work possess comparable algebraic growth. For example, the growth of the minimum covering algorithm described in Chapter III is on the order of N^4, as compared to methods of Norman and Rabin [15] which grow exponentially with N.

1.6 Topical Contents of Thesis

Chapter II gives Edmonds' solution to the problem of maximum matching, as well as notation and techniques used throughout the text. Chapter III presents the solution to the minimum covering problem.

The maximum k-cardinality matching problem is to find a maximum matching of exactly k edges. A parametric approach is introduced in Chapter IV, and is applied to the maximum k-cardinality matching problem. In Chapter V, this parametric technique is applied to the minimum k-cardinality covering problem.

Chapter VI explores the relationships between matching and covering for a given graph. In Chapter VII, the relationship is established between the minimum spanning tree algorithm of Kruskal [12] and the algorithm of Chapter V. Some of the implications of this relationship are examined. Chapter VIII presents some engineering applications of these techniques and ideas for future research.

Appendix A gives some results from linear programming theory used throughout the text. Appendix B shows a computer program for solving the maximum cardinality matching problem, together with some computational experience with that program. Appendix C relates a minimum covering to the concept of a reducing path.

The primary contributions of this dissertation are:

1) an exposition of Edmonds' maximum matching algorithm, together with some suggested improvements of that algorithm,
2) an efficient algorithm to solve the minimum covering problem,
3) an algorithm to obtain maximum k-cardinality matchings,
4) an algorithm to obtain minimum k-cardinality coverings, and
5) the minimum k-cardinality covering algorithm is shown to be an interesting generalization of Kruskal's minimum spanning tree algorithm [12].

CHAPTER II

MAXIMUM MATCHING

2.1 Introduction

This chapter will characterize solutions to the maximum matching problem and describe efficient algorithms to find these solutions. Section 2.2 presents notation and motivation for the maximum cardinality matching algorithm developed by Edmonds [7], and Section 2.3 examines that algorithm. Appendix B gives a computer program implementation of the algorithm, and computational experience with the program.

Section 2.4 describes matching "blossoms". These are subgraphs related to noninteger solutions to the maximum matching problem. Section 2.5 discusses the weighted matching problem, and Section 2.6 presents Edmonds' algorithm [8] for its solution. A proof that this algorithm achieves the optimum solution is given in Section 2.7.

This chapter constitutes a review of Edmonds' algorithm, prepared in collaboration and also appearing in the dissertation of Robert Urquhart.

2.2 Maximum Cardinality Matching

The maximum cardinality matching problem was stated and solved by Edmonds [7]. The following definitions are with reference to a given matching M in a weighted graph G.

If no edge of M is incident to vertex v_j, then v_j is called an exposed vertex relative to matching M. An alternating path is a path whose edges are alternately in M and not in M.

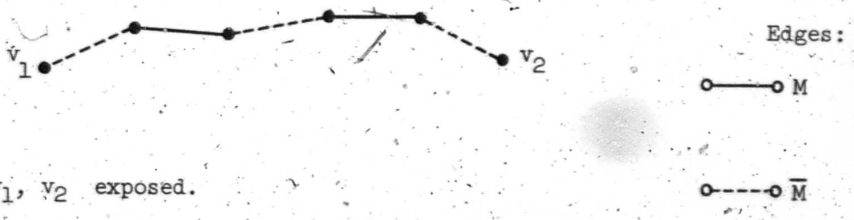
In order to indicate the operation of exchanging edges in a matching with those edges not in a matching along some alternating path P, the symmetric difference notation, $M \oplus P$, is used. If M is a matching, P an alternating path, then both are sets of edges, denote

$$M' = M \oplus P$$

as a new matching in the graph, where

$$M \oplus P = (M-P) \cup (P-M).$$

An augmenting path is an alternating path between two exposed vertices, relative to some matching in a graph. Such an augmenting path is shown in Figure 2.1.



Nodes v_1, v_2 exposed.

Figure 2.1 Augmenting Path

Berge [3] characterized maximum cardinality matchings by the use of augmenting paths.

Theorem 2.1 (Berge)

Given a matching M in a graph G. M is a maximum cardinality matching if and only if there exists no augmenting path in G.

Proof

Necessity

Suppose there exists an alternating path P in G. Form the matching $M' = M \oplus P$; but the cardinality of M' is larger than M since

$$|\overline{M} \cap P| > |M \cap P|.$$

Sufficiency

If M is not a maximum cardinality matching, then let M^* be any maximum cardinality matching. Examine the components of subgraph $M \oplus M^*$. Since both M and M^* are matchings, each component of $M \oplus M^*$ is an alternating path. Three types of paths are possible, as shown in Figure 2.2.

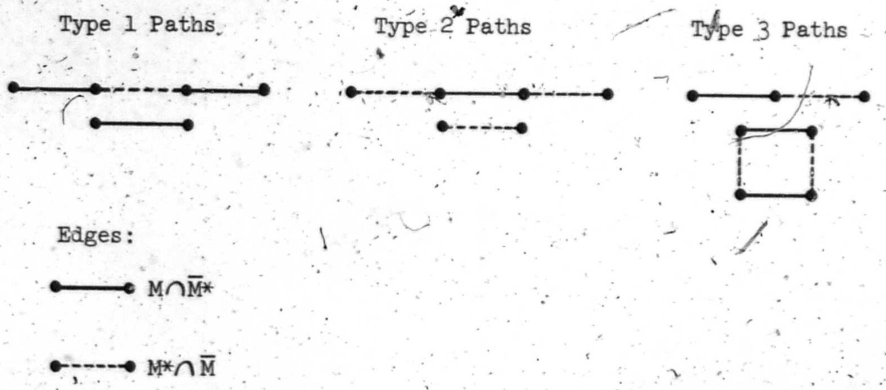


Figure 2.2 Components of Subgraph $M \oplus M^*$.

Since $|M^*| > |M|$, one of the components of $M \oplus M^*$ must have more edges of M^* than M, and hence that component is a type 2 path as in Figure 2.2. This demonstrates the existence of an augmenting path between two vertices exposed relative to matching M.

An alternating tree (Figure 2.3) relative to given matching M is a tree with the added conditions that:

(1) Exactly one vertex of the tree is exposed. This is called the root of the tree.

in graph G incident to an outer vertex are also incident to another vertex of the tree. A Hungarian tree is shown in Figure 2.5.

A simple blossom B with respect to a given matching M is a simple closed path consisting of $(2r + 1)$ edges and $(2r + 1)$ vertices, where r of the edges are in M. Thus there is one vertex of the simple blossom which is exposed relative to $M \cap B$.

When a simple blossom is found in the matching algorithm, it is shrunk to a simple vertex, thus forming a new graph. By shrinking simple blossoms in this manner, the concept of an alternating tree can be preserved in the resulting graph.

As a consequence of several such shrinkings, a vertex v_k in the resulting graph may be the image of several simple blossoms from previous graphs. If so, the preimage of v_k will be some odd set of vertices S_k in the original graph. The vertices S_k, together with all edges with both endpoints in S_k, will be called a blossom.

A blossom which is not contained within some other blossom will be called an outermost blossom. These blossoms will be particularly important in the weighted matching algorithm discussed in Section 2.5. A discussion of the structure of blossoms will be given in Section 2.4.

A blossomed tree is an alternating tree with one additional edge e_{12} which joins two outer vertices, v_1 and v_2, of the tree. Let P_1 be the path from v_1 to the root, and P_2 be the path from v_2 to the root; then $(P_1 \oplus P_2) \cup \{e_{12}\}$ is a blossom. This is shown in Figure 2.6.

2.3 Maximum Cardinality Matching Algorithm

The algorithm is initiated by specifying an initial matching (possibly empty), and proceeds by systematically searching for augmenting paths. Trees are rooted at an exposed vertex, and are grown by alternately adding edges in the matching and not in the matching as in Figure 2.3 until the tree either

(1) augments, as in Figure 2.4,

(2) blossoms, as in Figure 2.6, or

(3) becomes Hungarian, as in Figure 2.5.

See Figure 2.7 for a flow diagram of the algorithm.

If the tree augments, the cardinality of the matching can be increased by one. This is accomplished by tracing the augmenting path back to the root, and then interchanging the role of edges in the matching and not in the matching along that path. The tree is discarded following augmentation and a new tree is rooted at some remaining exposed vertex.

If the tree blossoms, the resulting blossom is identified by backtracing the two paths to the root. The blossom is then shrunk, the tree is retained, and further search for an augmenting path continues.

If the tree becomes Hungarian, then the maximum cardinality matching in this tree is independent of the remaining graph. This fact will be proved in Theorem 2.2.

The overall strategy of the algorithm is to either successively find augmenting paths, or to find Hungarian trees which can be eliminated from further consideration. Since each Hungarian tree eliminates one exposed vertex, and each augmentation eliminates two exposed vertices,

2.3 Maximum Cardinality Matching Algorithm

The algorithm is initiated by specifying an initial matching (possibly empty), and proceeds by systematically searching for augmenting paths. Trees are rooted at an exposed vertex, and are grown by alternately adding edges in the matching and not in the matching as in Figure 2.3 until the tree either

(1) augments, as in Figure 2.4,

(2) blossoms, as in Figure 2.6, or

(3) becomes Hungarian, as in Figure 2.5.

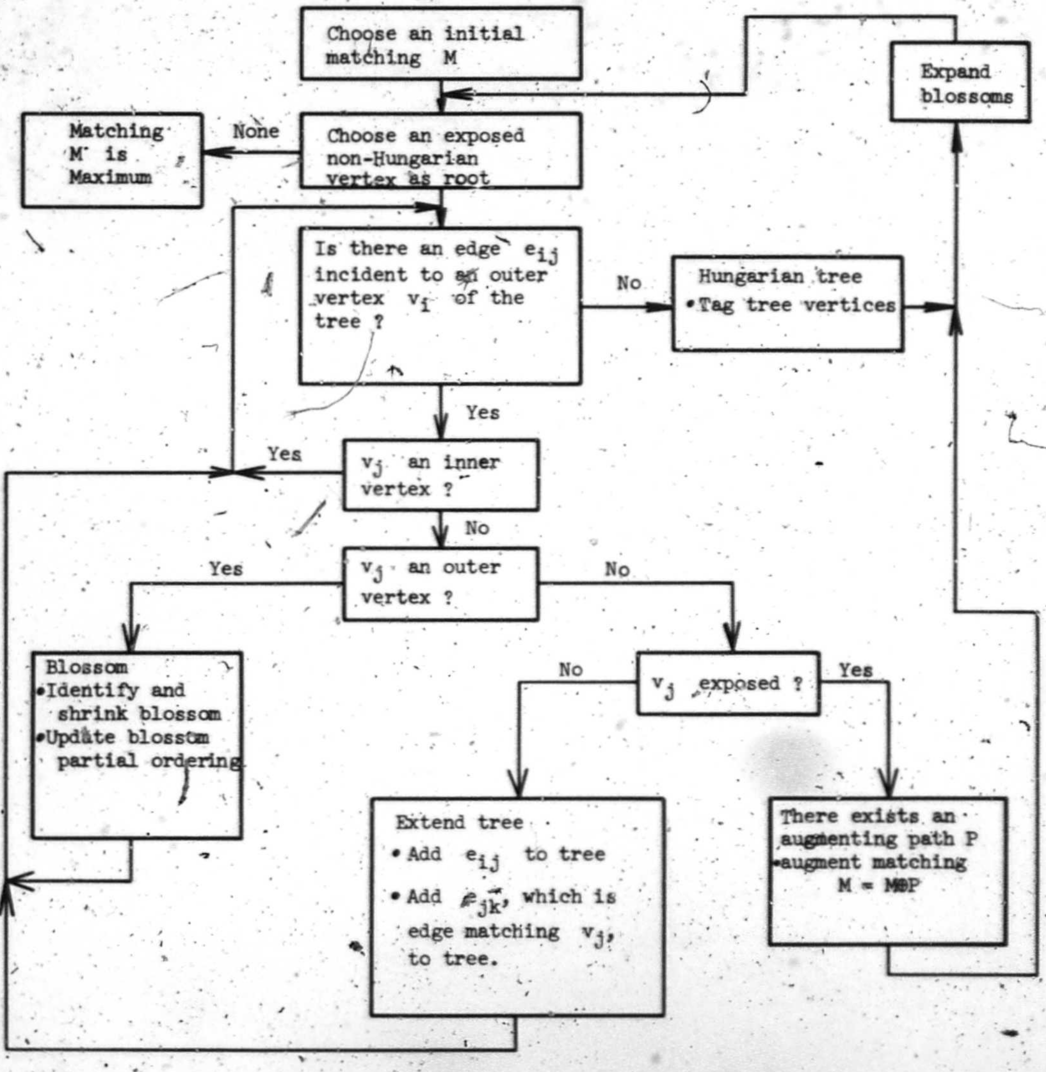
See Figure 2.7 for a flow diagram of the algorithm.

If the tree augments, the cardinality of the matching can be increased by one. This is accomplished by tracing the augmenting path back to the root, and then interchanging the role of edges in the matching and not in the matching along that path. The tree is discarded following augmentation and a new tree is rooted at some remaining exposed vertex.

If the tree blossoms, the resulting blossom is identified by backtracing the two paths to the root. The blossom is then shrunk, the tree is retained, and further search for an augmenting path continues.

If the tree becomes Hungarian, then the maximum cardinality matching in this tree is independent of the remaining graph. This fact will be proved in Theorem 2.2.

The overall strategy of the algorithm is to either successively find augmenting paths, or to find Hungarian trees which can be eliminated from further consideration. Since each Hungarian tree eliminates one exposed vertex, and each augmentation eliminates two exposed vertices,



Choose an initial matching M
Expand blossoms
Matching M is Maximum
None
Choose an exposed non-Hungarian vertex as root
Is there an edge e_{ij} incident to an outer vertex v_i of the tree ?
No
Hungarian tree
• Tag tree vertices
Yes
Yes
v_j an inner vertex ?
No
Yes
v_j an outer vertex ?
No
Blossom
• Identify and shrink blossom
• Update blossom partial ordering
No
v_j exposed ?
Yes
Extend tree
• Add e_{ij} to tree
• Add e_{jk}, which is edge matching v_j, to tree.
There exists an augmenting path P
• augment matching M = M⊕P

Figure 2.7 Maximum Cardinality Matching Algorithm

we can conclude that fewer than N' trees will have to be grown, where N is the number of vertices of the graph.

Edmonds computes an asymptotic upper bound on the growth of computation time with the number of nodes as N^4. This algorithm growth is examined in Appendix B, where a computer program for maximum cardinality matching is presented.

The following theorem proves that as a Hungarian tree forms relative to a particular matching M, the entire tree can be removed from further consideration without affecting the overall maximum cardinality matching.

Theorem 2.2

If J is a Hungarian tree in graph G with matching M_J, and M_{G-J} is any maximum cardinality matching of subgraph $(G-J)$, then $M_J \cup M_{G-J}$ is a maximum cardinality matching in G.

Proof

Let V_J be the set of vertices in the Hungarian tree J; partition the edges of G into three sets:

(1) E_J, the set of edges with both endpoints in V_J.

(2) $E_{J\bar{J}}$, the set of edges with one endpoint in V_J.

(3) E_{G-J}, the set of edges with neither endpoint in V_J.

Let K be any matching in G; then K can be partitioned relative to the edge partition above into

$$K = K_J \cup K_{J\bar{J}} \cup K_{G-J}.$$

Since M_{G-J} is a maximum cardinality matching in $(G-J)$,

1) $|M_{G-J}| \geq |K_{G-J}|$.

Every edge in $K_{J\bar{J}}$ touches exactly one inner vertex of J by definition of a Hungarian tree. Let $I(J)$ be the number of inner vertices of J. The removal of p inner vertices from the alternating tree J breaks it into $(p + 1)$ disjoint trees with a total of $[I(J)-p]$ inner vertices.

Since the maximum cardinality of any matching on an alternating tree is equal to the number of inner vertices, we have

$$I(J) - |K_{J\bar{J}}| \geq |K_J|$$

2) $|M_J| = I(J) \geq |K_J| + |K_{J\bar{J}}|$

Combining 1) and 2), we have $|M| \geq |K|$, but since K is an arbitrary matching we conclude M is of maximum cardinality.

2.4 Matching Blossoms

The maximum matching problem could be formulated as a linear programming problem except for the constraint of integrality. Removal of these constraints may result in a noninteger solution, as shown in Figure 2.8. The maximum solution without the integer constraint is $x_1 = x_2 = x_3 = x_3 = x_4 = x_5 = .5$, which has cost 2.5, while the largest cardinality of any matching in this graph is clearly 2.0.

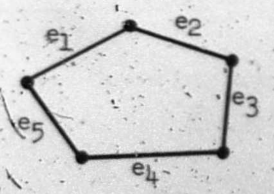


Figure 2.8 An Odd Cycle

The following theorem due to Balinski [1] shows that all vertices of the convex polyhedron specified by

$$Ax \leq 1, \quad x \geq 0,$$

have coordinate values 0, .5, or 1.

Theorem 2.3 (Balinski)

Any square nondecomposable submatrix of an incidence matrix of a graph has a determinant value 0, $\pm$ 1, or $\pm$ 2.

Since all such subdeterminants of matrix **A** are 0, $\pm$ 1, $\pm$ 2, any basis change encountered in a linear programming solution will produce the component values 0, .5, or 1.

The sources of noninteger solutions are odd cycles of the graph. If a graph contains no odd cycles, then it is a bipartite graph, and the maximum matching problem reduces to the classical assignment problem. Efficient solutions for this problem are well known [13].

Edmonds found that the following set of linear constraints were sufficient to guarantee integrality:

Let S_k be a set of $(2r_k + 1)$ vertices, and let R_k be a vector such that

$$R_{ki} = \begin{cases} 1, \text{ if edge } e_i \text{ has both endpoints in set } S_k \\ 0, \text{ otherwise.} \end{cases}$$

Then $R_k x \leq r_k$,

or if R is the odd vertex set-edge incidence matrix, this can be stated more generally as

$$Rx \leq r,$$

where r is the appropriate vector of odd vertex set indices. It should be noted that in the definition of a blossom in Section 2.2, the appropriate constraint in the above system is satisfied with equality, i.e.,

$$R_k x = r_k.$$

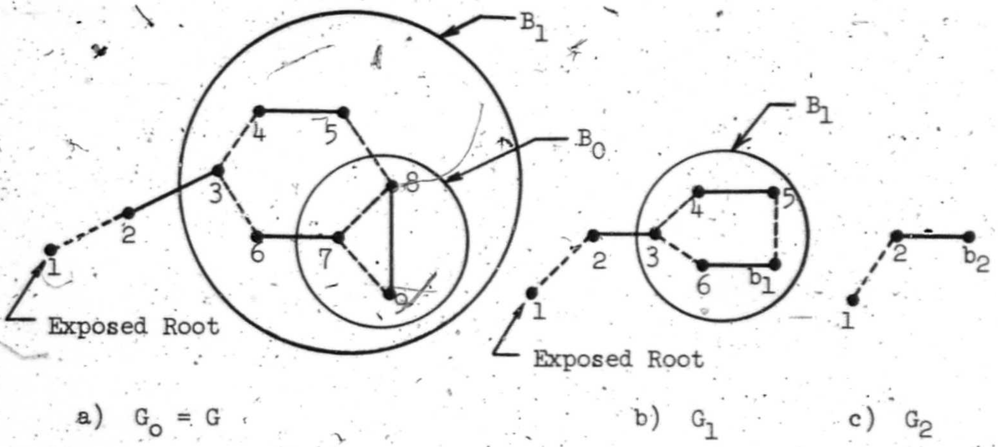
The sufficiency of these constraints to insure integrality will be proved constructively by the algorithm in Section 2.7.

When blossoms are detected in the algorithm, Edmonds shrinks them to a single vertex. This is a very convenient notation, in terms of retaining the alternating tree structure.

Notice that by our definitions of inner and outer vertices given in Section 2.2, each vertex within a blossom is both inner and outer. Edmonds suggests shrinking the vertices of a blossom to a single vertex, thus creating a new graph. The same result can be obtained by identifying which vertices are in a particular blossom, classifying them all as outer vertices, and continuing to grow trees using edges of the original graph. Thus the vertices of the blossom remain distinct, but edges between vertices of the same blossom are ignored in subsequent growth.

We will always talk of shrinking blossoms to a single vertex, thereby creating a new graph. That is, when a blossom B_i forms, we say B_i is shrunk to vertex b_{i+1} forming graph G_{i+1}, etc. We will not actually form G_{i+1}, but rather tag the vertices of B_i with the label b_{i+1}, and continue to use edges of $G_0 = G$ for subsequent growth.

In order to see some of the complexity which can arise in blossom structure, let us consider a case which may arise in the course of the algorithm. Suppose that after a blossom B_i has been formed and shrunk, another edge is found in the same tree which connects some vertex within blossom B_i to another outer vertex of the tree, as illustrated in Figure 2.9. This forms another blossom B_{i+1} which contains all the vertices of B_i along with a number of other vertices.



$S_o = \{7, 8, 9\}$, $P_o = (e_{78}, e_{89}, e_{97})$ in G_o

$S_1 = \{3, 4, 5, 6, 7, 8, 9\}$, $P_1 = (e_{34}, e_{45}, e_{5b_1}, e_{b_16}, e_{63})$ in G_1

Figure 2.9 Complex Blossom Structure

P_i is the simple closed path of the blossom B_i in graph G_i. Figure 2.9 a) depicts the blossoms B_o, B_1 in the original graph G, while Figures 2.9 b) and 2.9 c) show the effect of shrinking blossoms as they occur.

Assume that edge e_{78} was the last edge added to the tree.

Notice vertex v_4 was an inner vertex prior to the addition of e_{79}, but after that edge was added to the tree, v_4 is also an outer vertex. The path of even length from v_4 to the root v_1 is:

$$\{e_{45}, e_{58}, e_{89}, e_{97}, e_{76}, e_{63}, e_{32}, e_{21}\}.$$

In order to find the path of even length from any vertex within a blossom B_i back to the root, certain information is required about the structure of the tree when that blossom was formed. It is sufficient to remember the simple closed path P_k associated with each blossom B_k.

Notice that each subsequent blossom B_i is either disjoint from a previous blossom B_k, where $k < i$, or else B_i properly contains B_k. The odd node sets S_i, S_k induce a natural partial ordering on the blossoms.

$$S_i \supset S_k \Longrightarrow B_i > B_k.$$

This partial ordering is important when blossoms are subsequently expanded to induce the appropriate matching in G.

A vertex in graph G_i which is the image of some blossom structure can be treated as a simple vertex in G_i as far as tree growth and augmenting paths are concerned, but eventually the vertex must be expanded to obtain the actual matching in graph $G_0 = G$. The following theorem characterizes the freedom allowable in obtaining a matching in a complicated blossom structure.

<u>Theorem 2.4</u>

Given an odd set of $(2r_k + 1)$ nodes S_k, where S_k is the set of vertices of a blossom B_k in a graph G. If v is any vertex in S_k, then there exists a maximum cardinality matching in the subgraph on nodes S_k which leaves v exposed.

Proof

The blossom structure is expanded in any manner consistent with the partial ordering, i.e., B_i is expanded before B_j if and only if either $S_i \supset S_j$ or $S_i \cap S_j = \emptyset$. For an illustration of this process, one could refer to Figure 2.9.

At the first step, B_k is expanded by replacing B_k with the simple closed path P_k. One of the vertices of path P_k is either v or contains v. That vertex is left exposed, and a maximum cardinality matching is induced on the remaining vertices of P_k. But now we have the same situation relative to the vertex containing v as in the hypothesis, so eventually v is exposed in graph $G_0 = G$. At each step the vertices of P_i which do not contain v become matched.

If the vertex which is matched is a simple vertex, we are finished with it. If, however, it is the image of a blossom B_i, then the edge in the matching is actually incident to some vertex $v_k \epsilon S_i$, $v_k \neq v$. A maximum cardinality matching can be induced on the subgraph defined by vertices S_i by leaving the image of v_k exposed in this subgraph. Thus we return to the same situation as in the hypothesis, but with fewer blossoms. Since the number of blossoms is finite, the process eventually terminates with only the specified vertex v exposed.

2.5 Weighted Matching

The maximum matching problem in a weighted graph G can be formulated as an integer program:

1) Max cx subject to

2) $Ax \leq 1$

3) $x_{ij} = 0$ or 1.

As stated in Section 2.4, Edmonds replaced the explicit integrality constraints 3) by a set of linear constraints. The maximum matching problem can then be stated as a linear programming problem:

4) Max cx subject to

5) $Ax \leq 1$

6) $Rx \leq r$

7) $x \geq 0$

where R is an odd vertex set-edge incidence matrix of graph G. S_k is an odd set of $(2r_k + 1)$ vertices, and R_k is a vector such that

$$R_{ki} = \begin{cases} 1, & \text{if edge } e_i \text{ has both endpoints in set } S_k \\ 0, & \text{otherwise} \end{cases}$$

A typical constraint in 6) is

$$R_k x \leq r_k.$$

Where it is clear from context, we shall also use the symbol R_k to indicate that set of edges with both endpoints in vertex set S_k.

It can be seen that any matching satisfies 6). What is not clear is whether this set of constraints is <u>sufficient</u> to guarantee integrality. This proof will be given in Section 2.7.

Appendix A describes some results from the theory of linear programming, which will be needed here. By considering linear system 4), 5), 6), and 7) as a primal linear program, we can obtain the corresponding dual system as 8), 9), and 10).

8) $\text{Min } (\Sigma y_j + \Sigma r_k z_k)$

The dual variables y_j are summed over all vertices $v_j \in G$, and the products $r_k z_k$ are summed over all odd vertex sets S_k where $|S_k| = 2r_k + 1$. z_k is the dual variable associated with the odd vertex set S_k.

9) $A^T y + R^T z \geq c$

10) $y \geq 0, \ z \geq 0$

A typical dual equation in 9) is

11) $y_i + y_j + \sum_{e_{ij} \in R_k} z_k \geq c_{ij}$,

where R_k indicates the set of edges with both endpoints in odd vertex set S_k.

The algorithm is motivated by Theorem A.3 from Appendix A; Theorem A.3 guarantees optimality of the primal system if a primal-dual solution is both feasible and orthogonal. For this case, the primal-dual solution is a set of vectors x, y, z. A feasible set of such vectors must satisfy constraints 5), 6), 7), 9), and 10). The orthogonality conditions are

12) $x_{ij} > 0 \Longrightarrow y_i + y_j + \sum_{e_{ij} \in R_k} z_k = c_{ij}$,

13) $y_j > 0 \Longrightarrow A_j x = 1$,

i.e., vertex v_j is not exposed.

14) $z_k > 0 \Longrightarrow R_k x = r_k$,

i.e., an odd set of vertices S_k has a nonzero dual variable only if it is maximally matched using edges of R_k.

The algorithm, which will be described in Section 2.6, begins with and maintains feasible primal and dual variables, which are also orthogonal except for condition 13).

A feasible primal is the null matching, $x_{ij} = 0$, for all edges $e_{ij} \epsilon G$, which also satisfies condition 12). A feasible dual is obtained by setting $z_k = 0$ for all odd vertex sets S_k, and by choosing positive vertex variables y_j sufficiently large to satisfy conditions 9) and 10).

9) $y_i + y_j \geq c_{ij}$ for all $e_{ij} \epsilon G$.

10) $y \geq 0$

Condition 14) is satisfied, since all $z_k = 0$. Thus 13) is the only orthogonality condition which is not satisfied initially. We will show in Section 2.6 that every time a tree is grown, this tree eventually leads to a reduction in the number of vertices violating condition 13). Thus fewer than N trees are required.

As in the case of the maximum cardinality matching problem described in Section 2.2, we are searching for augmenting paths. These augmenting paths have a more complex structure than in the cardinality case. Define the weight $w(M)$ of a matching M as the sum of the edge weights of M, and the weight of an alternating path P relative to a matching M as

$$w(P \cap \overline{M}) - w(P \cap M).$$

A weighted augmenting path P relative to a matching M in a weighted graph G is an alternating path such that:

1) $M' = M \oplus P$ is a matching, and

2) $w(P \cap \overline{M}) - w(P \cap M) > 0$.

This definition leads to three different types of weighted augmenting paths P, shown in Figure 2.10, which are used throughout this thesis.

1) If $|P \cap \overline{M}| > |P \cap M|$, P is a strong augmenting path.

2) If $|P \cap \overline{M}| = |P \cap M|$, P is a weak augmenting path.

3) If $|P \cap \overline{M}| < |P \cap M|$, P is a deaugmenting path.

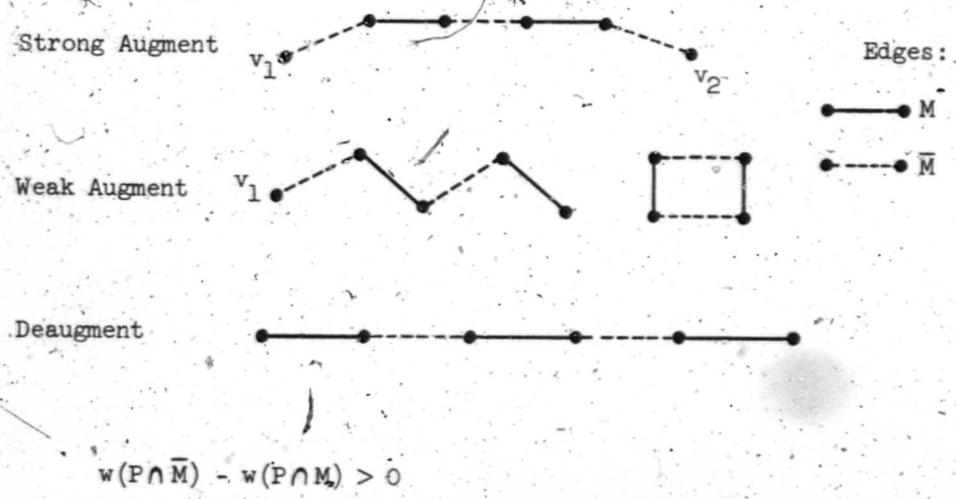


Figure 2.10 Weighted Augmenting Path Types

Theorem 2.5

A matching M is maximum in a weighted graph G if and only if there does not exist a weighted augmenting path in G.

Proof

Necessity

If a weighted augmenting path P exists in G, then form $M' = M \oplus P$. But since

$$w(P \cap \overline{M}) - w(P \cap M) > 0,$$

$$w(M') > w(M),$$

and M is not a maximum matching in G.

Sufficiency

Suppose M is not a maximum matching in G; then there exists some maximum matching M^* such that $w(M^*) > w(M)$.

Consider subgraph $M \oplus M^*$; each component of $M \oplus M^*$ is an alternating path. But since $w(M^*) > w(M)$, there exists at least one component P such that

$$w(M^* \cap P) > w(M \cap P).$$

Thus P is one of the three types of weighted augmenting paths shown in Figure 2.10.

2.6 Maximum Matching Algorithm

The maximum matching algorithm is based upon the primal and dual linear programs as formulated in Section 2.5. The algorithm is a generalization of the Hungarian method used by Kuhn [13] to solve the assignment problem.

Classify the vertices of G into four sets relative to an alternating tree:

1) Outer, $\emptyset$. This set contains all outer nodes in the tree.

2) Inner, I. This set contains all inner nodes in the tree.

3) Neutral, NU. This set contains nodes that are matched

and not in the tree.

4) Exposed, E. This set contains nodes that are exposed and not in the tree.

The maximum matching algorithm is shown in Figure 2.11, and can be described as follows:

(1) Begin the algorithm with no edges in the matching M, and sufficiently high vertex weights y_j such that $y \geq 0$, $y_i + y_j \geq c_{ij}$, for every edge $e_{ij} \epsilon G$. Set $z_k = 0$ for all odd vertex sets S_k.

(2) Define a subgraph G^*, which contains all the vertices of G and edge set E^*,

$$E^* = \{e_{ij} | y_i + y_j = c_{ij};\ v_i \text{ and } v_j \text{ not in the same blossom}\}.$$

(3) Choose a nonzero exposed node to root an alternating tree. The cardinality matching algorithm described in Section 2.2 is used to grow this alternating tree in subgraph G^*.

(4) A succession of blossoms may form, at which time all the vertices of the blossom are identified as outer vertices. Unlike the cardinality algorithm, blossoms formed in previous trees may appear as inner vertices.

(5) If a strong augmenting path occurs, it is used to augment the matching, and the tree is discarded.

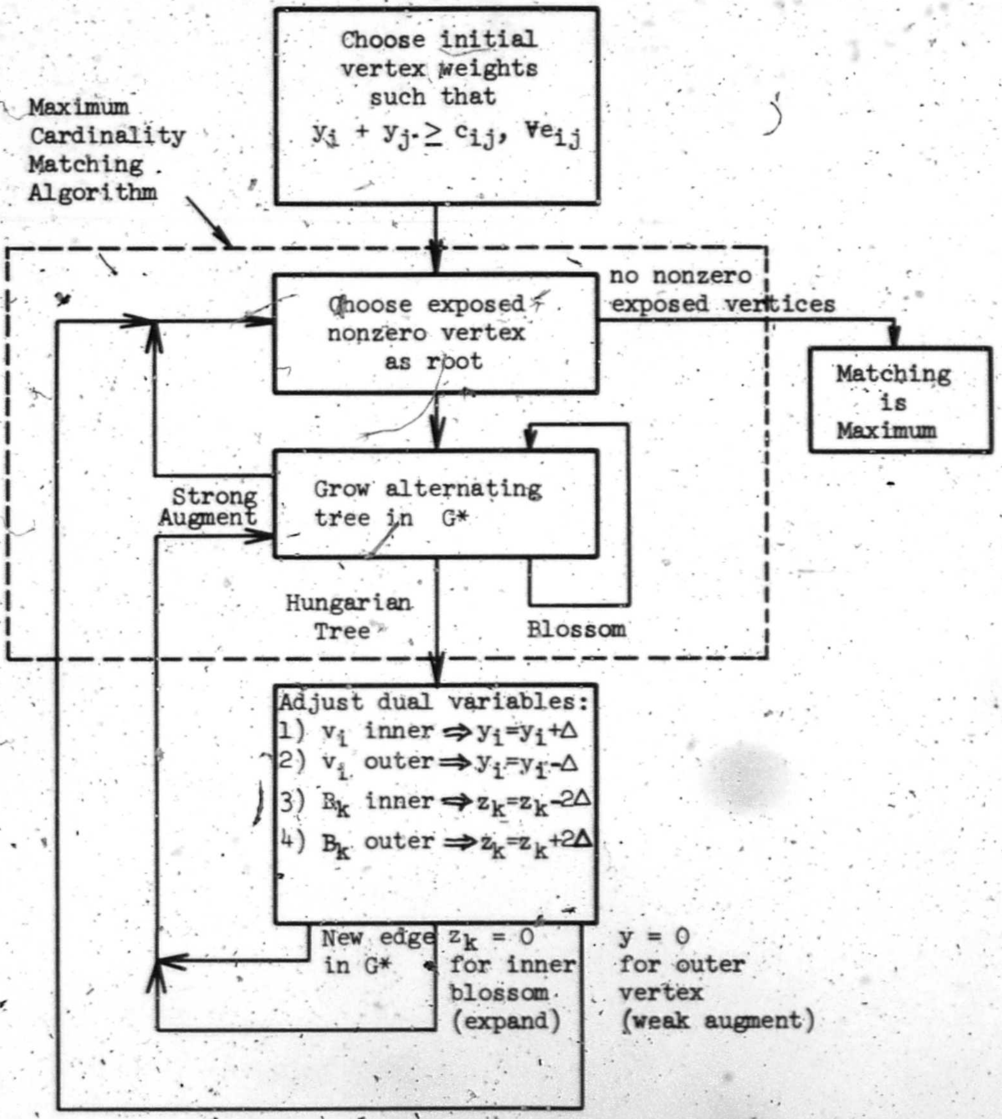

Choose initial vertex weights such that $y_i + y_j \geq c_{ij}, \forall e_{ij}$
Maximum Cardinality Matching Algorithm
Choose exposed nonzero vertex as root
no nonzero exposed vertices
Matching is Maximum
Grow alternating tree in G*
Strong Augment
Blossom
Hungarian Tree
Adjust dual variables:
1) v_i inner $\Rightarrow y_i = y_i + \Delta$
2) v_i outer $\Rightarrow y_i = y_i - \Delta$
3) B_k inner $\Rightarrow z_k = z_k - 2\Delta$
4) B_k outer $\Rightarrow z_k = z_k + 2\Delta$
New edge in G*
$z_k = 0$ for inner blossom (expand)
$y = 0$ for outer vertex (weak augment)

Figure 2.11 Maximum Matching Algorithm

(6) Eventually either an augmentation occurs, or the tree cannot be extended further, in which case the tree is called Hungarian as in Section 2.2. At this time leave the cardinality algorithm, and perform the following calculation:

$\Delta = \text{Min}\ (\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5)$ where

$$\Delta_1 = \underset{\substack{v_i \in \emptyset \\ v_j \in NU}}{\text{Min}} \{c_{ij} - y_i - y_j\}$$

$$\Delta_2 = \tfrac{1}{2}\underset{\substack{v_i \in \emptyset \\ v_j \in \emptyset}}{\text{Min}} \{c_{ij} - y_i - y_j\}$$

$\Delta_3 = \frac{1}{2}\text{Min}\ \{z_k\}$, over all outermost blossoms serving as inner nodes of the tree.

$$\Delta_4 = \underset{v_j \in \emptyset}{\text{Min}} \{y_j\}$$

$$\Delta_5 = \underset{\substack{v_i \in \emptyset \\ v_j \in E}}{\text{Min}} \{c_{ij} - y_i - y_j\}$$

Upon calculation of Δ, adjust dual variables:

a) Outer node weights y_j are decreased by Δ.

b) Inner node weights y_j are increased by Δ.

c) Dual variables z_k associated with outermost blossoms which serve as outer nodes of the tree are increased by 2Δ.

d) Dual variables z_k associated with outermost blossoms which appear as inner nodes of the tree are decreased by 2Δ.

(7) If $\Delta = \Delta_1$, a new edge enters subgraph G^*, and the tree can be grown further by returning to the cardinality algorithm.

(8) If $\Delta = \Delta_2$, a new edge enters subgraph G^* such that a new blossom forms. Return to the cardinality algorithm.

(9) If $\Delta = \Delta_3$, an outermost blossom B_k must be expanded, and the tree may be grown further by returning to the cardinality algorithm. In this case, the blossom B_k formed as an outer vertex in a previous tree has become involved in the present tree as an inner vertex. Further increase in Δ would have caused the dual variable z_k to become negative, and the dual solution would no longer be feasible. Thus the outermost blossom B_k is expanded, a matching induced, and the tree growth continued.

(10) If $\Delta = \Delta_4$, then y_j becomes zero for some outer vertex, and the tree is discarded. If this outer vertex should be the root of the tree, then clearly we have reduced the number of nonzero exposed vertices. If the weight of an outer vertex v_1, other than the root, becomes zero, we change the matching along the even path between v_1 and the root. This path is a weak augmenting path, since the cardinality of the matching is unchanged. Now the vertex v_1 becomes exposed. But since $y_1 = 0$, v_1

still satisfies the orthogonality condition 13), and we have succeeded in reducing the number of vertices violating 13). A diagram illustrating this change, as well as strong augments, is shown in Figure 2.12.

(11) If $\Delta = \Delta_5$, a strong augmenting path exists. The cardinality of the matching can be increased, and the tree discarded.

(12) If the tree is discarded, a new tree is rooted as in step (3). The net result is that each time a tree is rooted and grown, the number of vertices violating condition 13) of Section 2.5 is reduced. Thus after growing less than N trees, the orthogonality conditions are all satisfied, and the matching solution found is optimum.

Example

Find the maximum matching in the graph G shown in Figure 2.13.

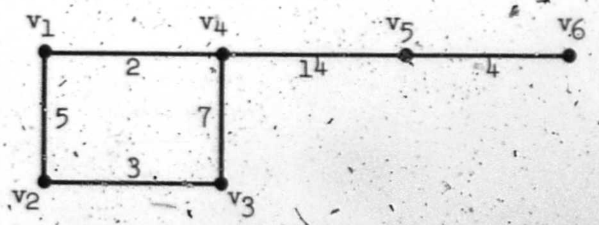


Figure 2.13 Weighted Graph G

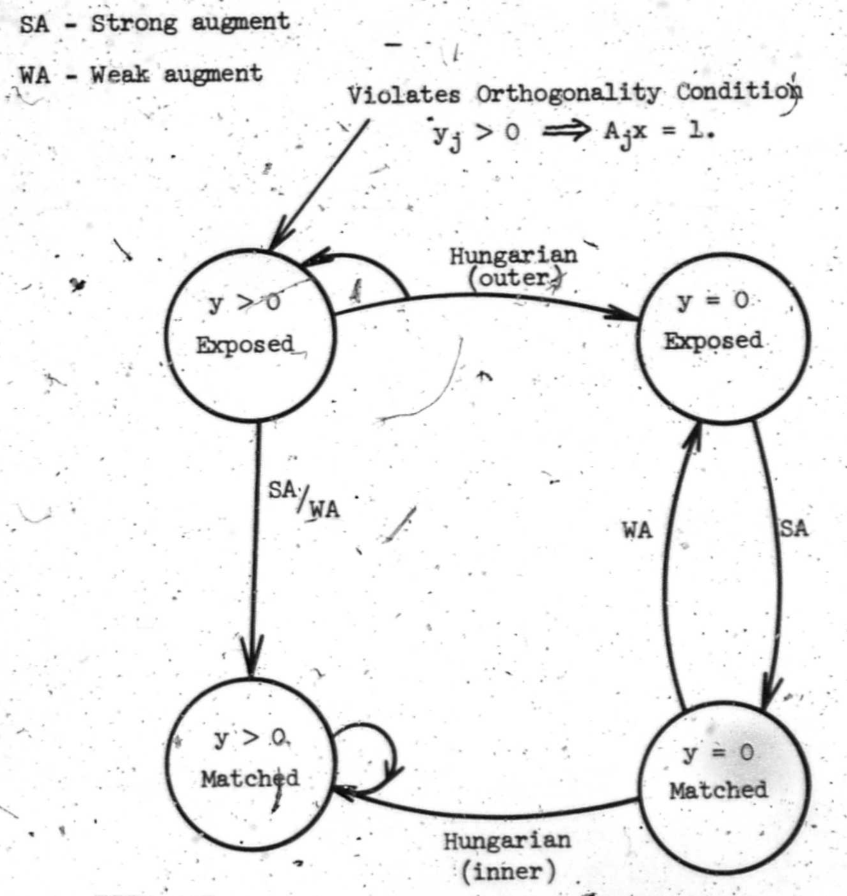


Figure 2.12 Vertex State Transition Diagram (Weighted Matching)

A valid set of dual variables is $y_j = 7$, for all $v_j \epsilon G$.

Tree 1

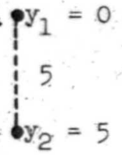
(1) Root at v_1; tree is Hungarian.

(2) Lower y_1 to $y_1 = 0$.

Tree 2

(1) Root at v_2; tree is Hungarian.

(2) Lower y_2 to $y_2 = 5$; grow edge e_{12}, augment.

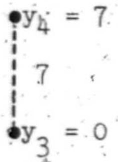


$M_1 = \{e_{12}\}$, $w(M_1) = 5$.

Tree 3

(1) Root at v_3; tree is Hungarian.

(2) Lower y_3 to $y_3 = 0$; grow edge e_{34}, augment.

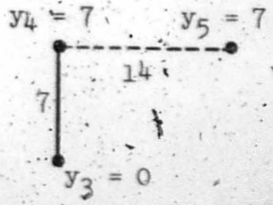


$M_2 = \{e_{12}, e_{34}\}$, $w(M_2) = 12$.

Tree 4

(1) Root at v_5; grow edges e_{54}, e_{43} in tree. Tree is then Hungarian.

(2) A weak augmenting path $P = \{e_{54}, e_{43}\}$ exists; implement the path P.



$M_3 = \{e_{12}, e_{54}\}$, $w(M_3) = 19$.

Tree 5

(1) Root at v_6; tree is Hungarian.

(2) Lower y_6 to $y_6 = 0$.

Thus M_3 is the maximum matching. The orthogonality conditions are now satisfied. As a check, compute:

$$W = cx = [5 + 14] = 19.$$

$$U = \Sigma\, y_j = [0 + 5 + 0 + 7 + 7 + 0] = 19.$$

2.7 Maximum Matching Algorithm Proof

The basis for the proof of the maximum matching algorithm was provided in Section 2.5. We must show that terminally the algorithm arrives at primal and dual variables which are both feasible and orthogonal, for then the solution is optimal by Theorem A.3 from Appendix A.

Thus we must show:

$$\left.\begin{array}{ll} 1) & x_{ij} \geq 0, \text{ for all } e_{ij} \in G \\ 2) & A_j x \leq 1, \text{ for all } v_j \in G \\ 3) & R_k x \leq r_k, \text{ for all odd} \\ & \qquad \text{vertex sets } S_k \end{array}\right\} \quad \text{Primal Feasibility}$$

$$\left.\begin{array}{ll} 4) & y_j \geq 0, \text{ for all } v_j \in G \\ 5) & z_k \geq 0, \text{ for all odd} \\ & \qquad \text{vertex sets } S_k \\ 6) & y_i + y_j + \displaystyle\sum_{e_{ij} \in R_k} z_k \geq c_{ij}, \\ & \qquad \text{for all } e_{ij} \in G \end{array}\right\} \quad \text{Dual Feasibility}$$

$$\left.\begin{array}{l} 7)\quad x_{ij} > 0 \Rightarrow y_i + y_j + \sum\limits_{e_{ij}\in R_k} z_k = c_{ij} \\ \\ 8)\quad y_j > 0 \Rightarrow A_j x = 1 \\ \qquad\qquad \text{(vertex } v_j \text{ not exposed)} \\ 9)\quad z_k > 0 \Rightarrow R_k x = r_k \\ \qquad\qquad \text{(vertex set } S_k \text{ maximally} \\ \qquad\qquad \text{matched by edge set } R_k\text{)} \end{array}\right\} \text{Orthogonality}$$

Since the algorithm solution is a matching, conditions 1), 2) and 3) are verified immediately. Since the algorithm always stops lowering vertex weights as an outer vertex y_j variable goes to zero, and all inner vertex weights are raised, 4) is always satisfied. z_k variables are always maintained nonnegative in the algorithm, so 5) is verified.

Our convergence argument was based on the systematic elimination of nonzero exposed vertices, thus orthogonality condition 8) is satisfied. The only way in which a nonzero z_k variable occurs is when a blossom forms, but then $R_k x = r_k$, and condition 9) follows. Thus the proof reduces to showing 6) and 7) are valid.

Proof of 6)

Let y_j^* be the final value of y_j when the algorithm terminates. If some edge e_{ij} does not belong terminally to any blossom edge set R_k, then the algorithm maintains the condition

$$y_i + y_j \geq c_{ij},$$

and 6) is verified immediately for this case.

But if edge e_{ij} is involved in a blossom structure, let y_i', y_j' be the weights of the endpoints of e_{ij} at the first time e_{ij} is absorbed into a blossom. Then at this point,

$$y_i' + y_j' \geq c_{ij},$$

and the final vertex weights are related

$$y_i^* = y_i' - \sum_{e_{ij} \in R_k} \tfrac{1}{2} z_k$$

$$y_j^* = y_j' - \sum_{e_{ij} \in R_k} \tfrac{1}{2} z_k .$$

Substituting this above, we obtain the general verification of condition 6),

$$6) \quad y_i + y_j + \sum_{e_{ij} \in R_k} z_k \geq c_{ij}.$$

Proof of 7)

If edge e_{ij} is in the matching, then it entered this set by forming a portion of an augmenting path in some tree. But its presence in the tree implies

$$y_i + y_j = c_{ij}.$$

Further, all edges in the matching which are not within a blossom, but in any tree are always between an inner and an outer vertex. Thus the Hungarian weight adjustment process maintains this equality. If the edge enters a blossom, then the proof of 6) can be used with the above equality holding instead of the inequality. Thus 7) is valid, and the algorithm has been shown to obtain an optimum solution.

In Appendix A, a convex polyhedron is defined, and it can be seen that the constraints

$$Ax \leq 1, \quad Rx \leq r, \quad x \geq 0,$$

form a convex polyhedron. We assumed a 0,1 integer solution vector x, corresponding to a matching, for an arbitrary weight vector c, and found this would always yield an optimum solution for this set of constraints. Thus by Theorems A.4 and A.5 from Appendix A, we conclude this convex polyhedron possesses only integer vertices. Since there are no noninteger vertices, this also proves the sufficiency of the linear constraints $Rx \leq r$ to eliminate noninteger vertices.

CHAPTER III

MINIMUM COVERING

3.1 Introduction

The minimum covering problem in a weighted graph G is to find a subset of edges of minimum weight such that each vertex is incident to at least one edge of the subset. Formulated as an integer problem:

$$\text{Min } cx \qquad \text{subject to } Ax \geq 1, \qquad x_{ij} = 0 \text{ or } 1.$$

This chapter will describe an algorithm which solves this problem. The algorithm is similar in many ways to that presented in Chapter II for solving the maximum matching problem.

Minimum cardinality coverings are investigated in Section 3.2. It is proved that the solution to the maximum cardinality matching problem implies the solution to the minimum cardinality covering problem.

Sections 3.3 and 3.4 describe and prove the algorithm for the special case of a bipartite graph, using the linear programming theory summarized in Appendix A.

The linear programming formulation for a general graph together with linear integrality constraints is given in Section 3.5. Section 3.6 discusses these integrality constraints in detail, and describes the topological nature of blossoms for the case of coverings. Section 3.7 presents the general algorithm, and Section 3.8 provides a proof of that algorithm using linear programming theory.

3.2 Minimum Cardinality Coverings

A star is a tree which has at most one vertex with degree greater than one. Any subgraph which consists of a union of disjoint stars is said to be a star subgraph. See Figure 3.1.

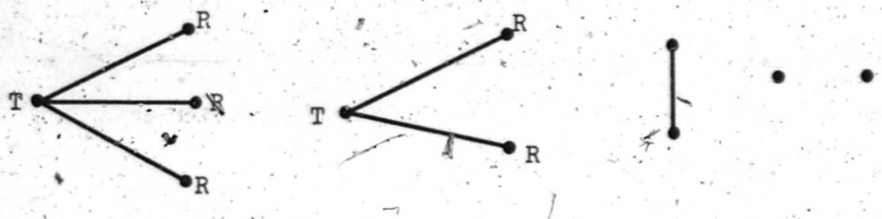


Figure 3.1 A Star Subgraph

A transmitter T of a star is defined as a vertex of degree greater than one. A receiver R is a vertex in a star which is adjacent to a transmitter. These definitions are unique to this thesis.

A forest is a subgraph of disjoint trees.

Lemma 3.1

A minimum cardinality covering in a graph G is a star subgraph.

Proof

A minimum cardinality covering contains no cycles, for otherwise some edge of the cycle could be removed, resulting in a covering of lower cardinality. Thus the minimum cardinality covering must be a forest.

But if the minimum cardinality covering is not a star subgraph, there must exist some path in a tree of the forest of length three or greater. Then there exist two adjacent vertices, both of degree two or greater. The edge between these two vertices can be removed, producing a covering of lower cardinality.

Let us consider two constructions which will produce a maximum cardinality matching from a minimum cardinality covering, and conversely.

Construction 1

Let C be a minimum cardinality covering in a graph G. From each component of C, arbitrarily choose one edge to be in a matching M.

Construction 2

Let M be a maximum cardinality matching in a graph G. Add one edge incident to each exposed vertex to obtain a covering C.

Theorem 3.1

Given a graph G, construction 2 produces a minimum cardinality covering C from a maximum cardinality matching M.

Proof.

Suppose there are a total of N nodes in graph G. By construction 2, $|C| = |M| + (N - 2\,|M|) = N - |M|$.

Now consider any minimum cardinality covering C^*, which must be a star subgraph by Lemma 3.1. From these component stars of C^*, extract a matching M^* by construction 1. But $|M^*| \leq |M|$, since M is assumed a maximum cardinality matching.

Then $|C^*| = |M^*| + (N - 2\,|M^*|) = N - |M^*|$,

by virtue of the star composition of C^*.

Hence $|C| \leq |C^*|$, and C is a minimum cardinality covering.

Corollary 3.1

Given a graph G, construction 1 produces a maximum cardinality matching M from a minimum cardinality covering C.

Theorem 3.1 also shows that covering C produced by construction 2 is a star subgraph, as shown in Figure 3.2. Note that any transmitter vertex must be an endpoint of an edge from matching M.

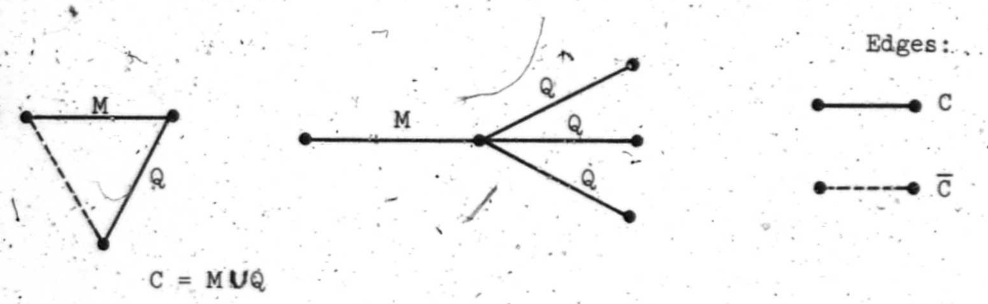


Figure 3.2 Minimum Cardinality Covering by Construction 2.

3.3 Minimum Covering Algorithm for a Bipartite Graph

For simplicity of exposition, let us consider the special case of a weighted bipartite graph G in the discussion and proof of the minimum covering algorithm. Since there is no difficulty with integrality, the linear program can be written:

Primal	Min cx	subject to $Ax \geq 1$, $x \geq 0$.
Dual	Max Σy_j	subject to $A^T y \leq c$, $y \geq 0$.

The algorithm will be similar to that presented in Section 2.6, and is motivated by Theorem A.3 in Appendix A. Assign nonnegative weights y_j to all nodes v_j. Define v_j to be a saturated vertex if there exists an edge e_{ij} incident to y_j such that

$$y_j = c_{ij}, \quad y_i = 0.$$

A vertex v_j is unsaturated if for all edges e_{ij} incident to v_j,

$$y_j < c_{ij}, \quad y_j \geq 0, \quad \text{and} \quad y_i + y_j \leq c_{ij}.$$

The algorithm is divided into two phases:

1) phase 1, in which a matching is obtained, and

2) phase 2, where edges are added to form a covering.

Unlike the case of matching discussed in Chapter II, the primal constraints are infeasible during phase 1 of the algorithm, and not until phase 2 terminates are these primal constraints satisfied.

During phase 1 of the algorithm, we need to classify the vertices into five sets relative to an alternating tree:

1) outer, Ø

2) inner, I

3) neutral, NU

4) exposed saturated, ES

5) exposed unsaturated, EU

An outer vertex is at the end of a path of even length from the root of the tree, and an inner vertex is at the end of a path of odd length from the root. A neutral vertex is incident to some edge of the matching, but not in the tree currently under construction. An exposed node is not incident to any edge of the matching, and is not in the tree. The meaning of classifications exposed saturated and exposed unsaturated should be evident.

A flow graph of the algorithm is given in Figure 3.3, and proceeds as follows:

1) Begin phase 1 with no edges in the matching C, and vertex weights y_j such that $y \geq 0$. $y_i + y_j \leq c_{ij}$, for every edge $e_{ij} \epsilon G$.

2) Define a subgraph G^* which contains all the vertices of G and edge set E^*,

   $E^* = \{e_{ij} | y_i + y_j = c_{ij}\}$.

   Choose an exposed unsaturated node to root an alternating tree. Search for an augmenting path through the growth of this tree in G^*.

3) If the tree becomes Hungarian as described in Section 2.6, perform the following calculation:

   $\Delta = \text{Min}\ (\Delta_1, \Delta_2, \Delta_3)$,

   $$\Delta_1 = \underset{\substack{v_i \epsilon \emptyset \\ v_j \epsilon NU}}{\text{Min}} \{c_{ij} - y_i - y_j\}$$

   $$\Delta_2 = \tfrac{1}{2} \underset{\substack{v_i \epsilon \emptyset \\ v_j \epsilon I}}{\text{Min}} \{c_{ij} - y_i - y_j\}$$

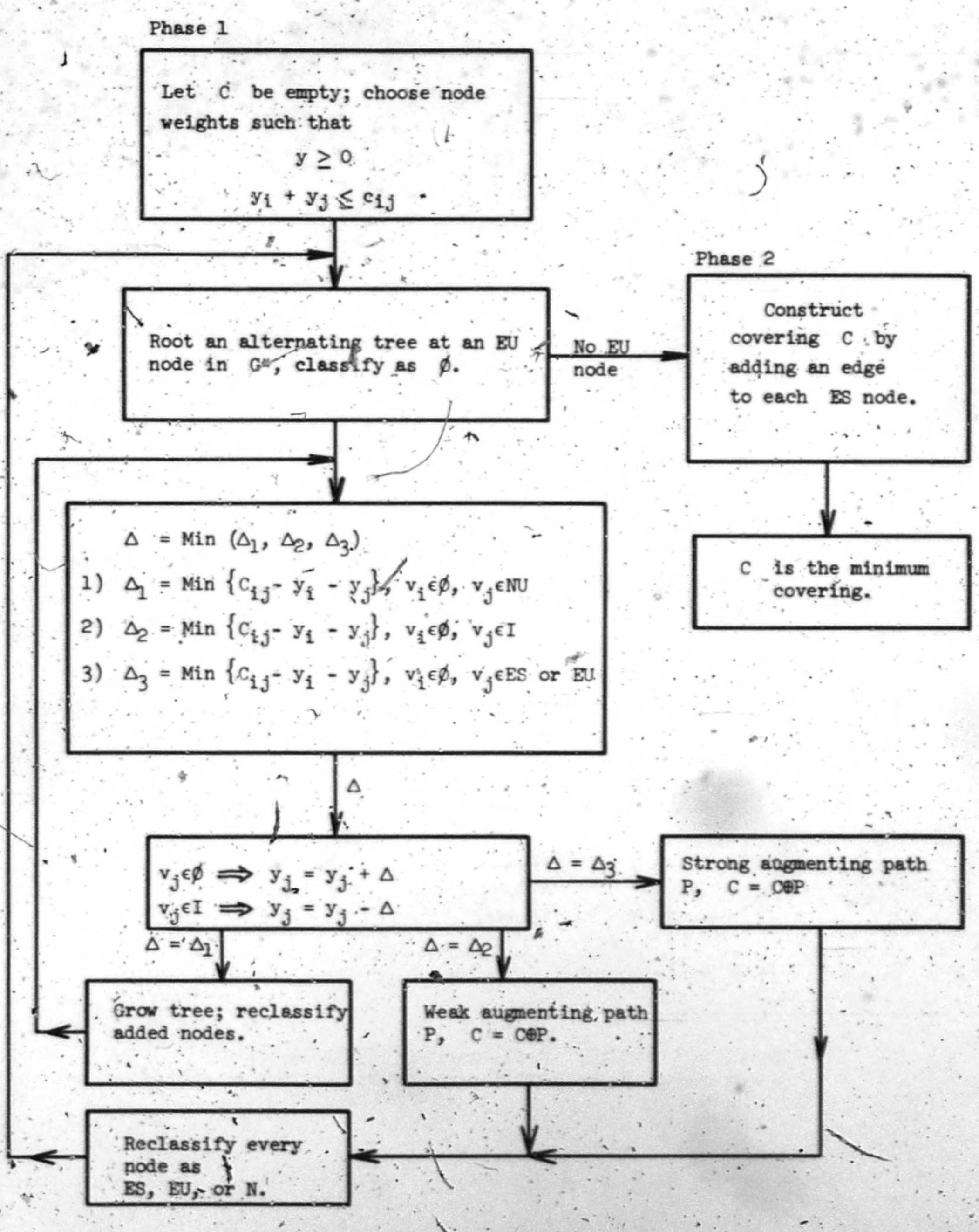

Phase 1
Let C be empty; choose node weights such that
$y \geq 0$
$y_i + y_j \leq c_{ij}$
Root an alternating tree at an EU node in $G^\#$, classify as $\emptyset$.
No EU node
Phase 2
Construct covering C by adding an edge to each ES node.
C is the minimum covering.
$\Delta = \text{Min}(\Delta_1, \Delta_2, \Delta_3)$
1) $\Delta_1 = \text{Min}\{c_{ij} - y_i - y_j\}$, $v_i \epsilon \emptyset$, $v_j \epsilon NU$
2) $\Delta_2 = \text{Min}\{c_{ij} - y_i - y_j\}$, $v_i \epsilon \emptyset$, $v_j \epsilon I$
3) $\Delta_3 = \text{Min}\{c_{ij} - y_i - y_j\}$, $v_i \epsilon \emptyset$, $v_j \epsilon ES$ or EU
Δ
$v_j \epsilon \emptyset \Rightarrow y_j = y_j + \Delta$
$v_j \epsilon I \Rightarrow y_j = y_j - \Delta$
$\Delta = \Delta_3$
Strong augmenting path P, $C = C \oplus P$
$\Delta = \Delta_1$
$\Delta = \Delta_2$
Grow tree; reclassify added nodes.
Weak augmenting path P, $C = C \oplus P$.
Reclassify every node as ES, EU, or N.

Figure 3.3 Algorithm for Minimum Covering in a Bipartite Graph

$$\Delta_3 = \min_{\substack{v_i \in \emptyset \\ v_j \in ES \text{ or } EU}} \{c_{ij} - y_i - y_j\}$$

Upon calculation of Δ, adjust dual variables:

a) Outer node weights are increased by Δ.

b) Inner node weights are decreased by Δ.

4) If $\Delta = \Delta_1$, a new edge enters subgraph G^*. Continue the growth of the tree.

5) If $\Delta = \Delta_2$, an outer node v_i becomes saturated, and some inner node weight y_j becomes zero. That is, in

$y_i + y_j = c_{ij}$,

$y_i = c_{ij}$, $y_j = 0$.

Expose this outer node v_i by implementing a weak augmenting path to the root of the tree, thus covering the root. Discard the tree.

6) If $\Delta = \Delta_3$, a strong augmenting path exists. The cardinality of the matching can be increased, and the tree discarded.

7) If the tree is discarded, a new tree is rooted as in step 2). Continue until all nodes are either matched or saturated.

8) Phase 2: Complete the covering C by adding an edge from each exposed saturated vertex to some zero-weighted vertex.

This algorithm converges because each tree reduces the number of exposed unsaturated vertices by at least one, so that no more than N trees need be grown. The convergence arguments are similar to those given in Section 2.6 for the case of matchings. A vertex state transition diagram is shown in Figure 3.4. Prior to the rooting of a new tree, the possible states of a vertex are EU, ES, or N. The diagram illustrates that intermediate Hungarian weight adjustments, as well as operations which terminate the subsequent tree growth, i.e., strong augments and weak augments, tend to reduce the number of EU vertices.

SA - Strong Augment	EU - Exposed Unsaturated
WA - Weak Augment	ES - Exposed Saturated
	N - Neutral

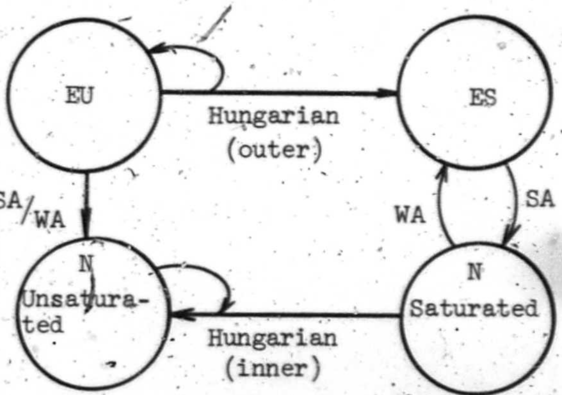


Figure 3.4 Vertex State Transition Diagram (Weighted Covering)

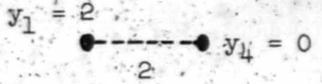
Example

Find the minimum covering in the graph G shown in Figure 3.5. A valid set of dual variables is $y_j = 0$, for all $v_j \in G$.

Tree 1

(1) Root at v_1; tree is Hungarian.

(2) Raise y_1 to $y_1 = 2$; grow edge e_{14}, augment.



$C_1 = \{e_{14}\}$, $w(C_1) = 2$.

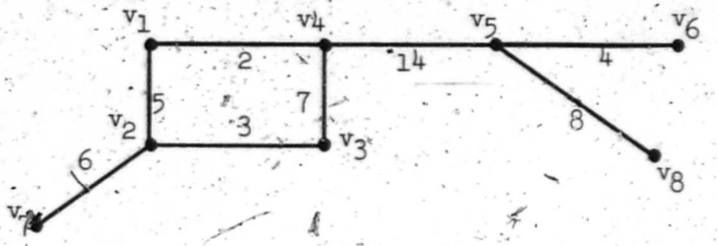
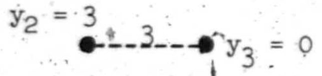


Figure 3.5 Weighted Graph G

Tree 2

(1) Root at v_2; tree is Hungarian.

(2) Raise y_2 to $y_2 = 3$; grow edge e_{23}, augment.

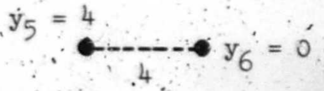


$C_2 = \{e_{14}, e_{23}\}$, $w(C_2) = 5$.

Tree 3

(1) Root at v_5; tree is Hungarian.

(2) Raise y_5 to $y_5 = 4$; grow edge e_{56}, augment.



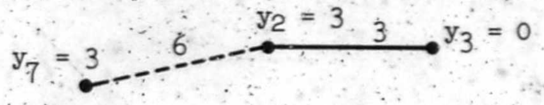
$C_3 = \{e_{14}, e_{23}, e_{56}\}$, $w(C_3) = 9$.

Tree 4

(1) Root at v_7; tree is Hungarian.

(2) Raise y_7 to $y_7 = 3$; grow edges e_{72}, e_{23}.

(3) Adjust weights; $y_7 = 6$, $y_2 = 0$, $y_3 = 3$. Discard tree.

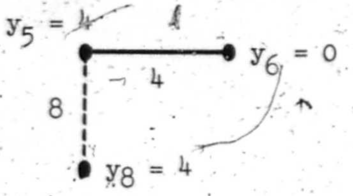


Tree 5

(1) Root at v_8; tree is Hungarian.

(2) Raise y_8 to $y_8 = 4$; grow edges e_{85}, e_{56}.

(3) Adjust weights; $y_8 = 8$, $y_5 = 0$, $y_6 = 4$. Discard tree.



Since all nodes are either covered by C_3 or saturated, the orthogonality conditions are satisfied. Now induce a covering from C_3 by adding edges e_{72} and e_{85}, both from saturated nodes to zero-weight nodes.

$$C_4 = \{e_{14}, e_{23}, e_{56}, e_{72}, e_{85}\}, \quad w(C_4) = 23.$$

C_4 is then the minimum covering of graph G. As a check, compute:

$$W = cx = [3 + 2 + 4 + 6 + 8] = 23.$$

$$U = \Sigma\, y_j = [2 + 0 + 3 + 0 + 0 + 4 + 6 + 8] = 23.$$

3.4 Proof of Algorithm for Bipartite Graph

The linear program for the bipartite graph case from Section 3.3 is:

Primal	Min cx	subject to $Ax \geq 1$, $x \geq 0$.
Dual	Max $\Sigma\, y_j$	subject to $A^T y \leq c$, $y \geq 0$.

Writing out the primal and dual feasibility and orthogonality conditions explicitly,

<u>Feasibility</u>

$$\left.\begin{array}{ll} 1) & A_j x \geq 1, \text{ for all } v_j \epsilon G \\ 2) & x_{ij} \geq 0, \text{ for all } e_{ij} \epsilon G \end{array}\right\} \text{Primal}$$

$$\left.\begin{array}{ll} 3) & y_i + y_j \leq c_{ij}, \text{ for all } e_{ij} \epsilon G \\ 4) & y_j \geq 0, \text{ for all } v_j \epsilon G \end{array}\right\} \text{Dual}$$

<u>Orthogonality</u>

5) $x_{ij} > 0 \Longrightarrow y_i + y_j = c_{ij}$.

6) $y_j > 0 \Longrightarrow A_j x = 1$.

From the linear programming theory presented in Appendix A, the covering solution is minimum if we can show that the algorithm of Section 3.3 attains conditions 1) through 6).

Condition 2) is always valid. Although the solution is primal infeasible during phase 1, condition 1) is terminally satisfied since phase 2 of the algorithm attains a covering.

When an edge e_{ij} enters covering C, condition

$$y_i + y_j = c_{ij}.$$

holds, and as long as that edge belongs to C, 5) is not violated.

Since C is terminally a covering, 6) states that every transmitter in C must have $y_j = 0$. Since edges are added from exposed nodes to vertices of zero weight in phase 2, 6) is valid.

This leaves conditions 3) and 4) to be verified. No inner node weight is ever lowered below zero. If some inner node weight became zero, then some outer or exposed node becomes saturated, and either a weak or strong augmenting path exists. Thus 4) is valid. The Hungarian weight adjustment maintains condition 3) as node weights change.

3.5 Linear Programming Formulation and Integrality

The minimum covering problem in a general weighted graph G is:

$$\text{Min } cx \qquad \text{subject to } Ax \geq 1, \ x_{ij} = 0 \text{ or } 1.$$

It is desired to add linear constraints in order to drop the explicit demand of integrality. In covering, as in matching, the source of difficulty in this regard is odd cycles. Consider the graph in Figure 3.6 with unity edge weights.

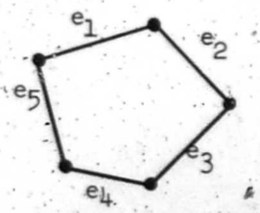


Figure 3.6 Odd Cycle Graph

The minimum solution to $Ax \geq 1$, $x \geq 0$, is 2.5, whereas a minimum integral covering has weight 3.0.

As in Section 2.4, there exist appropriate linear constraints which preclude noninteger solutions for these odd cycle configurations.

For every subset S_k of $(2r_k + 1)$ vertices in G, where r_k is a strictly positive integer,

$$\sum_{e_{ij}\epsilon T_k} x_{ij} \geq (r_k + 1)$$

where T_k is the set of edges having at least one endpoint in S_k.

It is clear that this constraint is satisfied by integer covering solutions. The sufficiency of the constraint to preclude noninteger solutions is difficult to show, and requires a proof using linear programming duality theory. The following modified linear program incorporating these constraints will be shown to yield only integer solutions.

Primal

$$\text{Min}\quad cx \qquad \text{subject to} \begin{bmatrix} A \\ T \end{bmatrix} x \begin{matrix} \geq \\ \geq \end{matrix} \begin{pmatrix} 1 \\ r+1 \end{pmatrix}, \quad x \geq 0,$$

where T is the matrix:

$$t_{ki} = \begin{cases} 1, & \text{if edge } e_i \text{ has at least one endpoint in the odd set } S_k \text{ of } (2r_k + 1) \text{ vertices} \\ 0, & \text{otherwise} \end{cases}$$

and where r_k is a positive integer.

Dual

$$\text{Min}\quad (\Sigma\, y_j + \Sigma\, (r_k + 1)\, z_k) \qquad \text{subject to}$$

$$A^T y + T^T z \leq c, \quad y \geq 0, \quad z \geq 0,$$

where

y_j corresponds to vertex v_j, and

z_k corresponds to odd node set S_k.

3.6 Linear Integrality Constraints and Blossoms

Blossoms for the matching problem were defined and discussed in Section 2.4. Many of the comments relative to matching blossoms are applicable to covering blossoms. A covering blossom B_k is defined as the odd set of vertices S_k, together with all edges with both endpoints in S_k. The corresponding integrality constraint from Section 3.5, $T_k x \geq (r_k + 1)$, is satisfied with equality, relative to some covering C.

A blossom may have one of two topological forms relative to a covering C as shown in Figure 3.7.

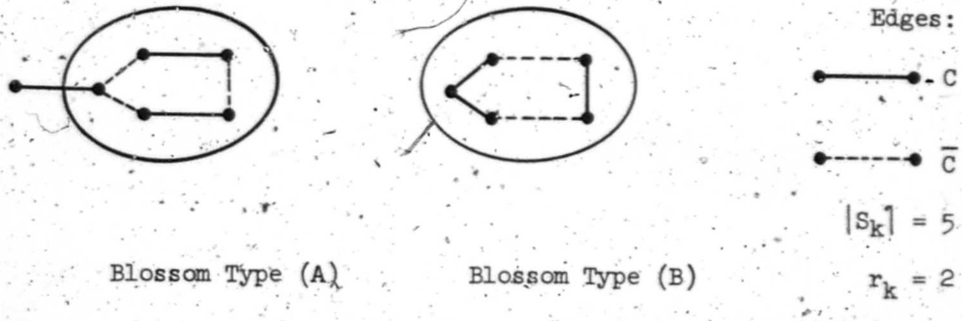


Figure 3.7 Covering Blossom Types

Note that in each case, an integrality constraint is satisfied with equality.

A type (A) blossom is formed in the same way as the matching blossoms of Chapter II. Moreover, a covering is induced on a blossom of this type exactly as described in Theorem 2.4.

Blossom type (B) arises when the dual variable y_j becomes zero for some vertex v_j within a blossom. Therefore v_j can become a transmitter without violating the orthogonality condition

$$y_j > 0 \Longrightarrow A_j x = 1.$$

Given a blossom which contains a vertex for which $y_j = 0$, we induce a covering such that v_j is a transmitter, and every other vertex within the blossom is incident to only one edge of the covering.

(1) Successively expand the blossom, inducing a matching as explained in Theorem 2.4 until v_j appears as a simple vertex in blossom B_k.

(2) Cover B_k as a type (B) blossom with v_j as the associated transmitter.

(3) Expand all remaining blossoms, and induce matchings as in Theorem 2.4.

The algorithm description in the next section will describe the various conditions which result in type (A) or type (B) blossoms.

3.7 Minimum Covering Algorithm

The algorithm to find a minimum covering in a general weighted graph is complicated by the existence of odd cycles. It is odd cycles which necessitate the consideration of blossoms. Unlike the algorithm of Section 3.3 for the case of bipartite graphs, it is not convenient to assign the dual variable y_j as the weight of vertex v_j. We shall associate a weight w_j to each vertex v_j. This vertex weight is related to the dual variable y_j by

$$y_j = w_j - \sum_{v_j \in S_k} z_k.$$

The algorithm consists of two phases, and is shown in the flow graph in Figure 3.8.

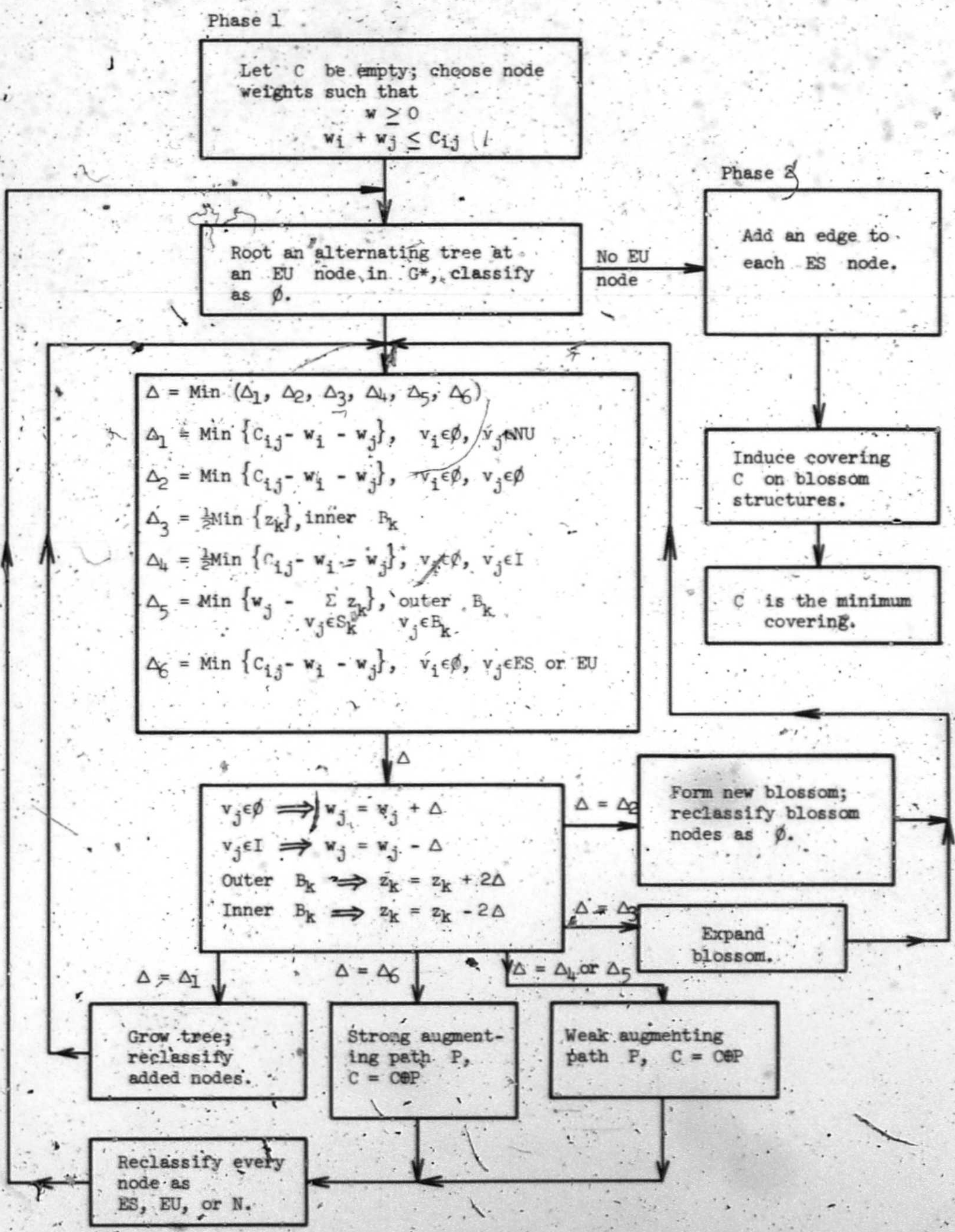

Phase 1
Let C be empty; choose node weights such that $w \geq 0$, $w_i + w_j \leq C_{ij}$
Root an alternating tree at an EU node in G*, classify as $\emptyset$.
No EU node
Phase 2
Add an edge to each ES node.
Induce covering C on blossom structures.
C is the minimum covering.
$\Delta = \text{Min}(\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5, \Delta_6)$
$\Delta_1 = \text{Min}\{C_{ij} - w_i - w_j\}, \quad v_i \in \emptyset, \; v_j \in NU$
$\Delta_2 = \text{Min}\{C_{ij} - w_i - w_j\}, \quad v_i \in \emptyset, \; v_j \in \emptyset$
$\Delta_3 = \frac{1}{2}\text{Min}\{z_k\}$, inner B_k
$\Delta_4 = \frac{1}{2}\text{Min}\{C_{ij} - w_i - w_j\}, \quad v_i \in \emptyset, \; v_j \in I$
$\Delta_5 = \text{Min}\{w_j - \sum_{v_j \in B_k} z_k\}_{v_j \in S_k}$, outer B_k
$\Delta_6 = \text{Min}\{C_{ij} - w_i - w_j\}, \quad v_i \in \emptyset, \; v_j \in ES$ or EU
Δ
$v_j \in \emptyset \Rightarrow w_j = w_j + \Delta$
$v_j \in I \Rightarrow w_j = w_j - \Delta$
Outer $B_k \Rightarrow z_k = z_k + 2\Delta$
Inner $B_k \Rightarrow z_k = z_k - 2\Delta$
$\Delta = \Delta_2$
Form new blossom; reclassify blossom nodes as $\emptyset$.
$\Delta = \Delta_3$
Expand blossom.
$\Delta = \Delta_1$
$\Delta = \Delta_6$
$\Delta = \Delta_4$ or Δ_5
Grow tree; reclassify added nodes.
Strong augmenting path P, $C = C \oplus P$
Weak augmenting path P, $C = C \oplus P$
Reclassify every node as ES, EU, or N.

Figure 3.8 Algorithm for Minimum Covering in a Weighted Graph

a) Phase 1: The algorithm finds successively higher cardinality matchings until all vertices not in blossoms are either matched or saturated.

b) Phase 2: An edge is added from each exposed saturated vertex to its associated zero-weight vertex. A final covering is induced on all blossom structures.

Let us describe this algorithm in greater detail:

(1) Begin phase 1 with no edges in the matching C, and vertex weights w_j such that

$w \geq 0$, and

$w_i + w_j \leq c_{ij}$ for every edge $e_{ij} \in G$.

Set $z_k = 0$ for all odd vertex sets S_k. Define a subgraph G^* which contains all the vertices of G, and edge set E^*:

$E^* = \{e_{ij} | w_i + w_j = c_{ij};\ v_i$ and v_j not in the same blossom$\}$.

(2) Choose an exposed unsaturated node to root an alternating tree. Search for an augmenting path by growing this alternating tree in G^*.

(3) If an edge is found between two outer vertices, form and identify the blossom as in Chapter II. Reclassify all nodes in the blossom as outer.

(4) If the tree becomes Hungarian, perform the following calculation:

$\Delta = \text{Min}\ (\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5, \Delta_6)$

$$\Delta_1 = \min_{\substack{v_i \in \emptyset \\ v_j \in NU}} \{c_{ij} - w_i - w_j\}$$

$$\Delta_2 = \tfrac{1}{2}\min_{\substack{v_i \in \emptyset \\ v_j \in \emptyset}} \{c_{ij} - w_i - w_j\}$$

$\Delta_3 = \frac{1}{2}\text{Min}\,\{z_k\}$, over all outermost blossoms B_k serving as inner nodes of the tree.

$$\Delta_4 = \tfrac{1}{2}\min_{\substack{v_i \in \emptyset \\ v_j \in I}} \{c_{ij} - w_i - w_j\}$$

$\Delta_5 = \text{Min}\,\{w_j - \sum_{v_j \in S_k} z_k\}$, over all nodes v_j in blossom serving as outer nodes of the tree.

$$\Delta_6 = \min_{\substack{v_i \in \emptyset \\ v_j \in EU \text{ or } ES}} \{c_{ij} - w_i - w_j\}$$

Upon calculation of Δ, the weight adjustments are as follows:

a) Outer node weights w_j are increased by Δ.

b) Inner node weights w_j are decreased by Δ.

c) Dual variables z_k associated with outermost blossoms which serve as outer nodes of the tree are increased by 2Δ.

d) Dual variables z_k associated with outermost blossoms which serve as inner nodes of the tree are decreased by 2Δ.

(5) If $\Delta = \Delta_1$, a new edge enters subgraph G^*, and the tree can be grown further.

(6) If $\Delta = \Delta_2$, an edge between outer vertices of the tree enters subgraph G^*. This forms a blossom which must be identified, and all nodes within this blossom are classified as outer. The tree can be grown further.

(7) If $\Delta = \Delta_3$, an outermost blossom B_k must be expanded and the tree grown further. In this case, the blossom B_k has an associated dual variable z_k which goes to zero.

(8) If $\Delta = \Delta_4$, then some outer node of the tree becomes saturated, and an inner node weight becomes zero. A weak augmenting path can be implemented, and the tree is discarded.

(9) If $\Delta = \Delta_5$, some blossom serving as an outer node of the tree contains a vertex v_j such that

$$W_j = \sum_{v_j \in S_k} z_k .$$

A weak augmenting path to the root is implemented.

(10) If $\Delta = \Delta_6$, an edge from an outer node to an exposed node (either EU or ES) enters subgraph G^*. This provides a strong augmenting path, and the tree is discarded.

(11) If the tree is discarded, a new tree is rooted as in Step (2). Since each tree reduces the number of exposed unsaturated nodes, fewer than N trees need be grown.

(12) At the beginning of phase 2, all vertices not in a blossom are either matched or saturated. An edge is added from each saturated exposed vertex not in a blossom to an associated vertex of zero weight. These zero weight vertices will serve as transmitters in the covering C.

(13) There exist three different classifications of blossoms. Induce a covering on each blossom as indicated below.

a) One edge of the matching is incident to some vertex in the blossom from phase 1. For this case, successively expand the blossom structure, and induce a covering as explained for a type (A) blossom in Section 3.6.

b) There exists a vertex in the blossom which is saturated relative to some vertex which is not in a blossom. Add the edge between these vertices. Then expand the blossom structure as in case a).

c) There exists a vertex v_j in the blossom such that $w_j = \sum_{v_j \in S_k} z_k$, which implies $y_j = 0$.

Use the construction for type (B) blossoms as described in Section 3.6 to induce a covering.

The set of edges C obtained by phases 1 and 2 of the algorithm form a minimum covering of the graph G. This will be proved in Section 3.8.

3.8 Proof of Minimum Covering Algorithm

The linear programming formulation of the covering problem was given in Section 3.5. Writing out the primal and dual feasibility and orthogonality conditions explicitly:

Feasibility

$$\left.\begin{array}{l} 1)\ A_j x \geq 1, \text{ for all } v_j \epsilon G \\ 2)\ x_{ij} \geq 0, \text{ for all } e_{ij} \epsilon G \\ 3)\ T_k x \geq (r_k + 1), \text{ for all odd node sets } S_k \end{array}\right\} \text{Primal}$$

$$\left.\begin{array}{l} 4)\ y_i + y_j + \sum\limits_{e_{ij} \epsilon T_k} z_k \leq c_{ij}, \text{ for all } e_{ij} \epsilon G \\ 5)\ y_j \geq 0, \text{ for all } v_j \epsilon G \\ 6)\ z_k \geq 0, \text{ for all } S_k \end{array}\right\} \text{Dual}$$

Orthogonality

7) $x_{ij} > 0 \Rightarrow y_i + y_j + \sum\limits_{e_{ij} \epsilon T_k} z_k = c_{ij}$.

8) $y_j > 0 \Rightarrow A_j x = 1$.

9) $z_k > 0 \Rightarrow T_k x = (r_k + 1)$.

Identify

10) $y_j = w_j - \sum\limits_{v_j \epsilon S_k} z_k$.

Using the results of Appendix A, if the algorithm achieves these conditions, then C is a minimum covering. This would also prove that the added linear constraints 3) introduced in Section 3.5 are sufficient to guarantee integrality.

Since C is a covering, 1), 2), and 3) are valid. Since

$z_k \geq 0$ in the algorithm, condition 6) holds. Definition

$$10)\ y_j = w_j - \sum_{v_j \in S_k} z_k$$

shows 5) holds, since by definition of a type (B) blossom structure, the algorithm always maintains

$$w_j \geq \sum_{v_j \in S_k} z_k,$$

together with the fact that $w_j \geq 0$.

Condition 8) is satisfied, since we only create transmitters at vertices for which $y_j = 0$. During the algorithm, $z_k > 0$ only for blossoms. Since blossoms satisfy the constraints $T_k x = (r_k + 1)$, 9) is verified.

Prior to the time an edge e_{ij} is absorbed into a blossom structure,

$$w_i + w_j \leq c_{ij}.$$

Let w_i', w_j' represent node weights at the time e_{ij} is absorbed into a blossom (if ever), and y_j^* be the final value of y_j when the algorithm terminates. Then

$$w_i' + w_j' \leq c_{ij},$$

$$y_i^* = w_i' - \sum_{e_{ij} \in T_k} \tfrac{1}{2} z_k$$

$$y_j^* = w_j' - \sum_{e_{ij} \in T_k} \tfrac{1}{2} z_k$$

By substituting these variables in the above inequality, we obtain a verification of condition

$$4)\ y_i + y_j + \sum_{e_{ij} \in T_k} z_k \leq c_{ij}.$$

Since an edge in the covering must satisfy $w_i' + w_j' = c_{ij}$ as it is absorbed in a blossom, this equality is maintained, and condition 7) follows similarly.

3.9 Summary

This chapter has investigated various forms of the minimum covering problem.

In Section 3.2, the minimum cardinality covering and maximum cardinality matching problems were shown equivalent. The problems were equivalent in the sense that one solution could be found from another by a simple construction.

An algorithm for obtaining a minimum covering in a bipartite graph was given in Sections 3.3 and 3.4. Here the principal features and motivations of this approach were examined without the complications of blossoms.

In Sections 3.7 and 3.8, the general minimum covering algorithm was presented. The growth of this algorithm can be shown to be N^4. This is the same order of complexity as for the maximum matching algorithm. However, the description of the minimum covering algorithm is more involved due to the increased complexity of covering blossoms.

The minimum covering algorithm will be used in Chapter V, and applications of this technique discussed in Chapter VIII.

CHAPTER IV

A PARAMETRIC APPROACH AND MAXIMUM MATCHING

4.1 Introduction

An extension of maximum matching discussed in Chapter II is the problem of obtaining a maximum k-cardinality matching in a weighted graph G:

$$\text{Max } cx \qquad \text{subject to } Ax \leq 1, \quad \sum_{e_{ij}\in G} x_{ij} = k, \; x_{ij} = 0 \text{ or } 1.$$

As will be explained in Chapter VIII, this extension was motivated by consideration of possible applications of maximum matchings.

The method of attack is to transform the weighted graph $[G, c_{ij}]$ to the graph $[G_\lambda, c_{ij} - \lambda]$ by subtracting λ from each edge weight of G. This will be called the parametric approach. There are two other ways to view this process.

1) From the viewpoint of linear programming, λ may be considered as a dual variable corresponding to the primal constraint

$$\sum_{e_{ij}\in G} x_{ij} \leq k,$$

2) Alternately λ may be viewed as a Lagrange multiplier, since

$$\text{Max}\left[\sum_{e_{ij}\in G}(c_{ij}x_{ij}) + \lambda\left(k - \sum_{e_{pq}\in G} x_{pq}\right)\right] = \text{Max}\left[\sum_{e_{ij}\in G}(c_{ij}-\lambda)x_{ij} + \lambda k\right].$$

This technique of using Lagrange multipliers to solve integer optimization problems is developed by Everett [11].

In Section 4.2, this parametric approach is explored for the maximum k-cardinality problem. It is shown that an appropriate value for the Lagrange multiplier λ always exists.

A direct algorithm is developed in Section 4.3, which utilizes this Lagrange multiplier concept. This direct algorithm solves the maximum k-cardinality matching problem, and also indicates the solution to the maximum matching problem. Section 4.3 develops this direct algorithm for the special case of bipartite graphs, while Section 4.4 applies the algorithm to general graphs.

A geometric interpretation of the parametric approach is given in Section 4.6. Section 4.7 shows the maximum matching values are concave with respect to cardinality k. An extension of the parametric technique to minimum matchings is indicated in Section 4.8.

4.2 The Parametric Method

Given a weighted graph $[G, c_{ij}]$, construct a graph $[G_\lambda, c_{ij} - \lambda]$ by subtracting λ from the weight of each edge. If M is a matching in G, let $w(M)$ represent the weight of that matching in G. Define $w_\lambda(M)$ to be the weight of matching M in graph G_λ. The maximum matching in graph G is in general different from the maximum matching in graph G_λ, as seen in Theorem 4.1.

Theorem 4.1

For a given λ, suppose that a maximum matching M in graph G_λ has cardinality k. Then M is a maximum k-cardinality matching in G.

Proof

Let M' be any k-cardinality matching in G. Then for any λ,

$w_\lambda(M) = w(M) - k\lambda$,

$w_\lambda(M') = w(M') - k\lambda$.

Thus

$w_\lambda(M) \geq w_\lambda(M')$ implies

$w(M) \geq w(M')$.

Since M' was any k-cardinality matching in G, M must be a maximum k-cardinality matching in G as claimed.

Theorem 4.1 suggests that we might be able to solve the maximum k-cardinality matching problem in G by using the maximum matching algorithm in G_λ. There exists a question of whether the cardinality of the maximum matching in G_λ is monotonic relative to λ. This question is answered in Lemma 4.1.

Lemma 4.1

Given λ_1, λ_2, such that $\lambda_2 > \lambda_1$. Let M_1 be a maximum matching in G_{λ_1}, where $|M_1| = k_1$, and M_2 a maximum matching in G_{λ_2}, where $|M_2| = k_2$. Then $k_1 \geq k_2$.

Proof

By assumption,

$w_{\lambda_1}(M_1) \geq w_{\lambda_1}(M_2)$, and

$w_{\lambda_2}(M_2) \geq w_{\lambda_2}(M_1)$.

Expanding,

$w(M_1) - k_1\lambda_1 \geq w(M_2) - k_2\lambda_1$,

$w(M_2) - k_2\lambda_2 \geq w(M_1) - k_1\lambda_2$.

Adding these inequalities, and rearranging yields,

$$k_1(\lambda_2 - \lambda_1) \geq k_2(\lambda_2 - \lambda_1).$$

But since $\lambda_2 > \lambda_1$ was assumed, we have the desired result

$$k_1 \geq k_2.$$

Though the cardinality k of the maximum matching in G_λ has been shown monotonic with λ by Lemma 4.1, not all values of k may be obtained as continuous values of λ are examined. Everett [11] refers to this problem as "gaps" in the Lagrange multiplier method. This means that many solutions correspond to the same value of the Lagrange multiplier. The following Theorem 4.2 shows that by the use of alternating paths, we can overcome this difficulty. Specifically, by the use of the parametric method, we can always find a maximum k-cardinality matching.

Theorem 4.2

Let m be the maximum cardinality of any matching in graph G. Given $k \leq m$, there exists a λ such that some maximum matching in G_λ is of k-cardinality.

Proof

We will prove this theorem by the construction of such a matching. Since k is monotonic with λ by Lemma 4.1, the only problem is when "gaps" occur as follows:

Suppose there exists a λ for which M_2 is a maximum matching in G_λ, where $|M_2| = k_2$, and M_1 is a maximum matching in $G_{(\lambda + 1)}$, where $|M_1| = k_1$. Further, suppose that k given in the hypothesis is such that

$$k_1 < k < k_2.$$

The components of $M_1 \oplus M_2$ in G_λ are alternating paths. Partition these alternating paths into one of three sets relative to the cardinality of M_1 and M_2 edges in the path. Define these paths as strong augments, deaugments, and weak augments as shown in Figure 4.1.

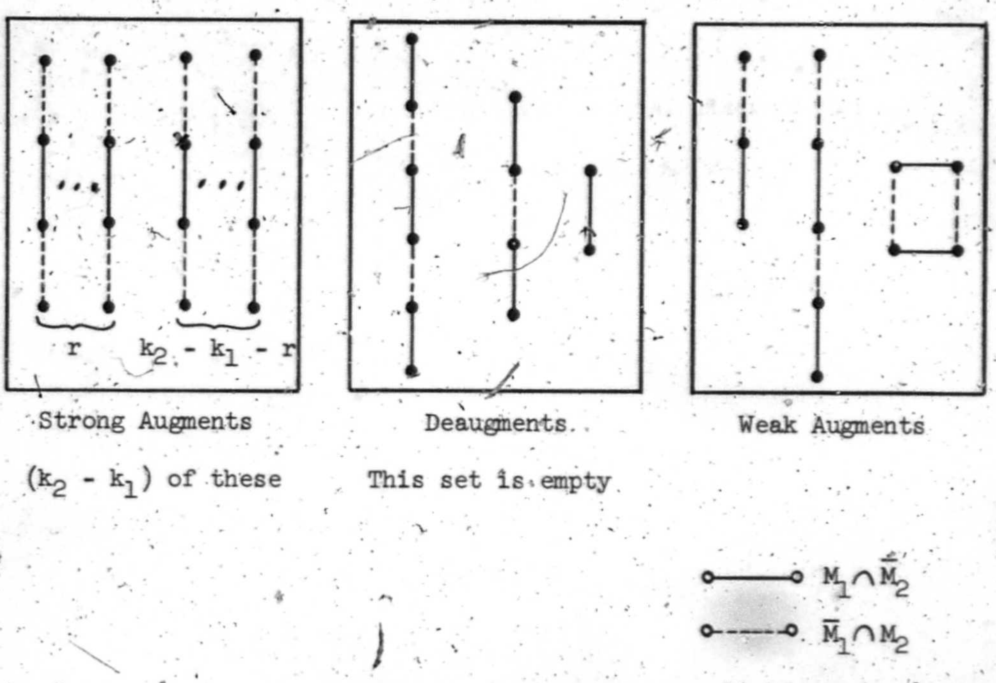


Figure 4.1 Alternating Paths in $M_1 \oplus M_2$

Let us consider each of these sets of paths in G_λ:

<u>(1) Deaugments</u>

There exist no deaugments in G_λ. To see this, note that for any path P in this set,

$$w_\lambda(P \cap M_1) - w_\lambda(P \cap M_2) \geq 0,$$

for otherwise this would contradict the assumption that M_1 is a maximum matching in G_λ. But if the above inequality holds, examine its implication in $G_{(\lambda + 1)}$:

$$w_{\lambda + 1}(P \cap M_1) - w_{\lambda + 1}(P \cap M_2) \geq 1.$$

But this violates the assumption that M_2 is a maximum matching in $G_{(\lambda + 1)}$.

<u>(2) Weak Augments</u>

For any path P in this set,

$$w_\lambda(P \cap M_1) - w_\lambda(P \cap M_2) = 0.$$

If this relationship were a strict inequality, it would violate either the assumption that M_1 is maximum in G_λ, or the assumption that M_2 is maximum in $G_{(\lambda + 1)}$.

<u>(3) Strong Augments</u>

Since it has been shown that there are no deaugments, there must be exactly $(k_2 - k_1)$ strong augments. For any strong augment P,

$$w_\lambda(P \cap M_1) - w_\lambda(P \cap M_2)$$

cannot be negative, as this would contradict the assumption that M_1 is a maximum matching in G_λ. Yet the inequality

$$w_\lambda(P \cap M_1) - w_\lambda(P \cap M_2) > 1$$

in G_λ implies that in graph $G_{(\lambda + 1)}$,

$$w_{\lambda + 1}(P \cap M_1) - w_{\lambda + 1}(P \cap M_2) > 0,$$

which violates the assumption that M_2 is a maximum matching in $G_{(\lambda + 1)}$. Therefore, if we consider only integer weights in graph G, for any strong augment P,

$$w_\lambda(P \cap M_1) - w_\lambda(P \cap M_2) = 0 \text{ or } 1.$$

Now if the theorem is true, there must exist a matching M which is maximum in either graph G_λ or $G_{(\lambda + 1)}$, where the cardinality of M is k.

Suppose r strong augments P_1 are such that

$$w_\lambda(P_1 \cap M_1) - w_\lambda(P_1 \cap M_2) = 0,$$

as shown in Figure 4.1. Then $(k_2 - k_1 - r)$ strong augments P_2 are such that

$$w_\lambda(P_2 \cap M_1) - w_\lambda(P_2 \cap M_2) = +1.$$

If $r \geq (k - k_1)$, the required matching M can be constructed by implementing $(k - k_1)$ of the strong augments P_1.

If $r < (k - k_1)$, there exist more than $(k_2 - k)$ strong augments P_2 such that

$$w_\lambda(P_2 \cap M_1) - w_\lambda(P_2 \cap M_2) = +1,$$

which implies

$$w_{\lambda+1}(P_2 \cap M_1) - w_{\lambda+1}(P_2 \cap M_2) = 0.$$

The required matching M can be constructed by implementing $(k_2 - k)$ of the strong augments P_2. Q.E.D.

As illustrated in Figure 4.2, it can be shown that there exists a value of k, i.e.,

$$\bar{k} = k_1 + r,$$

such that a $\bar{k}$-cardinality matching M is maximum in both graphs G_λ and $G_{(\lambda+1)}$.

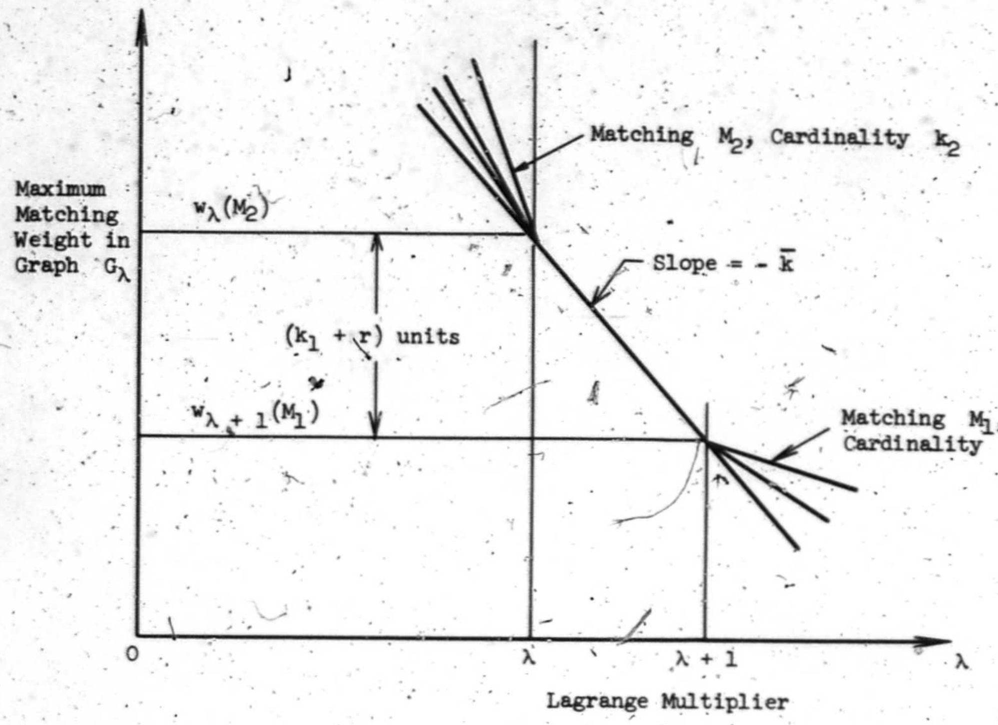
Thus if k is such that

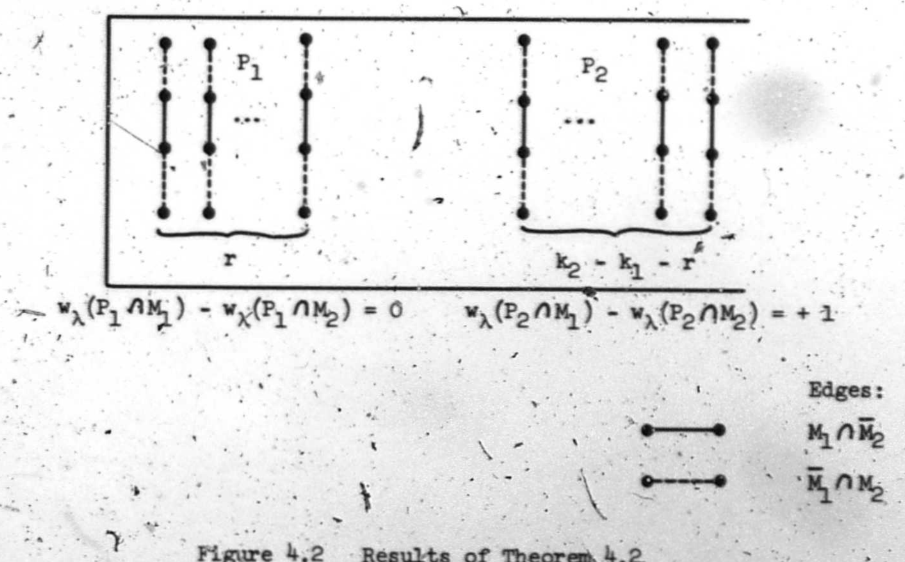
$$k_1 \leq k \leq \bar{k},$$

a maximum matching M in G_λ of k-cardinality can be constructed from matchings M_1 and M_2, using $(k - k_1)$ strong augments P_1.

But if k is such that

$$\bar{k} \leq k \leq k_2,$$





P_1
P_2
r
$k_2 - k_1 - r$
$w_\lambda(P_1 \cap M_1) - w_\lambda(P_1 \cap M_2) = 0$
$w_\lambda(P_2 \cap M_1) - w_\lambda(P_2 \cap M_2) = +1$
Edges:
$M_1 \cap \bar{M}_2$
$\bar{M}_1 \cap M_2$

Figure 4.2 Results of Theorem 4.2

a maximum matching M in $G_{(\lambda + 1)}$ of k-cardinality can be constructed from matchings M_1 and M_2, using $(k_2 - k)$ strong augments P_2.

Only if $k = \bar{k}$ is the matching M a maximum matching in both G_λ and $G_{(\lambda + 1)}$.

Theorem 4.2 thus shows that any systematic search for the appropriate Lagrange multiplier λ will always yield a maximum k-cardinality matching for any specified feasible k. If $k = m$, where m is the maximum cardinality of any matching in G, no search is necessary for this special case. Define

$$K = \underset{e_{ij} \in G}{\mathrm{Max}} \{c_{ij}\},$$

N = the number of nodes in G.

<u>Lemma 4.2</u>

For $\lambda = -\frac{N}{2} K$, a maximum matching M in graph G_λ has cardinality m.

<u>Proof</u>

Let M' be any matching in G, and of cardinality k. Then

$$w_\lambda(M) = w(M) + \frac{N}{2} Km$$

$$w_\lambda(M') = w(M') + \frac{N}{2} Kk .$$

If $k < m$, then $(k + 1) \leq m$, and

$$w(M') \leq kK < \frac{N}{2} K.$$

Thus

$$w_\lambda(M') < \frac{N}{2} K + \frac{N}{2} Kk \leq (k + 1) \frac{N}{2} K,$$

and

$$w_\lambda(M') < w_\lambda(M) \text{ as required.}$$

4.3 Direct Algorithm (Bipartite Graph)

The maximum k-cardinality matching problem is:

Max cx subject to $Ax \leq 1$, $\sum_{e_{ij}\epsilon G} x_{ij} = k$, $x_{ij} = 0$ or 1.

First let us consider bipartite graphs, in order to remove the difficulties of nonintegrality. A linear programming formulation of the above problem, together with its dual, is:

Primal

Max cx subject to $\begin{bmatrix} A \\ 1^T \end{bmatrix} x \begin{matrix} \leq \\ = \end{matrix} \begin{pmatrix} 1 \\ k \end{pmatrix}$, $x \geq 0$

Dual

Min $(\sum_{v_j \epsilon G} y_j + k\lambda)$ subject to $[A^T \ \ \underline{1}] \begin{pmatrix} y \\ \lambda \end{pmatrix} \geq c$, $y \geq 0$, λ unrestricted in sign. Subsequently we shall find that:

$\lambda > 0$ for cardinalities $k < m_o$,

$\lambda < 0$ for cardinalities $k > m_o$.

Note that the dual variable λ here is the Lagrange multiplier mentioned in Section 4.1, and used extensively in Section 4.2. If the value of λ were known, the k-cardinality constraint could be dropped from the primal, and in the dual constraint every edge weight could be considered as reduced by λ. This would result in an ordinary maximum matching problem as solved in Chapter II.

Now what we wish to do is develop an algorithm which at all times satisfies the primal-dual feasibility conditions:

1) $A_j x \leq 1$, for all $v_j \epsilon G$.

2) $x_{ij} \geq 0$, for all $e_{ij} \epsilon G$.

3) $y_i + y_j + \lambda \geq c_{ij}$, for all $e_{ij} \epsilon G$

4) $y_j \geq 0$, for all $v_j \in G$.

but terminally satisfies the constraint,

5) $\sum_{e_{ij} \in G} x_{ij} = k$

together with the orthogonality conditions,

6) $x_{ij} > 0 \Rightarrow y_i + y_j + \lambda = c_{ij}$.

7) $y_j > 0 \Rightarrow A_j x = 1$.

8) $\lambda \neq 0 \Rightarrow \sum_{e_{ij} \in G} x_{ij} = k$

It will be convenient for the description of the algorithm to assign a weight w_j to each vertex v_j. This vertex weight is related to the dual variable y_j by

9) $y_j = w_j - \frac{1}{2}\lambda$.

Consider the direct algorithm for the special case of bipartite graphs, shown in Figure 4.3:

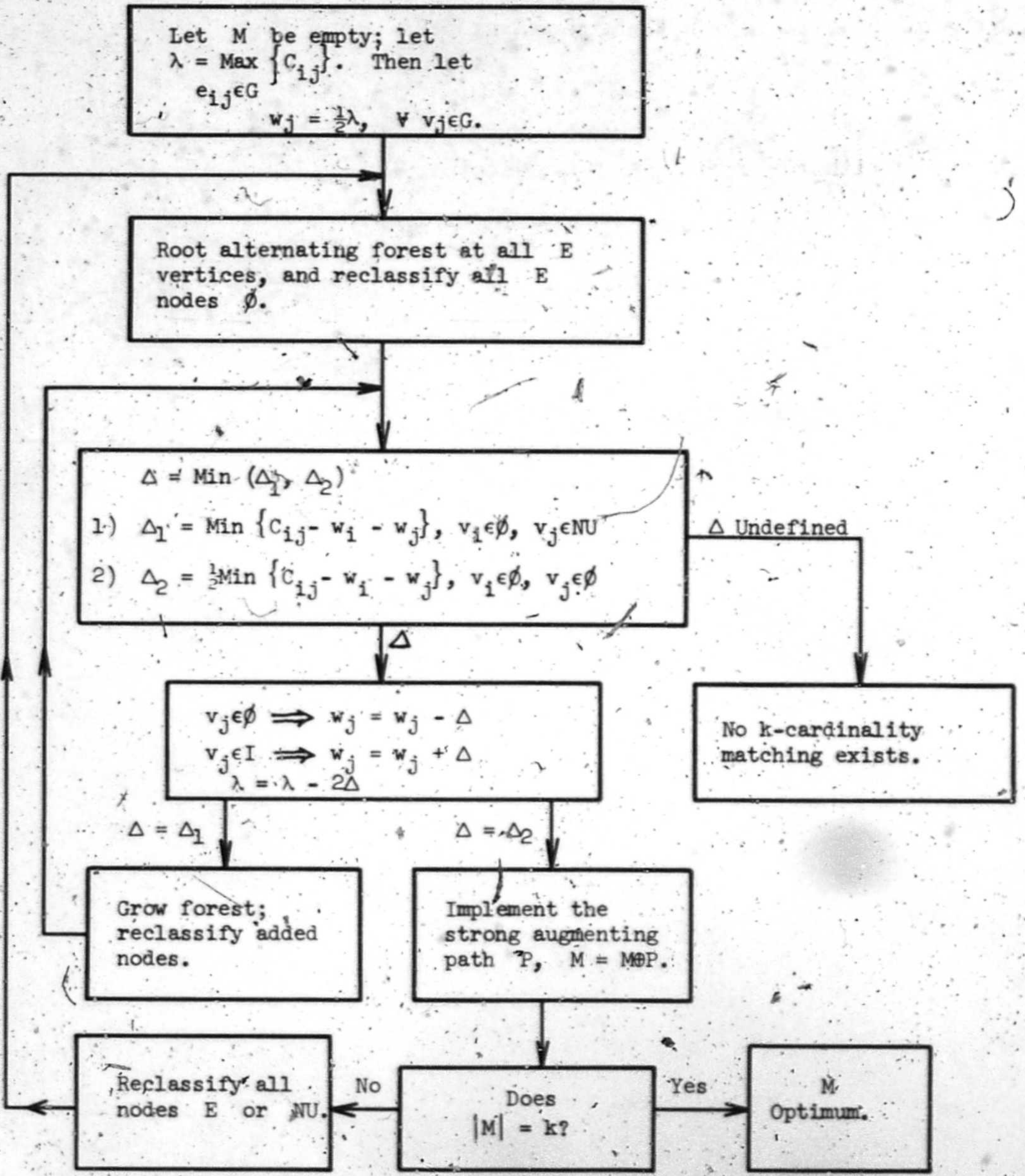
(1) Begin the algorithm with no edges in the matching M. Start all vertex weights at the same level,

$w_j = \frac{1}{2}\text{Max}_{e_{ij} \in G} \{c_{ij}\}$, for all $v_j \in G$.

(2) Define a subgraph G*, which contains all the vertices of G and edge set E*,

$E^* = \{e_{ij} | w_i + w_j = c_{ij}\}$.

(3) Grow a forest of alternating trees in G*, each rooted at an exposed node. Thus all exposed nodes are classified as outer during this forest growth.



Let M be empty; let $\lambda = \underset{e_{ij} \in G}{\mathrm{Max}} \{C_{ij}\}$. Then let $w_j = \frac{1}{2}\lambda, \quad \forall\, v_j \in G$.
Root alternating forest at all E vertices, and reclassify all E nodes $\emptyset$.
$\Delta = \mathrm{Min}\,(\Delta_1, \Delta_2)$
1) $\Delta_1 = \mathrm{Min}\,\{C_{ij} - w_i - w_j\},\ v_i \in \emptyset,\ v_j \in NU$
2) $\Delta_2 = \frac{1}{2}\mathrm{Min}\,\{C_{ij} - w_i - w_j\},\ v_i \in \emptyset,\ v_j \in \emptyset$
Δ Undefined
Δ
$v_j \in \emptyset \implies w_j = w_j - \Delta$
$v_j \in I \implies w_j = w_j + \Delta$
$\lambda = \lambda - 2\Delta$
No k-cardinality matching exists.
$\Delta = \Delta_1$
$\Delta = \Delta_2$
Grow forest; reclassify added nodes.
Implement the strong augmenting path P, $M = M \oplus P$.
Reclassify all nodes E or NU.
No
Does $|M| = k$?
Yes
M Optimum.

Figure 4.3 Direct Algorithm for Maximum k-Cardinality Matching — (Bipartite Graph)

(4) If a strong augmenting path occurs, it is used to augment the matching, and the forest is discarded.

(5) If all the trees of the forest become Hungarian, perform the following calculation:

$\Delta = \text{Min}\ (\Delta_1, \Delta_2)$ where

$$\Delta_1 = \underset{\substack{v_i \in \emptyset \\ v_j \in NU}}{\text{Min}} \{c_{ij} - w_i - w_j\}$$

$$\Delta_2 = \tfrac{1}{2}\underset{\substack{v_i \in \emptyset \\ v_j \in \emptyset}}{\text{Min}} \{c_{ij} - w_i - w_j\}$$

Upon calculation of Δ, adjust variables:

a) Outer node weights w_j are decreased by Δ.

b) Inner node weights w_j are increased by Δ.

(6) If $\Delta = \Delta_1$, a new edge enters subgraph G^*, and the forest can be grown further.

(7) If $\Delta = \Delta_2$, a strong augmenting path exists. The cardinality of the matching can be increased, and the forest discarded.

(8) If the forest is discarded, a new forest is rooted as in step (3). This process is iterated until a k-cardinality matching is obtained. At this terminal point, all exposed nodes have the same weight, and the Lagrange multiplier λ is identified as $\lambda = 2\, w_j$, for v_j exposed. All matched nodes have weight

$$w_j \geq \tfrac{1}{2}\lambda.$$

(9) In case $k > m$, the algorithm detects that there exists

no feasible matching of k-cardinality. This is detected in step (5) if the parameter Δ is undefined.

In Edmonds' algorithm described in Chapter II, a single alternating tree is rooted at an exposed node, and eventually this tree will be terminated with either a strong augmenting path or weak augmenting path. This algorithm, however, roots a forest of alternating trees, one tree at each exposed node. It can be proved that:

(a) An edge between outer nodes of the forest implies the existence of a strong augmenting path to be implemented. An edge between inner nodes of the forest cannot be used in an alternating path. An edge from an outer node to an inner node can be ignored, as it contributes no new information.

(b) There is no longer any need for weak augmenting paths to terminate the growth of a forest, for this growth will always end in a strong augmenting path.

(c) Some maximum k-cardinality matching is achieved at every step of the algorithm, as will be proved in Section 4.5.

(d) λ is monotone decreasing as the algorithm progresses.

<u>Example</u>

Find the maximum k-cardinality matchings in the graph G shown in Figure 4.4, and draw the maximum k-cardinality curve (Figure 4.5). A good set of initial vertex weights is $w_j = 7$, for all $v_j \epsilon G$.

<u>$k = 0$</u> M_0 = empty set $\lambda = 14$.

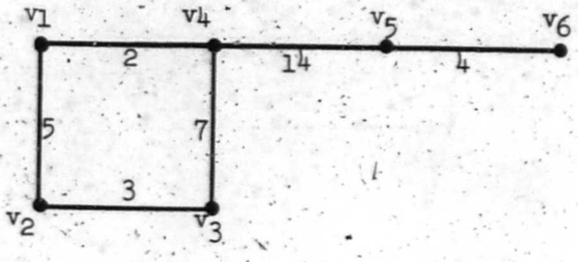
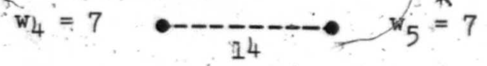


Figure 4.4 Weighted Graph G

k = 1 (1) Root forest at all exposed nodes.

(2) Augment edge e_{45}.

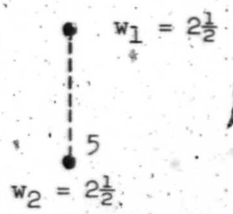


$M_1 = \{e_{45}\}$, $w(M_1) = 14$, $\lambda = 14$.

k = 2 (1) Root forest at all exposed nodes; forest is Hungarian.

(2) Lower all exposed node weights to $2\frac{1}{2}$.

(3) Augment edge e_{12}.



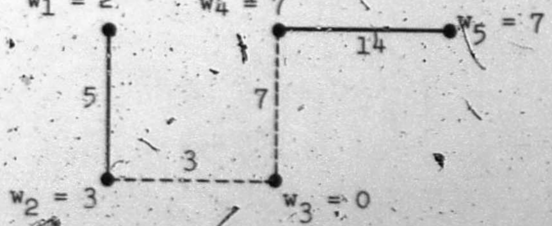
$M_2 = \{e_{12}, e_{45}\}$, $w(M_2) = 19$

$\lambda = 5$.

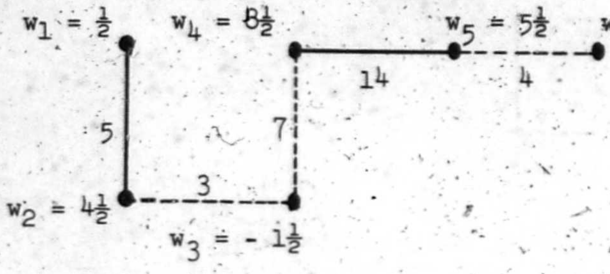
k = 3 (1) Root forest at exposed nodes v_3, v_6.

(2) Lower exposed node weights to $\frac{1}{2}$; grow edges e_{32}, e_{21}.

(3) Lower exposed node weights to 0; grow edges e_{34}, e_{45}.



(4) Lower exposed node weights to $-1\frac{1}{2}$; implement augmenting path.



$M_3 = \{e_{12}, e_{34}, e_{56}\}$

$w(M_3) = 16$

$\lambda = -3$

The maximum k-cardinality matching curve is shown in Figure 4.5. This approach should be compared to that of Edmonds' method used on the same example in Section 2.6.

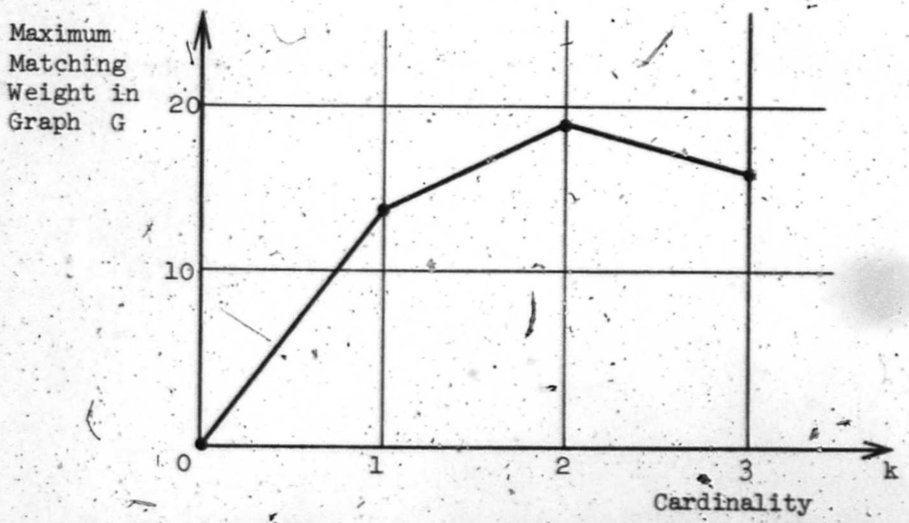


Figure 4.5 Maximum k-Cardinality Matching Curve for Graph G

4.4 Direct Algorithm (General Graph)

We are now in a position to generalize the algorithm from a bipartite to a general weighted graph, utilizing the discussion of blossoms from Chapter II.

Using Edmonds' constraint equations from Chapter II, we can state the linear programming formulation of the maximum k-cardinality matching problem, together with its dual:

Primal

Max cx subject to

$$\begin{bmatrix} A \\ R \\ \underline{1}^T \end{bmatrix} x \begin{matrix} \leq \\ \leq \\ = \end{matrix} \begin{pmatrix} 1 \\ r \\ k \end{pmatrix} \qquad x \geq 0$$

Dual

$$\text{Min } \left(\sum_{v_j \in G} y_j + \sum_{S_k} r_k z_k + k\lambda \right) \text{ subject to } [A^T, R^T\ \underline{1}] \begin{pmatrix} y \\ z \\ \lambda \end{pmatrix} \geq c,\ y \geq 0,\ z \geq 0,$$

λ unrestricted in sign.

The primary difference between this algorithm and that described in Section 4.3 is that now an outer node to outer node incidence does not necessarily indicate that a strong augmenting path exists. In case both outer nodes are in the same tree of the forest a blossom is formed. The direct maximum k-cardinality algorithm is shown in Figure 4.6. The basic difference between this direct algorithm and Edmonds' algorithm are:

(1) Instead of rooting a single tree to find a strong or weak augmenting path, a forest of trees is produced to find a strong augmenting path, one tree rooted at each exposed node. Philosophically, this might be viewed as taking advantage of parallel processing.

(2) No weak augmenting paths need be found during the course of the algorithm.

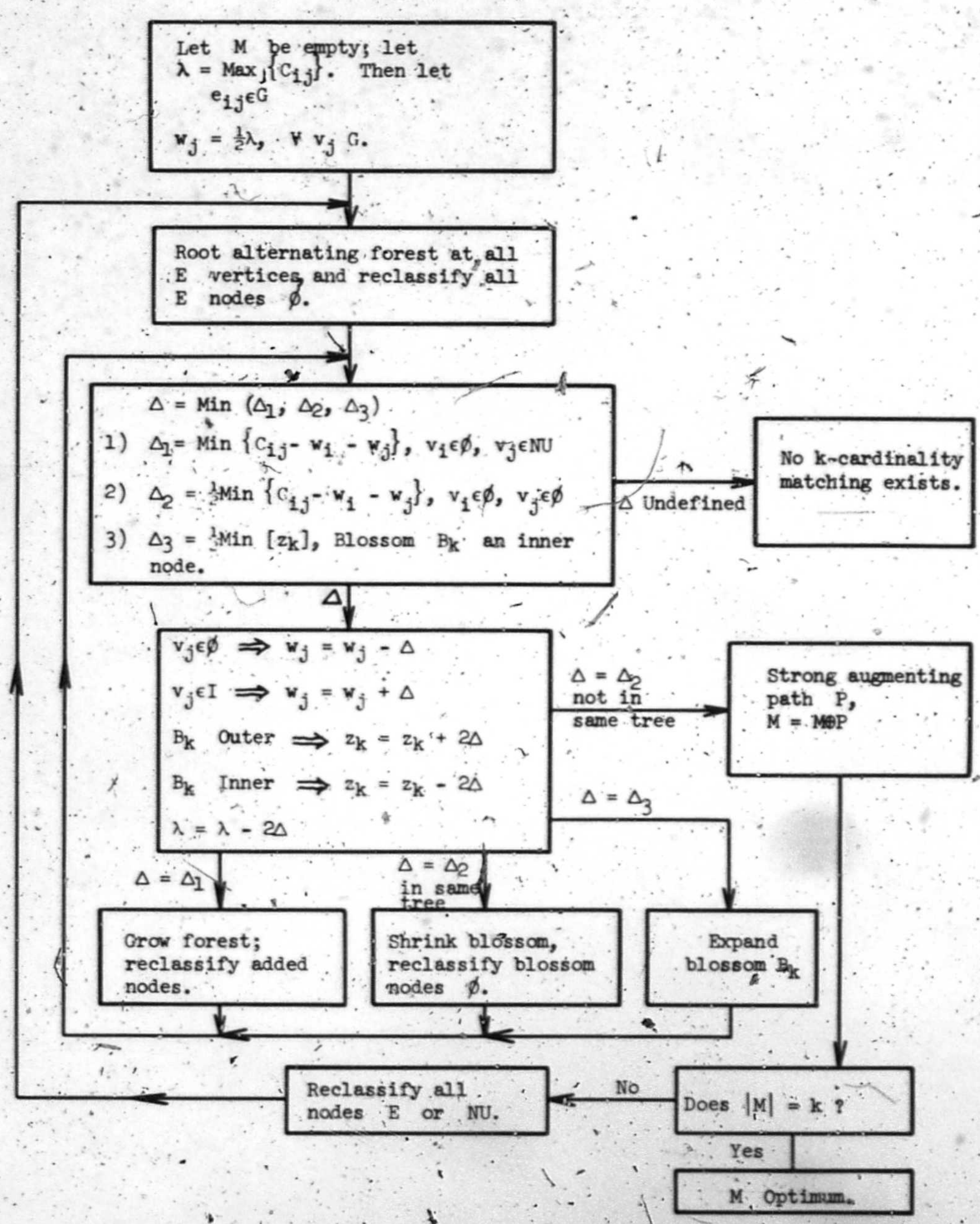

Let M be empty; let $\lambda = \text{Max}_{e_{ij} \epsilon G}\{C_{ij}\}$. Then let $w_j = \frac{1}{2}\lambda$, $\forall v_j$ G.
Root alternating forest at all E vertices, and reclassify all E nodes $\emptyset$.
$\Delta = \text{Min}(\Delta_1, \Delta_2, \Delta_3)$
1) $\Delta_1 = \text{Min}\{C_{ij} - w_i - w_j\}$, $v_i \epsilon \emptyset$, $v_j \epsilon NU$
2) $\Delta_2 = \frac{1}{2}\text{Min}\{C_{ij} - w_i - w_j\}$, $v_i \epsilon \emptyset$, $v_j \epsilon \emptyset$
3) $\Delta_3 = \frac{1}{2}\text{Min}[z_k]$, Blossom B_k an inner node.
Δ Undefined
No k-cardinality matching exists.
Δ
$v_j \epsilon \emptyset \Rightarrow w_j = w_j - \Delta$
$v_j \epsilon I \Rightarrow w_j = w_j + \Delta$
B_k Outer $\Rightarrow z_k = z_k + 2\Delta$
B_k Inner $\Rightarrow z_k = z_k - 2\Delta$
$\lambda = \lambda - 2\Delta$
$\Delta = \Delta_2$ not in same tree
Strong augmenting path P, $M = M \oplus P$
$\Delta = \Delta_3$
$\Delta = \Delta_1$
$\Delta = \Delta_2$ in same tree
Grow forest; reclassify added nodes.
Shrink blossom, reclassify blossom nodes $\emptyset$.
Expand blossom B_k
Reclassify all nodes E or NU.
No
Does $|M| = k$?
Yes
M Optimum.

Figure 4.6 Direct Algorithm for Maximum k-Cardinality Matching in a General Graph

(3) The algorithm systematically solves the maximum matching problem in the course of solving the maximum k-cardinality matching problem.

(4) The algorithm shows the geometrical nature of the maximum k-cardinality matching problem. This will be shown in Section 4:6.

(5) All exposed node weights are lowered uniformly, as contrasted with Edmonds' algorithm of Chapter II, where the weight of only one exposed node is decreased at any time.

(6) The algorithm is more deterministic, since there is no random choice of which exposed node to use as the root of a tree. Actually, the algorithm is totally deterministic, in that it will not depend on graph isomorphisms [4], except for inevitable ties in edge and path weights.

(7) Though the upper bound on the growth of the two algorithms is the same, i.e., N^4, the work and storage involved in growing a forest of trees is greater than growing and keeping track of a single tree.

4.5 Proof of the Direct Algorithm

To prove that the algorithm actually achieves the maximum k-cardinality matching at every step, we need only show that the following primal-dual feasibility and orthogonality conditions are satisfied:

Feasibility

$$\left.\begin{array}{ll}
1) & A_j x \leq 1, \text{ for all } v_j \in G \\
2) & R_k x \leq r_k, \text{ for all odd vertex sets } S_k \\
3) & x_{ij} \geq 0, \text{ for all } e_{ij} \in G \\
4) & \sum_{e_{ij} \in G} x_i = k
\end{array}\right\} \text{Primal}$$

$$\left.\begin{array}{ll}
5) & y_i + y_j + \sum_{e_{ij} \in R_k} z_k + \lambda \geq c_{ij}, \text{ for all } e_{ij} \in G \\
6) & y_j \geq 0, \text{ for all } v_j \in G \\
7) & z_k \geq 0, \text{ for all odd vertex sets } S_k
\end{array}\right\} \text{Dual}$$

Orthogonality

8) $x_{ij} > 0 \Longrightarrow y_i + y_j + \sum_{e_{ij} \in R_k} z_k + \lambda = c_{ij}$

9) $y_j > 0 \Longrightarrow A_j x = 1$

10) $z_k > 0 \Longrightarrow R_k x = r_k$

11) $\lambda \neq 0 \Longrightarrow \sum_{e_{ij} \in G} x_{ij} = k$

Identify the dual variables y and λ at the termination of the algorithm as:

12) $y_j = w_j - \frac{1}{2}\lambda$

Since we obtain a matching by the algorithm, conditions 1), 2), and 3) are satisfied.

For every node v_j, $w_j \geq \frac{1}{2}\lambda$ at the end of the algorithm, so by definition 12), condition 6) is verified. Since all exposed nodes have weight $w_j = \frac{1}{2}\lambda$ at the end of the algorithm, $y_j > 0$

guarantees node v_j is matched, so 9) is validated.

The dual variables z_k are always maintained nonnegative in the algorithm, so 7) is verified. The only way in which a nonzero z_k variable occurs is when a blossom forms, but then $R_k x = r_k$, and condition 10) follows.

Conditions 4) and 11) depend upon convergence of the algorithm. Thus if a k-cardinality matching is feasible, the algorithm generates a k-cardinality matching. Thus the proof reduces to showing 5) and 8) are valid.

Proof of 5)

If some edge e_{ij} does not belong terminally to any blossom B_k, then the algorithm maintains the condition

$$w_i + w_j \geq c_{ij},$$

and substitution of definition (12) shows that condition 5) is verified immediately for this case.

But if edge e_{ij} is involved in a blossom structure, let w_i', w_j' be the weights of the endpoints of e_{ij} at the first time e_{ij} is absorbed into a blossom, and at this point,

$$w_i' + w_j' \geq c_{ij},$$

then

$$y_i = w_i' - \tfrac{1}{2} \sum_{e_{ij} \in R_k} z_k - \tfrac{1}{2}\lambda,$$

$$y_j = w_j' - \tfrac{1}{2} \sum_{e_{ij} \in R_k} z_k - \tfrac{1}{2}\lambda.$$

Substituting above, we obtain the general verification of condition 5),

5) $y_i + y_j + \sum_{e_{ij} \in R_k} z_k + \lambda \geq c_{ij}$.

Proof of 8)

If edge e_{ij} is in the matching, then it entered this set by forming a portion of an augmenting path in the tree of the forest. But its presence in the tree implies

$$w_i + w_j = c_{ij}.$$

The Hungarian weight adjustment process maintains this equality. If the edge enters a blossom, then the proof of 5) can be used with the equality

$$w_i' + w_j' = c_{ij}$$

holding instead of the inequality. Thus 8) is valid, and the algorithm has been shown to obtain an optimum solution.

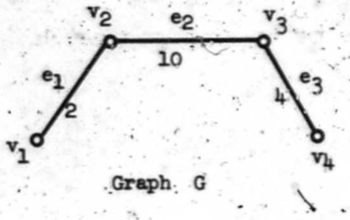
4.6 Geometric Interpretation

In Appendix A, the concept of a convex polyhedron is defined in terms of a linear programming formulation. Let us consider two examples.

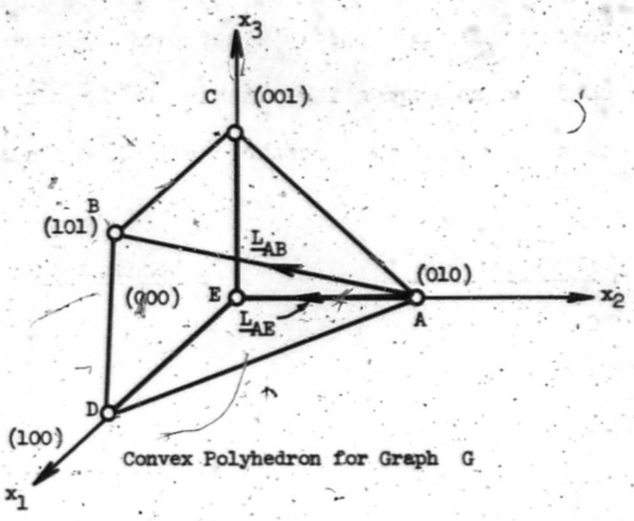
Example 1

Example 1 is shown in Figure 4.7. The maximum matching is $M_1 = \{e_2\}$, $w(M_1) = 10$. The maximum 2-edge matching is $M_2 = \{e_1, e_3\}$, $w(M_2) = 6$.

If p represents the number of edges in graph G, each vertex of the polyhedron is specified by p independent constraints satisfied with equality, and each edge of the polyhedron is specified



Graph G



Convex Polyhedron for Graph G

	Vertex Constraints	**Edges Constraints**
$\underline{v_1}$	$x_1 \leq 1$	$x_1 \geq 0$
$\underline{v_2}$	$x_1 + x_2 \leq 1$	$x_2 \geq 0$
$\underline{v_3}$	$x_2 + x_3 \leq 1$	$x_3 \geq 0$
$\underline{v_4}$	$x_3 \leq 1$	

Cost: $cx = 2x_1 + 10x_2 + 4x_3$

Figure 4.7 Example 1

by (p-1) independent constraints satisfied with equality. There are interesting relations between matching concepts in graph G and characteristics of the convex polyhedron. Each vertex of this polyhedron corresponds to a possible matching. Each edge of the polyhedron corresponds to an augmenting path in G by which one moves from one matching to another. A few instances are shown in Figure 4.8.

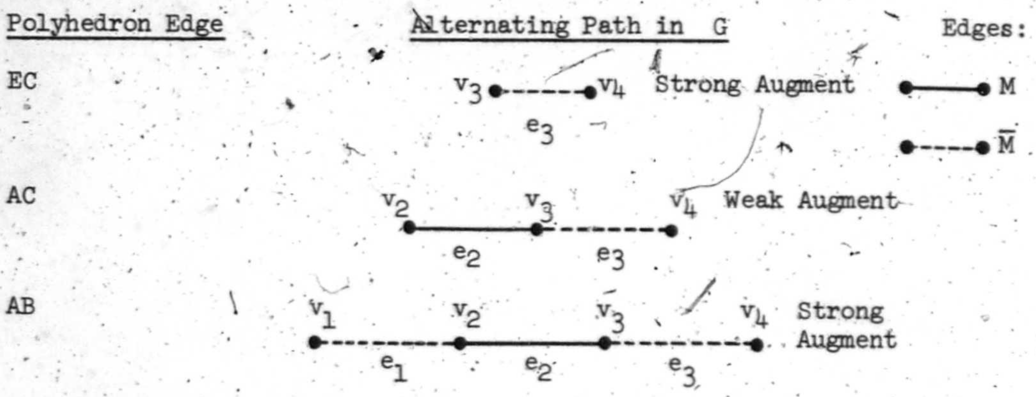


Figure 4.8 Correspondence of Alternating Paths and Polyhedron Edges

The edge weights in G determine a cost hyperplane. For Example 1, the equation of the plane is

$$2x_1 + 10x_2 + 4x_3 = W,$$

where (2, 10, 4) is a normal vector n of the plane, and W is the weight of the matching if vector x represents a feasible matching. The maximum matching value is the largest value of W such that the cost plane intersects the polyhedron. By the theory reviewed in Appendix A, a maximum value is always assumed at some vertex of the polyhedron. For this problem, from Figure 4.7, the maximum matching $M_1 = \{e_2\}$ corresponds to polyhedron vertex A, and W = 10.

Now consider the parametric technique. The matching constraints are unchanged, and thus the polyhedron remains the same. But as λ is subtracted from all the edge weights of graph G, the cost plane becomes:

$$(2 - \lambda)x_1 + (10 - \lambda)x_2 + (4 - \lambda)x_3 = W,$$

whose normal vector is $\hat{n} = (2 - \lambda, 10 - \lambda, 4 - \lambda)$. The effect of this change is to tilt the cost plane, so that as λ is varied, the optimum solution will occur at different polyhedron vertices. For sufficiently negative values of λ, the cost plane approaches the cardinality cost plane, and thus the maximum matching will also have maximum cardinality m, as proved in Lemma 4.2.

The value of λ for which there is a transition of the optimum solution from one polyhedron vertex to another occurs when the polyhedron edge between these vertices is in the cost plane. For Example 1, at $\lambda = -4$, the optimum matching will move from polyhedron vertex A to B. This occurs when polyhedron edge AB lies in the cost plane. In Figure 4.8, if $\underline{L}_{AB}$ is a vector along this edge, then

$$\hat{n} \cdot \underline{L}_{AB} = 0.$$

$$\underline{L}_{AB} = (1,0,1) - (0,1,0) = (1,-1,1).$$

Then

$$\hat{n} \cdot \underline{L}_{AB} = (2 - \lambda) - (10 - \lambda) + (4 - \lambda) = 0,$$

or

$$\lambda = -4.$$

Notice that $(-\lambda)$ is exactly the weight of the strong augmenting path in G corresponding to polyhedron edge AB. Thus for all $\lambda < -4$, the optimum polyhedron vertex is B, corresponding to a maximum matching in graph G_λ, an increase in cardinality.

For $\lambda > 0$, a deaugmenting path exists in G corresponding to polyhedron edge AE, a vector along which is $\underline{L}_{AE} = (0,-1,0)$. When vector $\underline{L}_{AE}$ is in the cost plane,

$$\hat{n} \cdot \underline{L}_{AE} = 0, \; -[10 - \lambda] = 0, \text{ or } \lambda = +10.$$

Thus the null matching is maximum in G_λ when λ forces all edges in G to be of negative weight.

For weak augments, since no cardinality change is made in moving between the appropriate polyhedron vertices, $\hat{n} \cdot \underline{L}$ is always constant. Thus the only situation in which $\hat{n} \cdot \underline{L} = 0$ is if the weight of the weak augmenting path is zero, which indicates ties in the maximum k-cardinality matching.

Example 2

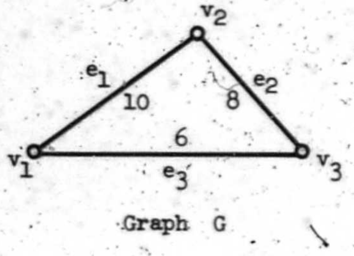
Example 2 is shown in Figure 4.9. The optimal solution is noninteger at polyhedron vertex $D(\frac{1}{2}\,\frac{1}{2}\,\frac{1}{2})$. Edmonds' constraint, $x_1 + x_2 + x_3 \leq 1$, eliminates polyhedron vertex D, so the maximum matching M_1 occurs at polyhedron vertex C, $M_1 = \{e_1\}$, $w(M_1) = 10$.

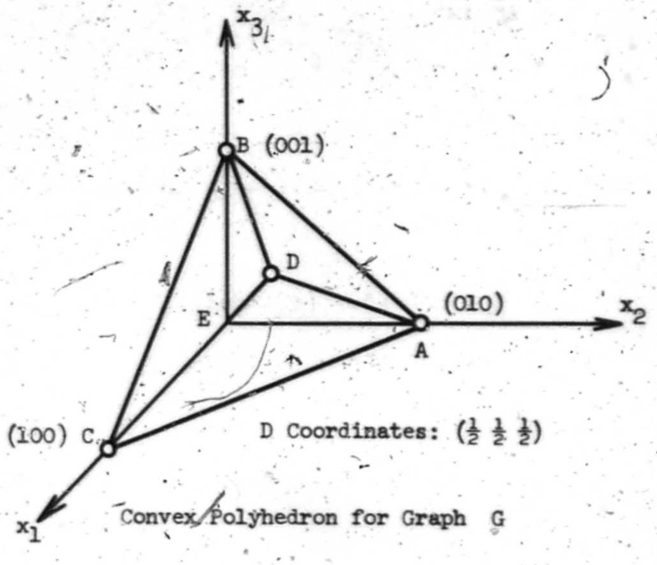
Assuming $c_1 > c_2, c_3$ for edge weights $c_1 > c_2 + c_3$, Edmonds' odd cycle constraint is not necessary for integer solutions.

Note that the maximum matching in G_λ continues to be M_1, corresponding to polyhedron vertex C, for all values of $\lambda < 10$. For values of $\lambda > 10$, all edge weights are negative, and the null matching is optimum.

4.7 The Maximum k-Cardinality Matching Curve

The curve representing the weight of the maximum k-cardinality





x_3
B (001)
D
E
(010)
A
x_2
(100) C
D Coordinates: $(\frac{1}{2}\ \frac{1}{2}\ \frac{1}{2})$
x_1
Convex Polyhedron for Graph G

	Vertex Constraints	Edge Constraints
$\underline{v_1}$	$x_1 + x_2 \leq 1$	$x_1 \geq 0$
$\underline{v_2}$	$x_1 + x_3 \leq 1$	$x_2 \geq 0$
$\underline{v_3}$	$x_2 + x_3 \leq 1$	$x_3 \geq 0$

Edmonds' Odd Cycle Constraint: $x_1 + x_2 + x_3 \leq 1$.

Figure 4.9 Example 2

matching in graph G as a function of cardinality k is shown in Figure 4.10. Define a <u>convex curve</u> or <u>convex function</u> as a function $f(x)$ such that for any two elements in the domain, $x_1, x_2 \epsilon X$, and $0 \leq \alpha \leq 1$, implies

$$f[\alpha x_1 + (1 - \alpha)x_2] \leq \alpha f(x_1) + (1 - \alpha) f(x_2).$$

Similarly, define a <u>concave curve</u> or <u>concave function</u> as a function $f(x)$ such that for any two elements in the domain, $x_1; x_2 \epsilon X$ and $0 \leq \alpha \leq 1$, implies

$$f[\alpha x_1 + (1 - \alpha)x_2] \geq \alpha f(x_1) + (1 - \alpha) f(x_2).$$

Linear functions are both convex and concave.

<u>Theorem 4.3</u>

The maximum k-cardinality matching curve is concave.

<u>Proof</u>

Let M_1 be the maximum k_1-cardinality matching in G for any $0 \leq k_1 \leq m$. Let M_2 be the maximum k_2-cardinality matching in G for any $0 \leq k_2 \leq m$, $k_2 \neq k_1$. Let λ_1 be the minimum value of λ for which M_1 is the maximum matching in $G_{\lambda 1}$, and let λ_2 be the minimum value of λ for which M_2 is the maximum matching in $G_{\lambda 2}$. (Note: let λ_1, λ_2 be the <u>maximum value</u> of λ in case k_1 or k_2 equals zero).

Assume $k_1 < k_2$, so $\lambda_1 \geq \lambda_2$, by Theorems 4.1 and 4.2. Now if M^* is any maximum k^*-cardinality matching in G, for $k_1 \leq k^* \leq k_2$, and if λ^* is the minimum value for which M^* is the maximum matching in G_{λ^*}, then

$$\lambda_2 \leq \lambda^* \leq \lambda_1.$$

$$w_{\lambda^*}(M^*) = w(M^*) - k^*\lambda^*.$$

Identify α in the definition of a concave curve as

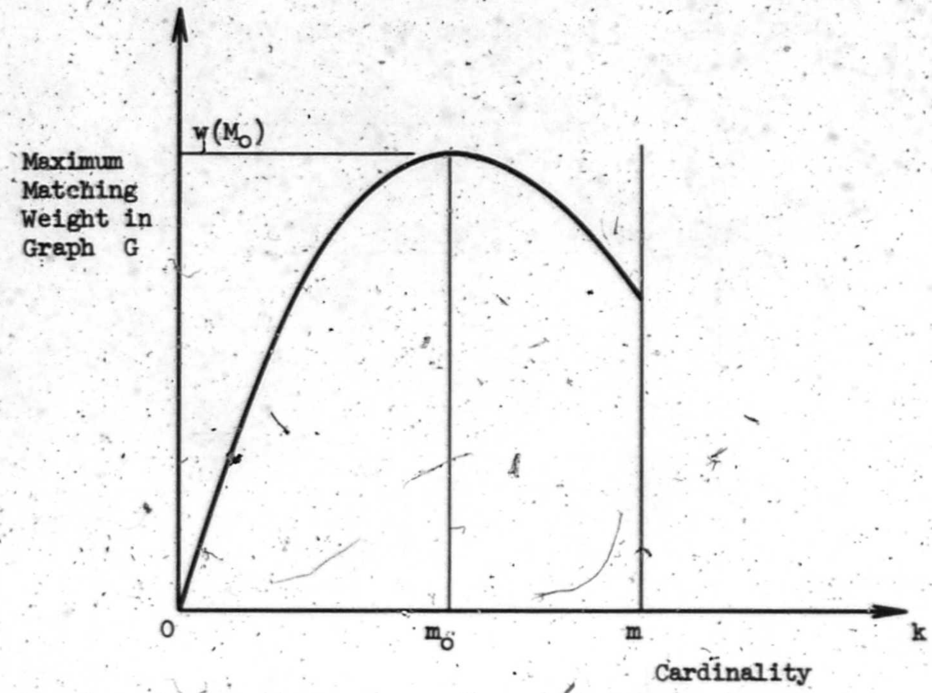


Figure 4.10 Maximum k-Cardinality Matching Curve

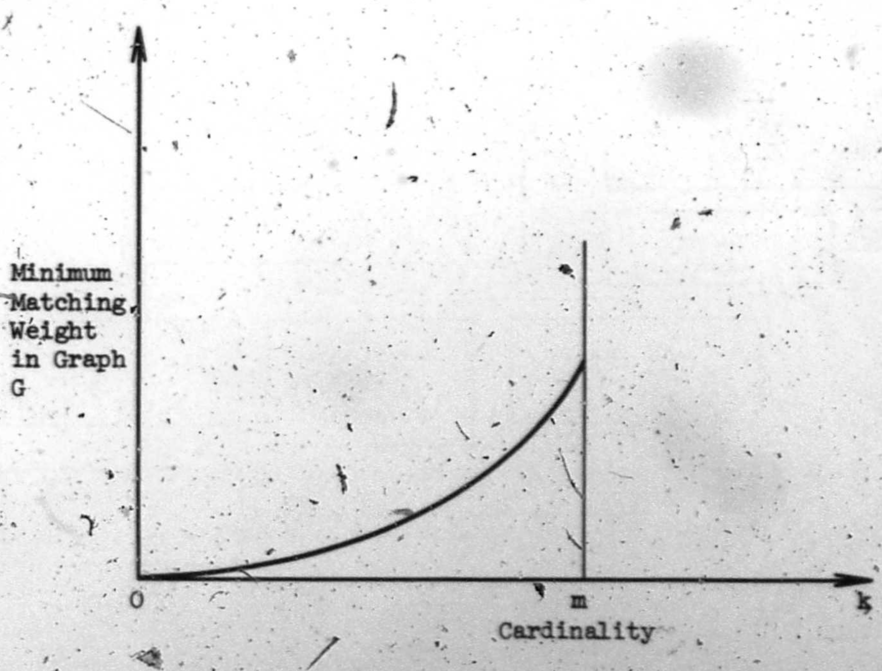


Figure 4.11 Minimum k-Cardinality Matching Curve

1) $\alpha = \dfrac{k_2-k^*}{k_2-k_1} \geq 0$, for then $k^* = \alpha k_1 + (1 - \alpha) k_2$.

2) $w_{\lambda*}(M^*) \geq w_{\lambda*}(M_1)$.

3) $w_{\lambda*}(M^*) \geq w_{\lambda*}(M\ )$

Expanding inequalities 2) and 3):

4) $w(M^*) - k\lambda^* \geq w(M_1) - k_1\lambda^*$,

5) $w(M^*) - k\lambda^* \geq w(M_2) - k_2\lambda^*$.

Since $0 \leq \alpha \leq 1$, multiply 4) by α, and 5) by $(1 - \alpha)$, and add the resulting inequalities, using 1), and note that all terms involving λ^* cancel, leaving 6) $w(M^*) \geq \alpha\, w(M_1) + (1 - \alpha)\, w(M_2)$, and the maximum k-cardinality matching curve is concave as claimed.

There exist alternate methods of proving Theorem 4.3 which better characterize the role of strong augmenting paths in determining the concavity of the curve. These methods utilize the concept of augmenting paths, motivated by Theorem 4.4, stated without proof.

Theorem 4.4

Let M_1 be a maximum k-cardinality matching in a weighted graph G. Select M_2 to be the maximum $(k + 1)$-cardinality matching such that $|M_1 \oplus M_2|$ is minimal. Then subgraph $M_1 \oplus M_2$ consists of one strong augmenting path.

It can then be shown that as maximum k-cardinality matchings are obtained for increasing values of cardinality k, the strong augmenting paths of Theorem 4.4 have monotonic nonincreasing values, as does the Lagrange multiplier λ. Thus proof of Theorem 4.3 by the use of strong augmenting paths is equivalent to the proof given utilizing the Lagrange multiplier λ.

4.8 Minimum Matching

Given a graph G with nonnegative edge weights, a minimum matching problem is trivial. However, the problem of a minimum k-cardinality matching is interesting, and can be solved by our present techniques with the use of the edge weight transformation

$$c_{ij}' = K - c_{ij}, \text{ for all } e_{ij} \in G,$$

where $K = \underset{e_{ij}\in G}{\text{Max}}\,[c_{ij}]$, which induces a graph $[G', c_{ij}']$.

A maximum k-cardinality matching M can be found in graph $[G', c_{ij}']$, and can be shown to be the desired minimum k-cardinality matching in graph $[G, c_{ij}]$, since

$$w'(M) = \sum_{e_{ij}\in M} c_{ij}' = \sum_{e_{ij}\in M} [K-c_{ij}] = kK - \sum_{e_{ij}\in M} c_{ij},$$

but

$$w(M) = \sum_{e_{ij}\in M} c_{ij},$$

so

$$w'(M) = kK - w(M).$$

Thus since kK is a constant, maximizing $w'(M)$ is equivalent to minimizing $w(M)$.

Moreover, the minimum k-cardinality matching curve as shown in Figure 4.11 is both convex and monotonic nondecreasing. In the equation,

$$w(M) = kK - w'(M),$$

the term kK is linear in k, and is thus a convex function; w'(M) has already been proved to be a concave function. It can be shown that if a concave function is subtracted from a convex function, that the result will be a convex function. Thus the minimum k-cardinality matching curve is convex. The curve can then be seen to be monotonic nonincreasing since it must pass through the origin, but this can be proved rigorously by consideration of the strong augmenting paths involved.

CHAPTER V

MINIMUM k-CARDINALITY COVERING

5.1 Introduction

The minimum covering problem is solved by an algorithm described in Chapter III. We wish to find a minimum weight covering of a specified cardinality k, and contemplate using parametric techniques similar to those of Chapter IV. Given a weighted graph G, this problem can be formulated as:

$$\text{Min } cx \quad \text{subject to } Ax \geq 1, \quad \sum_{e_{ij}\in G} x_{ij} = k, \quad x_{ij} = 0 \text{ or } 1.$$

The approach will be to transform the graph $[G, c_{ij}]$ to graph $[G_\lambda, c_{ij} - \lambda]$ by subtracting λ from each edge weight of G. The parameter λ may be interpreted as either the dual variable corresponding to the primal constraint

$$\sum_{e_{ij}\in G} x_{ij} = k,$$

or as a Lagrange multiplier [11].

Section 5.2 presents a characterization of minimum k-cardinality coverings.

A parametric approach to the problem is given in Section 5.3, and some of the implications of this approach are pointed out in Section 5.4. Section 5.5 offers a geometric interpretation of coverings similar to that of Section 4.7 for matchings.

Since a minimum k-cardinality covering will in general contain cycles for sufficiently large k, Section 5.6 explores the extension of restricting the optimum covering configuration to be a forest for $k \leq (N-1)$, where N is the number of nodes in graph G. Section 5.7 proves that the minimum k-cardinality covering curve is convex. The maximum k-cardinality covering problem is introduced in Section 5.8.

5.2 Characterization of Minimum k-Cardinality Coverings

A strong reducing path P relative to coverings C_1, C_2, where $|C_1| < k$, $|C_2| = k$, is a path P in G such that:

(1) P alternates in $C_1 \oplus C_2$ with respect to edges in C_1, C_2.

(2) $C' = C_2 \oplus P$, is a $(k-1)$-cardinality covering in G.

Theorem 5.1

Let C_1 be a minimum k-cardinality covering in a weighted graph G. Select C_2 to be the minimum $(k + 1)$-cardinality covering in G such that $|C_1 \oplus C_2|$ is minimal. Then subgraph $C_1 \oplus C_2$ consists of one strong reducing path P.

Proof

It can be shown by a graphical argument that a strong reducing path P always exists in $C_1 \oplus C_2$. Such a path P is illustrated in Figure 5.1. To show that P completely comprises $C_1 \oplus C_2$, form:

$C_1' = C_2 \oplus P$, a covering in G of cardinality k.

$C_2' = C_1 \oplus P$, a covering in G of cardinality $(k + 1)$.

Then

$$w(C_1') \geq w(C_1),$$
$$w(C_2') \geq w(C_2),$$

so

$$\left.\begin{array}{l} w(C_2 \cap \bar{P}) \geq w(C_1 \cap \bar{P}) \\ \\ w(C_1 \cap \bar{P}) \geq w(C_2 \cap \bar{P}) \end{array}\right\} \Longrightarrow \quad w(C_2 \cap \bar{P}) = w(C_1 \cap \bar{P})$$

By the minimality of $|C_1 \oplus C_2|$, it can be seen that path P completely comprises subgraph $C_1 \oplus C_2$.

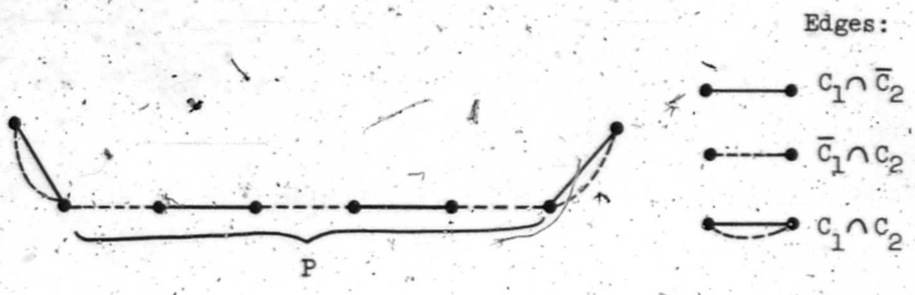


Figure 5.1 Strong Reducing Path

5.3 Parametric Approach

Given graph $[G, c_{ij}]$, construct graph $[G_\lambda, c_{ij} - \lambda]$.

Define

$$K = \underset{e_{ij} \in G}{\text{Max}} [c_{ij}]$$

$$L = \underset{e_{ij} \in G}{\text{Min}} [c_{ij}]$$

$w_\lambda(C)$ = Weight of covering C in G_λ.

The use of the parametric approach is complicated by the formation of negative edges. Consider λ in the range $\lambda < L$ in order to temporarily avoid this complication.

Define c_o as the cardinality of the minimum covering C_o in G, and c as the minimum cardinality of any covering.

<u>Theorem 5.2</u>

Given k, $c \leq k \leq c_o$, there exists a $\lambda \leq 0$ such that a minimum covering C in G_λ is of k-cardinality, and C is a minimum k-cardinality covering in G.

The proof of Theorem 5.2 is similar to the development of Chapter IV.

Similarly, if we obtain the minimum covering C in G_λ for $\lambda = L$, and the cardinality of C is p, then the parametric approach will obtain the minimum k-cardinality covering C in G_λ for $c_o \leq k \leq p$, using $0 \leq \lambda \leq L$.

For arbitrary positive values of λ, consider the following development.

Partition the nodes in G_λ into two sets:

1) V_N, nodes which are incident to an edge of negative weight.
2) V_P, nodes which are not incident to an edge of negative weight.

Define an edge partition of G_λ:

1) P_λ, all edges in G_λ with at least one endpoint in node set V_P. Define a <u>covering</u> of V_P as a subset of the edges of P_λ such that at least one of the edges of this subset is incident to each node in V_P.

2) N_λ, all edges in G_λ with both endpoints in V_N.

For an illustration of these concepts, see Figure 5.3.

Theorem 5.3

Define C as the edges of a minimum covering of V_P, together with all the negative edges of G_λ. If the cardinality of C is k, then C is the minimum k-cardinality covering in graph G.

Proof

Let D be any k-cardinality covering in G. Consider the weights of C and D in graph G_λ.

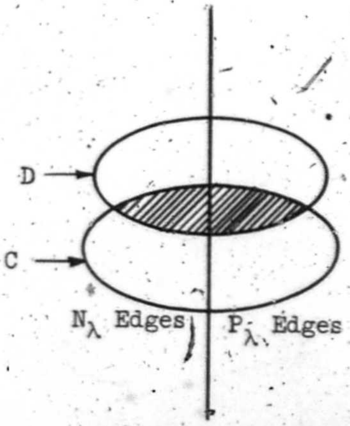


Figure 5.2 Venn Diagram for Proof of Theorem 5.3

$$w_\lambda(C) = w(C \cap P_\lambda) + w(C \cap N_\lambda)$$
$$w_\lambda(D) = w(D \cap P_\lambda) + w(D \cap N_\lambda)$$

But since we found a minimum covering of V_P,

$$w_\lambda(C \cap P_\lambda) \leq w_\lambda(D \cap P_\lambda),$$

and since C uses all the negative edges in G_λ,

$$w_\lambda(C \cap N_\lambda) \le w_\lambda(D \cap N_\lambda).$$

Thus

$$w_\lambda(C) \le w_\lambda(D).$$

But since $|C| = |D| = k$ by assumption,

$$w_\lambda(C) = w(C) - k\lambda, \text{ and}$$

$$w_\lambda(D) = w(D) - k\lambda.$$

Thus $w(C) \le w(D)$ in G,

and C is a minimum k-cardinality covering in G as claimed.

5.4 Implications of the Parametric Technique

We are now in an excellent position to answer many questions about minimum k-cardinality coverings. Consider Figure 5.3, which shows the decomposition of graph G_λ into edge sets P_λ and N_λ for some $\lambda > L$.

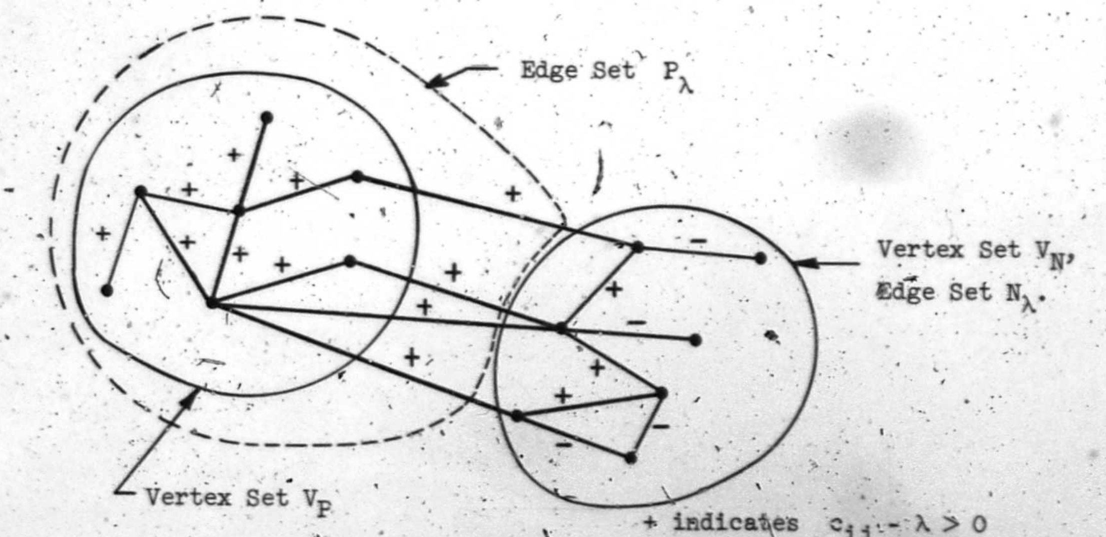

Edge Set P_λ
Vertex Set V_N, Edge Set N_λ.
Vertex Set V_P
+ indicates $c_{ij} - \lambda > 0$
- indicates $c_{ij} - \lambda < 0$

Figure 5.3 Decomposition of Graph G_λ into Edge Sets P_λ, N_λ

As λ increases, more edges become negative in G_λ, thus more nodes move out of node set V_P into V_N. Simultaneously the negative edges, together with positive edges with both endpoints in set V_N, move from edge set P_λ into N_λ, as seen in Figure 5.3.

The algorithm suggested by Theorem 5.3 solves the minimum k-cardinality covering problem by using all the negative edges, together with a minimum covering of V_P.

Thus the original problem is decomposed into an "easy" problem in N_λ, and a "hard" covering problem for node set V_P.

When λ becomes sufficiently large such that all nodes are in set V_N, the minimum k-cardinality covering problem becomes easy for all k above a corresponding value. The critical value λcrit for which this occurs is:

$$\lambda_{crit} = \max_{v_j} \left\{ \min_{\substack{\text{all } e_{ij} \\ \text{incident} \\ \text{to node } v_j}} [c_{ij}] \right\}$$

The implications of this result will be discussed in Chapter VII.

There are two ways in which the cardinality of the minimum covering in G_λ increases as λ increases:

(1) A zero weight strong reducing path appears in edge set P_λ, yielding a cardinality change.

(2) An edge becomes negative, thus ensuring that it will not only be in the next larger cardinality minimum covering, but in every subsequent minimum covering of greater cardinality.

Ties occur when several zero weight strong reducing paths appear in P_λ

or several edges become negative at the same value of λ.

One might inquire at what cardinality the minimum k-cardinality covering ceases to be a subgraph of disjoint stars. Since the minimum covering in P_λ can only consist of stars, edge set N_λ is responsible for providing non-star minimum coverings.

One also might ask at what cardinality the minimum k-cardinality covering will contain cycles. The answer can again be seen clearly in the edge set N_λ: it occurs at the lowest value of λ for which all edges of a cycle become negative in G_λ.

5.5 Geometrical Interpretation

The parametric process of Sections 5.3 and 5.4 may be interpreted geometrically.

Given a weighted graph G, the covering constraints

$$Ax \geq 1, \qquad x \geq 0,$$

are a function only of the topology of G. These constraints are associated with a convex polyhedron. The vertices of the convex polyhedron represent possible covering solutions, except for noninteger vertices which must be removed by additional constraints as described in Chapter III. The edges of the polyhedron correspond to strong and weak reducing paths, by which one moves from one covering solution to another.

The edge weights of the graph are relevant only to the cost, and determine a cost hyperplane,

$$cx = W,$$

where W is the weight of the covering if vector x represents a feasible covering. As λ is subtracted from each edge weight to form graph G_λ, the cost hyperplane tilts, and becomes

$$(c - \underline{1}\lambda)\, x = W,$$

whose normal vector is $\hat{n} = (c - \underline{1}\lambda)$.

Since negative edges play such an important role in Sections 5.3 and 5.4, note that edge e_i enters N_λ as a negative edge just when unit vector $\hat{x}_i$ lies in the hyperplane $(c - \underline{1}\lambda) = W$. The condition which must be satisfied for this to occur is

$$\hat{n} \cdot \hat{x}_i = 0.$$

This geometric interpretation is illustrated by the following simple examples.

Example 1

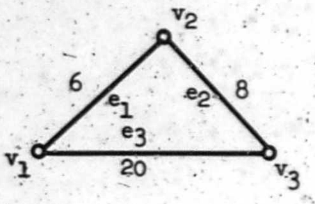
Example 1 is shown in Figure 5.4. The minimum covering is $\{e_1, e_2\}$ with cost $W = 14$, which corresponds to polyhedron vertex A. In general, if $c_3 > c_1 + c_2$, polyhedron vertex A is optimum, but if $c_3 < c_1 + c_2$, the noninteger vertex F is optimum. The constraint used to eliminate the noninteger vertex F is

$$x_1 + x_2 + x_3 \geq 2.$$

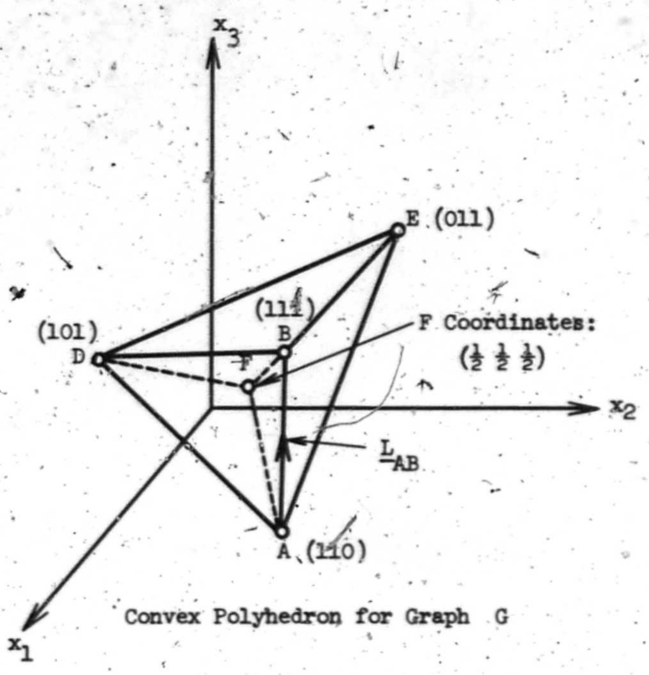
Consider this constraint added to the system, and subtract $\lambda > 0$ from each edge weight in graph G, tilting the cost plane to

$$(6 - \lambda)x_1 + (8 - \lambda)x_2 + (20 - \lambda)x_3 = W.$$

The polyhedron edge $\underline{L}_{AB} = (0,0,1)$ enters the cost plane when $\underline{L}_{AB} \cdot \hat{n} = 0$, or for $\lambda = 20$. Thus for $\lambda > 20$, the minimum covering in G_λ is $\{e_1, e_2, e_3\}$, which has cardinality three. This



Graph G



Convex Polyhedron for Graph G

	Vertex Constraints	Edge Constraints
$\underline{v_1}$	$x_1 + x_3 \geq 1$	$x_1 \geq 0$
$\underline{v_2}$	$x_1 + x_2 \geq 1$	$x_2 \geq 0$
$\underline{v_3}$	$x_2 + x_3 \geq 1$	$x_3 \geq 0$

Cost: $cx = 6x_1 + 8x_2 + 20x_3$

Odd Cycle Covering Constraint: $x_1 + x_2 + x_3 \geq 2$

Figure 5.4 Example 1

solution is represented by polyhedron vertex B. Subtracting $\lambda < 0$ from each edge weight maintains the solution at polyhedron vertex A for arbitrarily large negative λ.

Example 2

Example 2 is shown in Figure 5.5.

For any c_1, c_2, c_3, the minimum covering in G is $\{e_1, e_3\}$, represented by vertex A of polyhedron AB. However, as $\lambda > 0$ is subtracted from each graph edge weight, and as polyhedron edge AB comes into the cost plane, covering $\{e_1, e_2, e_3\}$ represented by polyhedron vertex B will become the minimum covering ~~in~~ G_λ. This occurs when

$$\underline{L}_{AB} \cdot \hat{n} = 0$$

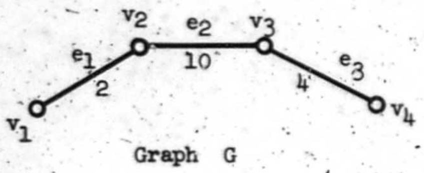
$\underline{L}_{AB} = (0,1,0)$, $\hat{n} = (2 - \lambda, 10 - \lambda, 4 - \lambda)$, or $\lambda = 10$.

5.6 The Minimum k-Cardinality Forest Covering Problem

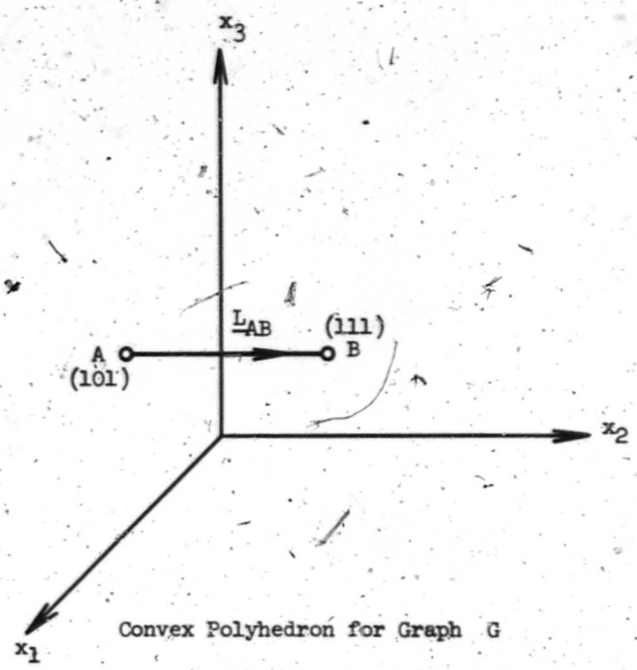
The solution to the minimum k-cardinality covering problem will contain cycles for sufficiently large k. We would like to consider a minimum covering solution which would contain no cycles for cardinalities up through $k = (N - 1)$.

Define a forest covering in a graph G as a forest such that some edge of the forest is incident to every vertex of G. A minimum forest covering of a graph G can be found by the use of the minimum spanning tree algorithm [12], which will be presented in Chapter VII.

The notation for Theorem 5.4 is the same as developed in Section 5.3.



Graph G



Convex Polyhedron for Graph G

	Vertex Constraints	Edge Constraints
$\underline{v_1}$	$x_1 \geq 1$	$x_1 \geq 0$
$\underline{v_2}$	$x_1 + x_2 \geq 1$	$x_2 \geq 0$
v_3	$x_2 + x_3 \geq 1$	$x_3 \geq 0$
$\underline{v_4}$	$x_3 \geq 1$	

Figure 5.5 Example 2

Theorem 5.4

Let C be the edges of a minimum covering of V_P, together with a minimum forest covering of V_N, using edge set N_λ. If the cardinality of C is k, then C is the minimum k-cardinality forest covering in graph G.

Proof

Let D be any k-cardinality forest covering in G. Consider the weights of C and D in graph G_λ.

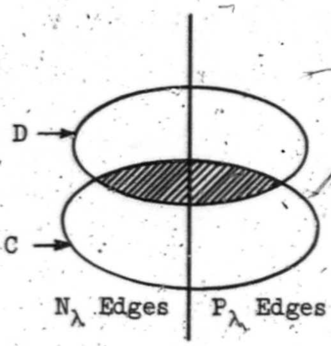


Figure 5.6 , Venn Diagram for Proof of Theorem 5.4

$$w_\lambda(C) = w_\lambda(C \cap P_\lambda) + w_\lambda(C \cap N_\lambda),$$
$$w_\lambda(D) = w_\lambda(D \cap P_\lambda) + w_\lambda(D \cap N_\lambda).$$

But since we found a minimum covering of V_P,

$$w_\lambda(C \cap P_\lambda) \leq w_\lambda(D \cap P_\lambda).$$

We formed $(C \cap N_\lambda)$ as the minimum forest covering of nodes V_N, so

$$w_\lambda(C \cap N_\lambda) \leq w_\lambda(D \cap N_\lambda).$$

Thus $w_\lambda(C) \leq w_\lambda(D)$,

and since $|C| = |D| = k$ was assumed,

$$w_\lambda(C) = w(C) - k\lambda,$$

$$w_\lambda(D) = w(D) - k\lambda.$$

So $w(C) \leq w(D)$ in G, and C is the minimum k-cardinality forest covering in G as claimed.

The major implications of this algorithm will be reserved for Chapter VII.

5.7 The Minimum k-Cardinality Covering Curve

The minimum k-cardinality covering curve is convex, as shown in Figure 5.7. The curve representing minimum k-cardinality forest coverings is also convex. These results can be proved using the monotonic behavior of the Lagrange multiplier λ. An alternate approach would be to utilize the concept of reducing paths, but reducing paths are more complex graph structures than are augmenting paths used in matching.

5.8 Maximum Covering

Given a graph G with nonnegative edge weights, a maximum covering problem would be trivial. A covering including all the edges of G would be of maximum weight. However, the problem of maximum k-cardinality covering is interesting, and can be solved by our present techniques. Consider the edge weight transformation:

$$c_{ij}' = K - c_{ij}, \quad \text{for all } e_{ij} \in G,$$

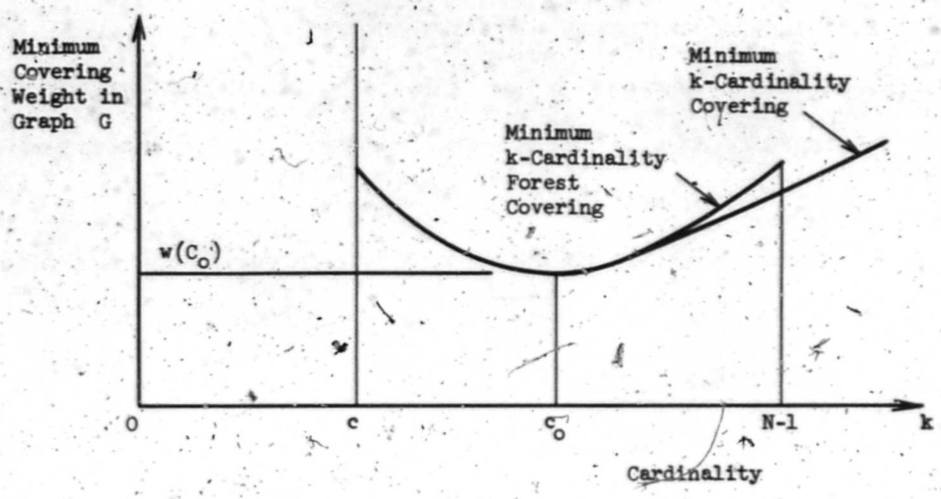


Figure 5.7 Minimum k-Cardinality Covering Curve

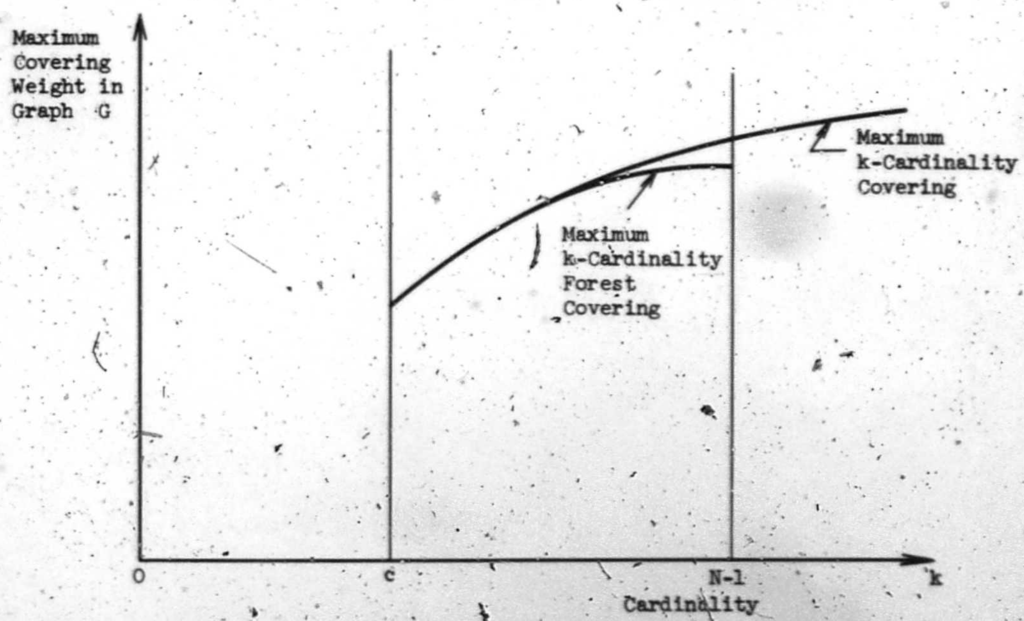


Figure 5.8 Maximum k-Cardinality Covering Curve

where $K = \underset{e_{ij} \in G}{\text{Max}} [c_{ij}]$, which induces a weighted graph $[G', c_{ij}']$.

A minimum k-cardinality covering C can be found in graph $[G', c_{ij}']$, and can be shown to be the desired maximum k-cardinality covering in graph $[G, c_{ij}]$. The proof is essentially the same as indicated in Section 4.8 for matching.

This maximum k-cardinality covering is shown in Figure 5.8, and can be proved to be both concave and monotonic nondecreasing.

CHAPTER VI

RELATIONSHIPS BETWEEN MATCHINGS AND COVERINGS

6.1 Introduction

Chapters IV and V dealt with the k-cardinality matching and covering problems in detail. The question of the relationship between these matchings and coverings naturally arises in the course of this parametric study; this chapter addresses itself to that question.

Section 6.2 introduces a new concept--a k,p-subgraph which allows us to examine matchings and coverings from a more general point of view. Optimum k,p-subgraphs are discussed in Section 6.3. This development motivates the consideration of a special case in Section 6.4: optimum k, (m + k)-subgraphs. Section 6.5 contributes an additional idea on the specific relationship between matchings and coverings.

6.2 Feasible k,p-Subgraphs

A k,p-subgraph S is a subgraph of a graph G such that

a) S contains k edges, and

b) exactly p of the vertices are covered by S.

Consider some special cases:

(1) Matching: a k,p-subgraph for which $p = 2k$, for $0 \leq k \leq m$, where m is the maximum cardinality of any matching.

(2) Covering: a k,p-subgraph for which $p = N$, for $k \geq c$, where c is the minimum cardinality of any covering, and N represents the number of nodes in G.

(3) $k,(m + k)$-Subgraph: a k,p-subgraph for which $p = (k + m)$, for $m \leq k \leq c$.

We would like to know which values of k and p represent feasible k,p-subgraphs. Figure 6.1 shows the feasible k-p region for a typical weighted graph G. The topology of G determines the parameters m and c. Note that

$$c = N - m$$

for all graphs, and thus since $m \leq \frac{N}{2}$ and $c \leq m$, $c = m$ only for special graphs G for which $c = m = \frac{N}{2}$. Such graphs are said to contain a <u>1-factor</u> or <u>perfect matching</u> [16].

The special cases of matchings, $k,(m + k)$-subgraphs, and coverings form an upper bound on the region of feasible k,p-subgraphs in the k-p plane as shown in Figure 6.1.

The lower bounds on the feasible region of k,p-subgraphs are more difficult to determine, and are complex functions of graph topology. Let us examine the constraints for low values of cardinality k, and then for some general classes of graphs.

<u>$k = 0$</u> $p = 0$ is the only feasible point.

<u>$k = 1$</u> $p = 2$ is the only feasible point, for each edge has two endpoints.

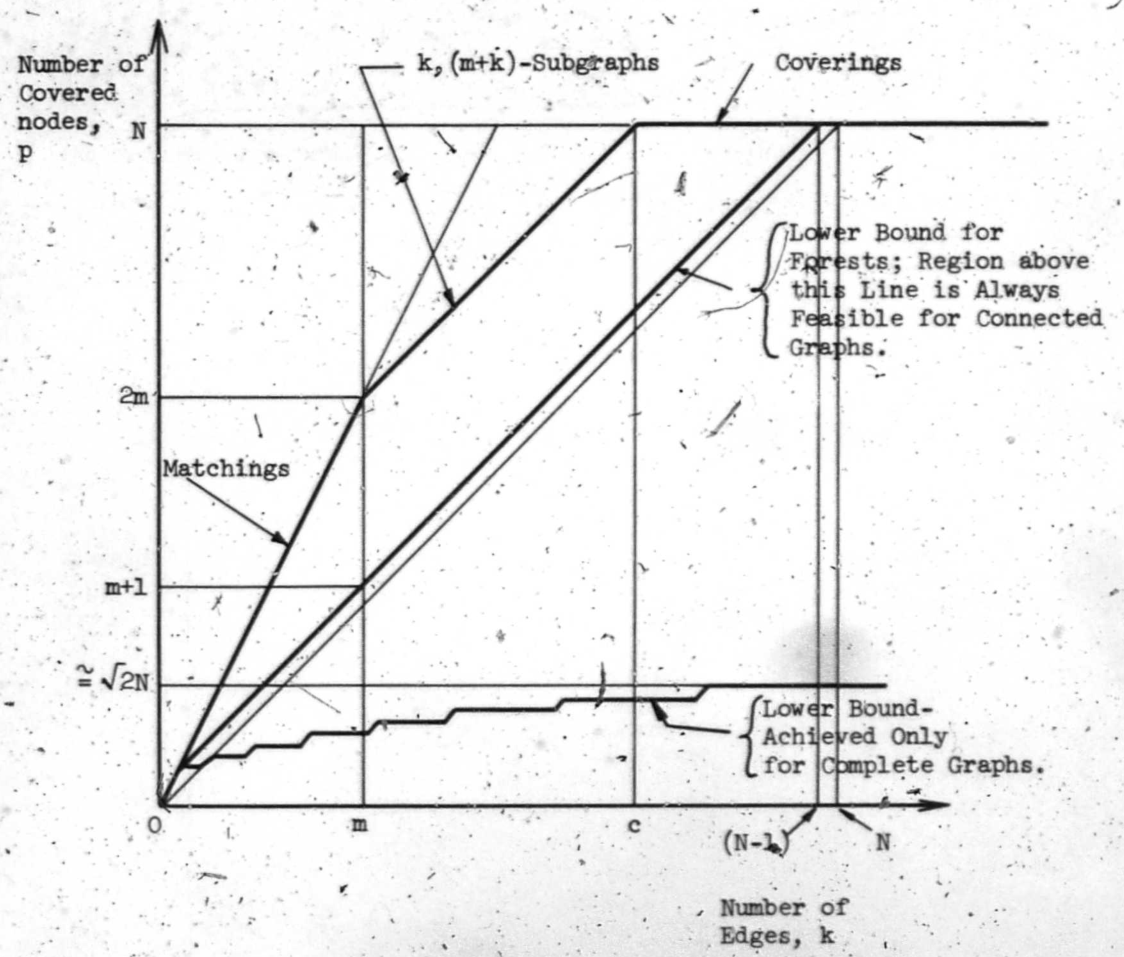
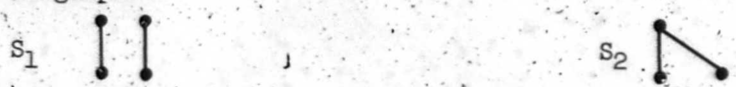
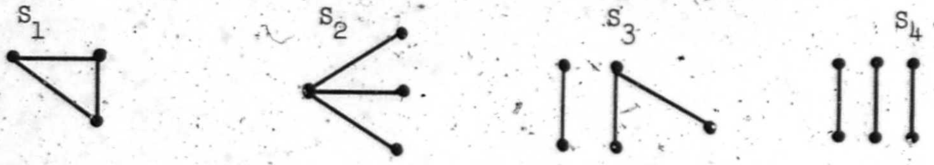

Number of Covered nodes, p
N
2m
m+1
≅ √2N
O
m
c
(N-1)
N
Number of Edges, k
k, (m+k)-Subgraphs
Coverings
Matchings
Lower Bound for Forests; Region above this Line is Always Feasible for Connected Graphs.
Lower Bound- Achieved Only for Complete Graphs.

Figure 6.1 Feasible k,p-Subgraphs

<u>k = 2</u> . p = 3,4 are the only feasible points, corresponding to subgraphs



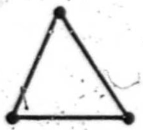
<u>k = 3</u> . p = 3, 4, 5, 6 are feasible points corresponding to subgraphs

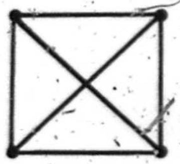


The lowest bounds on feasible p for k,p-subgraphs are valid for complete graphs, and arise from the existence of complete subgraphs, as shown below.

k = 3

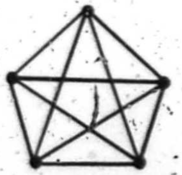
p = 3





k = 4, 5, 6

p = 4



k = 5, 6, 7, 8, 9, 10

p = 5

In general a complete subgraph in G on p nodes will have a lower bound of p for k in the range

$$\frac{(p - 1)(p - 2)}{2} \leq k \leq \frac{p(p - 1)}{2}$$

Thus for large N, the approximate lower bound of the feasible region at k = N is $P_{LB} \cong \sqrt{2N}$ for a complete graph.

The simplest estimate of a p-feasible region is above the line

$$p = k + 1, \text{ for } 1 \leq k \leq N - 1,$$

for connected graphs, and this is a strict lower bound for tree graphs.

6.3 Optimum k,p-Subgraphs

Having determined the values of k and p for which k,p-subgraphs are feasible, consider the question of obtaining the minimum (or maximum) k,p-subgraph.

The maximum k-cardinality matching problem was characterized and solved in Chapter IV using the concept of augmenting paths, as exemplified in Theorem 4.4.

Theorem 4.4

Let M_1 be a maximum k-cardinality matching in a weighted graph G. Select M_2 to be the maximum (k + 1)-cardinality matching such that $|M_1 \oplus M_2|$ is minimal. Then subgraph $M_1 \oplus M_2$ consists of one strong augmenting path.

The minimum k-cardinality covering problem of Chapter V possesses a similar characterization in Theorem 5.1.

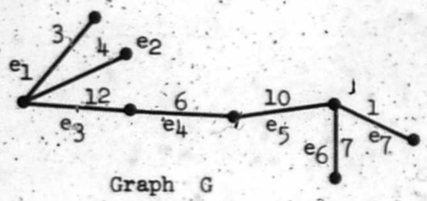
Theorem 5.1

Let C_1 be a minimum k-cardinality covering in a weighted graph G. Select C_2 to be the minimum (k + 1)-cardinality covering in G such that $|C_1 \oplus C_2|$ is minimal. Then subgraph $C_1 \oplus C_2$ consists of one strong reducing path.

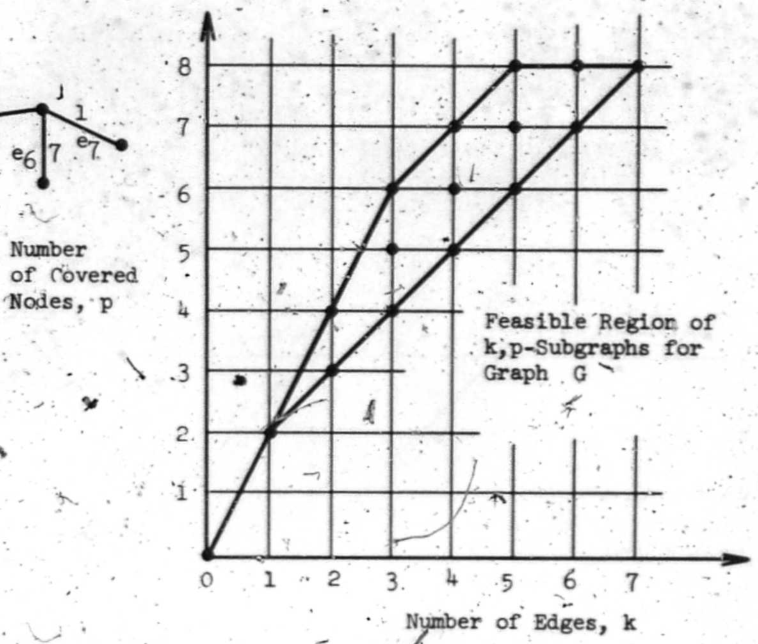
Since both augmenting paths and reducing paths are alternating paths, we might suspect that some type of alternating path would describe the characterization of optimum k,p-subgraphs, and would suggest algorithms for finding these optimum subgraphs. This is not true, however, as seen by the six characterizations below. In each characterization, alternating paths alternate with respect to edges of subgraphs S_1 and S_2.

1) Let S_1 be the unique minimum k,p-subgraph, and S_2 a minimum k,(p + 1)-subgraph, such that $|S_1 \oplus S_2|$ is minimal. Then in general, $S_1 \oplus S_2$ does not consist of a single alternating path.
2) Let S_1 be the unique minimum k,p-subgraph, and S_2 a minimum (k + 1),p-subgraph, such that $|S_1 \oplus S_2|$ is minimal. Then in general, $S_1 \oplus S_2$ does not consist of a single alternating path.
3) Let S_1 be the unique minimum k,p-subgraph, and S_2 a minimum (k + 1),(p + 1)-subgraph, such that $|S_1 \oplus S_2|$ is minimal. Then in general, $S_1 \oplus S_2$ does not consist of a single alternating path.
4) Let S_1 be the unique minimum k,p-subgraph with covered nodes Q_1. For every minimum k,p'-subgraph S_2 with covered nodes Q_2, where $p' > p$ and $|S_1 \oplus S_2|$ minimal, then in general Q_1 is not a subset of Q_2.
5) Let S_1 be the unique minimum k,p-subgraph with covered nodes Q_1. For every minimum k',p-subgraph S_2 with covered nodes Q_2, where $k' > k$ and $|S_1 \oplus S_2|$ minimal, then in general $Q_1 \neq Q_2$.
6) Let S_1 be the unique minimum k,p-subgraph with covered nodes Q_1. For every minimum $(k + \alpha)$, $(p + \alpha)$-subgraph S_2 with covered nodes Q_2, where $\alpha = 1, 2, \ldots$, and $|S_1 \oplus S_2|$ minimal, then in general Q_1 is not a subset of Q_2.

The examples shown in Figure 6.2 illustrate characterizations



Graph G



Edges below:

●——● $S_1 \cap \bar{S}_2$

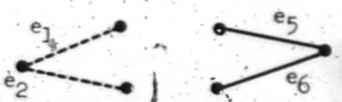
●- - -● $\bar{S}_1 \cap S_2$

a) Example of characterizations (1) and (4):

$k = 3$, $p = 4$, Min $S_1 = \{e_5, e_6, e_7\}$, $w(S_1) = 18$.

$k = 3$, $p = 5$, Min $S_2 = \{e_1, e_2, e_7\}$, $w(S_2) = 8$.

Then subgraph $S_1 \oplus S_2$ is:

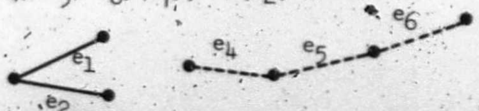


b) Example of characterizations (2) and (5):

$k = 3$, $p = 5$, Min $S_1 = \{e_1, e_2, e_7\}$, $w(S_1) = 8$.

$k = 4$, $p = 5$, Min $S_2 = \{e_4, e_5, e_6, e_7\}$, $w(S_2) = 24$.

Then subgraph $S_1 \oplus S_2$ is:



c) Example of characterizations (3) and (6):

$k = 2$, $p = 3$, Min $S_1 = \{e_1, e_2\}$, $w(S_1) = 7$.

$k = 3$, $p = 4$, Min $S_2 = \{e_4, e_5, e_7\}$, $w(S_2) = 17$.

Then subgraph $S_1 \oplus S_2$ is:

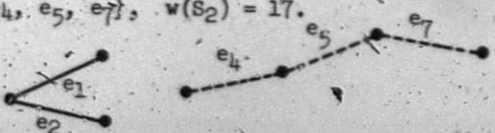


Figure 6.2 . Examples of Characterizations (1) Through (6)

(1) through (6). These examples prove that in general no single alternating path exists in the appropriate symmetric difference subgraph, but further show that the entire concept of an alternating path is no longer useful in describing optimum k,p-subgraphs.

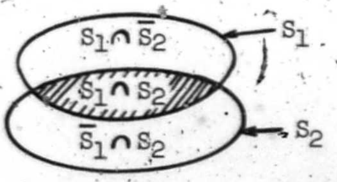
A method to solve for some k,p-subgraphs is given by the following theorem, using notation from Chapter V.

Theorem 6.1

For a given λ, partition the vertices of graph G_λ into two sets V_P, V_N, where V_N is the set of nodes for which at least one negative edge is incident in G_λ. If $|V_N| = p$, and there are k negative edges in G_λ, then these negative edges S_1 form a minimum k,p-subgraph in G.

Proof

Let S_2 represent an arbitrary k,p-subgraph in G. Then for the images of G_λ,



$$w_\lambda(S_1) = w_\lambda(S_1 \cap S_2) + w_\lambda(S_1 \cap \bar{S}_2),$$

$$w_\lambda(S_2) = w_\lambda(S_1 \cap S_2) + w_\lambda(\bar{S}_1 \cap S_2).$$

But by assumption, $w_\lambda(S_1 \cap \bar{S}_2) \leq 0$, $w_\lambda(\bar{S}_1 \cap S_2) \geq 0$, and $w_\lambda(S_1) \leq w_\lambda(S_2)$. Thus since $|S_1| = |S_2| = k$, $w(S_1) \leq w(S_2)$.

This algorithm solves one minimum k,p-subgraph problem for each k, where k = 1, 2, 3, ..., and p changes by zero, one, or two between successive problems. This process is illustrated for two examples in Figure 6.3, and the minimum k,p-subgraph problems solved in each example are noted in the respective feasible k-p region.

6.4 Optimum k,(m + k)-Subgraphs

In Section 6.2, three special types of k,p-subgraphs were defined:

(1) matchings,

(2) coverings, and

(3) k,(m + k)-subgraphs.

These subgraphs were found to form an upper bound on the region of feasible k,p-subgraphs in the k-p plane as shown in Figure 6.1.

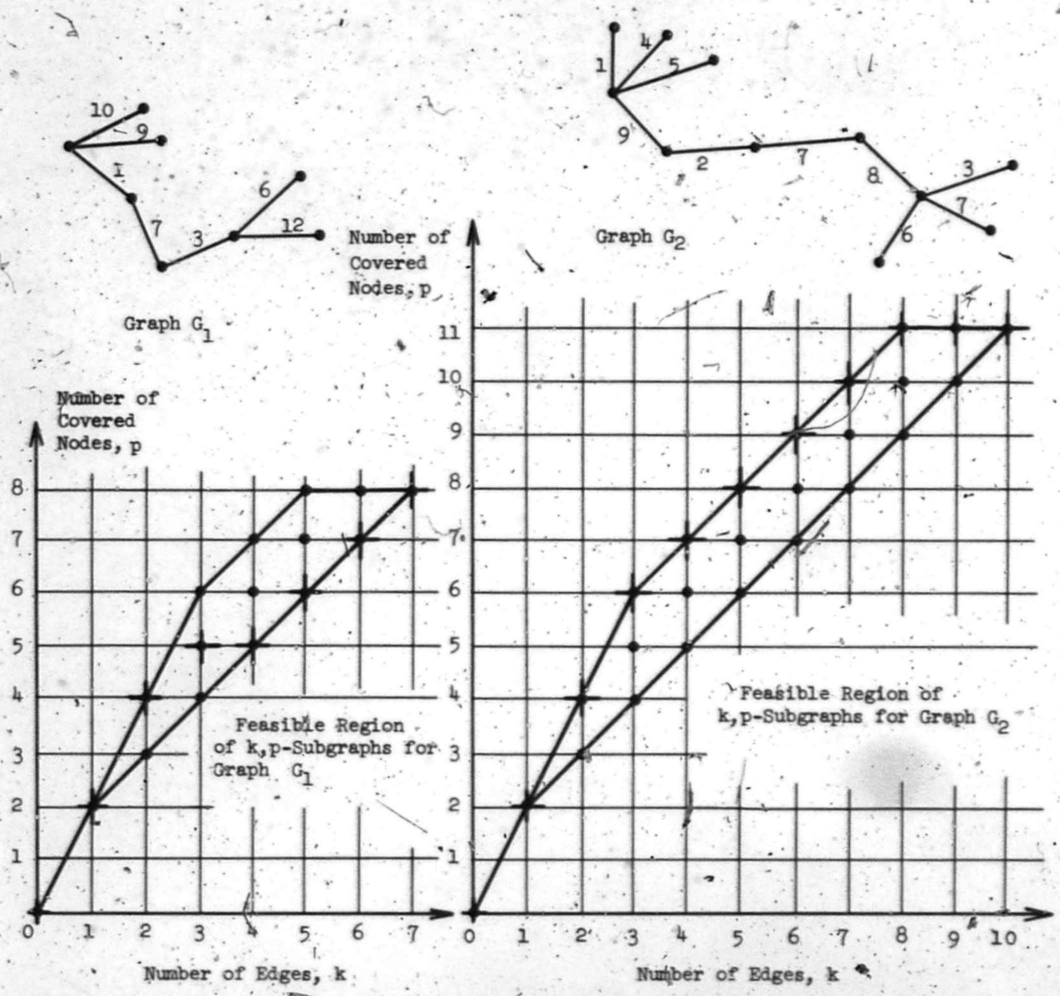
Although no general algorithm was found in Section 6.3 for obtaining optimum k,p-subgraphs, it would seem possible that k,(m + k)-subgraphs are so constrained that an algorithm should exist for the solution of this special case. The following development will show that this is indeed true.

Lemma 6.1

If S is a minimum k,(m + k)-subgraph in a graph G, then S is a star subgraph.

Proof

Suppose S possesses a path of length three or more. Construct a (k - 1), (m + k)-subgraph S' by deleting an intermediate edge in this path. Add (N - m - k) edges incident to the exposed nodes in S'



+ indicates those k,p values for which Theorem 6.1 obtains a minimum k,p-subgraph.

Figure 6.3 Minimum k,p-Subgraphs for Two Examples

to obtain a covering. The number of edges in this covering is then

$$(k - 1) + (N - m - k) = N - m - 1 = c - 1,$$

since $c = N - m$. But this contradicts the assumed minimum cardinality c of any covering, so S is a subgraph of star components.

A <u>$k,(m + k)$-subgraph augmenting path</u> P relative to subgraphs S_1 and S_2 is a path such that:

(1) S_1 is a $k,(m + k)$-subgraph.

(2) S_2 is a $k',(m + k')$-subgraph.

(3) $k' > k$.

(4) P is a path in subgraph $S_1 \oplus S_2$ which alternates with respect to S_1, S_2 edges.

(5) $w(P \cap S_2) - w(P \cap S_1) > 0$

(6) $S' = S_1 \oplus P$ is a $(k + 1),(m + k + 1)$-subgraph.

Throughout this section, we will simply refer to P as an <u>augmenting path</u> where the context of $k,(m + k)$-subgraphs is clear.

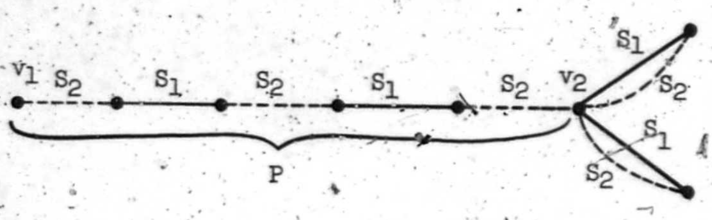
<u>Theorem 6.2</u>

Let S_1 be the unique minimum k, $(m + k)$-subgraph in a graph G, for $m \leq k < c$. Let S_2 be a minimum $(k + 1),(m + k + 1)$-subgraph, such that $|S_1 \oplus S_2|$ is minimal. Then subgraph $S_1 \oplus S_2$ consists of a single augmenting path, and the $(m + k)$ covered nodes of S_1 is a subset of the $(m + k + 1)$ covered nodes of S_2.

<u>Proof</u>

There must exist some node v_1 which is exposed relative to

S_1 but not S_2 in G. Thus there exists some alternating path P with endpoints v_1 and v_2, of maximal length in $S_1 \oplus S_2$. A constructive argument as to the nature of this path, together with examples of various structures, is given in Figure 6.4. The conclusion is that P is an augmenting path of the form:



To show path P comprises all of $S_1 \oplus S_2$, construct

$S_1' = S_2 \oplus P$, a $k, (m + k)$-subgraph.

$S_2' = S_1 \oplus P$, a $(k + 1), (m + k + 1)$-subgraph.

Then since S_1, S_2 are assumed minimum,

1) $w(S_1') \geq w(S_1)$

2) $w(S_2') \geq w(S_2)$.

Identify the weights of S_1', S_2' as

3) $w(S_1') = w(S_2 \cap \overline{P}) + w(S_1 \cap P)$

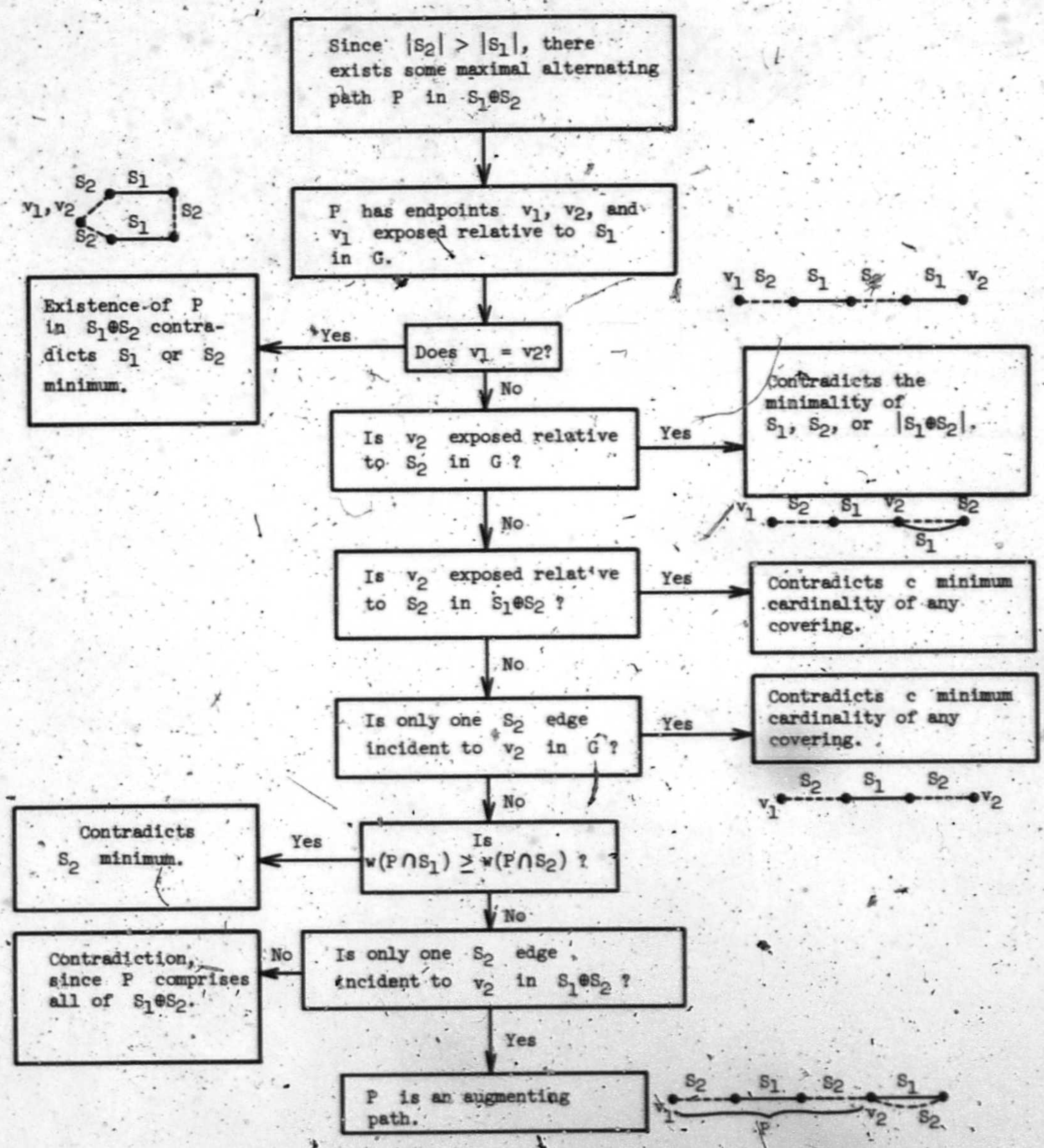
4) $w(S_2') = w(S_1 \cap \overline{P}) + w(S_2 \cap P)$.

Substituting 3) and 4) into 1) and 2), we obtain

5) $$\left.\begin{array}{l} w(S_2 \cap \overline{P}) + w(S_1 \cap P) \geq w(S_1) \\ \\ w(S_1 \cap \overline{P}) + w(S_2 \cap P) \geq w(S_2) \end{array}\right\} \Rightarrow \quad w(S_1 \cap \overline{P}) = w(S_2 \cap \overline{P})$$

By the minimality of $|S_1 \oplus S_2|$, it can be seen that subgraph $S_1 \oplus S_2$ consists only of path P.

Since no new exposed nodes relative to S_2 in G can be



Since $|S_2| > |S_1|$, there exists some maximal alternating path P in $S_1 \oplus S_2$
P has endpoints v_1, v_2, and v_1 exposed relative to S_1 in G.
S_2
S_1
v_1, v_2
S_2
S_1
S_2
Does $v_1 = v_2$?
Yes
Existence of P in $S_1 \oplus S_2$ contradicts S_1 or S_2 minimum.
No
v_1 S_2 S_1 S_2 S_1 v_2
Is v_2 exposed relative to S_2 in G ?
Yes
Contradicts the minimality of S_1, S_2, or $|S_1 \oplus S_2|$.
No
v_1 S_2 S_1 v_2 S_2
S_1
Is v_2 exposed relative to S_2 in $S_1 \oplus S_2$?
Yes
Contradicts c minimum cardinality of any covering.
No
Is only one S_2 edge incident to v_2 in G ?
Yes
Contradicts c minimum cardinality of any covering.
S_2 S_1 S_2
v_1 v_2
No
Is $w(P \cap S_1) \geq w(P \cap S_2)$?
Yes
Contradicts S_2 minimum.
No
Is only one S_2 edge incident to v_2 in $S_1 \oplus S_2$?
No
Contradiction, since P comprises all of $S_1 \oplus S_2$.
Yes
P is an augmenting path.
S_2 S_1 S_2 S_1
v_1 P v_2 S_2

Figure 6.4 Proof of Theorem 6.2

constructed, the set of $(m + k)$ covered nodes of S_1 is a subset of the $(m + k + 1)$ covered nodes of S_2.

A minimum m-cardinality matching M partitions the set of all nodes V into two sets.

V_E = The set of $(N - 2m)$ nodes exposed relative to M.

V_M = The set of 2m nodes matched relative to M.

Theorem 6.3

Assume that the minimum m-cardinality matching M in a graph G is unique. Then no node in V_E is a transmitter in any minimum $k,(m + k)$-subgraph S.

Proof

Figure 6.5 shows node sets V_E, V_M, and that $(k - m)$ nodes of V_E are to be covered by the $k,(m + k)$-subgraph S. As shown in the figure, construct $(k - m)$ alternating paths in subgraph $S \oplus M$ from the appropriate nodes in V_E. The alternating paths are either maximal in subgraph $S \oplus M$, or terminate as they interact with another alternating path. Note that there is one more S edge than M edge in each path, and that an even number of nodes from set V_M are involved in the paths. At least as many edges in S are required to cover the rest of the nodes in V_M as the number of edges in M incident to these nodes.

Thus we have already accounted for k edges in S. If any node in set V_E is a transmitter in S (such as v_o shown in Figure 6.5), this will contradict the assumption that S contains k edges.

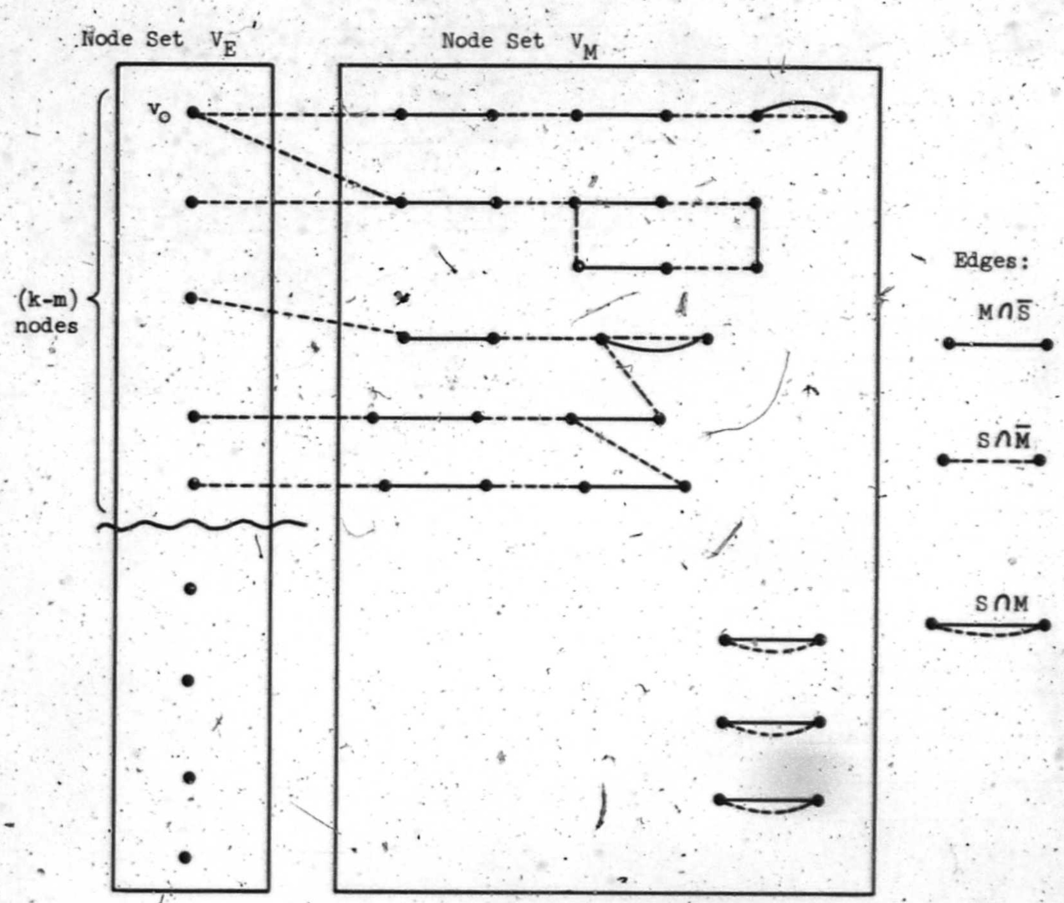


Figure 6.5 Construction for Proof of Theorem 6.3

Theorem 6.3 provides the key result needed to solve the optimum k,(m + k)-subgraph problem. Since the minimum m-cardinality matching can be found, the results of Theorem 6.3 allow a linear programming formulation to be written. This formulation can then be solved by primal-dual methods similar to that exhibited in Chapter IV.

6.5 Specific Relationships Between Matchings and Coverings

Consider the minimum k-cardinality matching problem discussed in Chapter IV, and compare it to the minimum k-cardinality covering problem of Chapter V. Figure 6.6 illustrates a specific relationship between these problems for a general weighted graph.

In Figure 6.6, note that construction 1 given in Section 3.2 for obtaining a matching from a covering shows that $B \geq A$. Construction 1 also shows that the weight of the minimum covering of cardinality $k = c + q$, $q = 0, 1, 2, \ldots$ forms an upper bound for the minimum weight matching of cardinality $k = m - q$. This is seen by comparing curve #3 to curve #1, where the former is simply a translated mirror image of curve #2.

Similar conclusions can be drawn for the case of maximum k-cardinality matching and covering problems.

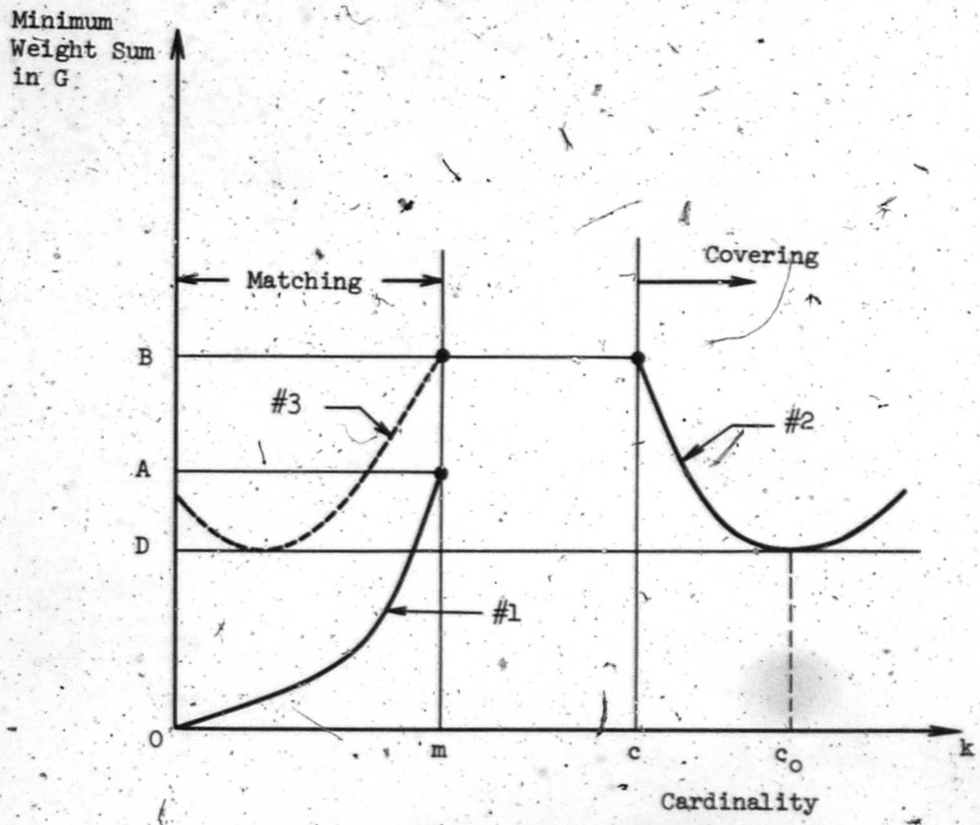


Figure 6.6 Minimum k-Cardinality Matching and Covering for a Weighted Graph G.

CHAPTER VII

MINIMUM SPANNING TREES, GREEDY ALGORITHMS, AND MATROIDS

7.1 Introduction

For the matching and covering problems discussed in this dissertation, the algorithms presented for their solution have been conceptually difficult and complex. One is impelled to search for simpler algorithms.

Edmonds [10] introduces the notion of a "greedy" algorithm. Define an algorithm as greedy if each element requires examination only once, and upon examination, can either be placed in the solution set or permanently discarded. Section 7.2 introduces a general system of independence, called a matroid, developed by Whitney [17]. As shown by Edmonds [9], greedy algorithms often exist for the solution of optimization problems in matroid systems. Section 7.2 also presents an interesting optimization problem and algorithm for such a matroid system.

An important concept in network theory is that of a spanning tree, defined to be a tree incident to every node of a given graph. Kruskal [12] presented an algorithm for finding a minimum spanning tree in a weighted graph. Section 7.3 examines that algorithm, and shows it to be a special case of the solution for matroids presented in Section 7.2.

Section 7.4 demonstrates that the minimum k-cardinality forest covering problem of Section 5.6 is a generalization of the minimum spanning tree problem, and how both problems are related to a greedy algorithm.

7.2 Matroids

Define a matroid M as a finite set M of elements together with a family of subsets, called independent, such that

(1) every subset of an independent set is independent, and

(2) for every subset A of M, all maximal independent subsets of A have the same cardinality, called the rank r(A) of A.

In the definition of a matroid, the concept of independence is defined axiomatically by conditions (1) and (2). It can be seen that the ordinary notion of linear independence satisfies (1) and (2). However, there are systems of independence satisfying axioms (1) and (2) which have no interpretation in the sense of linear independence. Consider the following example.

Example

Assume M a set of N elements, and define an independent set as any set consisting of p elements or less, where we assume $p < N$. This system of independence satisfies axioms (1) and (2), but has no relevance to the concept of linear independence.

Suppose we consider the edges of a graph as set M; is it true that matchings or coverings define appropriate independence systems in order that M might be a matroid? Unfortunately, both systems fail to define a matroid. Though a matching will satisfy axiom (1), axiom (2) fails to be valid. Defining a covering as the independence system insures that (2) holds, but (1) fails immediately. In the course of the development of this chapter, this may suggest from one point of view why the maximum matching and minimum covering problems are so difficult.

Let us consider the following optimization problem, and sub-sequently an algorithm for its solution together with a proof, in order to emphasize this point of view:

Problem

Given a matroid M as a finite set of elements $M = \{e_1, e_2, \ldots, e_N\}$ together with a family of independent subsets. Associate a weight c_i with each element $e_i \in M$ of the matroid. Find a maximal independent subset S_1 of minimum weight, where the weight of the subset S_1 is indicated

$$w(S_1) = \sum_{e_i \in S_1} c_i .$$

Algorithm

(1) Well order elements of M according to increasing weight as $\{e_1, e_2, \ldots, e_N\}$, so

$$i < j \Longrightarrow c_i \leq c_j$$

(2) Initially select S_1 as $\{e_1\}$.

(3) Add e_2 to S_1 if this forms an independent set; otherwise, permanently discard e_2.

(4) Continue adding minimum elements to S_1 in a similar fashion such that S_1 remains independent until

$$|S_1| = r(M).$$

(5) Claim that S_1 is the maximal independent subset of minimum weight of matroid M.

Before proving that this algorithm actually obtains the optimum solution to the problem, consider the following necessary result from Whitney [17].

<u>Theorem 7.1 (Whitney)</u>

Given a matroid M, if S is a maximal independent set, and D an independent set, then for some subset A of S, $(A \cup D)$ is a maximal independent set.

<u>Algorithm Proof</u>

If S_1 is not of minimum weight, then there exists some maximal independent set S_2 of minimum weight, and

$$w(S_2) < w(S_1).$$

If $S_1 \neq S_2$, there exists an element $e^* \epsilon \overline{S}_1 \cap S_2$. Of all such e^*, choose the minimum weight element. But for $w(S_2) < w(S_1)$, it must be that

$$c^* < \underset{e_i \epsilon S_1}{\mathrm{Max}} [c_i].$$

Then e^* must have been considered during the algorithm and rejected because it formed a dependent set; let the minimal such dependent set be

$$c = \{e^*, e_1, e_2, \ldots, e_p\}, \text{ such that}$$
$$c^* > c_i, \text{ for } e_i \epsilon S_1, i = 1, 2, \ldots, p.$$

Let $B = \{e_1, e_2, \ldots, e_p\}$ and note that B is an independent set.

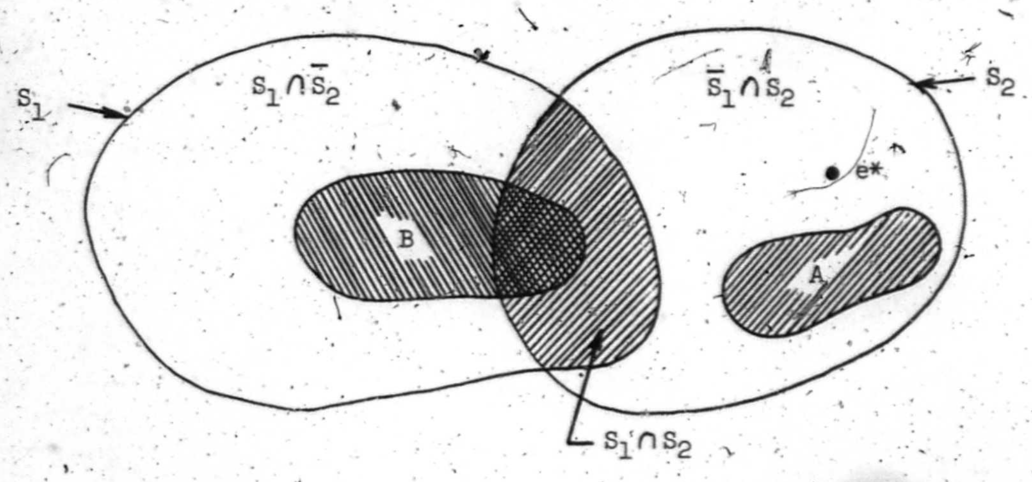


Figure 7.1 Venn Diagram for Algorithm Proof

At this point Theorem 7.1 is used to show that a new maximal independent set S can be constructed from sets B and S_2. Thus let $S = A \cup D$ be a maximal independent set, where $A \subset S_2$, $D = B \cup (S_1 \cap S_2)$. But $e^* \notin S$, for otherwise this would violate the assumption that S is independent.

Now set $(S_2 - S)$ is not a subset of S_1; by assumption, $c^* \leq c_i$, for every element $e_i \epsilon (\bar{S}_1 \cap S_2)$ and thus is true for every element $e_i \epsilon (S_2 - S)$.

Assigning weights,

$$w(S) = w[B - (S_1 \cap S_2)] + w[S_1 \cap S_2] + w(A),$$

$$w(S_2) = w(S_2 - S) + w[S_1 \cap S_2] + w(A).$$

But since sets $(S_2 - S)$, $[B - (S_1 \cap S_2)]$ are non-empty and of the same cardinality, and since for every

$$e_i \in [B - (S_1 \cap S_2)], \text{ and any}$$

$$e_j \in (S_2 - S),$$

$c_i < c^* \leq c_j$, then

$$w(S) < w(S_2).$$

This contradicts the assumption that S_2 was a maximal independent set of minimum weight, and the algorithm is proven.

7.3 Minimum Spanning Trees

Given a connected weighted graph $[G, c_i]$, the minimum spanning tree problem is to find a tree incident to every node of the graph, such that the weight of the tree is minimum. Kruskal [12] developed and proved the following algorithm for this problem:

(1) Well order the edges of G according to increasing weight as $\{e_1, e_2, e_3, \ldots, e_N\}$, so $i < j \Longrightarrow c_i \leq c_j$.

(2) Initially select a tree $T = \{e_1, e_2\}$.

(3) If the addition of e_3 to T forms a cycle, permanently discard e_3; otherwise, add e_3 to T.

(4) Continue adding minimum elements to T in a similar fashion, such that T remains a forest, until the

number of edges in tree T is $(N - 1)$, where N indicates the number of nodes of graph G.

(5) T is a minimum spanning tree of graph G.

This algorithm appears remarkably like the algorithm described in Section 7.2 for matroids. Let us try to couch the spanning tree problem in terms of matroids:

Define a matroid M as the set of edges of a connected weighted graph $[G, c_i]$. Define an independent set as any set of edges which do not form a cycle in graph G; these edges then form a subgraph which is a forest of trees. This definition of independence clearly satisfies matroid axiom (1), and since all maximal independent sets are spanning trees of cardinality $(N - 1)$, axiom (2) is also satisfied.

Thus the minimum spanning tree problem is just a special case of the matroid optimization problem described in Section 7.2, for which there exists a greedy algorithm.

7.4 Coverings and Spanning Trees

In Section 5.3, an algorithm for solving the minimum k-cardinality covering problem was presented, and some implications of that result were explored in Section 5.4. It was pointed out that the Lagrange multiplier λ partitioned the nodes of the graph G_λ into two sets, V_P and V_N, and the edges into sets, P_λ and N_λ. A minimum covering of V_P is then found using P_λ, together with all negative edges in N_λ. The assumption of all negative edges in N_λ can now be viewed as a greedy algorithm. Thus as λ increases, more nodes move into the V_N set. As λ exceeds a critical value λ_{crit}, where

$$\lambda_{crit} = \max_{v_j} \Big[\min_{\substack{\text{all } e_{ij} \\ \text{incident to} \\ \text{node } v_j}} \{c_{ij}\}\Big],$$

all nodes are in the set V_N. The algorithm for the minimum k-cardinality covering becomes entirely greedy, the appropriate solution being all negative edges in graph N_λ.

Section 5.6 presents a similar algorithm for solving the minimum k-cardinality forest covering problem, where a minimum covering of V_P is found in P_λ, together with a minimum forest covering of nodes V_N in N_λ. The minimum forest covering in N_λ can be found by Kruskal's minimum spanning tree algorithm by terminating as soon as a covering of the V_N nodes is obtained; this minimum forest covering will consist entirely of negative edges. Thus this portion of the overall algorithm can be regarded as greedy.

As λ exceeds the value λ_{crit}, the algorithm becomes equivalent to the minimum spanning tree algorithm. It is precisely as the edge of weight λ_{crit} enters Kruskal's constructed set that the edges of this set first form a covering in graph G. Thus it is clear how the minimum k-cardinality forest covering problem is a generalization of the minimum spanning tree problem, where for this special case $k = (N - 1)$, N the number of nodes of the graph.

Still another example of a greedy algorithm is that given for particular minimum k,p-subgraphs in Section 6.3. Though the general problem of optimum k,p-subgraphs remains unsolved, it is interesting that a greedy algorithm exists for these particular configurations.

CHAPTER VIII

APPLICATIONS AND FUTURE RESEARCH

8.1 Introduction

The formulation and efficient solution of optimum matching and covering problems in weighted graphs have been the objectives of this dissertation. In this chapter we would like to consider some applications of these problems and techniques, and to suggest some areas of future research.

Section 8.2 uses the concept of a maximum matching to solve a diagnostics problem for a communications system. The problem can be stated as the Chinese postman's problem, which is discussed in Busacker and Saaty [4]. Section 8.3 introduces an interesting problem of optimum data storage. Though this problem is not completely solved, the optimum matching and covering techniques do offer a partial solution.

In Section 8.4, a communications problem illustrates why a maximum k-cardinality matching would be desired. Section 8.5 applies the solution technique of a minimum k-cardinality covering to a problem of finding the optimum location of service centers.

The design of experiments is considered in Section 8.6, where the matroid theory of Chapter VII and minimum coverings are utilized to obtain an optimum design of experiments.

Some ideas for future research are indicated in Section 8.7.

8.2 A Diagnostics Problem

Given a connected weighted graph G, the Chinese postman's problem [4] is to find a minimum weighted tour. A tour of a graph is a sequence of adjacent edges such that every edge of the graph is in the sequence, and the initial and terminal vertices of the sequence are coincident. If the weighted graph represents a communication system, where an edge weight corresponds to message time delay or cost, the solution to this problem might provide a basis for a diagnostic technique of minimum time delay or cost. A test message could be sent from a particular terminal and routed over each link of the system. Return of the message to the originating terminal without error would give an indication of the present status of the network.

Another diagnostics formulation is relative to a directed graph. We can use this formulation to solve the following problem in a computer system. Associate each arc of the directed graph with a machine state transition together with a cost. Find the minimum cost input sequence such that all state transitions are implemented and checked out. This problem can be solved as a special case of the communications problem defined above, or as a network flow problem.

An Euler graph is a graph in which all vertices are of even degree. An Euler circuit is a tour of the graph in which no edges are repeated. It is well known that a connected graph G contains an Euler circuit if and only if G is an Euler graph. A proof of this result is given in Busacker and Saaty [4].

Thus the solution to the Chinese postman's problem is immediate for an Euler graph. For a general graph, we must decide which edges to duplicate in order that a minimum weighted tour is obtained. An alternate point of view is to duplicate a subset of the edges of minimum weight such that the result is an Euler graph.

Edmonds developed an algorithm which uses maximum matching to obtain the edges to be duplicated.

(1) Identify the set S of nodes of odd degree in the graph G.

(2) Find the shortest paths in G between every pair of nodes in set S.

(3) Create a complete weighted graph G^* using the nodes from S, and weight edges by the shortest path values found in (2).

(4) Find a minimum $\left|\frac{S}{2}\right|$ cardinality matching M in G^* as described in Section 4.9.

(5) Identify the shortest path in G corresponding to each edge in matching M in graph G^*.

(6) The edges of these $\left|\frac{S}{2}\right|$ paths are those to be duplicated for the solution of the Chinese postman's problem.

In a proof that the above algorithm solves the Chinese postman's problem, the principal argument is that all $\left|\frac{S}{2}\right|$ shortest paths selected in G are edge-disjoint. The value of the matching obtained in G^* is equal to the sum of the weights of the duplicated edges in G.

8.3 Optimum Data Storage

Suppose there are N computer locations, all of which demand access to a block of data. We would like to investigate the trade-off between the cost of storage of the data at all locations and transmission costs. Let b_i represent the data storage cost at location i, and c_{ij} the transmission cost between locations i and j. We might then think of this problem relative to a graph G where both nodes and edges are weighted.

Recall from Chapter III the definition of a <u>star subgraph</u> S: a subgraph in which each component is a tree with at most one vertex of degree greater than one. Modify the concept of a <u>transmitter</u> in a star subgraph.

(1) A vertex of degree greater than one is a transmitter.

(2) If a component consists of exactly one edge, assign the endpoint of minimum weight as the transmitter.

(3) A vertex exposed in S is a transmitter.

Define T as the set of all transmitters relative to a star subgraph S. A <u>receiver</u> R is a node in a star subgraph which is adjacent to a transmitter.

Define the cost of a star subgraph S as the sum of the edge weights in S and the sum of the transmitter weights in S, i.e.,

$$\text{cost} = \sum_{e_{ij}\in S} c_{ij} + \sum_{\substack{v_j\in T \\ \text{in } S}} b_j$$

The optimum data storage problem is then to find a star subgraph of minimum cost in a graph G where both nodes and edges are weighted. Balinski [2] proposes an integer program for a delivery problem which is similar to the above formulation. He points out that this is a natural generalization of the covering problem described in Chapter III.

The optimum data storage problem is at present unsolved, i.e., there exists no solution of algebraic growth. However, this dissertation does provide the solution to certain special cases of the problem, and may motivate alternate approaches to the problem.

Consider the special case of the data storage problem where each transmitter vertex is not adjacent to more than one receiver vertex. This restricted problem can then be solved by a maximum matching technique:

(1) Form a weighted graph G^* from G by using edge weights.

$$c^*_{ij} = \text{Max } (b_i, b_j) - c_{ij}.$$

Remove all negative weighted edges from consideration in G^*.

(2) Find a maximum matching M^* in G^*.

(3) The restricted star subgraph of minimum cost is M^*, where for each edge of M^*, the transmitter is the minimum weight vertex in G. The cost is

$$\sum_{e_{ij} \in M^*} c_{ij} + \sum_{v_j \in T} b_j$$

The algorithm can be proved by considering any other matching M.

$$\sum_{e_{ij} \in M^*} c_{ij}^* \geq \sum_{e_{ij} \in M} c_{ij}^*.$$

Then

$$\sum_{e_{ij}\in M^*} \text{Max}\,(b_i, b_j) - \sum_{e_{ij}\in M^*} c_{ij} \geq \sum_{e_{ij}\in M} \text{Max}\,(b_i, b_j) - \sum_{e_{ij}\in M} c_{ij}$$

Add $\sum_{v_j\in G} b_j$ to both side of this inequality and rearrange to obtain

$$\sum_{e_{ij}\in M^*} \text{Min}\,(b_i, b_j) + \sum_{\substack{v_j\in T \\ \text{in } M^*}} b_j \leq \sum_{e_{ij}\in M} \text{Min}\,(b_i, b_j) + \sum_{\substack{v_j\in T \\ \text{in } M}} b_j .$$

This shows M^* is the restricted star subgraph of minimum cost.

Another special case can be addressed by the use of the minimum k-cardinality covering algorithm described in Chapter V. Consider the minimum cost star subgraph problem where all vertex weights b_j are equal, i.e.,

$$b_j = b, \quad \text{for all } v_j\in G.$$

A good upper bound on the solution of this special case can be provided by the maximum k-cardinality covering algorithm of Section 5.3 as follows:

(1) Define a subset V^* of the nodes in graph G which have at least one edge e_{ij} incident for which $(b - c_{ij})$ is positive.

(2) Define a weighted graph G^* using only nodes V^*, and those edges e_{ij} of G for which $(b - c_{ij})$ is positive. Weight each of these edges in G^* by the value $(b - c_{ij})$.

(3) Find the maximum k-cardinality covering in G^* which is of star type by the algorithm of Section 5.3.

(4) The cost of this restricted star subgraph is a good upper bound for the optimum value.

It can be shown that the maximum k-cardinality covering in G^* of the star type is the restricted star subgraph of minimum cost, but the algorithm of Section 5.3 may not obtain this specific configuration. The reason is that at the corresponding cardinality value, the algorithm may obtain a forest which is not a star subgraph.

Further efforts should be made to solve both this special case and the problem of a minimum cost star subgraph.

8.4 A Communications Problem

Given a communications system represented by a weighted graph, where the weights on the edges correspond to channel capacity (bits/sec). Let us assume the system has no multiplexing capability, and thus messages cannot be sent and received simultaneously. Consider the following design questions relative to this system, and how the maximum matching algorithms provide the answers.

(1) What is the maximum number of messages which can be sent concurrently through the system? This can be interpreted as a matching problem, and solved by the maximum cardinality matching algorithm of Chapter II.

(2) What is the maximum channel capacity available for concurrent messages? This question can be answered by a maximum matching approach.

(3) Given the number k of channels desired, what is the

maximum channel capacity available for any k channels? What is the minimum channel capacity for any k channels? This information can provide both upper and lower bounds for any demand situation. Both problems can be solved by the maximum k-cardinality algorithm of Chapter IV.

Though this approach does not provide a basis for the design of an effective communications system, it can offer the evaluation of one more index of performance for an existing design. It was an application of this type which prompted the investigation of optimum k-cardinality configurations.

8.5 Optimum Location of Service Centers

Let a weighted graph be given, where the nodes represent customers to be served, and edge weights correspond to distances between customers. An interesting problem is to locate service centers at these nodes such that the total distance from customers to service centers is minimum. This is a special case of the data storage problem where the vertex weights are zero. This problem can be solved by the minimum covering algorithm of Chapter III.

The number of centers obtained by this formulation may be too large to be practical. With this solution as a lower bound, the minimum k-cardinality algorithm of Chapter V could be used to investigate the trade-off between a decrease in the number of centers and the associated increase in distance.

8.6 Design of Experiments

In the area of design of experiments, one problem is to find an optimum set of experiments such that each factor of interest is associated with some experiment in this set. We shall examine two formulations of this problem.

Consider the parameters of interest to form a Euclidean linear space E^N. Given a set A of p vectors, associate a cost c_i with each vector $\alpha_i \epsilon A$. The vector α_i is interpreted as an experiment, and c_i the cost of that experiment.

The problem is to find a minimum subset B of the experiments in A which spans the factor space E^N such that

$$\sum_{\alpha_i \epsilon B} c_i \text{ is minimum.}$$

It was pointed out in Chapter VII that linear independence satisfies the axioms of independence for matroids. Thus identify a matroid as a set of vectors A, where the independent subsets are those subsets which are linearly independent in the Euclidean space E^N. The problem of finding a minimum subset of experiments can then be solved by the algorithm for matroids given in Section 7.2.

Another formulation of the problem of finding an optimum set of experiments uses concepts of a covering. Let there be given a set of factors, $\Gamma = \{\gamma_1, \gamma_2, \dots, \gamma_N\}$ and a set of experiments $A = \{\alpha_1, \alpha_2, \dots, \alpha_p\}$, where a cost c_i is associated with each experiment α_i. Associate a subset of the factors in Γ with each experiment

α_i. The problem is then to find a subset B of the experiments in A such that each factor $\gamma_j \in \Gamma$ is associated with some experiment $\alpha_i \in B$, and

$$\sum_{\alpha_i \in B} c_i \text{ is minimum.}$$

This problem cannot be interpreted as a weighted graph problem, and is an example of the general covering problem [14].

Consider a special case, where each experiment $\alpha_i \in A$ is associated with two factors or less from set Γ. Form a weighted graph G as follows. Let a node correspond to each factor $\gamma_j \in \Gamma$, and let an edge correspond to each experiment α_i. If an experiment involves two factors γ_i, γ_j, place an edge e_{ij} between nodes v_i and v_j. If an experiment involves only one factor γ_i, place an edge e_{ij} between v_i and a new dummy node v_j. Weight each edge by the corresponding experiment cost c_i.

Then the following optimization problems can be solved for this special case using techniques from Chapters III and V:

(1) Find the fewest experiments from A which will involve all factors from set Γ. This problem can be solved by finding a minimum cardinality covering of Γ nodes in graph G.

(2) Find a subset B of the experiments from A which involves all factors from set Γ, and $\sum_{\alpha_i \in B} c_i$ is minimum. This problem can be solved by finding a minimum covering of Γ nodes in graph G.

(3) Given a specific number k of experiments to be performed, find those experiments which involve all factors from set Γ such that the cost is minimum. This problem can be solved by finding a minimum k-cardinality covering of Γ nodes in graph G.

8.7 Future Research

The purpose of this research was to characterize the solutions of matching and covering problems in weighted graphs and to develop efficient algorithms for finding these solutions. This work naturally suggests further research, primarily in the area of applying these techniques to more complex engineering problems.

(1) Obtain a solution to the general data storage problem. This problem is closely related to the theoretical problem of finding a maximum cardinality covering of the star type.

(2) Develop techniques to solve minimization problems for wiring and integrated circuit technology.

(3) Continue a theoretical investigation of the general covering problem [14]. The graphical covering algorithm of Chapter III solved a special case of the general covering problem. Further research may show that alternating path concepts can be used to solve other special cases of this important problem.

(4) Continue research on minimum coverings in weighted graphs. The minimum k-cardinality covering algorithm

of Chapter V does not utilize reducing paths. It would appear that further research into this problem might show that reducing paths can be used as a basis for an algorithm to find the minimum k-cardinality covering.

(5) Investigate optimum k,p-subgraphs further. It is interesting that the concept of an alternating path does not appear to be applicable here. Some technique might be devised to find optimum k,p-subgraphs, and thus give more insight into the class of matching and covering problems.

(6) Continue work on matroid theory, greedy algorithms, and their applications as described in Chapter VII. Research of this type might lead to the development of algorithms to solve many theoretical and applied problems.

APPENDIX A

LINEAR PROGRAMMING DUALITY THEORY [5]

Let A be a real-valued matrix and b, c, x, y be real-valued vectors.

A primal linear program is of the form:

1) Max cx subject to

2) $Ax \leq b$, $x \geq 0$

The dual linear program is defined relative to a given primal program. The dual of 1) and 2) is:

3) Min by subject to

4) $A^T y \geq c$, $y \geq 0$.

Let X be the set of all vectors which satisfy the linear inequality system 2), and Y the set of all vectors which satisfy 4). A vector $x \epsilon X$ or $y \epsilon Y$ is called a feasible solution with respect to the given constraints.

Given a primal-dual linear system, a pair of vectors x_o, y_o are called orthogonal if

$$x_o(A^T y_o - c) = 0$$

and

$$y_o(b - Ax_o) = 0$$

The following theorems from linear programming theory are used in the text:

Theorem A.1

If x, y are feasible solutions to the given primal-dual linear program, then $cx \leq by$.

Theorem A.2

If x, y are feasible solutions to the given primal-dual linear program, and $cx = by$, then x and y are optimum solutions to the primal and dual systems respectively.

Theorem A.3

If x, y are feasible solutions to the given primal-dual linear program, and x, y are orthogonal, then x and y are optimum solutions to the primal and dual systems respectively.

The set of vectors X is called a convex polyhedron; similarly Y is also a convex polyhedron. A vector x_o is a vertex of the convex polyhedron X if there do not exist two other vectors x_1, x_2 in X such that

$$x_o = \frac{x_1 + x_2}{2} .$$

The following results relate vertices of convex polyhedra to optimum solutions of linear programming problems:

Consider a convex polyhedron X defined by constraints

2) $Ax \leq b, \qquad x \geq 0,$

and some associated cost vector c.

Theorem A.4

The maximum value of cx occurs for $x = x_o$, where x_o is some vertex of the convex polyhedron X.

<u>Theorem A.5</u>

If x_o is a vertex of the convex polyhedron X, then there exists a vector c such that a unique maximum of cx occurs for $x = x_o$.

APPENDIX B

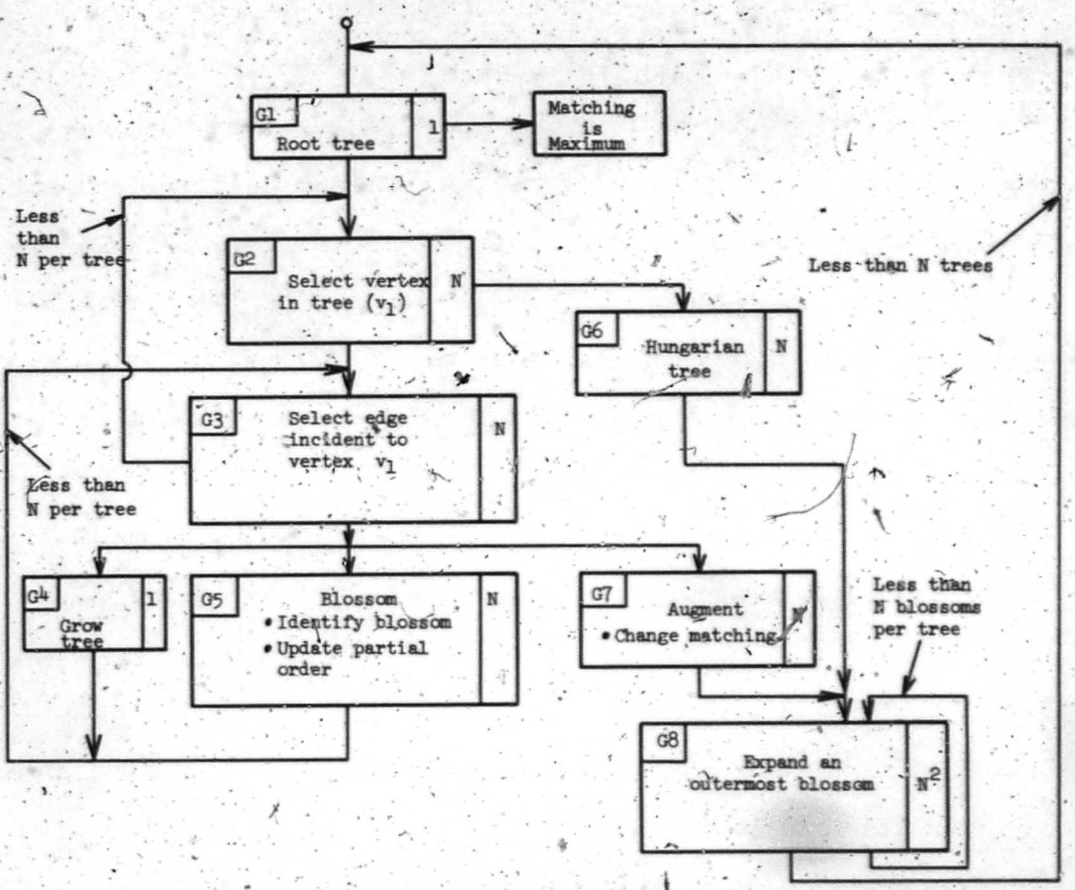
MAXIMUM CARDINALITY MATCHING COMPUTER PROGRAM

In Section 2.3, Edmonds' maximum cardinality matching algorithm [7] was presented. It was mentioned that Edmonds computed the asymptotic growth of the algorithm as N^4, where N represents the number of nodes of the graph. This appendix presents a confirmation of that growth figure, and gives a computer program for the solution of the maximum cardinality matching problem for a given graph G. This computer program was prepared by Robert Urquhart.

A flow chart of this algorithm was indicated in Figure 2.7 of Section 2.3; a breakdown of the algorithm is shown in Figure B-1, where the growth of each operation is indicated. The results of this growth study are also shown in the figure, and indicate the areas in the program of high growth. The program grows as N^4 only in one area, the expansion of blossoms.

In the expansion of blossoms; there is one operation within this routine which requires searching for a simple vertex in one blossom structure connected by an edge to a simple vertex, possibly in another blossom structure. This search is essentially of N^2 growth, and may be repeated N times per tree. Since N trees may be needed, this results in N^4 growth.

A series of complete graphs in the range of $N = 5$ to $N = 35$ were studied by the maximum cardinality matching computer program. Complete graphs were selected for study since the greatest number of blossoms would be necessary, and then subsequent expansion would require



Function	Function Growth	No. of Times Executed	Total Growth
G1	1	N	N
G2	N	N^2	N^3
G3	N	N^2	N^3
G4	1	N^2	N^2
G5	N	N^2	N^3
G6	N	N	N^2
G7	N	N	N^2
G8	N^2	N^2	N^4

Figure B-1 Growth Analysis of the Maximum Cardinality Matching Algorithm a relatively long running time. The results of this study are shown in Figure B.2 and show the growth to correspond to slightly less than N^3. This is due to the fact that a particular vector was stored as a 36-bit word, rather than allowing for a more general vector storage. Thus it is conjectured that N^4 growth is indeed a valid growth figure, but only large values of N would realize this asymptotic behavior.

Cardinality Matching Program

The matching program consists of six routines whose main functions are described below:

(1) Main

(a) Finds a root for the next tree.

(b) Finds an edge between two simple vertices such that the tree can be extended.

(c) Directs the flow of the program.

(d) Determines completion of algorithm and prints results.

(2) Grow

(a) Adds two simple vertices and two edges to the tree.

(b) Records backtrace information.

(3) Blossom

(a) Backtrace two paths from outer vertices to the root to determine Hamilton path HP.

(b) Reclassifies blossom vertices as outer vertices.

(c) Updates B, T, and M vectors.

(d) Computes vertex set (BV(K)) of new blossom.

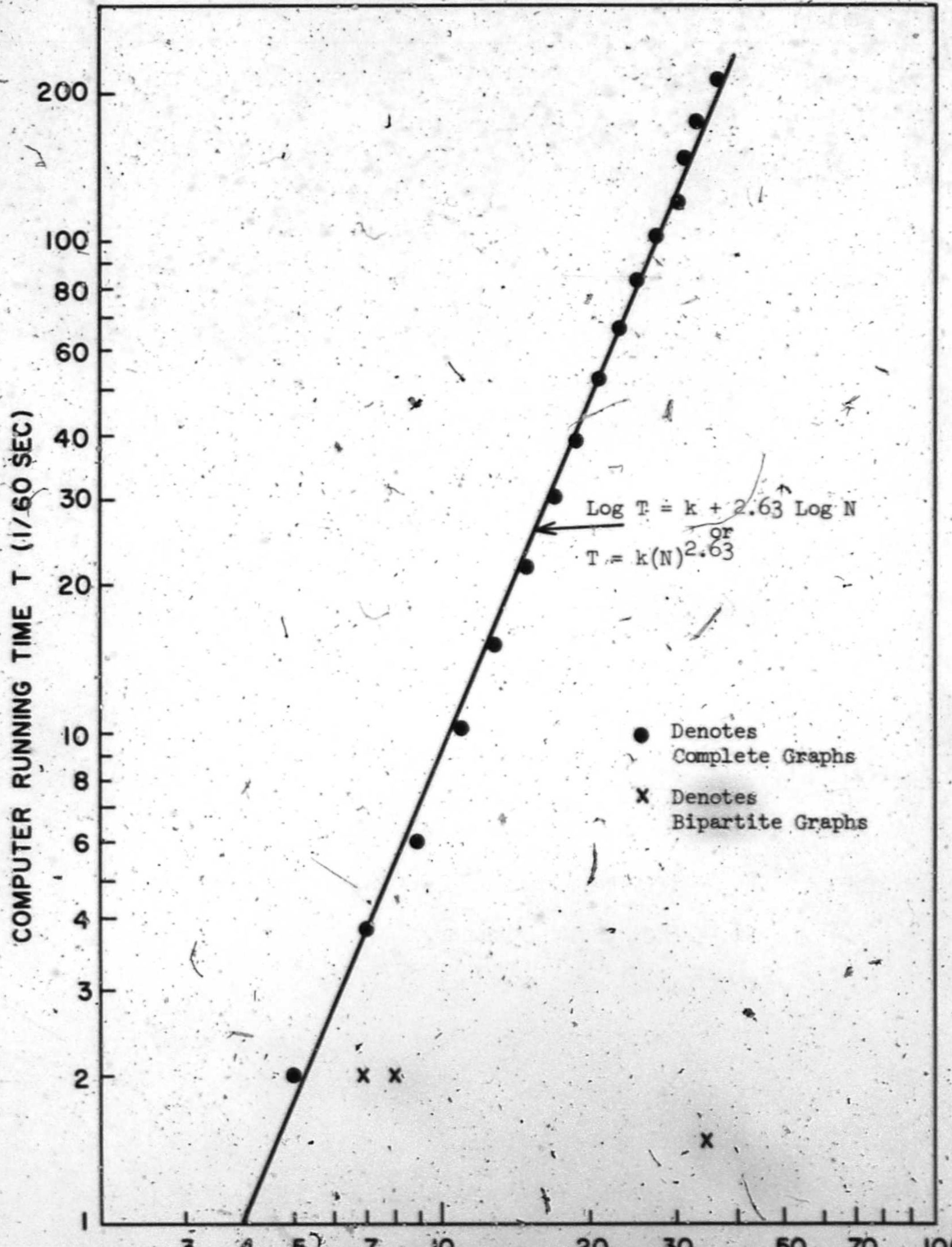


Figure B-2 Computer Study of the Maximum Cardinality Matching Algorithm

(4) Augment

(a) Backtraces path to root.

(b) Adjusts matching along path.

(5) Expand

Expands all blossoms into simple vertices.

(6) Hungarian

Tags vertices of Hungarian tree so they will not enter into any subsequent trees.

DATA STRUCTURES

<u>Input Data</u>

N - An integer from 1 to 36.

A_i - 0,1 vector of N bits; gives the vertices connected to vertex v_i.

<u>Internal Data</u>

C(I) - The classification of simple vertices:

E is exposed, H is Hungarian,

IN is inner, OUT is outer,

SOUT is scanned outer,

NU is neuter-this is a vertex which is matched, but is not in the tree.

T(I) - Backtrace information for the tree.

I runs from 1 to 2N; blossoms are numbered from N + 1 to 2N. One word per vertex or blossom.

B(I)	- Outermost blossom containing simple vertex I. If I is not in a blossom then B(I) = I. If a blossom K is an outermost blossom, then B(K) = K. If a blossom is not being used then B(K) = 0.
M(I)	- Matching information. One word per vertex or blossom.
HP(K,L)	- Hamilton path storage. N words reserved from each blossom. Each word is the vertex or blossom of the Hamilton Path.
BV(K)	- The simple vertices of blossom K.
LP(K)	- The length of the K^{th} Hamilton path.

```
$ COMPILE MAD, EXECUTE, PRINT OBJECT

MAD (06 JAN 1967 VERSION) PROGRAM LISTING ... ... ...

               DIMENSION A(36),C(36),T(72),M(72),HP(36*36),BV(72),LP(36)
               DIMENSION P(36), Q(36), B(72)
               NORMAL MODE IS INTEGER
 L0            READ AND PRINT DATA
               E=1
               H=2
               IN=3
               OUT=4
               SOUT=5
               NU=6
               ST=DAYTIM.(0)
               LM=1
                THROUGH L1, FOR I=1,1,I.G.2*N
               T(I)=0
               M(I)=0
               WHENEVER I.G.N, TRANSFER TO L1A
               B(I)=I
               C(I)=E
               BV(I)=LM
               LM=2*LM
 L1             CONTINUE
               WHENEVER CG.NE.1, TRANSFER TO LC1
               THROUGH LC2, FOR I=1,1,I.G.N
               A(I)=.N.BV(I)
 LC2           CONTINUE
 LC1           CG=0
               THROUGH L2, FOR EX=1,1,EX.G.N
               WHENEVER C(EX).E.E,TRANSFER TO L3
 L2            CONTINUE
               SP=DAYTIM.(0)
               DT=SP-ST
               PRINT COMMENT $ TIMES
               PRINT RESULTS DT
                THROUGH L52, FOR I=1,1,I.G.N
 L52           PRINT RESULTS M(I)
               TRANSFER TO L0
 L1A           B(I)=0
               BV(I)=0
               TRANSFER TO L1
 L3            ROOT=EX
               C(EX)=OUT
 L4             THROUGH L5, FOR IV=1,1,IV.G.N
               WHENEVER C(IV).E.OUT,TRANSFER TO L6
 L5            CONTINUE
               TRANSFER TO HUNG
 L6            THROUGH L8, FOR JV=1,1,JV.G.N
               WHENEVER A(IV).A.BV(JV).E.0,TRANSFER TO L8
                WHENEVER C(JV).E.E
                TRANSFER TO AUG
                OR WHENEVER C(JV).E.NU
                TRANSFER TO GROW
                OR WHENEVER C(JV).E.OUT
                TRANSFER TO L9
```

```
         OTHERWISE
         TRANSFER TO L8
         END OF CONDITIONAL
L8       CONTINUE
         C(IV)=SOUT
         TRANSFER TO L4
L9       WHENEVER B(IV).E.B(JV), TRANSFER TO L8
         TRANSFER TO BLOSS
GROW     AA=B(IV)
         BB=B(JV)
         CC=M(BB)
         T(BB)=AA
         T(CC)=BB
         C(BB)=IN
         C(CC)=OUT
         TRANSFER TO L8
HUNG     THROUGH L13, FOR I=1,1,I.G.N
         WHENEVER C(I).E.SOUT.OR.C(I).E.IN, TRANSFER TO L14
L13      CONTINUE
         TRANSFER TO EXPAND
L14      C(I)=H
         TRANSFER TO L13
AUG      C(JV)=OUT
         BB=B(IV)
         AA=B(JV)
L15      M(AA)=BB
         M(BB)=AA
         WHENEVER BB.E.ROOT, TRANSFER TO EXPAND
         AA=T(BB)
         BB=T(AA)
         TRANSFER TO L15
BLOSS    THROUGH L16, FOR KK=N+1,1,KK.G.2*N
         WHENEVER B(KK).E.0,TRANSFER TO L17
L16      CONTINUE
         PRINT COMMENT $ L16 ERROR$
         TRANSFER TO L0
L17      K=KK-N
         P(1)=B(IV)
         THROUGH L18, FOR IP=1,1,P(IP).E.ROOT
L18      P(IP+1)=T(P(IP))
         Q(1)=B(JV)
         THROUGH L19, FOR IQ=1,1,Q(IQ).E.ROOT
L19      Q(IQ+1)=T(Q(IQ))
L22      WHENEVER IQ.E.1.OR.IP.E.1,TRANSFER TO L20
         WHENEVER P(IP-1).NE.Q(IQ-1),TRANSFER TO L
L21      IP=IP-1
         IQ=IQ-1
         TRANSFER TO L22
L20      L=1
L20A     HP(K,L)=P(IP)
         WHENEVER IP.E.1, TRANSFER TO L23
         L=L+1
         IP=IP-1
         TRANSFER TO L20A
L23      I=0
L24      WHENEVER I.E.IQ-1,TRANSFER TO L25
         L=L+1
         I=I+1
         HP(K,L)=Q(I)
         TRANSFER TO L24
```

```
L25          LP(K)=L
             BV(KK)=0
L31          WHENEVER HP(K,1).E.ROOT, TRANSFER TO L32
             AA=HP(K,1)
             BB=T(AA)
             M(KK)=BB
             M(BB)=KK
             T(KK)=BB
L33          B(KK)=KK
             THROUGH L26, FOR I=1,1,I.G.L
             AA=HP(K,I)
             B(AA)=KK
             M(AA)=0
L26          T(AA)=0
             THROUGH L27, FOR J=1,1,J.G.2*N
              WHENEVER B(J).E.0, TRANSFER TO L27
             WHENEVER B(B(J)).E.KK, TRANSFER TO L28
             WHENEVER T(J).E.0, TRANSFER TO L27
             WHENEVER B(T(J)).NE.KK, TRANSFER TO L27
             T(J)=KK
L27          CONTINUE
             TRANSFER TO L8
L32          ROOT=KK
             TRANSFER TO L33
L28          B(J)=KK
             BV(KK)=BV(KK).V.BV(J)
             WHENEVER J.G.N, TRANSFER TO L27
             C(J)=OUT
             TRANSFER TO L27
EXPAND       THROUGH L34, FOR KK=N+1,1,KK.G.2*N
              WHENEVER B(KK).E.KK, TRANSFER TO L38
L34          CONTINUE
L35          THROUGH L36, FOR I=1,1,I.G.N
             WHENEVER C(I).E.E.OR.C(I).E.H,TRANSFER TO L36
             C(I)=NU
L36          CONTINUE
             THROUGH L37, FOR I=1,1,I.G.2*N
L37          T(I)=0
              TRANSFER TO L2
L38          K=KK-N
             WHENEVER M(KK).E.0, TRANSFER TO L39
             THROUGH L40, FOR J=1,1,J.G.N
L42          WHENEVER B(J).E.KK, TRANSFER TO L41
L40           CONTINUE
             PRINT COMMENT $ L40 ERROR$
             TRANSFER TO L0
L41          WHENEVER A(J).A.BV(M(KK)).E.0, TRANSFER TO L40
             THROUGH L43, FOR L=1,1,L.G.LP(K)
L44          BB=HP(K,L)
             WHENEVER BV(BB).A.BV(J).E.0, TRANSFER TO L43
             M(BB)=M(KK)
             M(M(KK))=BB
             M(KK)=0
             TRANSFER TO L46
L43          CONTINUE
             PRINT COMMENT $ L43 ERROR$
             TRANSFER TO L0
L46          B(KK)=0
              B(BB)=BB
             I=L
```

```
                  TRANSFER TO L47
    L39           BB=HP(K,1)
                  L=1
                  TRANSFER TO L46
    L47           WHENEVER I.E.LP(K), TRANSFER TO L55
    L49           I=I+1
                  WHENEVER I.E.L, TRANSFER TO L59
                  AA=HP(K,I)
                  WHENEVER I.E.LP(K), TRANSFER TO L50
    L51           I=I+1
                  BB=HP(K,I)
                  M(AA)=BB
                  M(BB)=AA
                   B(AA)=AA
                  B(BB)=BB
                  TRANSFER TO L47
    L55           I=0
                  TRANSFER TO L49
    L50           I=0
                  TRANSFER TO L51
    L59           THROUGH L56, FOR I=1,1,I.G.2*N
                  WHENEVER B(I).E.KK, TRANSFER TO L57
    L56           CONTINUE
                  TRANSFER TO EXPAND
    L57           THROUGH L58, FOR L=1,1,L.G.LP(K)
                  WHENEVER BV(I).A.BV(HP(K,L)).E.0, TRANSFER TO L58
                  B(I)=HP(K,L)
                  TRANSFER TO L56
    L58           CONTINUE
                  PRINT COMMENT $ L57 ERRORS
                  TRANSFER TO LO
                  END OF PROGRAM

THE FOLLOWING NAMES HAVE OCCURRED ONLY ONCE IN THIS PROGRAM.
COMPILATION WILL CONTINUE.

    L21            *099
    L31            *116
    L35            *146
    L42            *156
    L44            *162
```

APPENDIX C

REDUCING PATHS AND MINIMUM COVERINGS

The Berge [3] Theorem 2.1 involving augmenting paths provided the basis for Edmonds' maximum cardinality matching algorithm [7]. This notion is easily generalized to weighted augmenting paths, and Theorem 2.5 uses this concept to characterize maximum matchings. Augmenting paths are always simple.

A similar result exists for minimum coverings, but reducing paths are more complex, and lead to diverse structures. The complications arise when either the initial or terminal vertices of the path are coincident with intermediate vertices of the path. The minimum covering algorithm of Chapter III does not use reducing paths.

Define a <u>reducing path</u> P relative to a covering C in a graph G as a path such that:

1) Edges are alternately in C, and not in C.
2) $C' = C \oplus P$ is a covering in G.
3) The weights of the edges in P are such that $w(P \cap \overline{C}) - w(P \cap C) < 0$.
   Hence C' is a smaller weight covering than C.
4) P is minimal with respect to 1), 2), and 3).

The following theorem is adapted from a presentation by Edmonds in [6].

Theorem C.1

If C is not a minimum covering in a weighted graph G, then there exists a reducing path P in G.

Proof

Throughout the proof, we shall be making repeated reference to conditions (1), (2), (3), and (4) in the definition of reducing path.

Since $w(C)$ is not minimum over all possible coverings, there exists a covering K of smaller weight. Among those coverings K, choose one for which

$$w[(C-K) \cup (K-C)] \quad \text{is minimum.}$$

Later this will be shown to be an extremely judicious choice, leading to a covering K which differs from C only in a single reducing path.

Since K, C are not identical coverings, the set $(C-K) \cup (K-C)$ is non-empty. Thus an alternating path P satisfying condition (1) of the definition of reducing path can be constructed.

Also we can show by a Venn set diagram that relative to some path P satisfying (1), the following conditions are also satisfied:

$$(C \cap P) \subset (C-K) \quad \text{and} \quad (\bar{C} \cap P) \subset (K-C),$$

$$\text{also} \qquad C \cap P = \bar{K} \cap P \quad \text{and} \quad \bar{C} \cap P = K \cap P$$

Among such paths described above, choose one P with a maximum number of edges. We first show that P satisfies (2), i.e., that

$$C' = C \oplus P$$

formed from P actually is a covering. This would guarantee that some edge of C' is incident to every node of graph G.

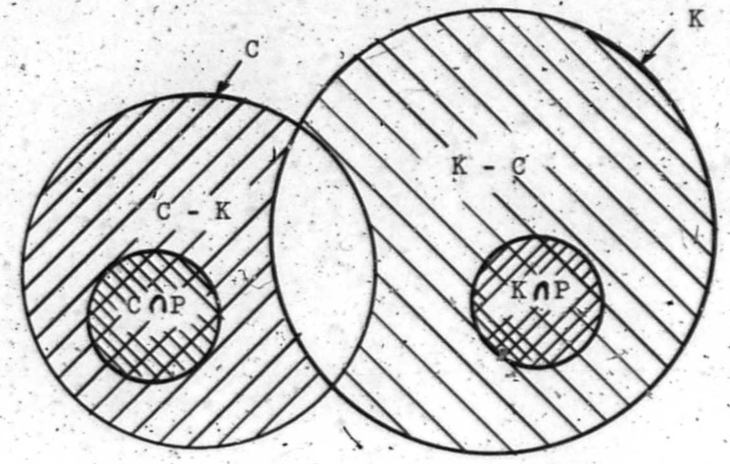


Figure C.1 Venn Diagram for C,K,P.

Consider a partition of the vertices of G relative to the path P as follows:

<u>Case i: Vertices not in path P</u>

Those vertices not in P are covered by edges in $C \cap \bar{P} \subset C'$, hence are covered by C'.

<u>Case ii: Intermediate vertices relative to path P</u>

From the definition of a reducing path, each intermediate vertex v is incident to an edge in $(\bar{C} \cap P) \subset C'$, hence are covered by C'.

<u>Case iii: Initial and terminal vertices of P</u>

Since both C and K are coverings, we can identify three cases which exhaust all possible ways in which an initial (or terminal) vertex v_1 can be incident to edges of C and K. All edges mentioned below are incident to v_1 (see Figure C.1).

a) There exists $e_1 \epsilon C \cap \overline{P}$.

In this case $e_1 \epsilon C'$, therefore v_1 is covered by C'.

b) There does not exist $e_1 \epsilon C \cap \overline{P}$, but there exists $e_2 \epsilon C \cap P$, and $e_3 \epsilon K \cap P$.

In this case $e_3 \epsilon (K \cap P) \subset C'$, thus v_1 is covered by C'.

c) There does not exist $e_1 \epsilon C \cap \overline{P}$, but there exists $e_2 \epsilon C \cap P$; there does not exist $e_3 \epsilon K \cap P$, but there exists $e_4 \epsilon K \cap \overline{C} \cap \overline{P}$. In this case e_4 can be added to P forming a larger reducing path, and thus violating the assumed maximality of P. Thus this case can not occur.

Thus C' is indeed a covering. Similarly, we can show that $K' = (K \cap \overline{P}) \cup (\overline{K} \cap P)$ is a covering.

If we suppose that condition 3) is <u>not true</u>, i.e.,

$$w(\overline{K} \cap P) = w(C \cap P) \leq w(K \cap P) = w(\overline{C} \cap P),$$

then K' as well as K would be a smaller weighted sum covering than C. To show this, expand

$$K' = (K \cap \overline{P}) \cup (\overline{K} \cap P).$$

But $(\overline{K} \cap P) = (C \cap P)$, and since this is a disjoint union (see Figure C.1),

$$w(K') = w(K \cap \overline{P}) + w(C \cap P).$$

Using the supposition,

$w(C \cap P) \leq w(K \cap P)$, we get

$$w(K') \leq w(K \cap \overline{P}) + w(K \cap P) = w(K) < w(C).$$

But this will contradict our original assumption that $w[(C-K) \cup (K-C)]$ is minimal over all possible coverings of weight less than $w(C)$.

To show this is a contradiction, recall K' is a covering, $K' = (K \cap \bar{P}) \cup (C \cap P)$, and form, using Figure C.1,

$$(C-K') \cup (K'-C) = \{[C - (K\cap\bar{P}) \cup (C\cap P)] \cup [(K\cap\bar{P}) \cup (C\cap P) - C]\}$$
$$= [C-K-P] \cup [K-C-P].$$

Now consider

$$w[(C-K') \cup (K'-C)] = w\{[C-K-P] \cup [K-C-P]\}.$$

But since the union is over disjoint sets, and we assume positive edge weights,

$$w[(C-K') \cup (K'-C)] = w[C-K-P] + w[K-C-P]$$
$$< w(C-K) + w(K-C).$$

We have arrived at a contradiction of the way in which K was selected! Thus condition (3) must hold.

Condition (4) follows from the requirement that covering K be such that $w[(C-K) \cup (K-C)]$ <u>is minimum</u>. However, this can more easily be seen from the corollary to follow. QED.

In the above theorem, it was only assumed that $(C\cap P) \subset (C-K)$, with the implication that given C, and our choice of K, that there might in general be more paths available than just P. We would like to ask if for the judicious choice of K which was made, i.e.,

$$\begin{cases} \text{Choose } K \text{ a covering such that } w(K) < w(C), \\ \text{and } w[(C-K) \cup (K-C)] \text{ as small as possible,} \end{cases}$$

is there only <u>one</u> such path P satisfying the conditions of Theorem

C.1? The following corollary answers this question in the affirmative, and shows that in the proof of the theorem, it was really not necessary to choose "the path with the greatest number of nodes". In reality there existed only one.

<u>Corollary</u>

Given a covering C in a weighted graph G, which is <u>not</u> minimum. Find a covering K such that

$$w(K) < w(C) \quad \text{and} \quad w[(C-K) \cup (K-C)] \quad \text{is minimum.}$$

Form a reducing path P as in Theorem C.1; then $(C \cap P) = (C-K)$.

Relative to the Venn diagram in Figure C.1, this implies that the path P elements of C "fill up" or occupy <u>all of set $(C-K)$</u>.

<u>Proof</u>

We know initially only that $(C \cap P) \subset (C-K)$ and $(K \cap P) \subset (K-C)$ are valid by the construction of the reducing path P.

Since $C' = (C \cap \overline{P}) \cup (K \cap P)$ was shown in the proof of Theorem C.1 to be a covering, form

$$\begin{aligned}(C-C') \cup (C'-C) &= \{[C - (C \cap \overline{P}) \cup (K \cap P)] \cup [(C \cap \overline{P}) \cup (K \cap P) - C]\} \\ &= [C \cap P] \cup [K \cap P].\end{aligned}$$

But by our selection of the covering K (instead of C'),

$$w[(C-C') \cup (C'-C)] \geq w[(C-K) \cup (K-C)].$$

By expanding on respectively disjoint sets from Figure C.1,

$$w(C \cap P) + w(K \cap P) \geq w(C-K) + w(K-C).$$

But since all edge weights are assumed positive, and

$$(C \cap P) \subset (C-K), \quad (K \cap P) \subset (K-C),$$

we have $w(C \cap P) \leq w(C-K)$, $w(K \cap P) \leq w(K-C)$.

By comparison of these inequalities, together with the assumption of no zero weights on any edges, equality must hold everywhere, and specifically we get:

$$(C \cap P) = (C - K), \quad (K \cap P) = (K - C) \qquad \text{QED.}$$

BIBLIOGRAPHY

1. M. L. Balinski, "Integer Programming: Methods, Uses, Computations," Management Science, Vol 12, No. 13, pp. 253-313 (1965).

2. M. L. Balinski and R. E. Quandt, "On an Integer Program for a Delivery Problem," Operations Research, Vol 12, pp. 300-304 (1963).

3. C. Berge, "Two Theorems in Graph Theory," Proc, Natl. Acad Sci US, 43, pp. 842-844 (1957).

4. R. G. Busacker and T. L. Saaty, "Finite Graphs and Networks," McGraw-Hill, Inc., New York, 1965.

5. G. B. Dantzig, "Linear Prog[illegible]ing and Extensions," Princeton University Press, 1963.

6. J. Edmonds, "Covers and Packings in a Family of Sets," Bull Amer Math Soc, 68, pp. 494-499 (1962).

7. J. Edmonds, "Paths, Trees, and Flowers," Canadian J Math, 17, pp. 449-467 (1965).

8. J. Edmonds, "Maximum Matching and Polyhedron with 0,1--Vertices," J Res NBS, Vol 69B, Nos. 1 and 2, pp. 125-130 (1965).

9. J. Edmonds, "Minimum Partition of a Matroid into Independent Subsets," J Res NBS, Vol 69B, Nos. 1 and 2, pp. 67-72 (1965).

10. J. Edmonds, "Optimum Arborescences," The International Seminar on Graph Theory and Its Applications, Rome, July, 1966.

11. H. Everett III, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," Operations Research, Vol 11, pp. 399-417 (1963).

12. J. B. Kruskal, Jr., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," Proc Amer Math Soc, 7, pp. 48-50 (1956).

13. H. W. Kuhn, "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, Vol 2, pp. 83-97 (1955).

14. E. L. Lawler, "Covering Problems: Duality Relations and a New Method of Solution," J. SIAM, Vol 14, pp. 1115-1132 (1966).

15. R. Z. Norman and M. O. Rabin, "An Algorithm for a Minimum Cover of a Graph," Proc Amer Math Soc, Vol 10, pp. 315-319 (1959).

16. W. T. Tutte, "The Factorization of Linear Graphs," J London Math Soc, Vol 22, pp. 107-111 (1947).

17. H. Whitney, "On the Abstract Properties of Linear Dependence," Amer J Math, Vol 57, pp. 509-533 (1935).