NEAR EAST UNIVERSITY

1988

FACULTY OF COMPUTER ENGINEERING DEPARTMENT

GRADUATION PROJECT

Submitted : GÜLSER DEMİR
Std. No : 940065

CONTENTS

1.1 SOME GRAPH THEORY CONCEPTS

A graph $G = (V, E)$ consists of a finite nonempty set V and a collection E of unordered pairs from V (i.e. two-element subsets of V). Every element in V ist two paths from u to i. If one and each unordered pair in E is called an edge of the graph. The edge $e = \{x, y\}$ is an edge between the two vertices x and y which is incident to both x and y. Two vertices are adjacent to e deletion of this arc will not affect the quasi-strong connectise is said to join the two vertices. Two or more edges that join the same pair of vertices are called parallel edges. A graph n that a weighted digraph has ansimple graph, in which case the collection E is a set. Otherwise G is multigraph. A graph G' is a subgraph of the graph G if every vertex of G' is a vertex of G and every edge of G' is an edge of G.

A directed graph or digraph consists of a finite set V of vertices and a collection A of ordered pairs from V called the arcs of the digraph. The t the greedy procedure and choos is a set. If $a = (x, y)$ is an ated to each vertex other than thrtex x to the vertex y and is incident (adjacent) from x and incident (adjacent) to y. Two vertices are nonadjacent if there is no arc from one to the other. The underlying graph G of a digraph D is the graph G obtained from D by replacing each arc (x, y) by an edge $\{x, y\}$.

In this book, unless otherwise mentioned, all graphs are simple graphs and all digraphs are simple digraphs. If we associate one or more real numbers with each edge (or arc) of a graph (or a digraph), the resulting structure is known as a weighted graph or a longer acyclic. Thus the problem is to obtain a procedure to obtain *a* minimum weights arboresch is a sequence *x1, e1, x2, e2, he subgraph obtained by* the gree*1, x2, x3,* . . are

vertices and *ek* is the edge between *xk- 1* and *xk* for *k =1, 2, . . . , r*. This path is represented as *(x 1, x2, x3, . . . , x,)* or as *x1-x2-x3-···-x*, without mentioning the edges explicitly. If *x1 = xr* it is a closed path. It is a simple path between *xl* and *xr* if *x1,x2, . . . , xr* are distinct. A circuit in a graph is a closed path in which the edges are all distinct. A cycle is a circuit in which the vertices are all distevery cycle C in H.

Proof

Obvio circuit. A cycle with *k* vertices *(k ≥ 3)* has *k* edges and is calcycle C in *H*.Let *e' = (p, q)* and *f' = (r, s)*. In T* there i Two vertices *x* and *y* in a graph are connected to each other if there is a path between them. A graph is a connected graph if there is a path between every pair of vertices in it. Otherwise it is a disconnected graph. A connected subgraph *H* of the graph *G* is called a component of *G* if there is no connected subgraph *H'* (other than *H*) such that *H* is a subgraph of *H'*.

If *G = (V, E)* and *E = E' ∪ {e}*, then *G' = (Vet of* arcs obtained from *A** by rned from *G* by deleting the edge *e* from *G*. An edge *e* in a connected graph *G* is a bridge if its deletion from *G* yields a disconnected subgraph. A directed path from a vertex *x1* to a vertex *x,* in a digraph is a sequence *x1, ai, x2, a2, . . . , ar- 1, xr* in which the *xi* are vertices and *ai* is the arc from *xi* to *xi+ 1*, where *i =1, 2, . . . , r -1*. This path may be represented as *x1 →x2→ · · ·→ xr* . This is a closed directed path if *x1= xr*. A simple directed path from *x* to *y* is a directed path in which all the vertices are distinct. A directed circuit is a closed directed path with distinct arcs, and a directed cycle is a closed directed path with distinct vertices. Two vertices *x* and *y* in a digraph form a strongly connected pair if there is a directed path from *x* to *y* and a

directed path from y to x. A digraph is a strongly connected digraph if every pair of vertices in the digraph is a strongly connected pair. A digraph is weakly connected if its underlying graph (multigraph) is connected.

A simple path in a graph is called a Hamiltonian path if it passes through every vertex of the graph. A closed Hamiltonian path is a Hamiltonian cycle. An Eulerian circuit in a graph is a circuit which contains every edge of the graph. The definitions in the case of digraphs are analogous.

Bipartite graphs and complete graphs

If the set V of vertices of a graph $G = (V, E)$ is partitioned into two subsets X and Y such that every edge in E is between some vertex in X and some vertex in Y, then the graph is called a bipartite graph and is denoted by $G = (X, Y, E)$.

Theorem 1.1

A graph with three or more vertices is bipartite if and only if it has no odd cycles.

Proof

If $G = (X, Y, E)$ is a bipartite graph and if C is a cycle in G, obviously it should have an even number of vertices since the vertices in C are alternately from X and Y.

On the other hand, suppose $G = (V, E)$ has no odd cycles. Observe that a graph G is bipartite if and only if every component of G is bipartite. So we assume without loss of generality that G is connected. The number of edges in a simple path P between two vertices u and v is the

length of P. A path P of minimum length between u and v is a shortest path and the number of edges in such a path is denoted $d(u, v)$.

Let u be any vertex in G. Define $X = \{x \in V: d(u, x)$ is even$\}$ and $Y = V - X$. We now show that whenever v and w are any two vertices in X (or in Y), there is no edge in E joining v and w.

Case l. Let v be any vertex in X other than u. Since there is a path of even length between u and v, there cannot be an edge between u and v since there is no odd cycle in G.

Case 2. Let v and w be two vertices in X other than u. Assume there is an edge e joining v and w. Let P be a shortest path of length $2m$ between u and v and Q be a shortest path of length $2n$ between u and w. If P and Q have no common vertices other than u, then these two paths and the edge e will constitute an odd cycle. If P and Q have common vertices, let u be the common vertex such that the subpath P' and $u$and v and the subpath Q' between u' and w have no vertex in common. Since P and Q are shortest paths, the subpath of P between u and u' is a shortest u-u' path. The subpath of Q between u and u' is also a shortest u- u' path. So both these shortest u-u' paths are of equallength k. Thus the length of P' is $2m - k$ and the length of Q' is $2n - k$. In this case P', Q' and the edge e together constitute a cycle of length $(2m - k) + (2n - k) + 1$ which is an odd integer.

Case 3. Suppose v and w are in Y. Then the length of a shortest path P between u and u is odd. As in case 2, the subpath P' from u' to u of length $(2m -1) - k$, the subpath Q' from u' to w of length $(2n -1) - k$ and the edge will form an odd cycle.

Case 4. Suppose u is the only vertex in X. Then every edge is between u and some vertex in Y.

A graph G = (V, E) is said to be complete if there is an edge between every pair of vertices in the graph. A bipartite graph *G = (V, W, E)* is complete if there is an edge between every vertex *v* in *V* and every vertex *w* in *W* A complete graph with *n* vertices is denoted by *Kn. If G = (V, W, E)* is a complete bipartite graph with *m* vertices in *V* and *n* vertices in *W*, then *G* is denoted by *Km,n.*

Degrees, indegrees and outdegrees

The number of edges incident at the vertex of a graph or a multigraph *G* is the degree of the vertex. A vertex of *G* is odd if its degree is odd. Otherwise it is an even vertex. In computing the degrees of the vertices, each edge is considered twice. Thus the sum of the degrees of all the vertices of *G* is equal to twice the number of edges in *G* and consequently the number of odd vertices of *G* is even.

The number of arcs incident to a vertex in a digraph is the indegree of the vertex, and its outdegree is the number of arcs incident from that vertex. The degree of a vertex in a digraph is the sum of its indegree and outdegree. In any digraph, the sum of the indegrees of all the vertices and the sum of the outdegrees of all the vertices are both equal to the total number of arcs in the digraph since each arc is incident from one vertex and incident to another vertex. As in the case of graphs, the sum of the degrees of all the vertices in a digraph is twice the number of arcs in it.

Incidence matrices and totally unimodular matrices

Suppose *G = (V, E)* is a graph where *V = {x1, x2, x3, . . . , xm}* and *E = {e 1, e2,e3, . . . , en}.* Then the incidence matrix of the graph *G* is the *m* x *n* matrix *A = [aik],* where each row corresponds to a vertex and each column

corresponds to an edge such that if *ek* is an edge between *xi* and *xj* then *aik* = *ajk* =*1*and all the other elements in column *k* are 0. The incidence matrix of a digraph *D=[V,E]* where *V={v1,v2z,...,vm}* and *E={a1,a2,...,an}* is the *mxn* matrix *[aik]* such that each row corresponds to a vertex and each column corresponds to an arc with the following property: if *ak* is the arc from vertex *vi* to vertex *vj* then *aik* = -1, *ajk* =1 and all other elements in column *k* which correspond to the arc *ak* are 0.

A matrix **A** with integer entries is called a totally unimodular (TU) matrix if the determinant of every (square) submatrix **B** of **A** is 0 or 1 or -1. The determinant of a square matrix **B** is denoted as det **B**. Obviously in a TU matrix every element is either 0 or 1 or -1.

Notice that if **A** is a TU matrix and if **B** is any nonsingular submatrix of **A**, the unique solution of the linear system **BX = C**, whenever **C** is an integer vector, is also an integer vector since each component of the solution vector is of the form *p/q,* where *p* is the determinant of an integer matrix and therefore an integer and *q* is the determinant of **B** which is either 1 or -1. TU matrices play an important role in network and combinatorial optimization problems in which we seek solution vectors with integer components.

Theorem 1.2

The matrix **A** = *[aij],* in which every element is 0 or 1 or - l, is totally unimodular if it satisfies the following two properties:

1. No column can have more than two nonzero elements.

2. It is possible to partition the set I of rows of **A** into sets *I*1 and *I* 2 such that if *aij* and *akj* are the two nonzero elements in column *j*, then row *i* and row *k* belong to the same subset of the partition if and only if they are of opposite sign.

Proof

Let C be any k x k submatrix of **A**. The proof is by induction on k. If $k = 1$, then the theorem is true. Suppose it is true for $k - 1$. We have to prove it is true for k. Let C' be any $(k-1) \times (k-1)$ submatrix of C. By induction hypothesis, det C' is 0 or 1 or -1. So we have to establish that the det C is 0 or 1 or -1.

There are three different possibilities: (i) C has a column such that all the elements in that column are zero and this implies det $C = 0$. (ü) C has a column with exactly one nonzero entry which could be either 1 or -1. Then expanding along this column, det C is either det C' or - det C'. Thus det C is 0 or 1 or - l. (iii) Every column of C has exactly two elements.

Suppose $E = \{r1, r2, \ldots, rk\}$ is the set of the rows of C. By hypothesis, this set of k rows is partitioned into two subsets $I1$ and $I2$. Without loss of generality, let us assume that $I1$ is the set of the first p rows and $I2$ is the set of the remaining $k - p$ rows. (It is possible that $p = 0$.) By the way these two sets are constructed, it is easy to see that $r1 + r2 + \cdots + rp = rp + 1 + rp+2 + \cdots + rk$. So C is linearly dependent and thus det $C = 0$.

Corollary 1.3

A matrix in which each element is 0 or 1 or -1 is totally unimodular if in each column there is at most one + 1 and at most one - l. In particular, the incidence matrices of digraphs and bipartite graphs are totally unimodular.

Theorem 1.4

The following properties are equivalent in a graph: (i) G is bipartite; (ü) G has no odd cycle; and (üi) the incidence matrix of G is totally unimodular.

Proof

The equivalence of (i) and (ü) has been already established by Theorem 1.1. If *G* is bipartite, its incidence matrix is a TU matrix by Corollary 1.3.

Suppose the incidence matrix of an arbitrary graph $G = (V, E)$ is a TU matrix. If the graph *G* is not bipartite, there should be at least one odd cycle in the graph. Let $V = 1, 2, 3, \ldots, n\}$. Assume the first $2k + 1$ vertices in *V* constitute an odd cycle:*1-2-3-· · ·- k + 1-l,* Let A be the incidence matrix of the graph in which the first $2k + 1$ rows correspond to the first $2k + 1$ vertices and the first $2k + 1$ columns correspond to the edges $\{1, 2\}, \{2, 3\}, \{3, 4\}, \ldots, \{2k, 2k + 1\}$, {2k + l,1 }, respectively. Let *C* be a submatrix formed by the first $2k + 1$ rows and the first $2k + 1$ columns of the incidence matrix. Then the determinant of *C* is *2*. (In fact, the determinant of the incidence matrix of an odd cycle is always *2* or *-2*.) This contradicts the fact that **A** is totally unimodular.

1.2 SPANNING TREES

Forests and trees

A graph in which no subgraph is a cycle is called an acyclic graph or a forest. A connected acyclic graph is a tree.

Theorem 1.5

A graph is a tree if and only if there is a unique simple path between every pair of vertices in the graph.

Proof

Let G be a graph such that between every pair of vertices there is a unique simple path. So G is connected. If G is not a tree, then there is at least one cycle in G creating two simple paths between every pair of vertices in this cycle. So G is a tree. On the other hand, suppose G is a tree and x and y are two vertices in G. Since G is connected there is a simple path P between x and y. Suppose P' is another simple path between x and y. If the two paths are not the same, then let $e = \{vi, vi+1\}$ be the first edge in P that is not in P' as we go from x to y in the graph along the edges. Let W and W' be the set of intermediate vertices between vi and y in P and P', respectively. If W and W' have no vertices in common, there is a cycle consisting of all the vertices in W and W' and the

vertices vi and y. If W and W' have common vertices, then let w be the first common vertex as we go from vi to y along either P or P'. Then we have a cycle in G using the vertices in P between vi and w and the vertices in P' between w and vi. Thus in any case G has a cycle, which is a contradiction

.

Theorem 1.6

A graph is a tree if and only if every edge in it is a bridge.

Proof

Suppose $e = \{x, y\}$ is any edge in a graph G. Then (x, e, y) is a path between x and y in G. If G is a tree, this is the only path between x and y, and if this path is deleted then the vertices x and y will not be connected. Thus in a tree every edge is a bridge. On the other hand, suppose G is a graph in which every edge is a bridge. So G is connected. Suppose G is not a tree. Then there is at least one cycle C in G. Let x and y be adjacent vertices in C. Then the edge $e = \{x, y\}$ cannot be a bridge in G since there is another path between x and y using the other vertices of the cycle. This contradicts the hypothesis that every edge is a bridge. Thus G is a tree.

Theorem 1.7

(i) A connected graph with n vertices is a tree if and only if it has $n-1$ edges.

(ii) An acyclic graph with n vertices is a tree if and only if it has $n-1$ edges.

Proof

(i) Suppose G is a tree with n vertices. We prove by induction on n that G has $n-1$ edges.

This is true when $n = 1$. Suppose it is true for all m, where $1 < m < n$. If we delete an edge $e = \{x, y\}$ from G, we get two subgraphs $H = (V, E)$ and $H' = (V', E')$, with k and k' vertices, respectively, such that there is no vertex common to V and V'. Since both k and k' are less than n H has $k-1$ edges and

H' has $k'-1$ edges. So both H and H' together have $k+,k'-2=n-2$ edges. If we combine H and H' by using the edge e, we get the graph G. So G has $(n-2)+1=n-1$ edges.

On the other hand, let $G=(V, E)$ be a connected graph with n vertices and $n-1$ edges. Suppose G is not a tree. Then there is an edge in G which is not a bridge. If we delete this edge we have a connected subgraph $G'=(V, E')$. Continue this process till we get a connected subgraph $H=(V, F)$ in which every edge is a bridge. So H is a tree with n vertices. So H has $n-1$ edges leading to a contradiction, since G has $n-1$ edges.

(ii) If an acyclic graph with n vertices is a tree, then it has $n-1$ edges by (i) above.

On the other hand, let $G=(V, E)$ be an acyclic graph with n vertices and $n-1$ edges. Suppose G is not connected. Then there are r connected subgraphs $G1, G2, \cdots, Gr$ where $Gi=(Vi, Ei)$, the cardinality of Vi is ni for each i and $(V1, V2, \cdots, Vr)$ is a partition of V. Each Gi is a connected component of G. Since G is acyclic, each Gi is acyclic and therefore a tree with $ni-1$ edges. Thus the total number of edges in G is $(n1-1)+(n2-1)+\cdots+(ni-1)=n-r$. But the number of edges in G is $n-1$. Thus $r=1$, which implies that G is connected and hence a tree.

Spanning trees

If $G=(V, E)$ is a graph and $T=(V, F)$ is a subgraph which is also a tree, then T is a spanning tree in the graph. An edge in E which is not in F is called a chord of T. The following two theorems are immediate consequences of Theorems 1.5-1.7.

Theorem 1.8

A graph is connected if and only if it has a spanning tree.

Theorem 1.9

Suppose $G = (V, E)$ is a simple graph with n vertices and $H = (V, E)$ is a subgraph. If H satisfies any two of the following three properties then it satisfies the third property also:

(i) H is connected.

(ii) H is acyclic.

(iii) H has $n - 1$ edges.

If a graph $G = (V, E)$ is connected, starting from any vertex it is possible to visit all the other vertices by searching these vertices. There are two ways of accomplishing this: the breadth-first search (BFS) and the depth-first search (DFS).

The BFS method proceeds as follows. Starting from any vertex v, visit all the vertices which are adjacent to v using the edges whichjoin v and these adjacent vertices. Suppose these vertices are $v1, v2, \ldots, vk$. Now start from $v1$ and visit all the unvisited vertices which are adjacent to $v1$ using the edges joining $v1$ and these adjacent vertices. Suppose these vertices are $v11$ $v12 \cdots, v1k$ Then we start from $v2$ and continue this process. Proceed likewise till we visit all the vertices (as yet unvisited) adjacent to vk. Now start from vIl and visit all the unvisited vertices adjacent to ull. Then from $v12$ and so on. Continue this process till all vertices of G are visited. If F is the set of the edges used to visit all the vertices of G, then (V, F) is a spanning tree in G known as the BFS spanning tree rooted at $v1$.

In the DFS method, start from any vertex vo and use an edge eol to visit an adjacent vertex *v1* . Then use an edge *e12*to visit the vertex *v2* which is adjacent to *v1* In general, when we are at a vertex *vi* we use an edge *vij* to visit an adjacent vertex *vj* if *vj* has not been visited. If *vi* has no unvisited adjacent vertex, we backtrack to *vi-1* and explore further. Eventually this backtracking takes us back to the vertex *v* from which we started. If *F* is the set of the edges used in this procedure, then *(V,F)* is a spanning tree known as the DFS spanning tree rooted at *v0.*

Disconnecting sets and cutsets

If $G = (V, E)$ is a connected graph, then any subset D of E is a disconnecting setif he deletion of all the edges in D from the graph makes it disconnected. A disconnecting set D is a cutset if no proper subset of D is a disconnecting set. Suppose the set V is partitioned into two subsets X and Y. If $D = (X, Y)$ is the set of all edges in E of the form $\{x, y\}$ where $x \in X$ and $y \in Y$, then D must be a disconnecting set. But D need not be a cutset.

When will a partition of the set of vertices of a connected graph define a cutset? We have the following theorem in this context.

Theorem 1.10

A disconnecting set $D = (X, Y)$ in a graph $G = (V, E)$ will be a cutset in G if $G' = (V, E - D)$ has exactly two connected components.

Proof

If D is not a cutset, then there is a proper subset D' of D which is a cutset. Suppose $e = \{x, y\}$, where $x \in X$ and $y \in Y$ is an edge in D but not in D'. If v is any vertex in X and w any vertex in Y, by hypothesis there is a path between v and x passing through vertices exclusively from X and there is a

path between y and w passing through vertices exclusively from Y. Consequently the vertex v in X and the vertex w in Y are connected as long as the edge e remains undeleted. In other words, D' is not a disconnecting set and so it cannot be a cutset.

Corollary 1.11

If e is any edge of a spanning tree T in a connected graph G, the deletion of e from the tree defines a partition of the set of vertices of T (and therefore of G) into two subsets X and Y such that $D = (X, Y)$ is a cutset of the graph G.

Observe that even though an arbitrary partition of the set V of the connected graph $G = (V, E)$ into two subsets need not define a cutset, a cutset D in G always defines a partition of the set of vertices into two sets. Specifically, if the deletion of the edges of a cutset D from the graph creates two connected subgraphs $G' = (V', E')$ and $G'' = (V'', E'')$ then $\{V'', V''\}$ is a partition of V and D is the disconnecting set (V', V'').

Theorem 1.12

Suppose C is a cycle, D is a cutset and T is a spanning tree in a graph G. Then: (i) the number of edges common to C and D is even; (ii) at least one edge of C is a chord of T; and (iii) at least one edge of D is an edge of T.

Proof

(i) Let $D = (X, Y)$. If all the vertices of C are exclusively in X (or exclusively in Y) then C and D cannot have any edges in common. Suppose this is not the case. Then C has two vertices x and y where $x \in X$ and $y \in Y$. Then the cycle C which starts from x and ends at x will have to use the edges from D an even number of times. (ii) A chord of T is by definition

an edge of the graph G which is not an edge T. If no edge of C is a chord of T, then C is a subgraph of T. But C is a cycle. This is a contradiction. (iii) If no edge of D is an edge of T, the deletion of all the edges belonging to D will not disconnect T. So T will continue as a spanning tree of G, implying that D is not a cutset.

If $e = \{x, y\}$ is a chord of T, the unique cycle in G consisting of the edge e and the edges of the unique path in T between x and y is called the fundamental cycle of G relative to T with respect to e and is denoted by $CT(e)$, or by $C(e)$ if the tree T is fixed beforehand.

It follows from Corollary 1.11 that corresponding to each edge e of a spanning tree T of a connected graph G there is a unique cutset called the fundamental cutset of T with respect to the edge e which is denoted by $DT(e)$, or by $D(e)$ if the tree is fixed beforehand.

Thus every spanning tree of a connected graph G with n vertices and m edges defines a set of $n-1$ fu ndamental cutsets and a set of $m - n + 1$ fundamental cycles.

There are two theorems relating the concepts of fundamental cycles and fundamental cutsets of a spanning tree.

Theorem 1.13

Let e be an edge of a spanning tree T in a graph and f be any edge (other than e) in the fundamental cutset $D(e)$ defined by e. Then: (i) f is a chord of T and e is an edge of the fundamental cycle $C(f)$ defined by f; and (ii) e is not an edge of the fundamental cycle $C(e')$ defined by any chord e' which is not in $D(e)$.

Proof

(i) If f is not a chord of T, then it is an edge of the spanning tree. In that case T cannot be acyclic, which is a contradiction. So f is a chord of T defining a fundamental cycle $C(f)$. Now $D(e)$ is the union of $\{e, f\}$ and a set A of chords of T, and $C(f)$ is the union of $\{f\}$ and a set B of edges of T. The intersection of A and B is empty. By the previous theorem the intersection of $D(e)$ and $C(f)$ should have an even number of edges. So e is in B and therefore in $C(f)$.

(ii) Now $C(e')$ is the union of $\{e'\}$ and a set X of edges of T. The set $D(e)$ is the union of $\{e\}$ and the set A as in (i) above. Suppose e is in $C(e')$. This implies that e is in X. Since $C(e')$ and $D(e)$ have an even number of edges in common and since the intersection of A and X is empty, the edge e' is *in* $D(e)$, contradicting the hypothesis that e' is not in $D(e)$.

Theorem 1.14

Let e be a chord of a spanning tree T in a graph, and f be any edge (other than e) in the fundamental cycle. $C(e)$ defined by e. Then: (i) f is an edge of T and e is an edge of the fundamental cutset $D(f)$ defined by f; and (ii) e is not an edge of the fundamental cutset $D(e')$ defined by any edge e' of T which is not in $C(e)$.

Proof

The proof is similar to that of Theorem 1.13.

1.3 MINIMUM WEIGHT SPANNING TREES

Three basic theorems

If w is a mapping from the set of edges of a graph $G = (V, E)$ to the set of real numbers, the graph equipped with the mapping (the weight function) is known as a weighted graph or a network. The number $w(e)$ is called the weight of the edge e. An edge e in a set F of edges in the graph is a minimum weight edge in the set F if $w(e) \leq w(f)$ (or a maximum weight edge if $w(e) \geq w(f)$) for every f in F. If H is a subgraph of a weighted graph, the weight of H is the sum of the weights of all the edges of H and is denoted by $w(H)$. A spanning tree T in a connected network is a minimum weight spanning tree or a minimal spanning tree (MST) if there is no other spanning tree T' such that $w(T')$ is less than $w(T)$.

Under what conditions will a spanning tree be a minimum spanning tree? We have two theorems in this context - one involving fundamental cutsets and the other involving fundamental cycles. Loosely speaking, these theorems assert that from the perspective of a minimal spanning tree, every edge in the tree is an edge worthy of inclusion in the tree (cutset optimality condition) whereas every edge not the tree is an edge that can be ignored (cycle optimality condition).

Theorem 1.15

A spanning tree T in a weighted graph is a minimum weight spanning tree if and only if every edge in the tree is a minimum weight edge in the fundamental cutset defined by that edge.

Proof

Let e be an edge of an MST. Suppose $w(e) > w(f)$ for some edge f in the cutset $D(e)$. Consider the spanning tree T' obtained by deleting e from T and adjoining f to it. Then $w(T') < w(T)$, contradicting the fact that T is an MST. Thus the condition is necessary for a spanning tree to be a minimal spanning tree.

On the other hand, suppose T is a spanning tree which satisfies the given condition. Let T' be any minimum weight spanning tree in the graph. If T' and T are not the same, there will be an edge $e = \{ i, j \}$ in T which is not an edge in T'. Let the fundamental cutset $D(e)$ with respect to T be the set of all edges (in the graph) between the vertices in X and the vertices in Y, where $i \in X$ and $j \in Y$. If we adjoin the edge e to the tree T', a unique cycle $C(e)$ is created which contains an edge f (other than e) joining a vertex in X and a vertex in Y.

Now $w(e) \leq w(f)$ by our hypothesis. If $w(e) < w(f)$, we can construct a spanning tree T'' by adjoining e to T' and deleting f from it; the weight of T'' is less than the weight of the minimal spanning tree T'. Thus $w(e) = w(f)$. Now if we adjoin the edge f to T and delete e from T we obtain a new spanning tree $T1$ such that $w(T) = w(T1)$. If $T1 = T'$, *then* $w(T) = w(T')$, showing that T is an MST. Otherwise we consider an edge in $T1$ which is not in T' and repeat the process till we construct a spanning tree Tk such that the weights of T, Tk and T' are all equal.

Corollary 1.16

If e is an edge of any cycle in a connected graph G such that $w(e) \geq w(f)$ for every edge f in that cycle, then there is an MST in G which does not

contain the edge *e*. In particular, if $w(e) > w(f)$, no MST in *G* can have *e* as an edge.

Proof

Suppose e is an edge in every MST in *G* and let *T* be an MST. Since *e* is an edge in *T*, according to Theorem 1.15, $w(e) > w(e')$ for every edge e' in the cutset $D(e)$. Now if we adjoin an edge e' (other than *e*) from $D(e)$ to *T*, we obtain a cycle $C(e')$ which contains the edge *e*.

So according to the hypothesis, $w(e) \geq w(e')$. Thus $w(e) = w(e')$. Now if we adjoin e' to *T* and delete *e* from *T*, we obtain a spanning tree T' such *that* $w(T') = w(T) + w(e') - w(e) = w(T)$. In other words, T' is an MST which does not contain the edge, contradicting the assumption that *e* is an edge in every MST. So there is at least one MST which does not contain *e*.

Suppose the hypothesis is the strict inequality $w(e') < w(e)$. Suppose there exists a minimal spanning tree *T* which contains *e*. As above, we can construct a spanning tree T' such that $w(T') = w(T) + w(e') - w(e) < w(T)$, implying that *T* is not an MST.

Theorem 1.17

A spanning tree *T* in a graph *G* is a minimal spanning tree if and only if every chord of the tree is a maximum weight edge in the unique fundamental cycle defined by that edge.

Proof

Let *T* be a minimal spanning tree in *G*. Suppose there is a chord e of *T* such that $w(e) < w(f)$ for some edge *f* in the cycle $C(e)$. Then the tree T' obtained by adjoining *e* and deleting *f* is indeed a spanning tree such

that $w(T') = w(T) + w(e) - w(f) < w(T)$, violating the assumption that T is an MST. Thus the condition is necessary.

Let T be a spanning tree in G satisfying the given condition and let f be any edge in this tree T Now any edge e (other than f) in the fundamental cutset $D(f)$ is a chord of T defining the fundamental cycle $C(e)$, and hence by the hypothesis $w(e) \geq w(f)$. This inequality should hold for every e in the cutset $D(f)$. In other words, $w(f) \leq w(e)$ for every e in $D(f)$. This is precisely the optimality condition established in Theorem 1.15 and hence T is an MST. Thus the condition is a sufhcient condition.

Corollary 1.18

Let e be an edge incident at a vertex x of a connected graph. If $w(e) \leq w(f)$ for every other edge f incident at x, then there exists a minimal spanning tree in the graph in which e is an edge. In particular, if $w(e) < w(f)$, every MST in G has e as an edge.

Proof

Suppose $e = \{x, y\}$ is not an edge in any MST. Let T be an MST. Since e is not in T, e is a chord of T. Let $e' = \{x, z\}$ be the other edge in the fundamental cycle $C(e)$ which is incident at the vertex x. In the fundamental cycle $C(e)$, the edge e is a chord and all the other edges are edges in the tree. So by the cycle optimality criterion, $w(e) \geq w(f)$ for every f in the cycle $C(e)$. In particular, $w(e) \geq w(e')$. But $w(e') \geq w(e)$ by our hypothesis. Thus $w(e) = w(e')$. Now the weight of the spanning tree T' obtained by adjoining e to T and deleting e' is $w(T') = w(T) + w(e) - w(e') = w(T)$. Thus there exists a spanning tree T' which contains e, contradicting the assumption.

Suppose $w(e) < w(f)$ and e is not an edge in any MST. Let T be any MST. As before we have the spanning tree T' such that $w(T') = w(T) + w(e) - w(f) < w(T)$, which is a contradiction since T is an MST. Thus e is an edge in every MST.

The following theorem gives us a procedure for constructing a minimal spanning tree in a connected graph G from a subgraph of a known minimal spanning tree in G.

Theorem 1.19

If V' is the set of vertices of a component of any subgraph H of a minimal spanning tree T in $G = (V, E)$ and if e is an edge of minimum weight in the disconnecting set $D = (V', V- V')$, then there exists an MST in G which contains e as an edge and H as a subgraph.

Proof

If e is an edge in T, then the result is obvious. So let us assume that e is a chord in T. In that case $T \cup \{e\}$ has a unique cycle C. The edge e is common to both C and D. So there should be an edge f (other than e) in $(C \cap D)$ since $| C \cap D |$ is even.

Now e is a minimum weight edge in D and f is an edge in D. So $w(e) \leq w(f)$. Every edge in C other than e is an edge in T. In particular, f is an edge in T defining the fundamental cutset $D(f)$. Since $f \in (C \cap D(f))$ and $|(C \cap D(f))|$ is even there should be at least one more edge common to both C and $D(f)$. No edge in $D(f)$ other than f can be an edge in T. The only edge in C which is not in T is the edge e. Thus e is necessarily an edge in $D(f)$.

Since T is an MST, by Theorem 1.15, $w(f) \leq w(e)$. Thus $w(e) = w(f)$. If $T^* = T - \{f\} + \{e\}$, then T^* is a spanning tree with weight $w(T^*) = w(T)$. In other words, T^* is an MST in G which contains e as an edge and H as a subgraph.

Two MST algorithms

Using Theorems 1.15,1.17 and 1.19, we can derive two methods of obtaining an MST in a connected graph. Both procedures are `greedy' in the sense that at every stage a decision is made to make the best possible choice of an edge for inclusion in the MST without violating any rules. In the course of obtaining a tree the only rule is that at no stage in the decision-making process should the choice of the best possible edge (the `choicest morsel') for inclusion in an MST produce a cycle.

One greedy procedure (known as Kruskal's greedy algorithm) is as follows. The set of edges of the connected graph G with n vertices is listed in nondecreasing weight order. We construct an acylic subgraph T, examining these edges one at a time in the order they are arranged. An edge will be added to T if its inclusion does not yield a cycle. The construction terminates when T has $n-1$ edges. There are four steps in the algorithm:

Step 1. Arrange the edges in nondecreasing order in a list L and set T to be the empty set. (T is the set of edges in an MST.)

Step 2. Add the first edge in L to T.

Step 3. If every edge in L is examined, stop and report that G is not connected. Otherwise take the first unexamined edge in L and include it in T if it does not form a cycle with the edges already in T. If the edge is added to T, go to step 4. Otherwise repeat step 3.

Step 4. Stop if T has $n-1$ edges. Otherwise go to step 3.

The resulting subgraph T must be a spanning tree in G. The fact that T is indeed an MST can be established using Theorem 1.17. Observe that in Kruskal's procedure, an edge e was discarded (in favor of an edge of larger weight) with respect to the tree T because it created a cycle C with some or all of the edges already included in the list L. At the same time, the weight of the discarded edge is greater than or equal to the weight of any other edge in the cycle C because the edges are examined one at a time according to their weights in nondecreasing order. Thus the spanning tree T satisfies the optimality condition as stated in Theorem 1.17.

Another procedure (known as Prim's greedy algorithm) to construct an MST is as follows. We start from an arbitrary vertex and add edges one at a time by maintaining a spanning tree T on a subset W of the set of vertices of the graph such that the edge adjoined to T is a minimum weight edge in the cutset $(W, V - W)$. The correctness of this procedure follows directly from the sufficient condition established in Theorem 1.19. There are three steps in this procedure:

Step l. Select an arbitrary vertex of G and include it in the tree T.

Step 2. Let W be the set of vertices in T. Find an edge of minimum weight in the disconnecting set $(W, V - W)$ and add it to T. If an edge cannot be added, report that G is not connected.

Step 3. Stop if T has $n-1$ edges. Otherwise repeat step 2.

Notice that the procedures described above are applicable even when the graph is not connected. If the input is an arbitrary network G with n vertices, the output will be either an MST with n -1 edges or a message that G is not connected.

Prim's algorithim (matrix method)

Since a spanning tree in a connected graph passes though every vertex of the graph, we can construct an MST by applying Prim's algorithm starting from any vertex.

In particular if $V= \{1,2,...,n\}$ is the set of vertices of a connected graph G we can start from vertex 1. And create a subtre of the MST by adjoining one edge at a time till we use exactly $n-1$ edges of the graph. We now outline a matrix formulation of this procedure.

Let **D**= $[d(I,j)]$ be the nxm matrix where n is the number of vertices of the graph $G=(V,E)$ and $d=(I,j)$ is the weght of the edge $\{I,j\}$ if there is an edge between i and j. Otherwise $d(i, j)$ is plus infinity. Initially delete all elements of column 1 and check row 1 with a check mark √. All elements are uncircled initially. Each iteration has two steps as follows.

Step l. Select a smallest element (ties are broken arbitrarily) from the uncircled entries in the rows which are checked. Stop if no such element exists. The edges which correspond to the circled entries constitute an MST.

Step 2. If $d(i, j)$ is selected in step 1, circle that entry and check row j with a check mark. Delete the remaining elements in column j. Go to step 1.

1.4 THE TRAVELİNG SALESMAN PROBLEM

Optimal Hamiltonian cycles

In general, it is not asy to determine whether an arbitrary graph or digraph is Hamiltonian since there is no known practical characterization of such graph and digraphs. If a network is Hamiltonian, a problem of interest is to obtain a Hamiltonian cycle of minimum weight, known as an optimal Hamiltonian cycle, in the network. Unfortunately there is no efficient procedure to obtain such an optimal Hamiltonian cycle. Finding an optimal Hamiltonian cycle in G by a process of enumeration is a hopelessly inefficient task since the number of Hamiltonian cycles could be very large even when the number of vertices is small. In the parlance of theoretical computer science, we say that the time complexity of (any known procedure to solve every instance of) the problem of finding an optimal Hamiltonian cycle in a network is exponential in the worst case.

The celebrated traveling salesman problem (TSP) is the problem of obtaining an optimal Hamiltonian cycle in a connected network if the network is Hamiltonian. A problem of practical importance, however, is the problem of finding a closed path (not necessarily simple) of minimum weight which passes through every vertex at least once in a connected network. Many practical problems in network optimization can be formulated as salesman problems. The real significance of the TSP is not that it has a wealth of applications but that it is a generic problem which captures the essence of several problems in combinatorial optimization.

Even though it is not easy to find an optimal Hamiltonian cycle in a connected graph, we can use the minimal spanning tree algorithm to obtain a lower bound for the weight of such a cycle provided the cycle exists. In this

section we assume that the weight w(e) is nonnegative for every edge e in the network under consideration.

Suppose T' is any MST in a network G and T is the spanning tree obtained by deleting an edge from an arbitrary Hamiltonian cycle C. Then $w(T') \leq w(T) \leq w(C)$. So the weight of an MST in the graph is a lower bound for the weight of an optimal Hamiltonian cycle.

In fact we can obtain a better lower bound using the minimum spanning trees of certain subgraphs of the graph. Suppose C is an optimal Hamiltonian cycle in a graph $G = (V, E)$ with n vertices. If we delete a vertex v and the two edges e and f incident to v in the cycle C, we obtain a spanning tree in the subgraph $G' = (V', E')$ where $V' = V - \{v\}$ and $E' = E - \{e, f\}$.

Suppose T' is an MST in G' with weight $w(T')$. If p and q are two edges incident at the vertex v, of smallest possible weights, then $w(T') + w(p) + w(q)$ is a lower bound for $w(C)$, the weight of the optimal Hamiltonian cycle. Since the graph has n vertices, there will be at most n such (not necessarily distinct) lower bounds. The largest of these lower bounds is a lower bound for the weight of the optimal Hamiltonian cycle if such a cycle exists.

A quick method to obtain an approximate solution

We now turn our attention to the following question. Suppose it is knowpn that a Hamiltonian cycle exists in a given network G and thus an optimal Hamiltonian cycle C^* with weight $w(C^*)$ exists in G. Is it possible (without too much computational work) to obtain a Hamiltonian cycle C in G such that the gap $w(C) - w(C^*)$ is not too wide? Since an optimal Hamiltonian cycle is a `lowest weigth ' Hamiltonian cycle any Hamiltonian

cycle whose weight does not exceed the lowest weight by too much may be called a low weight Hamiltonian cycle. For this purpose we restrict our attention to the class of complete graphs with additional property known as the triangle property: if *i, j* and k are any three vertices in a complete graph and if the weights of the edges *{i, j}, { j, k}* and *{k, i}* are *a, b* and *c,* then *a + b* cannot be less than *c.* We then have a theorem which gives an upper bound for this gap.

Theorem 1.20

If complete network satisfies the triangle property, there exists a Hamiltonian cycle such that its weight is less than twice the weight of the optimal Hamiltonian cycle.

Proof

We prove this by actually constructing a Hamiltonian cycle C in the network with the desired property.

Step l. Choose any vertex as the initial cycle *C1* with one vertex.

Step 2. Let Ck be a cycle with k vertices. Find the vertex wk which is not in *Ck* that is closest to a vertex *vk* in *Ck.*

Step 3. Let *Ck+ 1* be the new cycle obtained by inserting *wk* just prior to *vk* in *Ck.*

Step 4. Repeat steps 2 and 3 till a Hamiltonian cycle C is found.

We now prove $w(C) \leq 2w(C^*)$, where C^* is any optimal Hamiltonian cycle in *G.* Suppose the vertices are *1, 2, 3, . . . , n.* Assume without loss of generality the cycle consisting of the edges *{l, 2}, {2, 3}, . . . , {n-1, n}, {n,1}* is an optimal Hamiltonian cycle C^*. Then the path *1-2-3- · -n* is a Hamiltonian path *P* with weight *w(P)* which is at most equal to $w(C^*)$.

Our initial cycle *C1* consists of vertex 1 and no edges. With this cycle we associate the set *E1* of all the edges in the Hamiltonian path *P*.

The weight of the edge *{i, j}* is denoted by *w(i, j)* or *w(j, i)*.

To construct *C2* , with two vertices, we choose the vertex (denoted *i*) which is closest to vertex 1 in the network. *C2* thus has path *1-i-1*, with weight $2w(1, i) \leq 2w(1, 2)$. Since we selected the edge *{1, i}* and did not select *{1, 2}*, we delete the edge *{1,2}* from the path *P*. Thus with cycle *C2* we associate $S2 = \{\{2, 3\}, \{3, 4\}, \ldots, \{n-1, n\}\}$.

Let *j* be the vertex not on *C2* closest to a vertex in *C2*. If the edge of smallest weight incident at vertex 1 is $e = \{1, p\}$ and the edge of smallest weight incident at *i* is $f = \{i, q\}$ and if $w(e) < w(f)$, we take $p = j$, *C3* is given by *1-i-j-1*. If $w(e) > w(f)$, we take $q = j$ so that *C3* has path *1-j-i-1*. (In the case of equality we take either *p* or *q*.)

Now

$$w(C3) - w(C2) = [w(i, j) + w(1, j) + w(1, i)] - [2w(1, i)]$$
$$= w(i,j) + w(l,j) - w(l,i)$$

But $w(1, j) \leq w(i, j) + w(1, i)$ because of the triangle property. Thus

$$w(l,j) - w(l,i) \leq w(i,j)$$

Hence

$$w(C3) - w(C2) \leq 2w(i, j)$$

We choose the edge *{i, j}* for inclusion in the updated cycle *C3*. At this stage we locate the first edge in the path *P* from *i* (the new vertex in *C2*) to *j* (the new vertex in *C3*) and delete this edge from the set *S2* to obtain *S3* which is associated with *C3*. This deleted edge is of the form $\{i, i+1\}$ or $\{i, i-1\}$. Since *w(i, j)* cannot exceed the weight of the deleted edge, *w(C3) - w(C2)* is less than or equal to twice the weight of the edge deleted from *P*.

Proceeding like this, we have *w(Ck) - w(Ck-1)* less than or equal to twice the weight of a unique edge from *P*. Let *Cn* be denoted by *C*. Hence

$$w(C) - W(Cn-1) \leq 2wn-1$$

$$w(Cn-1)-w(Cn-2) \leq 2wn-2$$

$$w(C3) - w(C2) \leq 2w2$$

$$w(C2) \leq 2w1$$

where *wl, w2, ..., wn-1* are the wei hts of the *n -1* ed es of the path *P*.

Addition yields

$$w(C) \leq 2(wl + wz + \ldots + wn-1) = 2w(P) \leq 2w(C^*)$$

Thus we obtain a Hamiltonian cycle such that $w(C) \leq 2w(C^*)$ and so the gap $w(C) - w(C^*)$ cannot exceed $w(C^*)$. Incidentally, we also obtain $1/2w(C)$ as a lower bound for $w(C^*)$.

Another lower bound for an optimal Hamiltonian cycle

We now describe another algorithm for obtaining a low weight Hamiltonian cycle *C* in a complete graph $G = (V, E)$ which satisfies the triangle property, using an MST in the graph. Suppose $T = (V, E')$ is an MST in *G*. Let $H = (V, F)$ to be the multigraph obtained by duplicating all the edges of *T*. Then it can be proved that *H* has an Eulerian circuit and so it is possible to start from any vertex and return to that vertex by using each edge of *H* exactly once

We start from any arbitrary vertex in T and conduct the following DFS of the vertices of the tree which is a systematic method of visiting all the vertices of the graph. We move along the edges of *H* (using each edge exactly once) from vertex to vertex, assigning labels to each vertex, without

revisiting any vertex as long as possible. When a vertex is reached, each of whose adjacent vertices had been already visited, the search revisits the vertex v that was visited immediately before the previous visit to v. This procedure creates a circuit C' in H which uses each edge of H exactly once. Thus $w(C) = w(H) = 2w(T)$. Suppose the vertices in C' appear in the order $v1, v2, \ldots, vn$, which is a permutation of the n vertices of the graph.

Now consider the cycle C'' given by $v1\text{-}v2\text{-}v3\text{-}\cdots\text{-}vn\text{-}v1$. This Hamiltonian cycle may have edges which are not edges in the circuit C'. But by the triangle property, $w(C'') \leq w(C')$. Thus $w(C'') \leq 2w(T) \leq 2w(C)$, where C is any Hamiltonian cycle in the graph. Thus we are able to construct a Hamiltonian cycle C'' in the graph, the weight of which does not exceed twice the weight of any optimal Hamiltonian cycle in the graph.

The algorithm has three steps:

Step 1. Find a minimal spanning tree in the graph.

Step 2. Conduct a depth-first search of the vertices of T

Step 3. If the order in which the vertices appear in the search is $v1,v2,\ldots,vn$, then the Hamiltonian cycle $v1\text{-}v2\text{-}v3\text{-}\cdots\text{-}vn\text{-}v1$ is a low weight Hamiltonian cycle.

(Since we can start the DFS (in a fixed MST) from any vertex, it may happen that the same tree T may yield more than one low weight Hamiltonian cycle. It is also possible that there will be more than one MST in a given graph.)

1.5 MINIMUM WEIGHT ARBORESCENCES

A digraph is a directed forest if its underlying graph is a forest. A branching is a directed forest in which the indegree of each vertex is at most 1. A vertex in a branching is a root if its indegree is 0. An arborescence is a branching which has exactly one root. If a subgraph $T = (V, A')$ of a digraph $G = (V, A)$ is an arborescence with its root at the vertex r, then T is an arborescence rooted at r in the digraph G.

A subdigraph H of a digraph G is a directed spanning tree rooted at u if H is an arborescence in G with root at vertex v. If a directed spanning tree rooted at v is obtained as a result of a breadth-first search (depth-first search), then it is a BFS (DFS) directed spanning tree rooted at v. If a digraph is not strongly connected, a BFS or DFS starting from an arbitrary vertex v may not result in an arborescence rooted at that vertex.

Obviously, if a digraph is strongly connected it has an arborescence rooted at every vertex. But strong connectivity is not a necessary condition for the existence of an arborescence in a digraph. For example, the digraph $G = (V, A)$, where $V = \{1, 2, 3\}$ and $A = \{(1, 2), (2, 3), (2,1)\}$, has an arborescence rooted at vertex 1 even though the graph is not strongly connected. We now seek to obtain a suff cient condition to be satisfied by a digraph so that it will have an arborescence.

A digraph is quasi-strongly connected if for every pair of vertices u and v in the digraph, there exists a vertex w such that there are directed paths from w to u and from w to v. (The vertex w could be u or v.) Strong connectivity obviously implies quasi-strong connectivity. But the converse is not true, as can be seen from the example given above. If a digraph has an arborescence, then it is obviously quasi-strongly connected. It turns out that

quasi-strong connectivity is also a sufbcient condition for the existence of a spanning arborescence in a digraph.

Theorem 1.21

A digraph has an arborescence if and only if it is quasi-strongly connected.

Proof

If there is an arborescence in *G,* then *G* is quasi-strongly connected. So it is a necessary condition.

On the other hand, suppose $G = (V, A)$ is a quasi-strongly connected digraph, where $V = \{ 1, 2, 3, \ldots, n\}$. Let $H = (V, A')$ be a maximal quasi-strongly connected subgraph of G so that the deletion of one more arc from A will destroy the quasi-strong connectivity of *H.*

There exists a vertex *x1* in *V* such that there are paths (in *H*) from *x1* to the vertices 1 and 2.

There exists a vertex *x2* such that there are paths (in *H*) from *x2* to the vertices 3 and *x1.*

Finally, there exists a vertex *xn-1* such that there are paths (in *H*) from *xn-1* to vertices *n* and *xn-2.* In other words, there exists a vertex *v* such that there are paths in *H* from *v* to every vertex.

Let *i* be any other vertex in *H.* Since there is a directed path from *v* to *i,* the indegree of i is at least 1. Suppose the indegree of *i* is more than 1. Then there are at least two distinct vertices *j* and *k* such that both *(j,* i) and *(k, i)* are arcs in *H.* So there are at least two paths from *u* to *i.* If one of the arcs *(j, i)* or *(k, i)* is deleted, then the quasi-strong connectivity of H is not affected. This contradic tion shows that the indegree of i is 1.

If there is an arc directed to v, the deletion of this arc will not affect the quasi-strong connectivity of H. Thus the indegree of v is 0. So H is an arborescence in G rooted at v. This completes the proof.

Suppose it is known that a weighted digraph has an arborescence rooted at a vertex r of the digraph. The minimum weight arborescence problem is that of finding an arborescence of minimum weight rooted at r. In the remaining part of this section we consider weighted digraphs. We assume that all arcs in a digraph have different weights.

Suppose $G = (V, A)$ has an arborescence rooted at r. If we adopt the greedy procedure and choose an arc of minimum weight directed to each vertex other than the root, and if the resulting subgraph H is acyclic, then H is indeed an arborescence rooted at r. For example, in the digraph $G = (V, E)$ where $V = \{ 1, 2, 3, 4\}$ and $A = \{ (1, 2), (1, 3), (2, 4), (3, 2), (4, 3) \}$ with weights $3, 5, 4, 6, 7$, Respectively, this greedy procedure to obtain a minimum weight arborescence rooted at vertex 1 chooses the arcs $(1, 2)$, $(2,4)$ and $(1, 3)$, giving a minimum weight arborescence in G with a total weight of $3 + 4 + 5 = 12$. But if the arc weights are $6, 7, 4, 3, 5$, respectively, the resulting subgraph is no longer acyclic.

Thus the problem is to obtain a procedure to obtain a minimum weight arborescence rooted at a vertex r when the subgraph obtained by the greedy method is not acyclic. The procedure described here is based on the treatment of the same topic by Gondran and Minoux (1984).

Theorem 1.22

Let $T^* = (V, A^*)$ be a minimum weight arborescence rooted at v in the weighted digraph $G = (V, A)$ with distinct arc weights, and let $H = (V, A')$ be the subgraph obtained from G by choosing the arc of minimum weight

directed to each vertex other than the root v. Then the set $C - A^*$ has exactly one arc for every cycle C in H.

Proof

Obviously, $C - A^*$ is not empty. Suppose $(C - A^*) = \{e', f'\}$ for some cycle C in H. Let $e' = (p, 9)$ and $f' = (r, s)$.

In T^* there is a unique directed path from the root v to q in which the last arc is not (p, q). Let this arc be $e = (p', 9)$. Similarly, there is a unique directed path in T^* from the root v to the vertex s the last arc of which is not (r, s) but the arc $f = (r', s)$. The weight of any arc e is $w(e)$. Since the arcs have distinct weights, by our construction of the subgraph H, $w(e') < w(e)$ and $w(f') < w(f)$. Let A'' be the set of arcs obtained from A^* by replacing e by e'. Then $T' = (V, A'')$ is an arborescence rooted at v the weight of which is less than the weight of T^*. But T^* is an arborescence with minimum weight. If more than two arcs of C are not in A^* we arrive at the same contradiction. Thus exactly one arc of C is in A^*.

Construction of a condensed digraph from *G*

Suppose $G = (V, A)$ has an arborescence rooted at v. Let $H = (V, A')$ be the subgraph obtained by the greedy method. Suppose C is a cycle in H. Let $T^* = (V, A^*)$ be a minimum weight arborescence rooted at r. Let $C - A^*$ be the arc $e' = (k, j)$ as in Theorem 1.22. So there is a unique arc $e = (i, j)$ in A^*, where i is not a vertex in the cycle C.

Let $G1 = G/C = (V1, A1)$ be the graph obtained by shrinking all the vertices of C into a single vertex. Each arc in $A1$ corresponds to a unique arc in A. So $A1$ can be considered as a subset of A. Then $T^*1 = (V1, A^* \cap A1)$ is an arborescence in $G1$. We have $w(T^*1) = w(T^*) + w(C) - w(e')$.

We now define a weight function *w1* on the set *A1*. If *e* is an arc which is not incident to a vertex in *C*, we define *wl(e) = w(e)*. Otherwise *wl(e) = w(e) - w(e')*. Thus *w1* is fixed once *C* is fixed. Moreover, *wl(T*1) = w(T*1) - w(e')*. Thus *w(T*)-w1(T*1)=w(C)*. Hence *T** is a minimum weight arborescence in *G* if and only if *T** is a minimum weight arborescence in a condensed graph *G1*.

Thus we have the following algorithm to obtain a minimum weight arborescence.

(a) Step *i = 0. Go = G, wo(e) = w(e)* for each *e* in the graph *G*.

(b) At step *i*, using the weight function *wi* , construct the subgraph *Hi* of *Gi* by selecting the arc of smallest weight directed to every vertex (other than the root) of *Gi*.

(c) If there is no directed circuit in *Hi*, then *Hi* is an arborescence in *Gi* from which a minimum weight arborescence of *Go* can be derived. Otherwise go to (d).

(d) If *Hi* has a directed circuit *C*, then define *Gi+1= Gi/C* and *wi+1(e) = wi(e)* for *e = (i, j)* where $j \notin C$, and *wi+ 1 (e) = wi(e) - wi(e')* for *e = (i, j)* where $j \in C$ and $i \notin C$ and *e' = (k, j)* is an arc in *C*. (If *Hi* contains cycles *C1, C2, ..., Ct,* first condense *Gi* with respect to *C1* to obtain *Gi/C1*, then condense *Gi /C1* with respect to *C2,* and so on.) Set *i = i + 1* and return to (b).

If we assume at each step of the algorithm that all arcs of the current digraph have different weights, then the minimum weight arborescence is unique. When there are some arcs with equal weights, the algorithm remains valid since the difference *w(T*) – w1(T*1)* is still equal to the weight of the condensed cycle. If *C* has two arcs *e' = (p, q)* and *f' = (r, s)* which are not in

A^*, then there are arcs e and f in A^* directed to q and s, respectively. Then $w1(e) = w(e) - w(e')$ and $w\,1(f) = w(f) - w(f')$.

1.6 MAXIMUM WEIGHT BRANCHINGS

An acyclic subgraph B of a digraph G is a branching in G if the indegree (in B) of each vertex is at most l. If w is a real-valued weight function defined on the set of arcs of the digraph, an optimization problem of interest is that of finding a branching B in G such that the sum $w(B)$ of the weights of all the arcs in B is as large as possible. This problem is known as the maximum weight branching problem.

An attempt to solve the maximum weight branching problem by the greedy method need not be successful, as can be seen from the following example. Consider $G = (V, A)$, where $V = \{1, 2, 3, 4\}$ and A is the set $\{(1, 2), (2, 3), (3, 4), (4, 3)\}$ with weights 9, 8, 5 and 10, respectively. The greedy method will give a branching with weight $10 + 9 = 19$. But the weight of the maximum weight branching is $9 + 8 + 5 = 22$.

In this section we obtain a procedure for solving the maximum weight branching problem. The algorithm is known as Edmonds branching algorithm. The discussion here is based on a combinatorial proof of this algorithm by Karp (1971).

If $e = (i, j)$ is.an arc in the digraph $G = (V, A)$, th en the vertex i which is the source of e is denoted by $s(e)$. Likewise the vertex j which is the terminal of e is denoted by $t(e)$. Thus both s and t are mappings from A to V. An arc e from , vertex p to vertex q in a digraph $G = (V, A)$ with weight function w is a critecal arc if $w(e) > 0$and $w(e') \leq w(e)$ for every arc e' where $t(e') = t(e)$. A spanning subgraph $H = (V, A')$ of G is a critical graph if each arc

in A' is critical and the indegree of each vertex is at most 1. No two cycles in a critical graph can have and edge or a vertex in common.

If a critical graph $H= (V' A')$in a graph G is acyclic, then obviously H is in a maximum weight branching. For example, consider the digraph with $V=\{1,2,3,4\}$ and arcs $(1,2),(1,4),(2,3),(3,4)$and $(4,2)$, with weights $6,5,3,2$ and 4, respectively. The critical graph is $H= (V ,A')$where the arcs in A' are $(1,2)$, $(1, 4)$ and $(2, 3)$. H is acyclic and is a maximum weight branching . On the other hand if the arc weights are respectvely 3, 2, 9, 6 and 8, the the critical graph H is (V, A'), where the arcs in A are $(2, 3)$, $(3, 4)$ and $(4, 2)$. In this case H is not acyclic.

Thus the crux of the problem is to obtain an algorithm to solve the maximum weight branching problem when the critical subgraph is not acyclic. The procedure is as follows. Each cycle in the critical graph is replaced by a single vertex and the weight function is appropriately redefined. We continue this process till we get an acyclic critical graph in which a maximum weight branching is easily discernible. Once we obtain a maximum weight branching in this acyclic graph, we unravel the cycles and revert to the original problem and obtain a maximum weight branching in the given digraph.

Construction of a condensed digraph

Let $G = (V, A)$ be a digraph with weight function w, and H be a critical graph in G. Suppose the cycles in H are Ci $(i = l, 2, \ldots, k)$. Let $W = \{v \in V : v$ is not a vertex in any of these k cycles$\}$. Replace each cycle Ci by a vertex Xi. Let $V1=\{X 1, X 2, \ldots, Xk\} \cup W$. If e is an arc of the digraph G which is not an arc of Ci, and if $t(e)$ is a vertex of Ci, then we define $w1(e)=$ $w(e) - w(f) + w(ei)$, where f is the unique arc in Ci with $t(f)=t(e)$ and ei is an arc of minimum weight in the cycle Ci. If t(e) is not a vertex in any of these k cycles, then $w1(e)=w(e)$. the multidigraph $G1$ thus obtained from H with $V1$ as the set of vertices is called the condensed graph of the weighted graph. A critical graph in $G1$ is denoted by $H1$. Thus from the pair (G,H) we move to the pair $(G1,H1)$. We continue this process till we reach the pair (Gm,Hm), where Hm is acyclic.

Theorem 1.23

An arc e in a digraph is B-eligible if and only if there is no directed path in the branching B from $t(e)$ to $s(e)$.

Proof

Let $B = (V', A')$ be a branching in $G = (V, A)$. The set

$$A'' = A' \cup_v \{e\} - \{ f \in A' : t(e) = t(f) \}$$

will form a branching if and only if it does not contain a cycle since B is a branching. Since B is acyclic, any cycle in A'' should contain the arc e. In other words, C is a cycle in A'' if and only if $C - \{e\}$ is a directed path in B from $t(i)$ to $s(i)$. Thus e is B-eligible if and only if there is no directed path in B from $t(e)$ to $s(e)$.

Suppose $B = (V', A')$ is a branching in $G = (V, A)$. Let C be the set of arcs in a cycle in G. Since C cannot be a subset of A', the set $C - A'$ is nonempty. If $C - A'$ consists of one arc e, then e is not B-eligible. The converse also is true, and this is the content of the next theorem.

Theorem 1.24

If $B = (V', A')$ is a branching in $G = (V, A)$ and if C is the set of arcs in a cycle such that no arc in $C - A'$ is B-eligible, then $C-A'$ contains exactly one arc.

Proof

Suppose $C - A' = \{ei: i = 1, 2, \ldots, k\}$. Assume that these arcs appear clockwise in the cycle such that $ei+1$ follows immediately after ei for $i = 1, 2, \ldots, k-1$. Hence $t(ei-1) = s(ei)$ or there is a path in $C \cap A'$ from $t(ei-1)$ to

s(ei) for *i* =2, 3, ..., *k*. Also, either *t(ek)* = *s(e1)* or there is a path in $C \cap A'$ from *t(ek)* to *s(e1)*.

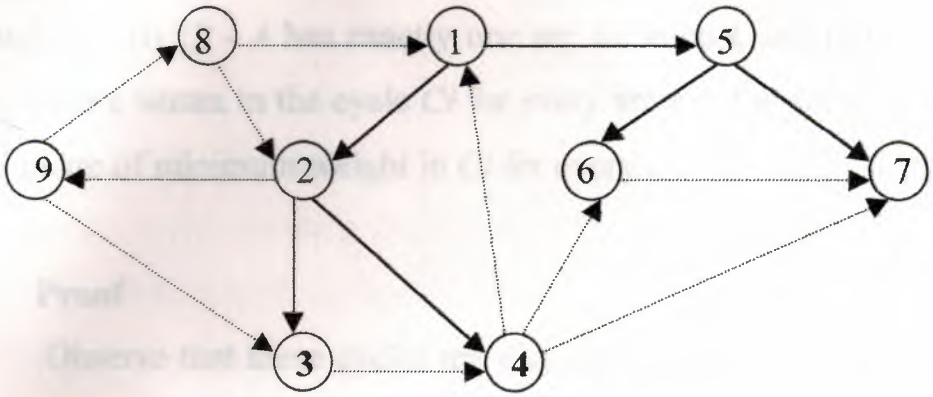
Suppose no arc in $C - A'$ is B-eligible. Then there exists a path in *B* from *t(ei)* to *s(ei)* for each *i*. Now there is a path in $C \cap A'$ from *t(ei-1)* to *s(ei)*. There is also a path in A' from *t(ei)* to *s(ei)*. So there is a path in A' from *t(ei-1)* to *t(ei)* or there is a path in A' from *t(ei)* to *t(ei-1)*. In the former case, there is a path *t(ei-1) t(ei) s(ei)* in the branching A'. But the unique path in A' from *t(ei-1)* to *s(ei)* is in $C \cap A'$. So the path from *t(ei-1)* to *t(ei)* is also in $C \cap A'$. So there should be an arc in $C \cap A'$ directed to *t(ei)*. But there is already the arc *ei* directed to *t(ei)* in the cycle. This arc *ei* is not B-eligible and therefore it is not in A'. Thus we have two arcs in the cycle - one in A' and one in its complement - such that both are directed to the vertex *t(ei)*. So there is no path in A' from *t(ei-1)* to *t(ei)* for each *i*. Hence there is a path in A' from *t(ei)* to *t(ei-1)* for *i* =1, 2, . . . , *k* and there is a path in A' from *t(e1)* to *t(ek)*. This situation creates the following cycle C' in A':

t(ek) t(ek -1) . . . t(e1) t(ek)

leading to a contradiction since A' is acyclic.

In the figure below, the arcs in the branching *B* are thick lines and the arcs not in *B* are dotted lines. The arc (8,1) is eligible and there is no path from 1 to 8 in *B*. By adjoining this arc to *B*, the cardinality of the set A' of arcs in the branching is increased by one. Likewise, the arc (2, 9) is eligible and there is no path from 9 to 2. The arc (l, 6) is B-eligible and if it enters *B*, the arc (5, 6) has to leave *B*. The arc (4,1) is not eligible and there is a path from 1 to 4 in *B*. The cycle $1 \rightarrow 2 \rightarrow 9 \rightarrow 8 \rightarrow 1$ has (1, 2) in the branching

and the three remaining arcs are B-eligible. In the cycle $1 \rightarrow 2 \rightarrow 4 \rightarrow 1$, the only arc not in the branching is(4,1) which is not B-eligible.



Theorem 1.25

If H is a critical subgraph of a digraph $G = (V, A)$, there exists a maximum weight branching $B = (V, A')$ in G such that, for every cycle C in H, the set $C - A'$ has exactly one arc.

Proof

Let $H = (V, A1)$ be a critical graph. From the collection of all maximum weight branchings, choose that branching $B = (V, A')$ which has the maximum number of arcs in common with the critical graph H. Let $e = (i, j)$ be any arc in A, but not in A'. Suppose e is B-eligible. Then the arcs in

$$A'' = A' \cup \{e\} - \{f : f \in A' \text{ and } t(e) = t(f)\}$$

form a maximum weight branching which has more arcs from the critical graph than A' has. So no arc in H is B-eligible. In particular, no arc in a cycle C in H is B-eligible. Hence by Theorem 1.24, the cardinality of the set $C - A'$ is 1.

Theorem 1.26

Suppose the cycles in a critical subgraph H of a digraph G are $C1$, $C2$, ... , Ck. Then there exists a maximum weight branching $B = (V, A')$ in G such that (i) Ci - A has exactly one arc for each i, and (ii) if arc e is not directed to a vertex in the cycle Ci for every arc e in the set A' - Ci, then Ci - A' is an arc of minimum weight in Ci for every i.

Proof

Observe that these cycles are disjoint in the sense that no two cycles have a vertex or an arc in common.

Let ei be an arc ofminimum weight in the cycle Ci. Let $S = \{e1, e2, ... , ek\}$. By Theorem 1.25, there exists (not uniquely) a maximum weight branching (which depends on H but not on the individual cycles) such that every arc except one of each Ci is an arc of this branching. Choose a maximum spanning branch $B = (V, A)$ of this kind which contains the minimum number of arcs from the set S. This branching B satisfies (i).

Suppose this branching B does not satisfy property (ii). So there exists a cycle Cj $(1 \leq j \leq k)$ where this property does not hold. No arc of A' - C is directed to a vertex in Vj and ej is not the arc Cj - B. So the arc ej is an arc in the branching B. Let e be the arc Ci - B. So $w(e) \leq w(ei)$. Then A' - $(ej) \cup \{e\}$ is a maximum weight branching which satisfies property (i) and has fewer edges than A' from the set S. This is a contradiction. So B satisfies property (ii). Thus B is the desired maximum weight branching in the digraph.

Theorem 1.27

There is a one-to-one correspondence between the set of maximum weight branchings in a weighted digraph G satisfying properties (i) and (ii) of Theorem 1.26 and the set of maximum weight branchings in a condensed digraph in G.

Proof

Let $G = (V, A)$ be a digraph with a weight function w. Let H be a critical graph in G with cycles Ci $(i = 1, 2, ..., k)$ and let $G1=(V1, A1)$ be the corresponding condensed graph.

We can define a weight function $w1$ on this condensed graph by the way the arcs in $A1$ are classified. The set $A1$ consists of two categories of arc: an arc e is in the first category when $t(e)$ is not a vertex in any of the cycles of H, in which case $w1(e) = w(e)$. Otherwise the arc e is in the second category in which case $w1(e) = w(e) - w(f) + w(ei)$, where f is the unique arc in the cycle Ci which is directed to $t(e)$ and ei is an arc of minimum weight in Ci.

Let $B = (V, A')$ be any branching in G satisfying properties (i) and (ii) of Theorem 1.26 using these cycles Ci.

An arc e in A' such that both $s(e)$ and t(e) are not in the same cycle in H defines a unique arc in $A1$; let $D1$ be the set of arcs thus defined. Then $B1=(V1, D1)$ is a branching in $G1$. (We may say that $D1$ is the `intersection' of A' and $A1$.) Thus once a critical graph is fixed in G, a branching in G defines a unique branching in the condensed graph $G1$ defined by the critical graph.

Now consider the condensed graph $G1= (V1, A1)$ defined by a critical graph H in a digraph G. Let the cycles in H be Ci $(i = 1, 2, \ldots, k)$. Let $B1= (V1, A1)$ be a branching in $G1$.

If the indegree in $B1$ of the condensed vertex corresponding to the cycle Ci is 0, let $Ci = Ci - \{ei\}$.

If this indegree is 1, there is a unique arc f (belonging to G) in Ci which is responsible for this. In this case let $Ci = Ci - \{f\}$. Thus from each cycle in H, we take all the arcs except one. Let X be the set of all arcs thus obtained from the k cycles.

Now consider arcs in B, which are directed to vertices which are not condensed vertices. Each such arc corresponds to a unique arc in G. Let Y be the set of arcs in G thus obtained. Then the union of X and Y constitutes a branching B in G.

Now we turn our attention to optimality. Let B and $B1$ be as described above. Let P be the sum of the weights of the k cycles in H. Let Y be the sum of the weights of the minimum weight arcs in these k cycles, taking exactly one arc from each cycle. It is a simple exercise to verify that $w(B) - w1(B1) = P - Q$.

Thus there is a one-to-one correspondence between the set of maximum weight branchings in a digraph G and the set of maximum weight branchings in a condensed digraph obtained by condensing each cycle in a critical graph H in G into a vertex.

Observe that Theorem 1.27 gives us a procedure for obtaining a maximum weight branching in G. First construct a critical subgraph H. If H is acyclic, then H is a maximum weight branching and we are done. Otherwise, we shrink each cycle into a vertex and readjust the weight function to obtain a condensed graph $G1$. We continue this process till we reach an acyclic critical raph H of G. first construct a critical subgarph H. ıf H is acyclic, then h is a maximum weight branching and we are done. Otherwise, we shrink each cycle into a vertex and readjust the weight

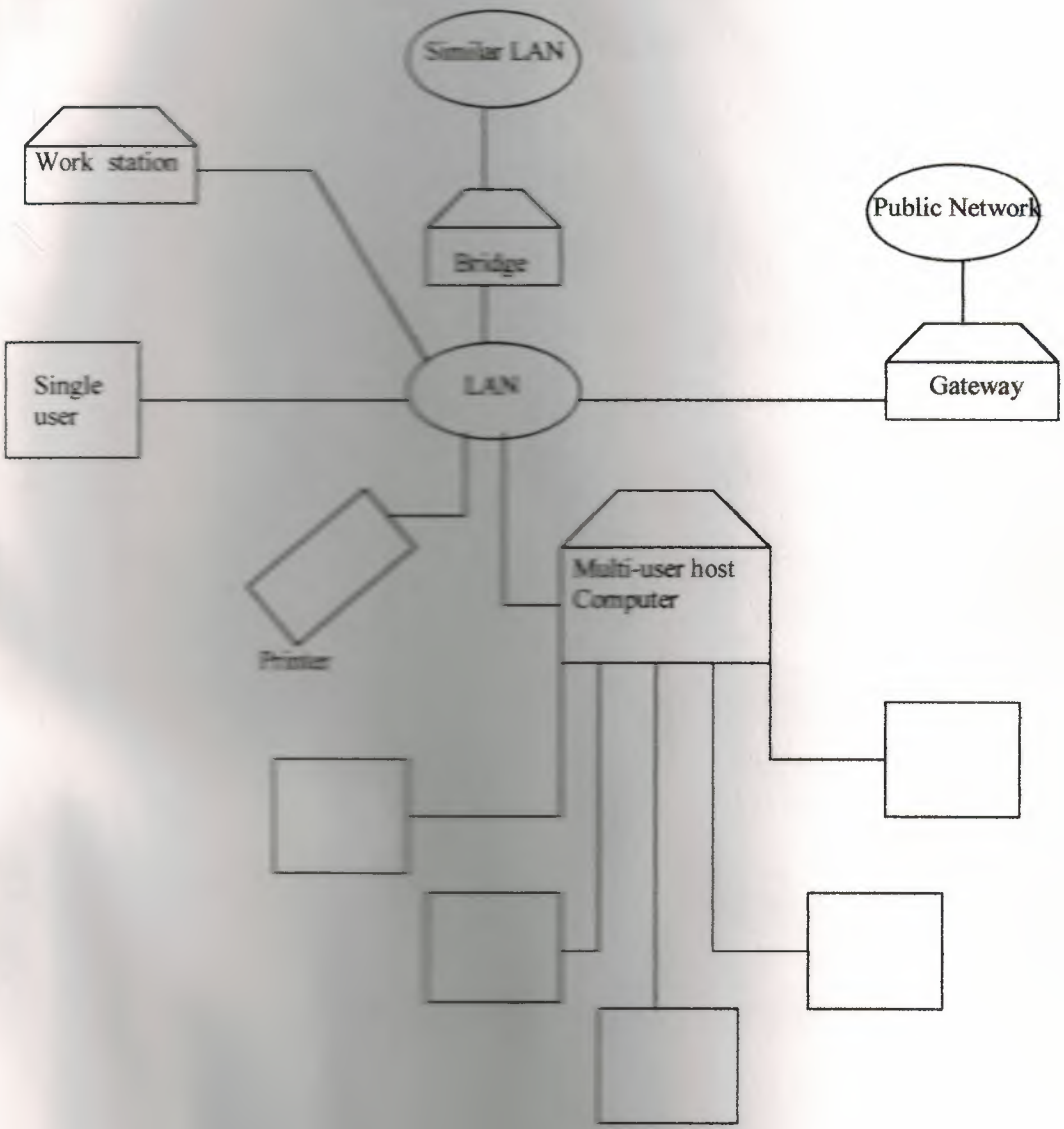
function to obtain a consensed graph *G1*. We continue this process till we reach an acyclic critical graph Hm of the condensed graph *Gm*. Thus we moe from *(G,H)* to *(Gm,Hm)*. Now the unraveling starts. *Hm* gives a maximum weight brancing in *Gm*. We expand the consensed vertices in *Hm-1* and obtain a maximum weight branching in *Gm-1*. We continue this process of expanding consensed cycles till we get a maximum weight brancing in the digraph *G*.

We can summarize the of maximum weight branching algorithm as follows:

Step 1(the condensation process). The input is *G=G0*. Construct *Gi* from *Gi-1* by consending cycles of a critical subgraph *Hi-1* of *Gi-1* and by modifiying the weight of the arcs. *Gk* is the first digraph in the sequence with a critical acyclic subgraph *Hk*.

Step 2(the unraveling process). The graph *Hk* is a maximum weight brancing in *Gk*. Let *Bk =Hk*. Construct *Bi-1* from *Bi* by expanding the condensed cycles. *Bi* is a maximum weight branching in *Gi* for i=*k, k-1,....,0*. *B=B0* is the output.

NETWORK


Similar LAN
Work station
Bridge
Public Network
Single user
LAN
Gateway
Printer
Multi-user host Computer