T.C

YAŞAR UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

MATHEMATICS

MASTER THESIS

THE ROLE OF GRAPH THEORY IN LOGISTICS

Sertaç CERRAHOĞLU

Supervisor

Asst. Prof. Dr. Gökşen BACAK TURAN

İzmir, 2011

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Asst. Prof. Dr. Gökşen BACAK TURAN, for her excellent guidance, patience, and always smiling face. From finding an appropriate subject in the begining to the process of writing thesis, she offers her unserved help and guidance and lead me to finish my thesis step by step. Also, I would like to thank her for scientific advice and knowledge suggestions.

I would like to thank my master's thesis committee for their support and helpful reviews: Asst. Prof. Dr. Gökşen BACAK TURAN, Prof. Dr. Alpay KIRLANGIÇ and Asst. Prof. Dr. Shahlar MAHARRAMOV.

Finally, I thank my parents for supporting me throughout all my studies at university.

YEMİN METNİ

Yüksek Lisans Tezi olarak sunduğum "THE ROLE OF GRAPH THEORY IN LOGISTICS" adlı çalışmanın, tarafımdan bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın yazıldığını ve yararlandığım eserlerin bibliyografyada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve bunu onurumla doğrularım.

..../..../.......

Sertaç CERRAHOĞLU

ABSTRACT

Master Thesis

THE ROLE OF GRAPH THEORY IN LOGISTICS

Sertaç CERRAHOĞLU

Yaşar University

Institute of Natural and Applied Sciences

Master of Mathematics

This thesis consists of six chapters. In the first chapter, an introductory approach is given. In the second chapter, the basic notions of graph theory contained in this thesis are introduced. In chapter three, the concept of network is defined and some network flow problems are mentioned. In the fourth chapter some information about logistics and transportation are given. The definition and history of logistics are mentioned and the role of transportation in logistics is explained. In chapter five some applications of graph theory are applied to logistics problems. Finally, the conclusion is given.

Keywords: Network Flows, Shortest Path Problem, Maximum Flow Problem, Transportation Problem.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Graph theory is the branch of mathematics that concerns with graphs, which are the constructs consisting of vertices and edges. The paper written by Leonhard Euler on the Seven Bridges of Königsberg and published in 1736 is regarded as the first paper in the history of graph theory. Graph theory has proven to be particularly useful to a large number of rather diverse fields, such as computer science, biology, physics, sociology, logistics, etc. The exiciting and rapidly groving area of graph theory is rich in the theoritical results as well as applications to real-world problems.

Many real-life situations can be described by means of a diagram of a set of points with lines joining certain pairs of points. A graph structure can be extended by assigning a weight to each edge of the graph. Graph with weights, or weighted graphs, are used to represent structures in which pairwise connections have some numerical values. For example if a graph represents a road network, the weights could represent the length of each road. Graphs model many situations like the connections of wires/leads, logistics/transportation problems, pipelines between points with known capacities, family trees, organizational charts, etc. The weights of the edges can be express the distance between two locations, the journey time, traffic expenses and so on.

This thesis aims to review the literature how graph theory can be applied to logistics. Firstly basic notaions of graph theory are given, then networks and flows in networks are taken into consideration. Shortest path problems, transportation problems, and maximum flow problems are discussed and examined. An introductory information about logistics and transportation are given. As applications of graph theory Dijkstra’s algorithm, Ford-Fulkerson algorithm and North-West Corner method are applied to some logistics problems.

CHAPTER 2

PRELIMINARIES

Graph Theory is now a major tool in mathematical research, electrical engineering, computer programming and networking, business administration, sociology, economics, marketing, and communications; the list can go on and on. In particular, many problems can be modelled with paths formed by traveling along the edges of a certain graph. For instance, problems of efficiently planning routes for mail delivery, garbage pickup, snow removal, diagnostics in computer networks, and others, can be solved using models that involve paths in graphs.

2.1 Graphs

A **graph** $G = (V,E)$ is a finite, nonempty set $V(G)$, together with a (possibly empty) set $E(G)$ of 2-elements subsets of $V(G)$. The elements of V are called **vertices**, while those of E are called **edges**. The number of vertices in a graph G is called the **order** of G, denoted by $p = |V(G)|$, while the number of edges in G is called the size of G, denoted by $q = |E(G)|$. A graph of order p and size q is often referred to as a (p,q) - graph. If the unordered pair $e = \{u,v\}$ is an edge of the graph G, informally written as $e = uv$, it is said that the vertices u and v are **adjacent** in G and that the edge e joins u and v. The edge e is said to be **incident** with the vertices u and v. A graphical representation of an order 7 graph G_1 of size 8 is shown in Figure 2.1. The vertex set is $V(G_1) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and edge set is $E(G_1) = \{v_1v_6, v_1v_7, v_2v_4, v_3v_5, v_3v_6, v_3v_7, v_4v_5, v_5v_6\}$. The vertices v_1 and v_6 are adjacent in G_1, while v_1 and v_2 are not (West, D.B, 2001).

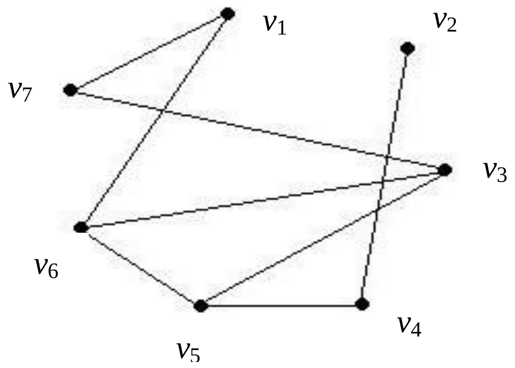


Figure 2.1: Graphical representation of a (7,8) − graph, G_1.

The **open neighbourhood** of a vertex v in a graph G is defined as the set

$$N_G(v) = \{u \in V(G) : uv \in E(G)\} ,$$

while the **closed neighbourhood** of v in G is defined as

$$N_G[v] = N_G(v) \cup \{v\} .$$

The open neighbourhood of a set S is defined as $N(S) = \{N(v) : v \in S\}$, while the closed neighbourhood of a set S is defined as $N[S] = \{N[v] : v \in S\}$. For any vertex v in a graph G, the number of vertices adjacent to v, i.e. $|N_G(v)|$, is called the degree of v in G, denoted by $\deg_G v$. Note that if the reference to a graph G is clear from the context, the subscript is often omitted, hence written as $\deg v$ only.

A vertex is **odd** if its degree is odd and **even** if its degree is even. If the degree of a vertex is 0, it is called an **isolated vertex**, while if the degree is 1, it is called an **end-vertex**. The **minimum degree** of vertices in G is denoted by $\delta(G)$, while the **maximum degree** of the vertices is denoted by $\Delta(G)$. Referring to the graph G_1 in Figure 2.1, the open neighbourhood of the vertex v_5 is $N_{G1}(v_5) = \{v_3, v_4, v_6\}$, while its closed neighbourhood is $N_{G1}[v_5] = \{v_3, v_4, v_5, v_6\}$. The graph has no isolated vertices, but v_2 is, in fact, an end-vertex. The minimum degree of G_1 is therefore $\delta(G_1) = 1$, while the maximum degree is $\Delta(G_1) = 3$.

The **Fundamental Theorem of Graph Theory**, is probably one of the most well -known results in the discipline and relates the sum total of the degrees and the size of any graph.

Theorem 2.1 (Chartrand, G. and Lesniak, L. 2000). Let G be a (p, q) - graph, with $V(G) = \{v_1, v_2, \ldots, v_p\}$.

$$\sum_{i=1}^{p} \deg_G v_i = 2q$$

Proof: When the degrees of all the vertices are summed, each edge is counted twice, once for each of the vertices that it joins. ■

When $G = (V, E)$ and $H = (W, F)$ are graphs, we say H is a **subgraph** of G when $W \subseteq V$ and $F \subseteq E$.We say H is an **induced subgraph** when $W \subseteq V$ and $F = \{xy \in \text{E}: x, y \in W\}$. In Figure 2.2, we show a graph, a subgraph and an induced subgraph.

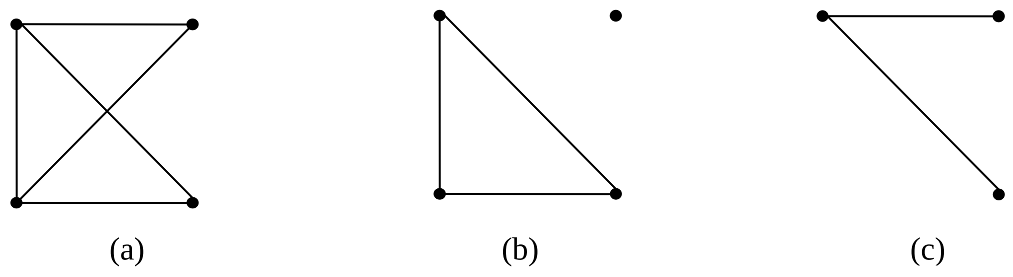


Figure 2.2 : (a) a graph, (b) a spanning subgraph, (c) an induced subgraph.

2.2 Connectedness

A graph $G = (V, E)$ is called a **complete graph** when xy is an edge in G for every distinct pair $x, y \in V$. Conversely, G is an independent graph if $xy \notin E$, for every distinct pair $x, y \in V$. It is customary to denote a complete graph on n vertices by K_n and an independent graph on n vertices by I_n (Gross, L.J and Yellen, J., 2006).

A **walk** is an alternating sequence of vertices and edges, beginning and ending with a vertex, where each vertex is incident to both the edge that precedes it and the edge that follows it in the sequence, and where the vertices that precede and follow an edge are the end vertices of that edge. A walk is **closed** if its first and last vertices are the same, and **open** if they are different.

The **length** l of a walk is the number of edges that it uses. For an open walk, $l = n–1$, where n is the number of vertices visited (a vertex is counted each time it is visited). For a closed walk, $l = n$ (the start/end vertex is listed twice, but is not counted twice).

A **trail** is a walk in which all the edges are distinct. A closed trail has been called a **tour** or **circuit**, but these are not universal, and the latter is often reserved for a regular subgraph of degree two.

A graph $G = (V, E)$ on $n \geq 1$ vertices is called a **path** when the elements of the vertex set can be labelled as $\{x_1, x_2, \ldots , x_n\}$ so that $E = \{x_i x_{i+1} : 1 \leq i < n\}$.Similarly, if $n \geq 3$, G is called a **cycle** when $E = \{ x_i x_{i+1} : 1 \leq i < n\} \cup \{x_n x_1\}$.It is customary to denote a path on n vertices by P_n, while C_n denotes a cycle on n vertices. The length of a path or a cycle is the number of edges it contains. Therefore, the length of P_n is $n - 1$ and the length of C_n is n (Chartrand, G. and Lesniak, L. 2000).

A graph G is **connected** when there is a path from x to y in G, for every $x, y \in V$; else G is **disconnected**. The induced subgraph on such an equuivalence class is called a **connected component** or just **component** of the graph. A graph is connected if there is just one equivalence class, that is, if every pair of vertices is connected.

A graph is **acyclic** when it does not contain any cycle on three or more vertices. Acyclic graphs are also called forests. A connected acyclic graph is called a **tree**. When $G = (V, E)$ is a connected graph, a subgraph $H = (W, F)$ of G is called a **spanning tree** when $W = V$ and H is a tree. In Figure 2.3, we show a graph and its spanning tree.

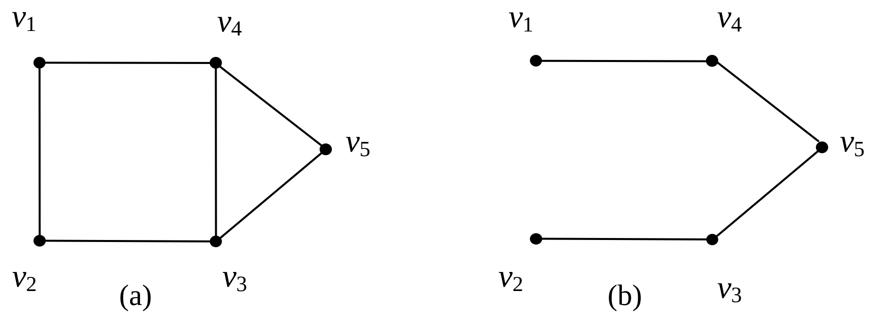
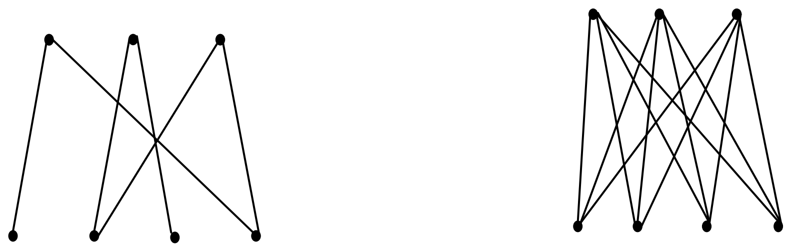


Figure 2.3 : (a) a graph, (b) a spanning tree.

A graph is called **bipartite**, if the corresponding node set can be split into two sets N_1 and N_2 in such a way that each member of S joins a node of N_1 to a node of N_2. A **complete bipartite** graph is a bipartite graph in which each node N_1 is joined to each node of N_2 by exactly one member. If the number of nodes in N_1 and N_2 are denoted by *r* and *s*, respectively, then a complete bipartite graph is denoted by Kr,s. Examples of bipartite and complete bipartite graphs are shown in Figure 2.4 (Chartrand, G. and Lesniak, L. 2000).



(a) A bipartite graph (b) A complete bipartite graph $K_{3,4}$

Figure 2.4: Two bipartite graphs. (a) A bipartite graph. (b) A complete bipartite graph $K_{3,4}$

A graph can be represented in various forms. Some of these representations are of theoretical importance, others are useful from the programming point of view when applied to realistic problems. One of them is matris representation. Every graph has associated with it an adjacency matrix, which is a binary $n \times n$ matrix A in which $a_{ij} = 1$ and $a_{ji} = 1$ if vertex v_i is adjacent to vertex v_j, and $a_{ij} = 0$ and $a_{ji} = 0$ otherwise. The adjacency matrix is a matrix of size $V \times V$ such that

$$Mij = \begin{cases} 1, \text{ if there is an edge between } i \text{ and } j \\ 0, \text{ otherwise} \end{cases}$$

The figure 2.3 (a) graphical representation of an adjacency matrix is a table, such as shown in Figure 2.5

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	0	1	0
v_2	1	0	1	0	0
v_3	0	1	0	1	1
v_4	1	0	1	0	1
v_5	0	0	1	1	0

Figure 2.5: Matrix representation of a graph

2.3 Eulerian and Hamiltonian Graphs

Graph theory began with Euler's study of a particular problem: the SevenBridges of Königsberg. During the eighteenth century the city of Königsberg (in East Prussia) was divided into four sections (including the island of Kneiphof) by the Pregel river. Seven bridges connected these regions and it was said that residents spent their Sunday walks trying to find a starting point so that they could walk about the city, cross each bridge exactly once, and return to their starting point.

Eulerian Graphs

The following problem, often referred to as the bridges of Königsberg problem, was first solved by Euler in the eighteenth century. The problem was rather simple - the town of Königsberg consists of two islands and seven bridges. Is it possible, by beginning anywhere and ending to the same point, to walk through the town by crossing all seven bridges but not crossing any bridge twice?

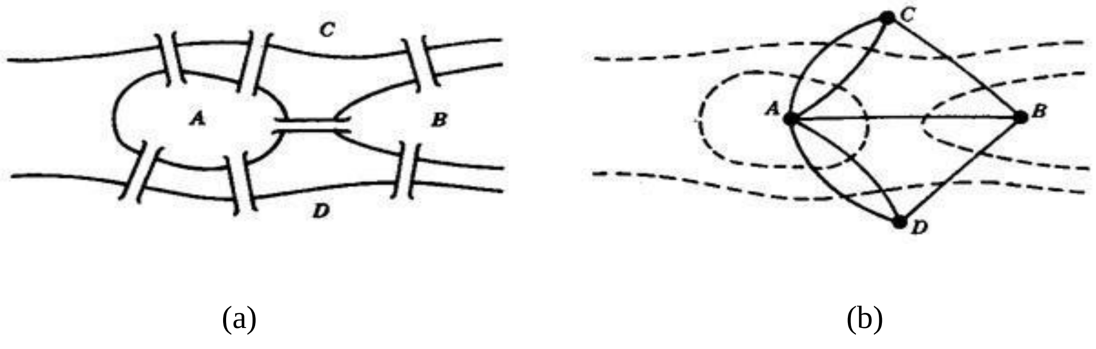


Figure 2.6 : (a) is Königsberg in 1736, (b) is Euler's graphical representation

This problem was solved in 1736 by the Swiss mathematician Euler, in the earliest known paper on graph theory, who studied the famous problem of "the bridges of Königsberg". Euler proved it is impossible to take a walk crossing all seven bridges of the river Pregel exactly once (West, D.B., 2001).

Eulerian trail: An Eulerian trail is a trail that visits every edge of the graph once and only once. It can end on a vertex different from the one on which it began. A graph of this kind is said to be traversable.

Eulerian Cycle: An Euler cycle is a cycle that traverses every edge of a graph exactly once. If there is an open path that traverse each edge only once, it is called an Euler path.

Eulerian Graph: A graph is called Eulerian when it contains an Eulerian circuit.

An Eulerian trail exists in a connected graph if and only if there are either no odd vertices or two odd vertices. For the case of no odd vertices, the path can begin at any vertex and will end there; for the case of two odd vertices, the path must begin at one odd vertex and end at the other. Any finite connected graph with two odd vertices is traversable. A traversable trail may begin at either odd vertex and will end at the other odd vertex (Gallier, J., 2005).

Hamiltonian Graphs

In 1859, the Irish mathematician Sir William Rowan Hamilton developed a game that he sold to a Dublin toy manufacturer. The game consisted of a wood regular dodecahedron with the twenty corner points (vertex) labelled with the names of prominent cities. The object of the game was to find a circuit along the edges of the solid so that each city on the cycle exactly once.

Hamiltonian Cycle: A Hamiltonian cycle in a graph is a closed path that visits every vertex in the graph exactly once. A Hamiltonian cycle ends up at the vertex from where it started (West, D.B., 2001).

Hamiltonian graphs are named after the nineteenth-century Irish mathematician Sir William Rowan Hamilton(1805-1865). This type of problem is often referred to as the traveling salesman or postman problem.

Hamiltonian Graph: If a graph has a Hamiltonian cycle, then the graph is called a Hamiltonian graph. We represent the solid by a graph: the vertices of the graph correspond to the vertices of the solid and the edges similarly correspond:

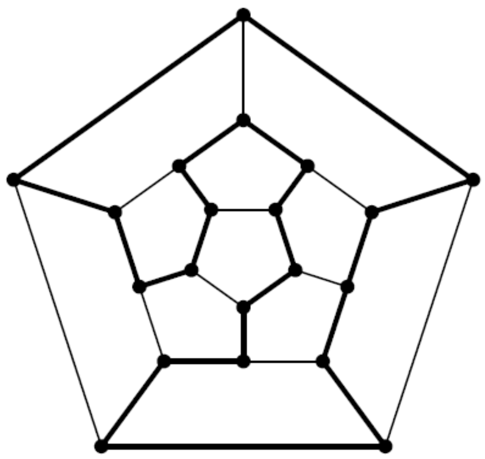


Figure 2.7 : Hamiltonian graph

2.4 Digraphs

A **directed graph** (or just digraph) *D* consists of a non-empty finite set *V*(*D*) of elements called **vertices** and a finite set *E*(*D*) of ordered pairs of distinct vertices called **arcs**. We call *V* (*D*) the vertex set and *E*(*D*) the arc set of *D*. We will often write *D* = (*V, E*) which means that *V* and *E* are the vertex set and arc set of *D,* respectively. The order (size) of *D* is the number of vertices (arcs) in *D*; the order of *D* will be sometimes denoted by |*D*| . For example, the digraph *D* in Figure 2.6 is of order and size 6; *V*(*D*) = {*u*, *v* ,*w*, *x*, *y*, *z* }, *E*(*D*) = {(*u*, *v*) , (*u*,*w*), (*w*, *u*), (*z*, *u*), (*x*, *z*), (*y*, *z*)}. Often the order (size) of the digraph under consideration is denoted by *n* (or *m*).

For an arc (*u*, *v*) the first vertex *u* is its tail and the second vertex *v* is its head. We also say that the arc (*u*, *v*) leaves *u* and enters *v*. The head and tail of an arc are its end-vertices; we say that the end-vertices are adjacent, *u* is adjacent to *v* and *v* is adjacent to *u*. If (*u*, *v*) is an arc, we also say that *u* dominates *v* (or *v* is dominated by *u*) and denote it by *u*→*v*. We say that a vertex *u* is incident to an arc *e* if *u* is the head or tail of *a:* We will often denote an arc (*u*, *v*) by *uv* (Chartrand, G. and Lesniak, L., 2000).

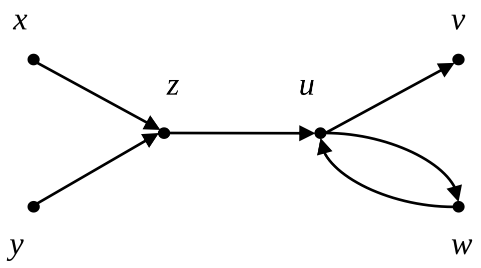


Figure 2.8 : A digraph *D*

For a pair *X, Y* of vertex sets of a digraph *D*, we define

$$(X, Y)_D = \{xy \in E(D) : x \in X,\ y \in Y\},$$

i.e. $(X, Y)_D$ is the set of arcs with tail in *X* and head in *Y* . For example, for the digraph *H* in Figure 2.6, $(\{u,v\}, \{w,z\})_H = \{uw\}$, $(\{w,z\}, \{u,v\})_H = \{wv\}$, and $(\{u,v\},\{u,v\})_H = \{uv,vu\}$.

An **undirected graph** (or a graph) $G = (V,E)$ consists of a non-empty finite set $V = V(G)$ of elements called vertices and a finite set $E = E(G)$ of unordered pairs of distinct vertices called edges.

In other words, if each edge of the graph G has no direction then the graph is called **undirected graph.**

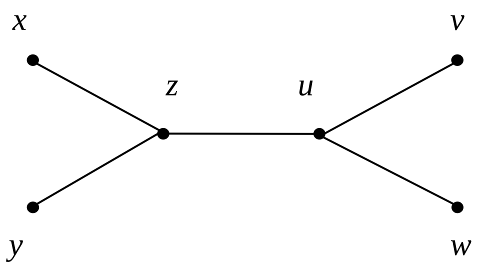


Figure 2.9 : An undirected graph *D*

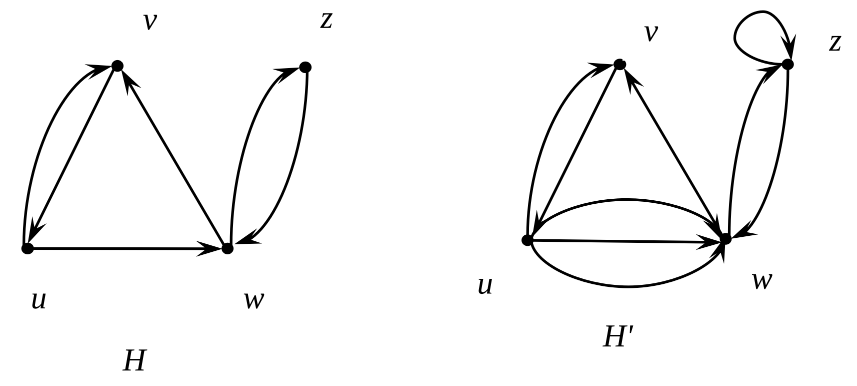


Figure 2.10 : A digraph H and a directed pseudograph H'

The above definition of a digraph implies that we allow a digraph to have arcs with the same end-vertices (for example, *uv* and *vu* in the digraph *H* in Figure 2.10), but we do not allow it to contain parallel (also called multiple) arcs, that is, pairs of arcs with the same tail and the same head, or loops (i.e. arcs whose head and tail coincide). When parallel arcs and loops are admissible we speak of **directed pseudographs**; directed pseudographs without loops are **directed multigraphs**. In Figure 2.8 the directed pseudograph *H'* is obtained from *H* by appending a loop *zz* and two parallel arcs from *u* to *w*. Clearly, for a directed pseudograph *D*, *E*(*D*) and $(X, Y)_D$ (for every pair *X*, *Y* of vertex sets of *D*) are multisets (parallel arcs provide repeated elements). We use the symbol $\mu_D(x, y)$ to denote the number of arcs from a vertex *x* to a vertex *y* in a directed pseudograph *D*. In particular, $\mu_D(x, y) = 0$ means that there is no arc from *x* to *y* (Bang-Jensen, J and Gutin, G., 2007).

We will sometimes give terminology and notation for digraphs only, but we will provide necessary remarks on their extension to directed pseudographs, unless this is trivial.

The **indegree** of a vertex ($d^{-}(v)$) in a directed graph is the number of edges directed into it; its **outdegree** ($d^{+}(v)$) is the number of edges directed away from it; its **degree** is the sum of its indegree and outdegree. The sum of the in degree and out degree of a vertex is called the **total degree** of the vertex. A vertex with zero in degree is called a **source** and a vertex with zero out degree is called **a target.** Since each edge has an initial vertex and terminal vertex.

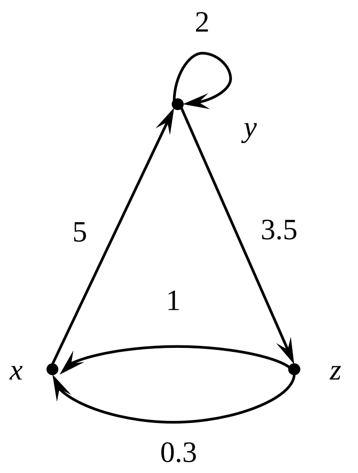
Proposition 2.1. (Gross, L.J and Yellen, J., 2006) For any digraph *D,*

$$\sum_{v \in V(D)} d^{-}(v) = \sum_{v \in V(D)} d^{+}(v) = |E(D)|$$

■

A digraph *H* is a subdigraph of a digraph D if *V* (*H*) ⊆ *V* (*D*), *E*(*H*) ⊆ *E*(*D*) and every arc in *E*(*H*) has both end-vertices in *V* (*H*). If *V* (*H*) = *V* (*D*), we say that *H* is a **spanning subdigraph** (or a factor) of *D*. The digraph *L* with vertex set {*u, v, w, z*} and arc set {*uv, uw, wz*} is a **spanning subdigraph** of *H*. If every arc of *E*(*D*) with both end-vertices in *V* (*H*) is in *E*(*H*), we say that *H* is induced by *X* =*V*(*H*) and call *H* an induced subdigraph of *D*.

A **weighted directed pseudograph** is a directed pseudograph *D* along with a mapping *c*: *E*(*D*) → *R*. Thus, a weighted directed pseudograph is a triple *D* = (*V*(*D*), *E*(*D*), *c*).We will also consider **vertex-weighted directed pseudographs**, i.e. directed pseudographs *D* along with a mapping *c*: *V*(*D*) → *R*. (See Figure 2.11) If *a* is an element (a vertex or an arc) of a weighted directed pseudograph *D* = (*V* (*D*), *E*(*D*), *c*), then *c*(*a*) is called the **weight** or the **cost** of *a* . An (unweighted) directed pseudograph can be viewed as a (vertex-)weighted directed pseudograph whose elements are all of weight one. For a set *B* of arcs of a weighted directed pseudograph *D* = (*V*, *E*, *c*), we define the weight of *B* as follows: *c* (*B*) = $\sum_{a \in B} c(a)$. Similarly, one can define the weight of a set of vertices in a vertex-weighted directed pseudograph. The **weight of a subdigraph** *H* of a weighted (vertex-weighted) directed pseudograph *D* is the sum of the weights of the arcs in *H* (vertices in *H*). For example, in the weighted directed pseudograph *D* in Figure 2.9 the set of arcs {*xy*, *yz*, *zx* } has weight 9.5 (here we have assumed that we used the arc *zx* of weight 1). In the directed pseudograph *H* in Figure 2.9 the subdigraph *U* = ({*u*,*x*,*z*},{*xz*,*zu*}) has weight 5 (Bang-Jensen, J and Gutin, G., 2007).



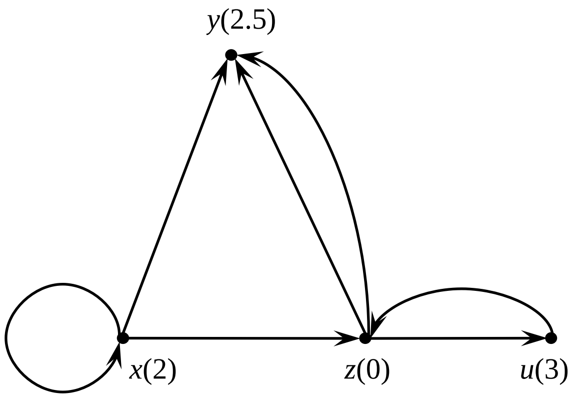


Figure 2.11:Weighted and vertex-weighted directed pseudographs (the vertex weights are given in brackets).

Walk, Trail, Path and Cycle

In the following, D is always a directed pseudograph, unless otherwise specified. A **walk** in D is an alternating sequence $W = x_1a_1x_2a_2x_3 \dots x_{k-1}a_{k-1}x_k$ of vertices x_i and arcs a_j from D such that the tail of a_i is x_i and the head of a_i is x_{i+1} for every $i = 1, 2, \dots, k - 1$. A walk W is closed if $x_1 = x_k$, and open otherwise. The set of vertices $\{x_1, x_2, \dots, x_k\}$ is denoted by $V(W)$; the set of arcs $\{a_1, a_2, \dots, a_{k-1}\}$ is denoted by $E(W)$. We say that W is a walk from x_1 to x_k or an (x_1, x_k)-walk. If W is open, then we say that the vertex x_1 is the **initial vertex** of W, the vertex x_k is the **terminal vertex** of W, and x_1 and x_k are **end-vertices** of W. The **length** of a walk is the number of its arcs. Hence the walk W above has length k-1. A walk is even (odd) if its length is even (odd). When the arcs of W are defined from the context or simply unimportant, we will denote W by $x_1x_2 \dots x_k$ (West, D.B., 2001).

A **trail** is a walk in which all arcs are distinct. Sometimes, we identify a trail W with the directed pseudograph $(V(W), E(W))$, which is a subdigraph of D. If the vertices of W are distinct, W is a) **path**. If the vertices $x_1, x_2, \dots, x_{k-1}$ are distinct, $k \geq 3$ and $x_1 = x_k$, W is a **cycle** (Harju, T.,2007).

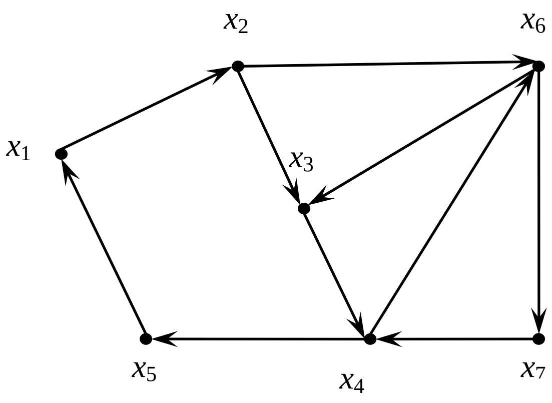


Figure 2.12 : A directed graph H .

The path $x_1x_2x_3x_4x_6$ is an (x_1, x_6) -path and $x_2x_3x_4x_6x_3$ is an (x_2, x_3) -trail. The cycle $x_1x_2x_3x_4x_5x_1$ is a 5-cycle in D.

Eulerian Cycles

The following graph is a directed graph version of the Königsberg bridge problem, solved by Euler in 1736. The vertices A,B,C,D correspond to four areas of land in Königsberg and the edges to the seven bridges joining these areas of land. The problem is to find a closed path that crosses every bridge exactly once and returns to the starting point.

In fact, the problem is unsolvable, as shown by Euler, because some vertices do not have the same number of incoming and outgoing edges (In the undirected version of the problem, some vertices do not have an even degree.)

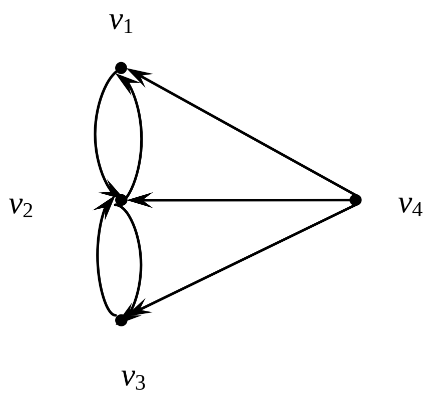


Figure 2.13: A directed graph modeling the Königsberg bridge problem

Hamiltonian Cycles

A Hamiltonian cycle, also called a Hamiltonian circuit, Hamilton cycle, or Hamilton circuit, is a graph cycle (closed loop) through a graph that visits each node exactly once (Skiena 1990, p. 196). By convention, the trivial graph on a single node is considered to posses a Hamiltonian cycle, but the connected graph on two nodes is not. A graph possessing a Hamiltonian cycle is said to be a Hamiltonian graph. The Hamiltonian cycle is named after Sir William Rowan Hamilton, who devised a puzzle in which such a path along the polyhedron edges of an dodecahedron was sought (the Icosian game)

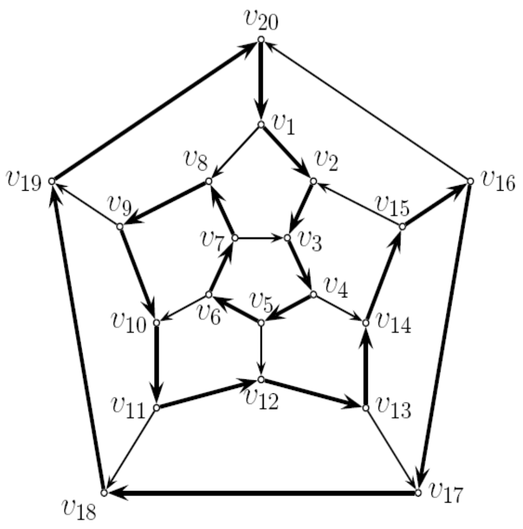


Figure 2.14 : A Hamiltonian cycle in *D*

Although graph theory is one of the younger branches of mathematics, it is fundamental to a number of applied fields, including operations research, computer science, and social network analysis. In this chapter we discussed the basic concepts of graph theory from the point of view of network analysis.

CHAPTER 3

FLOWS IN NETWORKS

In this chapter we consider an important algorithmic problem called the Network Flow Problem. Network flow is important because it can be used to express a wide variety of different kinds of problems. So, by developing good algorithms for solving network flow, we immediately will get algorithms for solving many other problems as well. A wide variety of engineering and management problems involve optimization of network flows – that is, how objects move through a network. Examples include coordination of trucks in a transportation system, routing of packets in a communication network, and sequencing of legs for air travel. Such problems often involve few indivisible objects, and this leads to a finite set of feasible solutions. Surprisingly, as we will see in this chapter, network flows problems can often be solved by algorithms.

Graph theory provides a framework for discussing systems in which it is possible to travel between discrete vertices. If we extend a directed graph to a network flow by assigning a capacity and a flow value to every edge, then this flow can be used to model any number of systems in which a resource travels from one point to another, e.g. the spread of data in a network, traffic along roads, water in pipes, and so on.

3.1 Networks

In graph theory, a flow network is a directed graph where each edge has a **capacity** and each edge receives a flow. The amount of flow on an edge cannot exceed the capacity of the edge. Often in Operations Research, a directed graph is called a **network**, the vertices are called **nodes** and the edges are called **arcs**. A network can be used to model traffic in a road system, fluids in pipes, currents in an electrical circuit, or anything similar in which something travels through a network of nodes. Figure 3.1 offers a visual representation of a directed graph with nodes labelled 1 through 8. We will denote an edge pointing from a node *i* to a node j by (*i*, *j*). In this notation, the graph of Figure 3.1 can be characterized in terms of a set of nodes

$V = \{s, u, v, w, x, y, z, t\}$ and a set of edges $E = \{(s, u), (s, y), (s, z), (u, v), (y, x), (x, z), (v, t), (z, v), (z, w), (w, t)\}$.

Graphs can be used to model many real networked systems. For example, in modelling air travel, each node might represent an airport, and each edge a route taken by some flight. Note that, to solve a specific problem, one often requires more information than the topology captured by a graph. For example, to minimize cost of air travel, one would need to know costs of tickets for various routes (West, D.B., 2001).

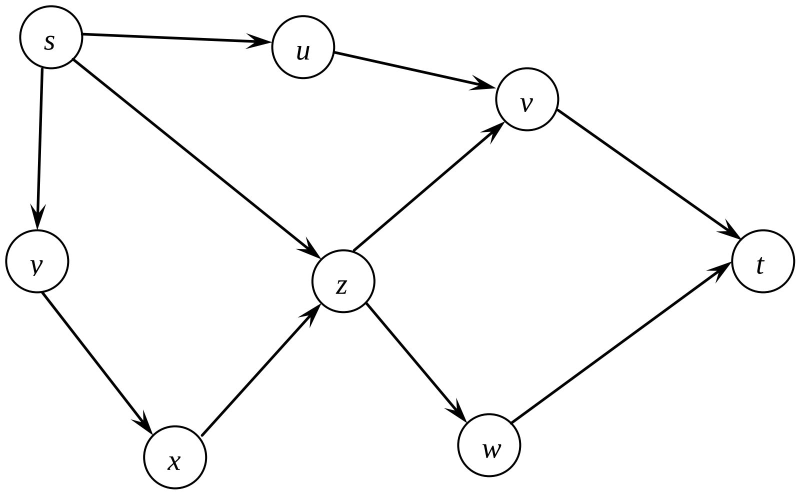


Figure 3.1: A directed graph

3.2 Network Flow Problems

Applications of graph theory, together with concepts from optimization theory, to practical problems is the domain of network flow problems. For example, the network flow problem related with Euler tours is the Chinese Postman Problem (after the Chinese mathematician, Kwan Mei-Ko, who discovered it in early 1960's). The problem is the following. Suppose the distances of the bridges are known and we have to cross each bridge at least once. We would like to find a route that comes back to the starting point which minimizes the total distance traveled on the bridges. Clearly, Euler tour, if it exists, is the optimal solution to the Chinese Postman

Problem. But if the network we have on hand does not satisfy the Euler's conditions, Chinese Postman Problem attempts to find a tour with minimal repetition of the edges. The Chinese Postman Problem is only one among many network flow problems.The traveling salesman problem is a well-known problem in the area of network and combinatorial optimization. This problem is easy to state: Starting from his home base, node 1, a salesman wishes to visit each of several cities represented by nodes 2,…,*n*, exactly once and return home, doing so at the lowest travel cost.The simplicity of this problem and its complexity to solve have attracted the attention of many researchers over a long period of time. The first mathematical model related to the traveling salesman problem was studied in the 1800s. Researchers have paid attention to this problem because it is a generic core model that captures the combinatorial essence of most routing problems (Ahuja, R.K., Magnanti, T.L., and Orlin, J.B, 1993).

Networks for the transmission of information, the transportation of people, and the distribution of goods and energy are part of our everyday life. Think about how all the utilities and the telephone, Internet, and cable-TV services made available in our homes and offices, how our mobility is made possible by the highway, rail, and airline networks. Without effective distribution and logistics networks, we could not have all the goods and services that are available now at affordable prices. The general problem types, namely, the minimum cost flow problems, the shortest path problems, the maximum flow problems, transportation problems and assignment problems.

We focus on the shortesth path, maximum flow and transportation problems in logistics.

3.2.1 The Shortest Path Problem

This network flow problem is one that we all use in our daily lives: what is the fastest route to take between two locations in the city during the rush hour, what is the "most" scenic route to drive, or the cheapest route to fly, between two cities in our vacation. The street intersections or the cities are the nodes, and the street or highway segments in between the intersections, or non-stop flights between the cities are the arcs. There are numbers associated with the arcs. These numbers can be the estimates of how long will it take to drive through a street segment during the rush hour, or a measure of "scenic pleasure" one expects to obtain traveling on a certain highway segment, or the cost of flying between two cities. The objective is to find a series of arcs connected with correct directions such that one can start at the origin and arrive at the destination traversing the arcs and that the sum of the numbers on the arcs are either minimum (in the case of the fastest or the cheapest route).

Given a weighted, directed graph *G*, a start node s and a destination node *t*, **the *s-t* shortest path problem** is to output the shortest path from *s* to *t*. The single-source shortest path problem is to find shortest paths from *s* to every node in *G*. The (algorithmically equivalent) single-sink shortest path problem is to find shortest paths from every node in *G* to *t*.

The shortest-path problem is a particular network model that has received a great deal of attention for both practical and theoretical reasons. The essence of the problem can be stated as follows: Given a network with distance c_{ij} (or travel time, or cost, etc.) associated with each arc, find a path through the network from a particular origin (source) to a particular destination (target) that has the shortest total distance. The simplicity of the statement of the problem is somewhat misleading, because a number of important applications can be formulated as shortest- (or longest-) path problems where this formulation is not obvious at the outset. These include problems of equipment replacement, capital investment, project scheduling, and inventory planning. The theoretical interest in the problem is due to the fact that it has a special structure, in addition to being a network, that results in very efficient solution procedures. The problem is also sometimes called the **single-pair shortest path problem**, to distinguish it from the following generalizations:

- The **single-source shortest path problem**, in which we have to find shortest paths from a source vertex *v* to all other vertices in the graph.
- The **single-destination shortest path problem**, in which we have to find shortest paths from all vertices in the graph to a single destination vertex *v*. This can be reduced to the single-source shortest path problem by reversing the edges in the graph.
- The **all-pairs shortest path problem**, in which we have to find shortest paths between every pair of vertices v_i, v_j in the graph.

In general, the (linear programming) formulation of the shortest-path problem is as follows:

$$\text{minimize } \sum_i \sum_j c_{ij} x_{ij},$$

subject to:

$$\sum_j x_{ij} - \sum_k x_{ki} = \begin{cases} 1 & \text{if } i = s \text{ (source)}, \\ 0 & \text{otherwise}, \\ -1 & \text{if } i = t \text{ (target)} \end{cases}$$

$$x_{ij} \geq 0 \text{ for all } i\text{-}j \text{ in the network.}$$

We can interpret the shortest-path problem as a network-flow problem very easily.

There exist many techniques for solving the shortest path problem, some of the better known algorithms are Dijkstra’s Algorithm (Dijkstra, E.W, 1959) and Bellman-Ford Algorithm (Bellman, R.,1957).

Dijkstra's Algorithm

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959, (Dijkstra, E.W., 1959) is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms.

For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities . Dijkstra's algorithm finds the length of an **optimal** path between two vertices in a graph. (Optimal can mean shortest or cheapest or fastest or optimal in some other sense: it depends on how you choose to label the edges of the graph.) The algorithm characterizes each vertex (node) by its state. The state of a node consists of two features: **distance value** and **status label**. Distance value of a node is a scalar representing an estimate of the its distance from node *s*. Status label is an attribute specifying whether the distance value of a node is equal to the shortest distance to node *s* or not.

- The status label of a node is **Permanent** if its distance value is equal to the shortest distance from node *s*
- Otherwise, the status label of a node is **Temporary**

The algorithm maintains and step-by-step updates the states of the nodes. At each step one node is designated as **current**

- d_l denotes the distance value of a node *l*.
- *p* or *t* denotes the status label of a node, where *p* stand for permanent and *t* stands for temporary
- c_{ij} is the cost of traversing link (*i*, *j*) as given by the problem.

The state of a node *l* is the ordered pair of its distance value d_l and its status label.

Algorithm Steps

Step 1. (Initialization)

- Assign the zero distance value to node *s*, and label it as **Permanent**. (The state of node *s* is (0, *p*))
- Assign to every node a distance value of ∞ and label them as **Temporary**. (The state of every other node is (∞,*t*))
- Designate the node s as the **current** node.

Step 2. (Distance Value Update and Current Node Designation Update)

Let *i* be the index of the current node.

- Find the set J of nodes with temporary labels that can be reached from the current node *i* by a link (*i*, *j*). Update the distance values of these nodes. For each $j \in J$, the distance value *dj* of node *j* is updated as follows.

$$new\ \ d_j = \min \{ d_j, d_i + c_{ij}\}$$

where c_{ij} is the cost of link (*i*, *j*), as given in the network problem.

- Determine a node *j* that has the smallest distance value d_j among all nodes j ∈ J, find *j** such that

$$\min_{j \in J} d_j = d_{j*}$$

- Change the label of node j^* to permanent and designate this node as **the current node.**

Step 3. (Termination Criterion)

- If all nodes that can be reached from node s have been permanently labeled, then stop - we are done.
- If we cannot reach any temporary labeled node from the current node, then all the temporary labels become permanent - we are done.
- Otherwise, go to Step 2.

For example, below is a network with the arcs labelled with their lengths. The example will step though Dijkstra's Algorithm to find the shortest route from the source *s* to the target *t*.

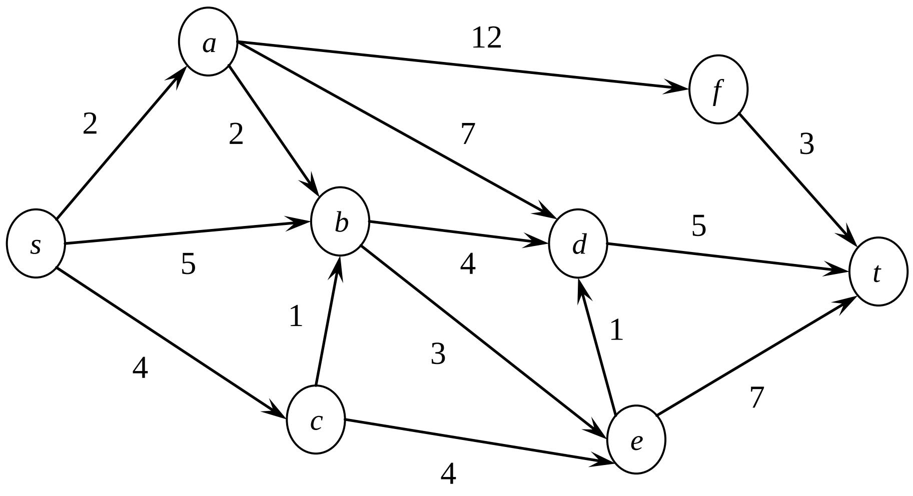


Figure 3.2: The shortest path problem (a)

Step 1.

- Node *s* is designated as the current node.
- The state of node *s* is (0,*p*).
- Every other node has state (∞,*t*).

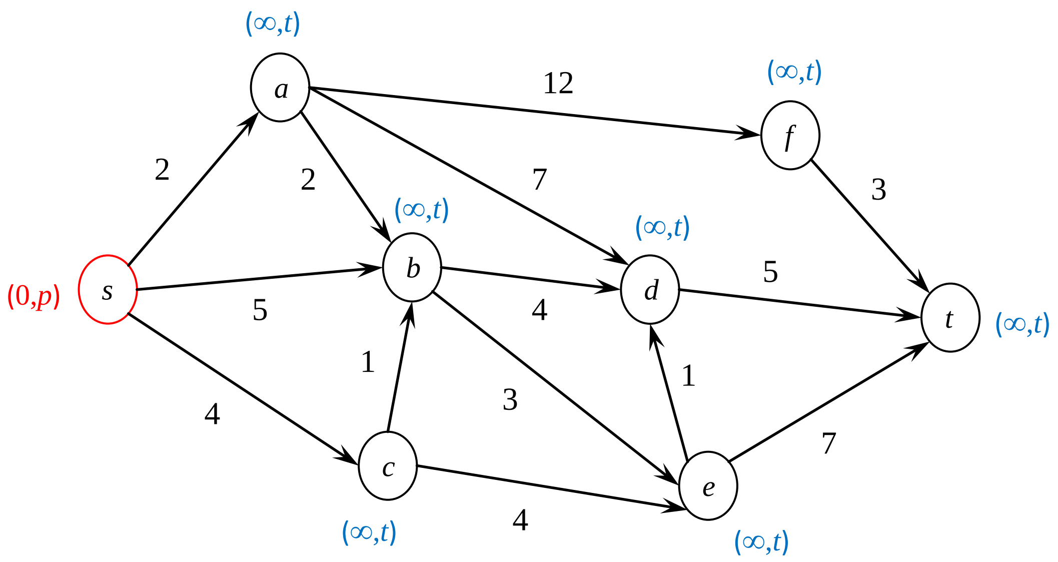


Figure 3.3: The shortest path problem (b)

Step 2.

- Nodes *a*, *b*, and *c* can be reached from the current node s.
- Update distance values for these nodes

$$d_a = \min\{\infty, 0 + 2\} = 2$$
$$d_b = \min\{\infty, 0 + 5\} = 5$$
$$d_c = \min\{\infty, 0 + 4\} = 4$$

- Now, among the nodes *a*, *b*, and *c* ; node *a* has the smallest distance value.
- The status label of node *a* changes the permanent, so its state is (2, *p*), while the status of *b* and *c* remain temporary.
- Node *a* becomes the current node.

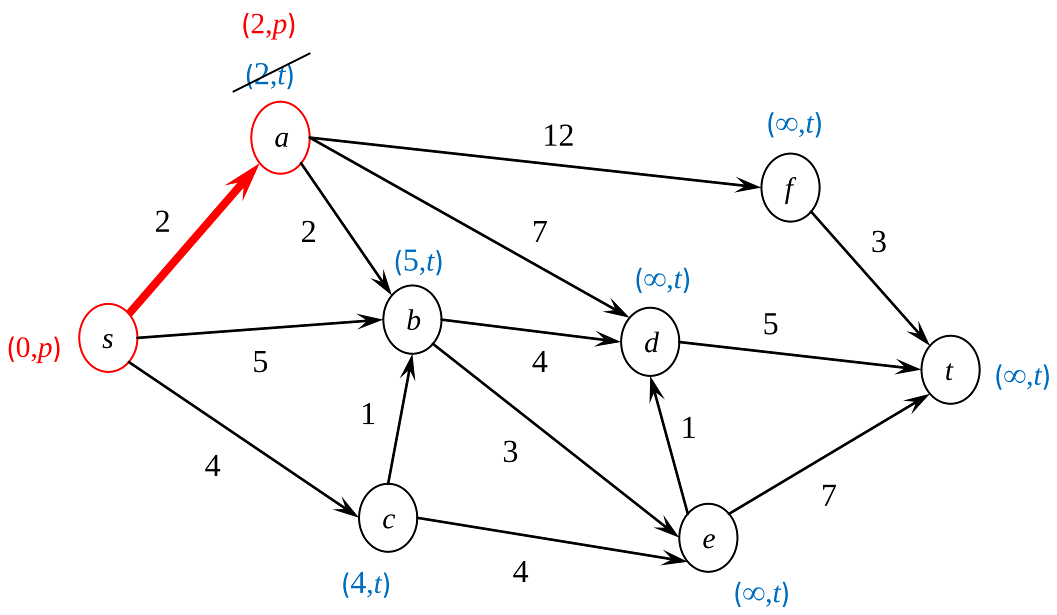


Figure 3.4: The shortest path problem (c)

<u>Step 3.</u>

- Graph at the end of step 2.
- We are not done, not all nodes have been reached from s, so we perform another iteration (back to step 2).

<u>Another Implementation of Step 2.</u>

- Nodes *b*, *d*, and *f* can be reached from the current node *a*.
- Update distance values for these nodes

$$d_b = \min\{5, 2 + 2\} = 4$$

$$d_d = \min\{\infty, 2 + 7\} = 9$$

$$d_f = \min\{\infty, 2 + 12\} = 14$$

- Now, among the nodes *b*, *d*, and *f* node *b* has the smallest distance value.
- The status label of node *b* changes the permanent, while the status of *d* and *f* remain temporary.
- Node *b* becomes the current node. We are not done (step 3 fails), so we perform another step 2.

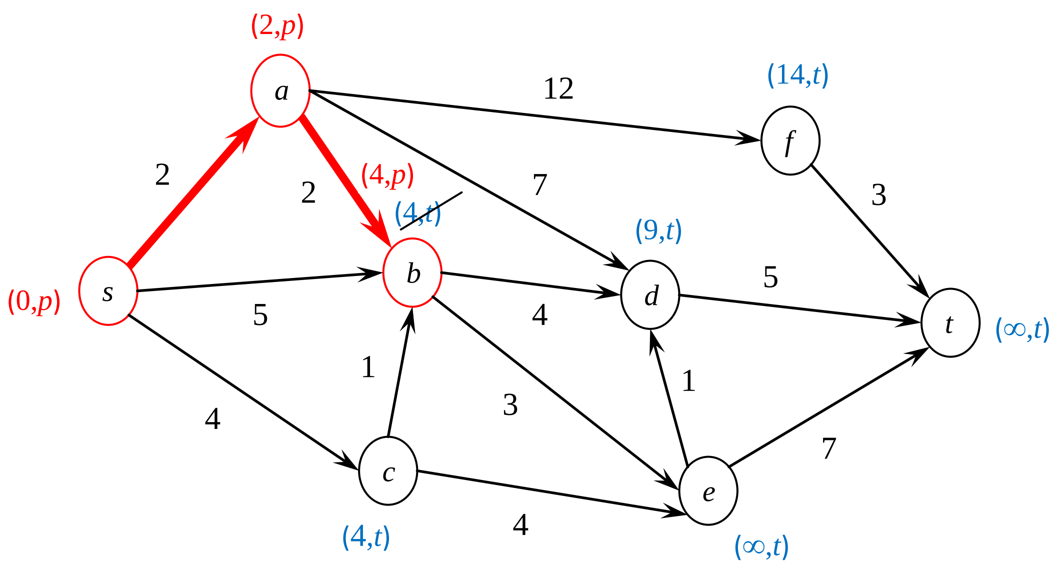


Figure 3.5:The shortest path problem (d)

<u>Another Step 2.</u>

- Node d and e can be reached from the current node b.
- Update distance values for them

$$d_d = \min\{9, 4 + 4\} = 8$$
$$d_e = \min\{\infty, 4 + 3\} = 7$$

- Now, between the nodes d and e node e has the smallest distance value. The status label of node e changes to permanent, while the status of d remains temporary.
- Node e becomes the current node. We are not done (step 3 fails), so we perform another step 2.

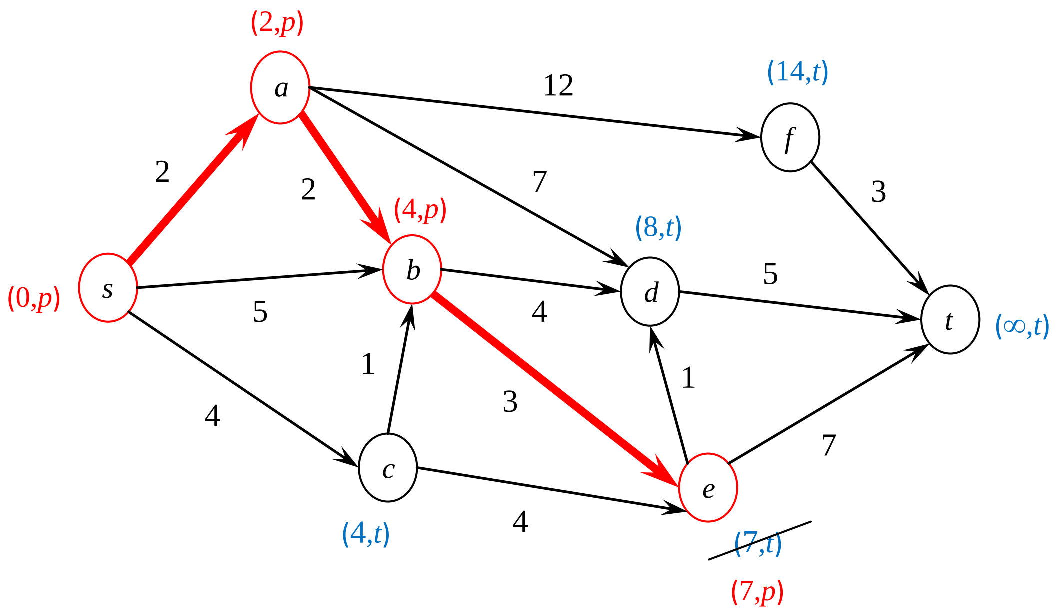


Figure 3.6 : The shortest path problem (e)

<u>Another Step 2.</u>

- Nodes *d* and *t* can be reached from the current node e.
- Update distance values for them

$$d_d = \min\{8, 7 + 1\} = 8$$
$$d_e = \min\{\infty, 7 + 7\} = 14$$

- Now, between the nodes *d* and *t*, node d has the smallest distance value. The status label of node *d* changes to permanent, while the status of *t* remains temporary.
- Node d becomes the current node. We are not done (step 3 fails), so we perform another step 2).

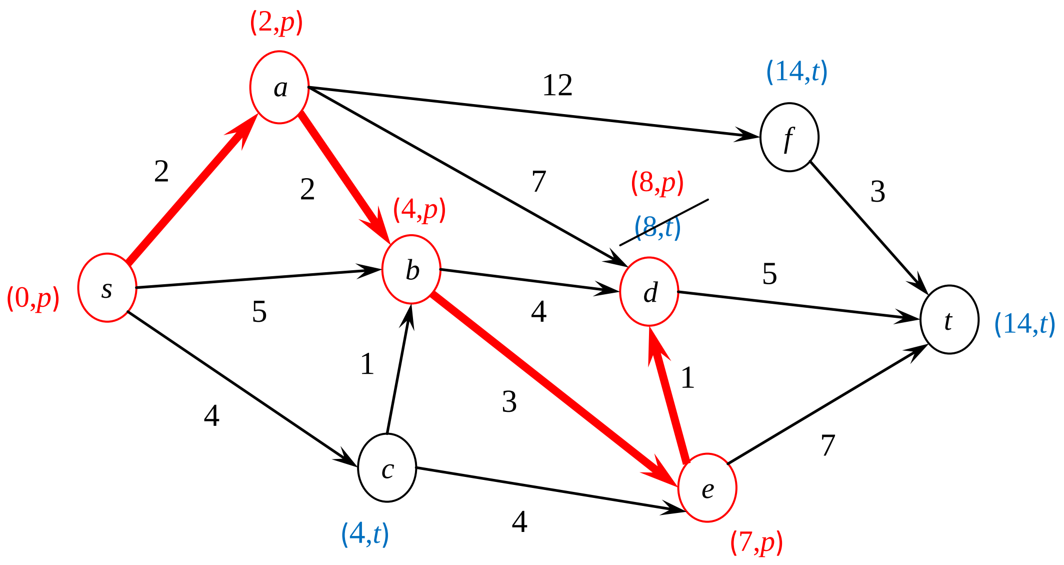


Figure 3.7: The shortest path problem (f)

__Another Step 2.__

- Node t can be reached from the current node d.
- Update distance value for node t

$$d_t = \min\{14, 8 + 5\} = 13$$

its status changes to permanent.

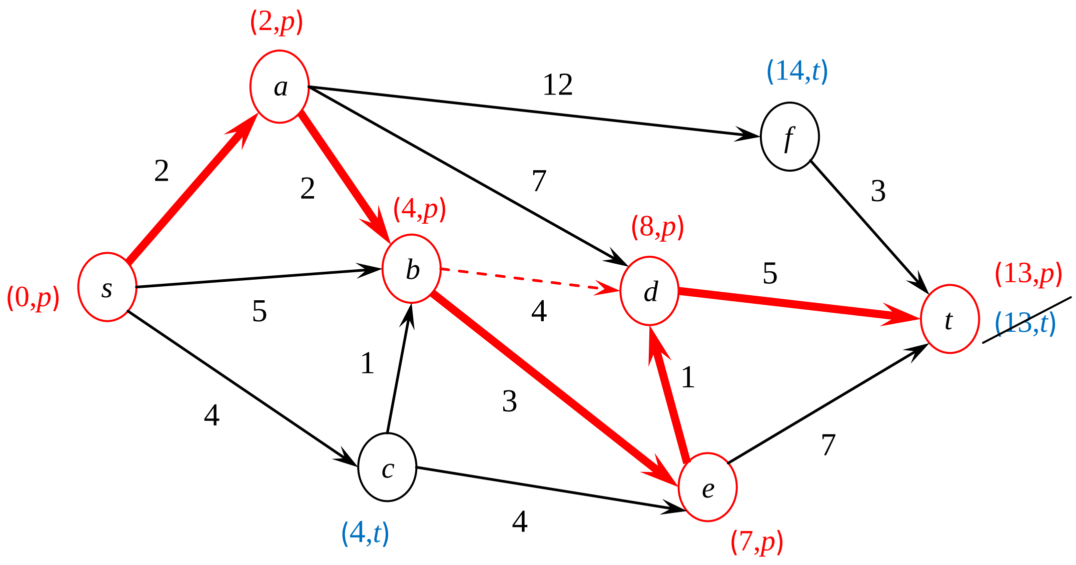


Figure 3.8: The shortest path problem (g)

Now, node *t* is the destination node, therefore we are done. The shortest route (*s-a-b-e-d-t*) from *s* to *t* has a distance of 13. The another shortest route is (*s-a-b-d-t*).

3.2.3 Maximum Flow Problem

Maximum flow originally developed as a means for studying rail transportation networks:

"Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other" (Harris, 1955).

The Soviet rail system was studied in a classified report by Harris and Ross from 1955 entitled "Fundamentals of a Method for Evaluating Rail Net Capacities". Rail network not modeled exactly due to its size and (probably) due to inexact information. Modeled with nodes as small regions that are connected to neighboring regions. Connections between neighboring regions assigned a capacity which is the tonnage (in 1000 tons) that can be transported between the nodes on a daily basis.

The maximum flow problem has a long history. Although other documents (A.W. Boldyreff,1955) define similar problems in the same time period, G. B. Dantzig is credited with the creation of the general maximum flow problem in 1951 (G.B. Dantzig, 1951). Ford and Fulkerson (L.R Ford, Jr. and D.R. Fulkerson, 1956 & 1962) created the first known algorithm in 1955. Since 1955, new algorithms were created using more elegant methods to find the maximum flow. With the elegant methods came more complex sub-steps within these algorithms. Although more complex sub-steps, these new design techniques led to a decrease in running time for maximum flow. Different graph characteristics also play a significant role in the running time of each algorithm.

In the maximum flow problem, we are given a directed or undirected graph, most commonly directed in real world applications, where one vertex is considered a source and another is the destination or commonly referred to as the target. Some object then flows along the edges of the graph from the source to the target. Each edge along the path is given a maximum capacity that can be transported along that route. The maximum capacity can vary from edge to edge in which case the remainder must either flow along another edge towards the target or remain at the current vertex for the edge to clear or be reduced. The goal of the maximum flow problem is to determine the maximum amount of throughput in the graph from the source to target. In real world applications determining the maximum throughput allows the source to know exactly how much of something to produce and send along the path without creating waste.

A network is a directed graph $G = (V,E)$ with a source vertex $s \in V$ and a target vertex $t \in V$. Each edge $e = (v,w)$ from v to w has a defined capacity, denoted by $c(e)$ or $c(v,w)$. (Each edge e in G has an associated non-negative capacity $c(e)$, where for all non-edges it is implicitly assumed that the capacity is 0). For example, consider the graph in Figure 3.9 below.

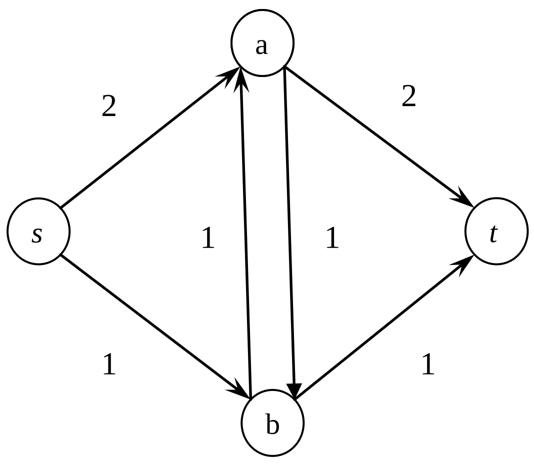


Figure 3.9: An example of a network with $n = 4$ vertices and $m = 6$ edges. The capacities of the edges are shown.

In a network flow problem, we assign a flow to each edge. There are two ways of defining a flow: raw (or gross) flow and net flow.

Raw flow is a function *r* (*v*,*w*) that satisfies the following properties:

- **Capacity constraint**: The flow along any edge must be positive and less than the capacity of that edge. $0 \le r(v,w) \le c(v,w)$.

- **Flow conservation**: For any vertex $v \notin \{s, t\}$, flow in equals flow out:

$$\underbrace{\sum_{w\in V} r(w,v)}_{\text{incoming flow}} - \underbrace{\sum_{w\in V} r(v,w)}_{\text{outgoing flow}} = 0 \quad \text{, for all } v \in V \setminus \{s,t\}.$$

$$\sum_{w\in V} r(w,v) = \sum_{w\in V} r(v,w) .$$

With a raw flow, we can have flows going both from *v* to *w* and flow going from *w* to *v*. In a net flow formulation however, we only keep track of the difference between these two flows.

Net flow is a function that satisfies the following conditions:

- **Skew symmetry**: $f(v,w) = -f(w,v)$.
- **Conservation**: $\sum_{w\in V} f(v,w) = 0$, for all $v \in V \setminus \{s, t\}$.
- **Capacity constraint**: $f(v,w) \le c(v,w)$ for all $v,w \in V$.

A raw flow *r* (*v*,*w*) can be converted into a net flow via the Formula;

$$f(v,w) = r(v,w) - r(w,v).$$

For example, if we have 7 units of flow from *v* to *w* and 4 units of flow from *w* to *v*, then the net flow from *v* to *w* is $f(v,w) = 3$. Skew symmetry follows directly from this formula relating raw flows and net flows. Although skew symmetry relates *f* (*v*,*w*) and *f* (*w*,*v*), it is important to note that the capacity in one direction *c*(*v*,*w*) is independent of the capacity in the reverse direction, *c*(*w*, *v*).

The value of a flow is the sum of the flow on all edges leaving the source *s*. We later show that this is equivalent to the sum of all the flow going into the sink *t*. The value of a flow represents how much we can transport from the source to the sink. The value of a flow f is defined as

$$|f| = \sum_{v\in V} f(s,v).$$

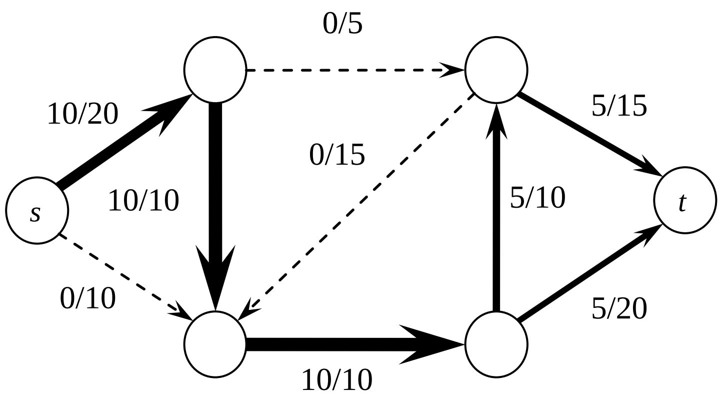


Figure 3.10: An example network showing an (s, t)-flow f of value $|f|$ = 10. In each edge label, the numerator is the flow on the edge and the denominator is the capacity. Dashed edges are avoided by f.

Cuts

In a network flow problem, it is useful to work with a cut of the graph, particularly an s-t cut. An ***s-t* cut** of network G is a partition of the vertices V into 2 groups: S and $\bar{S} = V \setminus S$ such that $s \in S$ and $t \in \bar{S}$.

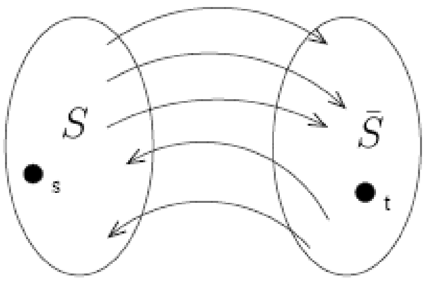


Figure 3.11 : An illustration of an s-t cut. There might be both edges from S to $\bar{S}$ and from $\bar{S}$ to S.

We will usually represent a cut as the pair (S, $\bar{S}$), or just S. We generalize the concept of the net flow and the capacity of an edge to define the net flow and capacity of a cut.

The **net flow** along cut (S, $\bar{S}$)) is defined as $f(S) = \sum_{v \in S} \sum_{w \in \bar{S}} f(v, w)$.

The **value** (or **capacity**) of a cut is defined as $c(S) = \sum_{v \in S} \sum_{w \in \bar{S}} c(v, w)$.

In summary, the value (or capacity) of a cut is the sum of all values (capacities) of edges that go from S to $\overline{S}$.

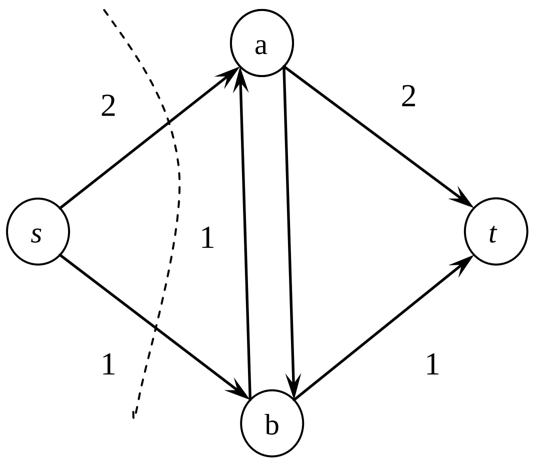


Figure 3.12 : An example of a cut in a network. The *s-t* cut is represented by a dashed line. The value (capacity) of the cut is equal to 3. This is one of the minimum *s-t* cuts.

Lemma 3.1 (Brossard, E.,2010) Given a flow f, for any cut S, $f(S) = |f|$. In other words, all *s-t* cuts carry the same flow: the value of the flow *f*.

Proof: We can prove the lemma by induction on the size of the sets S. For $S = s$, the claim is true. Now, suppose we move one vertex v from $\bar{S}$ to S. The value $f(S)$ changes in the following way:

- $f(S)$ increases by $f(v, \bar{S})$.
- $f(S)$ decreases by $f(S, v) = -f(v, S)$.

In conclusion, the total change in the value of f (S) after moving the vertex v from S to $\bar{S}$ is equal to $f(v, \bar{S}) + f(v, S) = f(v, V) = 0$ (by conservation of flow).

$$\begin{aligned} f(S) &= f(S,V) - f(S,S) \\ &= f(S,V) \\ &= f(s,V) + f(S-s,V) \\ &= f(s,V) \\ &= |f|. \end{aligned}$$

■

Lemma 3.2 (E.Demaine, D., Karger, 2003) If f is a flow, then $|f| \leq c(S)$ for any cut S.

Proof: For all edges e, $f(e) \leq c(e)$, so $f(S) \leq c(S)$ (the flow across any cut S is not more than the capacity of the cut). By Lemma 2, $|f| = f(S)$, so $|f| \leq c(S)$ for any cut S. ■

Let f be a feasible flow on a network G. The corresponding **residual network**, denoted G_f, is a network that has the same vertices as the network G, but has edges with capacities $c_f(v,w) = c(v,w) - f(v,w)$. Only edges with non-zero capacity, $c_f(v,w) > 0$, are included in G_f.
Note that the feasibility conditions imply that $c_f(v,w) \geq 0$ and

$$c_f(v,w) \leq c(v,w) + c(w, v) .$$

This means all capacities in the residual network will be non-negative.

Lemma 3.3 (E.Demaine, D. Karger, 2003) Let $G = (V,E)$ be a flow network with source s, target t, and flow function f. Let G_f be the residual network of G induced by f, and let f' be a flow function of G_f. Then the flow sum $f + f'$ is a flow in G with value $|f + f'| = |f|+|f'|$.

Proof: Need to show that the three flow properties are fulfilled.

Capacity constraint: For all $u, v \in V : (f' + f)(u, v) \leq c(u, v)$

Skew symmetry: For all $u, v \in V : (f + f')(u, v) = -(f + f')(v, u)$

Flow conservation: For all $u \in V \setminus \{s, t\} : \sum_{v \in V} (f + f')(u, v) = 0$

Value of flow: $|f + f'| = \sum_{v \in V} (f + f')(s, v)$

$$= \sum_{v \in V} (f(s, v) + f'(s, v))$$

$$= \sum_{v \in V} f(s, v) + \sum_{v \in V} f'(s, v)$$

$$= |f| + |f'| .$$ ■

An **augmenting path** is a directed path from the node s to node t in the residual network G_f.

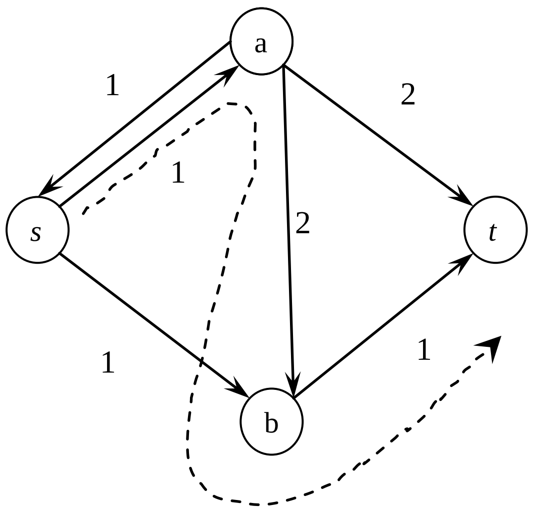


Figure 3.13 : An example of a residual network. This residual network corresponds to the network depicted in Figure 3.9 and the flow in Figure 3.10. The dashed line corresponds to a possible augmenting path.

The edges in a residual network either indicate flow that is still under an original edge's capacity, or flow that has already been used. The existence of an augmenting path indicates that more flow can be achieved without violating capacity limitations by increasing traffic pushed along edges with capacity remaining and/or decreasing traffic along a currently used edge. The idea of the Ford-Fulkerson scheme is to keep finding augmenting paths, using each to its capacity, recalculating the residual network, and repeating until no more augmenting paths exist.

Note that if we have an augmenting path in G_f , then this means we can push more flow along such a path in the original network G. To be more precise, if we have an augmenting path (s, v_1, v_2, . . . v_k , t), the maximum flow we can push along that path is min{c_f (s, v_1), c_f (v_1, v_2), c_f (v_2, v_3), . . . c_f (v_{k-1}, v_k), c_f (v_k, t)}. Therefore, for a given network G and flow f, if there exists an augmenting path in G_f , then the flow f is not a maximum flow.

More generally, if f' is a feasible flow in G_f, then $f + f'$ is a feasible flow in G. The flow $f + f'$ stil satisfies conservation because flow conservation is linear. The flow $f + f'$ is feasible because we can rearrange the inequality $f'(e) \leq c_f(e) = c(e) - f(e)$ to get $f'(e) + f(e) \leq c(e)$. Conversely, if f' is a feasible flow in G, then the flow $f - f'$ is a feasible in G_f .

Straightforward to formulate the maximum flow problem as a linear programming problem:

$$\text{maximize} \quad \sum_{v \in V} f(s,v)$$

$$\begin{aligned} \text{subject to:} \quad & f(u, v) \le c(u, v), \quad \forall\, u, v \in V \\ & f(u, v) = -f(v, u), \forall\, u, v \in V \\ & \sum_{v \in V} f(s,v) = 0 \quad , \quad \forall\, u \in V \setminus \{s, t\} \end{aligned}$$

Note that this formulation is not in standard form: There exist equality constraints in the formulation, and there are no nonnegativity constraints.

Theorem 3.1: ***Max-flow min-cut theorem*** (E.Demaine, D. Karger, 2003) In a flow network G, the following conditions are equivalent:

1. A flow f is a maximum flow.
2. The residual network G_f has no augmenting paths.
3. $|f| = c(S)$ for some cut S.

These conditions imply that the value of the maximum flow is equal to the value of the minimum s-t cut: $\max_f |f| = \min_S c(S)$, where f is a flow and S is a s-t cut .

Proof: We show that each condition implies the other two.

1 ⇒ 2: If there is an augmenting path in G_f , then we previously argued that we can push additional flow along that path, so f was not a maximum flow. 1 ⇒ 2 is the contrapositive of this statement.

2 ⇒ 3: If the residual network G_f has no augmenting paths, s and t must be disconnected. Let S = {vertices reachable from s in G_f }. Since t is not reachable, the set S describes a s-t cut.

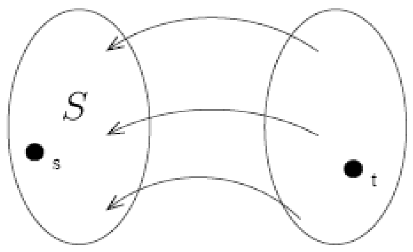


Figure 3.14: Network G_f is disconnected. The set S contains all the nodes that are reachable from s.

By construction, all edges (v,w) crossing the cut have residual capacity 0. This means in the original network G, these edges have $f(v,w) = c(v,w)$. Therefore, $|f| = f(S) = c(S)$.

3 ⇒ 1: If for some cut S, $|f| = c(S)$, we know f must be a maximum flow. Otherwise, we would have a flow g with $|g| > c(S)$, contradicting Lemma 3.3.

From (1) and (3), we know that the maximum flow cannot be less than the value of the minimum cut, because for some S, $|f| = c(S)$ and $c(S)$ is at least as big as the minimum cut value. Lemma 3.3 tells us that the maximum flow can not be greater than the minimum cut value. Therefore, the maximum flow value and the minimum cut value are the same. ■

Ford-Fulkerson Algorithm

The Ford-Fulkerson algorithm solves the problem of finding a maximum flow for a given network. The description of the algorithm is as follows:

1. Start with $f(v,w) = 0$.
2. Find an augmenting path from s to t in G_f (using, for example, a depth first search or similar algorithms).
3. Use the augmenting path found in the previous step to increase the flow.
4. Repeat until there are no more augmenting paths in G_f.

If the capacities are all integers, then the running time is $O(m|f|)$. This is true because finding an augmenting path and updating the flow takes $O(m)$ time, and every augmenting path we find must increase the flow by an integer that is at least 1. In general, if we have integral capacities, then our solution satisfies an integrality property: there exists an integral maximal flow. This happens because every augmenting path increases flows by an integer amount.

For example, There are a bunch of junctions (nodes in the graph) and a bunch of pipes (edges in the graph) connecting the junction. The pipe will only allow water to flow one way (the graph is directed). Each pipe has also has a capacity (the weight of the edge), representing the maximum amount of water that can flow through the pipe. Finally, we pour an infinite amount of water into the source vertex. The problem is to find the maximum flow of the graph - the maximum amount of water that will flow to the target. Below is an example of a network flow graph:

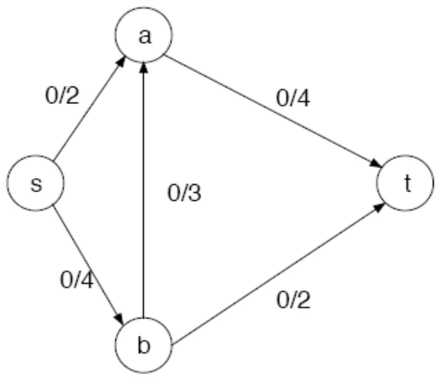


Figure 3.15 : A simple network flow graph. There are currently no water flowing.
It is fairly easy to see that the maximum flow in the above figure is 6.

We can flow 2 units of water from $s \to a \to t$, 2 units of water from $s \to b \to a \to t$, and 2 units of water from $s \to b \to t$. This gives a flow of 6 and since all incoming edge to the sink are saturated, this is indeed the maximum flow. Note that in the example, not all pipe are saturated with water.

For Ford Fulkerson Method, it was easy to see the solution in the above example, but how do we find the solution in general? One idea is to keep finding path from *s* to *t* along pipes which still has some capacities remaining and push as much flow from s to t as possible. We will then terminate once we can’t find any more path. This idea seem to work since it is exactly how we found the maximum flow in the example. However, there is one problem - we cannot guarantee which path we’ll find first. In fact, if we picked the wrong path, the whole algorithm will go wrong. For example, what happens if the first path we found was $s \to b \to a \to t$. If we push as much flow as possible, then we end up with the following:

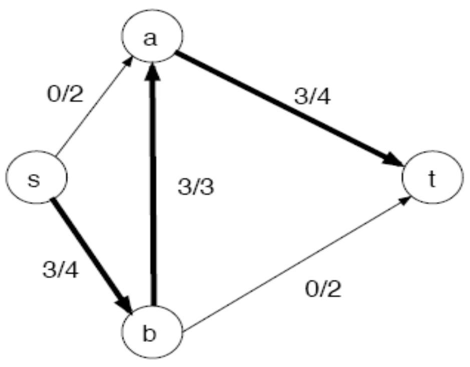
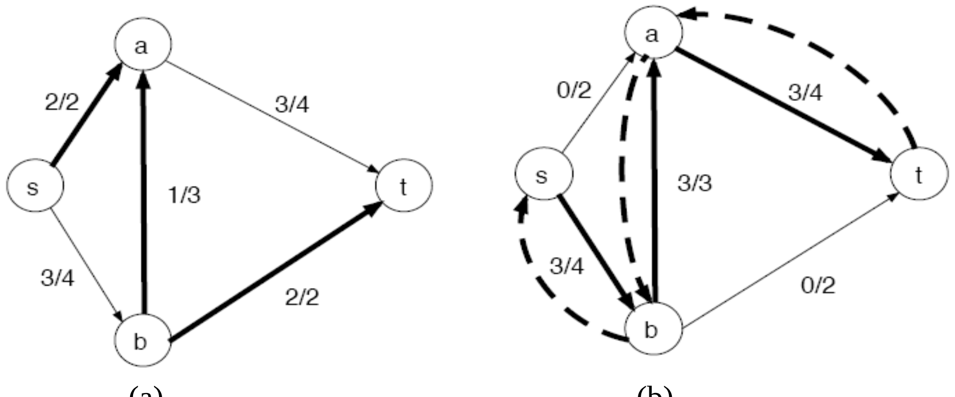


Figure 3.16 : We have pushed 3 flow along the path $s \to b \to a \to t$.

Now we have run into a problem: our only options left are to push 1 unit of water along the path s → a → t and 1 unit of water along s → b → t. After that, we won't be able to find any more path from s to t. Yet, we have only found 5 flow, which is not the maximum flow. Thus, the idea is not optimal since it depends on how we picked our path. While we can try to find a "good" path-picking algorithm, it would be nice if the algorithm is not dependent on the paths we chose.

A crucial observation is that there is actually another path from s to t other than the two that we mentioned above! Suppose we redirect 2 units of water from b → a → t to b → t, this will decrease the amount of water running through the pipe (a, t) to 0. Now we have a path from s → a → t in which we can flow 2 units of water. The graph now looks as follows:



(a) (b)

Figure 3.17 : (a) The bolded edge indicate which pipe has their flow adjusted.(b) The dashed edges are the edges we added in the the residual graph. The capacity of the dashed edge is the same as the amount of water carried by the solid edge in the opposite direction.

It is now easy to see that we can push 1 unit of water along $s \rightarrow b \rightarrow a \rightarrow t$ to obtain the maximum flow of 6. If we carefully study the above figure, what we have essentially done is to flow 2 unit of water along $s \rightarrow a$, and then push back 2 unit of water from $a \rightarrow b$, and finally redirect the pushed back flow along $b \rightarrow t$ to the target. So the key to completing the algorithm is the idea of pushing back flow if we have x units of water flowing in the pipe (u,v), then we can pretend there is a pipe (v,u) with capacity x when we are trying to find a path from s to t.

This is the concept of residual graph. The residual graph of a network flow is essentially the network graph except that for every edge (u, v) that currently carries x unit of water, there is an edge (v, u) with capacity x in the residual graph. The following figure shows the residual graph after finding our first path in Figure 3.11 (b).

3.2.4 Transportation Problem

One of the most important and successful applications of quantitative analysis to solving business problems has been in the physical distribution of products, commonly referred to as **transportation problems**. Basically, the purpose is to minimize the cost of shipping goods from one location to another so that needs of each arrival area met and every shipping location operates within its capacity. (Reeb, J. and Leavengood, S., 2002). The problem was formalized by the French mathematician Gaspard Monge in 1781. In the 1920s A.N. Tolstoi was one of the first to study the transportation problem mathematically. In 1930, in the collection *Transportation Planning Volume I* for the National Commissariat of Transportation of the Soviet Union, he published a paper "Methods of Finding the Minimal Kilometrage in Cargo-transportation in space" (Schrijver, A., 2003).

Transportation problems deal with the determination of a minimum-cost plan for transporting a commodity from a number of sources to a number of destinations. To be more specific, let there be m sources (or origins) that produce the commodity and n destinations (or targets) that demand the commodity. At the i-th source, $i = 1, 2, \dots, m$, there are s_i units of the commodity available. The demand at the j-th destination, $j = 1, 2, \dots, n$, is denoted by d_j. The cost of transporting one unit of the commodity from the i-th source to the j-th destination is c_{ij} . Let x_{ij}, $1 \leq i \leq m;$ $1 \leq j \leq n,$ be the numbers of the commodity that are being transported from the i-th source to the j-th destination. Problem is to determine those x_{ij} that will minimize the overall transportation cost (Hiller, F.S. and Lieberman, G.J.,1995).

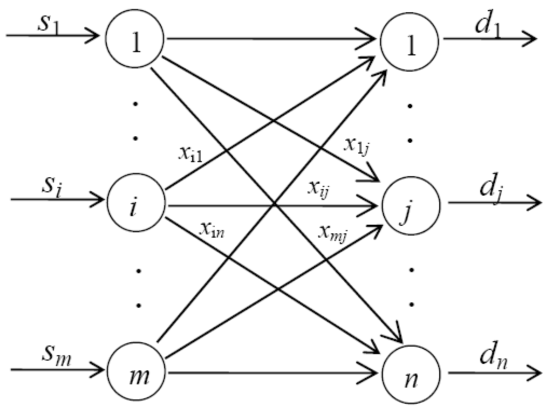


Figure 3.18: Transportation Problem

We note that at the i-th source, we have the i-th source equation

$$\sum_{j=1}^{n} x_{ij} = s_i , \qquad 1 \leq i \leq m,$$

while at the j-th destination, we have the j-th destination equation

$$\sum_{i=1}^{m} x_{ij} = d_j , \qquad 1 \leq j \leq n.$$

Notice that if the total demand equals the total supply, then we have the following **balanced transportation** equation:

$$\sum_{i=1}^{m} s_i = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij} = \sum_{j=1}^{n} d_j .$$

and the model is said to be balanced.

In the case of an unbalanced model, i.e. the total demand is not equal to the total supply, we can always add dummy source or dummy destination to complement the difference. In the following, we only consider balanced transportation models. They can be written as the following linear programming problem:

$$\text{minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{subject to} \quad \begin{cases} \sum_{j=1}^{n} x_{ij} = s_i & 1 \le i \le m, \\ \sum_{i=1}^{m} x_{ij} = d_j & 1 \le j \le n, \\ x_{ij} \ge 0 & 1 \le i \le m \;\; 1 \le j \le n, \end{cases}$$

where $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$.

Notice that there are *mn* variables but only *m*+*n* equations.

The transportation table:

	D_1	D_2	…		D_n	**supply**
S_1	c_{11}	c_{12}	…	…	c_{1n}	s_1
S_2	c_{21}	c_{22}	…	…	c_{2n}	s_2
…	…	…	…	…	…	…
S_m	c_{m1}	c_{m2}	…	…	c_{mn}	s_m
requirement	d_1	d_2	…		d_n	

The North – West Corner Rule

The North-West corner rule is a method for computing a basic feasible solution of a transportation problem where the basic variables are selected from the North West corner. Steps involved in this method are as follows (Reeb, J. and Leavengood, S., 2002):

Step 1: The first assignment is made in the cell occupying the upper left-hand (North West) corner of the transportation table. The maximum feasible amount is allocated there, i.e; $x_{11} = \min(s_1, d_1)$.

Step 2: If $d_1 > s_1$, the capacity of origin (source) S_1 is exhausted but the requirement at D_1 is not satisfied. So move downs to the second row, and make the second allocation:

$x_{21} = \min (s_2 \ , d_1 - x_{11})$ in the cell (2,1).

If $s_1 > d_1$, allocate $x_{12} = \min (s_1 - x_{11}, d_2)$ in the cell (1,2) .

Continue this until all the requirements and supplies are satisfied.

For example, the North – West Corner method solution is as follows.

	D_1	D_2	D_3	**Supply**
S_1	7	5	4	20
S_2	6	2	6	10
Demand	8	8	14	

Figure 3.19: The North – West Corner Rule (a)

	D_1	D_2	D_3	**supply**
S_1	7 8 $\rightarrow$	5 8 $\rightarrow$	4 4 $\downarrow$	~~20~~ ~~12~~ ~~4~~ 0
S_2	6	2	6 10	~~10~~ 0
Demand	~~8~~ 0	~~8~~ 0	~~14~~ ~~10~~ 0	

Figure 3.20: The North – West Corner Rule (b)

Now all requirements have been satisfied and hence an initial basic feasible solution to the transportation problem has been obtained by North –West Corner rule. Since the allocated cells do not form a loop, the feasible solution is non-degenerate. Total transportation cost with this allocation is:

$$z=7\times 8+5\times 8+4\times 4+10\times 6=172\,.$$

CHAPTER 4

LOGISTICS and TRANSPORTATION

The operation of transportation determines the efficiency of moving products. The progress in techniques and management principles improves the moving load, delivery speed, service quality, operation costs, the usage of facilities and energy saving. Transportation takes a crucial part in the manipulation of logistic. Reviewing the current condition, a strong system needs a clear frame of logistics and a proper transport implements and techniques to link the producing procedures. The objective of this chapter is to define logistics and the role of transportation in logistics.

4.1 Logistics

Logistics is the science of planning, organizing and managing activities that provide goods or services (MDC, LogLink / Logistics World, 1997). It is the management of the flow of goods and services between the point of origin and the point of consumption in order to meet the requirements of customers. Logistics is now an important part of the supply chain for many businesses and seems a modern concept. Logistics involves the integration of information, transportation, inventory, warehousing, material handling, and packaging, and often security.

The Council of Supply Chain Management Professionals (CSCMP) has defined logistics as "…that part of Supply Chain Management that plans, implements, and controls the efficient, effective forward and reverse flow and storage of goods, services and related information between the point of origin and the point of consumption in order to meet customers' requirements." (Tseng, Y.,Yue, W.L. and Taylor, M., 2005).

4.2 History of Logistics

Logistics was initially a military activity concerned with getting soldiers and munitions to the battlefront in time for flight, but it is now seen as an integral part of the modern production process. The main background of its development is that the recession of America in the 1950s caused the industrial to place importance on goods circulations. The term, logistics, was initially developed in the context of military activities in the late 18th and early 19th centuries and it launched from the military logistics of World War II. The probable origin of the term is the Greek logistikos, meaning ‘skilled in calculating’ (BTRE, 2001). Military definitions typically incorporate the supply, movement and quartering of troops in a set. And now, a number of researches were taken and made logistics applications from military activities to business activities. Business logistics was not an academic subject until the 1960s. A key element of logistics, the trade-off between transport and inventory costs, was formally recognized in economics at least as early as the mid-1880s (BTRE, 2001).

The further tendency of logistics in the early 21st century is logistics alliance, Third Party Logistics (3PL) and globalised logistics. Third-party logistics (3PL) involves using external organizations to execute logistics activities that have traditionally been performed within an organization itself (Baziotopoulos, 2008). According to this definition, third-party logistics includes any form of outsourcing of logistics activities previously performed in-house. If, for example, a company with its own warehousing facilities decides to employ external transportation, this would be an example of third-party logistics. Logistics is an emerging business area in many countries. Logistics circulation is an essential of business activities and sustaining competitiveness, however, to conduct and manage a large company is cost consuming and not economic. Therefore, alliance of international industries could save working costs and cooperation with 3PL could specialize in logistics area (Proceedings of the Eastern Asia Society for Transportation Studies, 2005).

Military Logistics

In military science, maintaining one's supply lines while disrupting those of the enemy is a crucial some would say the most crucial element of military strategy, since an armed force without resources and transportation is defenseless. The defeat of the British in the American War of Independence and the defeat of the Axis in the African theatre of World War II are attributed to logistical failure. The historical leaders Hannibal Barca, Alexander the Great, and the Duke of Wellington are considered to have been logistical geniuses. Militaries have a significant need for logistics solutions, and so have developed advanced implementations. Integrated

Logistics Support (ILS) is a discipline used in military industries to ensure an easily supportable system with a robust customer service (logistic) concept at the lowest cost and in line with (often high) reliability, availability, maintainability and other requirements as defined for the project. In military logistics, logistics officers manage how and when to move resources to the places they are needed (Chang, Y.H.,1998).

Business Logistics

Logistics as a business concept evolved in the 1950s due to the increasing complexity of supplying businesses with materials and shipping out products in an increasingly globalized supply chain, leading to a call for experts called supply chain logisticians. Business logistics can be defined as "having the right item in the right quantity at the right time at the right place for the right price in the right condition to the right customer", and is the science of process and incorporates all industry sectors. In business, logistics may have either internal focus, or external focus covering the flow and storage of materials from point of origin to point of consumption. The main functions of a qualified logistician include inventory management, purchasing, transportation, warehousing, consultation and the organizing and planning of these activities. Logisticians combine a professional knowledge of each of these functions to coordinate resources in an organization. There are two fundamentally different forms of logistics: one optimizes a steady flow of material through a network of transport links and storage nodes; the other coordinates a sequence of resources to carry out some project.

4.3 Logistics and Industry

Logisticians work in virtually every industry in the business world, including:

Aerospace, Airlines, Auto Industry, Clothing & Apparel, Courier & Messaging, Engineering, Food & Grocery, Forestry, Freight Forwarding, Government, Hotel and Hospitality, International Trade, Military and National Defence, Natural Gas, Public Transportation, Oil, Railways, Shipping, etc.

4.4 Transportation Planning in Logistics

Transport or transportation is the movement of people and goods from one location to another. Transport is performed by various modes, such as air, rail, road, water, cable, pipeline and space. The field can be divided into infrastructure, vehicles, and operations. Transport is important since it enables trade between peoples, which in turn establishes civilizations. Transport infrastructure consists of the fixed installations necessary for transport, and may be roads, railways, airways, waterways, canals and pipelines, and terminals such as airports, railway stations, bus stations, warehouses, trucking terminals, refueling depots (including fueling docks and fuel stations), and seaports. Terminals may be used both for interchange of passengers and cargo and for maintenance. Vehicles traveling on these networks may include automobiles, bicycles, buses, trains, trucks, people, helicopters, and aircraft.(from Wikipedia, "Transport," http://en.wikipedia.org/wiki/Transportation).

Logistics is concerned with the efficient flow of raw materials, of work in process inventory, and of finished goods from supplier to customer. In addition to transportation, logistics entails inventory control, warehousing, materials handling, order processing, and related information activities involved in the flow of products.

The globalization of business has increased the need for global supply chains that arelonger, more complex, and inherently costlier. Businesses will seek logistics service suppliers who can meet their global logistics needs. This development will spur the growth of global third-party logistics (3PL) providers who provide a full portfolio of logistics services, including transportation. It also will encourage the development of modern and efficient transport infrastructures to minimize the cost of transport operations on major trade routes. These infrastructures include right of way, intermodal facilities, and communications links for all modes. Pull processes require fast, frequent, and reliable transportation systems with shipment visibility. This requirement has fueled the growth of time-sensitive transport alternatives such as air freight and priority ground transport. Transport suppliers must be able to provide shipment visibility by adopting mobile communication, e-commerce, vehicle status, and other Technologies (Fair, M.L. and Williams, E.W, 1981).

Maritime Logistics

Maritime industry plays an important role in international freight. It can provide a cheap and high carrying capacity conveyance for consumers. Therefore, it has a vital position in the transportation of particular goods, such as crude oil and grains. Its disadvantage is that it needs longer transport time and its schedule is strongly affected by the weather factors. To save costs and enhance competitiveness, current maritime logistics firms tend to use largescaled ships and cooperative operation techniques. The operation of maritime transport industry can be divided into three main types. Liner shipping, the business is based on the same ships, routes, price, and regular voyages. Tramp shipping, the characters of this kind of shipping are irregular transport price, unsteady transport routes, and schedule. Industry shipping, the main purpose of industry shipping is to ensure the supply of raw materials.

Figure 4.1 : The role of transportation in logistics

Air Freight Logistics

Air freight logistics is necessary for many industries and services to complete their supply chain and functions. It provides the delivery with speed, lower risk of damage, security, flexibility, accessibility and good frequency for regular destinations. Reynolds-Feighan (2001) said air freight logistics is selected ‘when the value per unit weight of shipments is relatively high and the speed of delivery is an important factor'. The characteristics of air freight logistics are that: (1) airplanes and airports are separated. Therefore, the industries only need to prepare planes for operation; (2) it allows to speed delivery at far destinations; (3) air freight transport is not affected by landforms (Reynolds-Feighan, 2001).

Land Logistics

Land logistics is a very important link in logistics activities. It extends the delivery services for air and maritime transport from airports and seaports. The most positive characteristic of land logistics is the high accessibility level in land areas. The main transport modes of land logistics are railway transport, road freight transport and pipeline transport. Railway transport has advantages like high carrying capacity, lower influence by weather conditions, and lower energy consumption while disadvantages as high cost of essential facilities, difficult and expensive maintenance, lack of elasticity of urgent demands, and time consumption in organizing railway carriages. Road freight transport has advantages as cheaper investment funds, high accessibility, mobility and availability. Its disadvantages are low capacity, lower safety, and slow speed. The advantages of pipeline transport are high capacity, less effect by weather conditions, cheaper operation fee, and continuous conveyance; the disadvantages are expensive infrastructures, harder supervision, goods specialization, and regular maintenance needs (Tilanus, B., 1997).

Transportation systems are commonly represented using networks as an analogy for their structure and flows. Transport networks belong to the wider category of spatial networks. Transport networks are better understood by the usage level (e.g. number of passengers, tons, vehicles, capacity) than by their sole topology based on a binary state (i.e. presence or absence of links). Inequalities between locations can often be measured by the quantity of links between nodes and the related revenues generated by traffic flows.

The efficiency of a network can be measured through **graph theory** and **network analysis.** These methods rest on the principle that the efficiency of a network depends partially on the lay-out of nodes and links. Obviously some network structures have a higher degree of accessibility than others, but careful consideration must be given to the basic relationship between the revenue and costs of specific transport networks.

CHAPTER 5

APPLICATIONS OF GRAPH THEORY IN LOGISTICS

Transportation is a critical part of any global logistics effort because of the long distances that can separate a firm from its customers. A transportation system can be inbound and outbound. A transportation system must fit within other logistics activities. Historically, national governments have exercised tight economics control over transport organizations, either through direct company ownership or through laws intended to regulate the way those businesses were run. This governmental involvement in the business of transportation is gradually waning as nations move to privatize state-owned businesses and deregulate privately held firms. For the logistics manager, the competitive nature of goods movement today means greater opportunities for obtaining better service and / or lower costs for transport providers. The five primary modes of transportation are rail, road, pipeline, water and air. Each has different economic and service characteristics that are summarized in the following table (see figure 5.1).

	Rail	**Road**	**Water**	**Air**	**Pipeline**
Price	Low	High	Very low	Very high	Very low
Speed	Slow	Fast	Very slow	Very fast	Slow
Door	Sometimes	Yes	Sometimes	No	Sometimes
Reliability	Medium	Medium	Low	Very high	Very high
Packing needs	High	Medium	High	Low	Nil
Risk of loss and damage	High	Medium	Medium	Low	Very low
Flexibility	Low	High	Low	Very low	Very low
Environmental impact	Low	High	Low	Medium	Low

Figure 5.1 : Economics and service characteristics

The transportation industry facilitates the movement of goods for the purposes of trade, production and consumption. Good transportation systems are often described as satisfying several quality factors, such as cost, time and length. The relationships among the elements are often associated with linear transport routes or networks.

The usual representation is that network junctions are represented as nodes and routes between them are represented by arcs (links) as used in planar graphs.

<u>**Example 5.1:**</u> **(The Shortest Path Problem)** A gold company should carry some gold stocks from Bergama to stores in İzmir. One of its current job is to move gold stocks by using shortest routes.

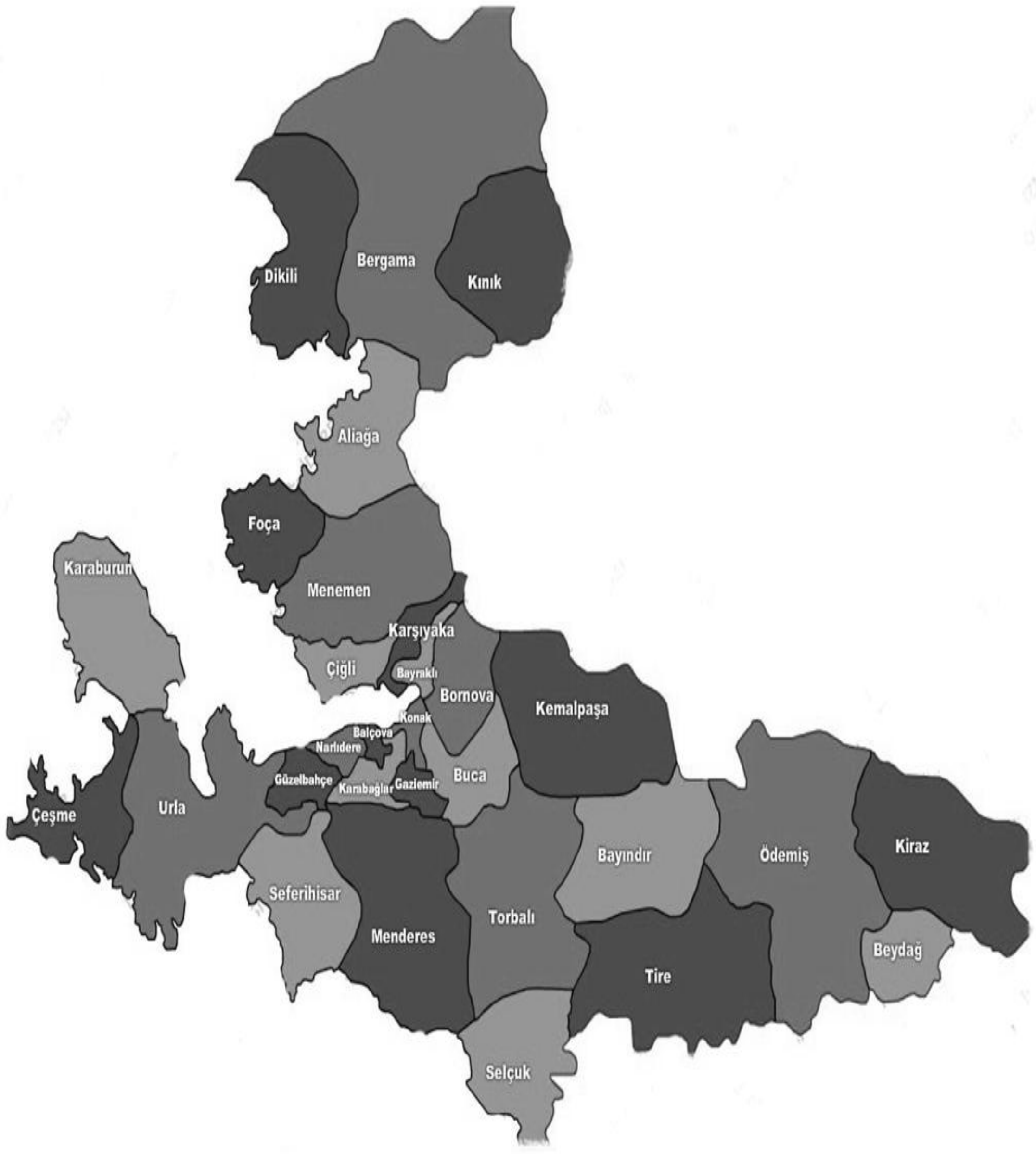


Figure 5.2: Map of İzmir

Node	Node	City	City	Distance(km)
v_1	v_2	Bergama	Dikili	29
v_1	v_3	Bergama	Kınık	19
v_1	v_4	Bergama	Aliağa	47
v_4	v_5	Aliağa	Foça	34
v_4	v_6	Aliağa	Menemen	26
v_5	v_6	Foça	Menemen	36
v_6	v_7	Menemen	Çiğli	17
v_6	v_8	Menemen	Karşıyaka	36
v_7	v_8	Çiğli	Karşıyaka	24
v_8	v_9	Karşıyaka	Bayraklı	9
v_8	v_{10}	Karşıyaka	Bornova	13
v_9	v_{10}	Bayraklı	Bornova	6
v_9	v_{11}	Bayraklı	Konak	9
v_{10}	v_{11}	Bornova	Konak	13
v_{10}	v_{15}	Bornova	Buca	14
v_{10}	v_{16}	Bornova	Kemalpaşa	29
v_{11}	v_{12}	Konak	Balçova	9
v_{11}	v_{14}	Konak	Gaziemir	14
v_{11}	v_{15}	Konak	Buca	3
v_{11}	v_{18}	Konak	Karabağlar	23
v_{12}	v_{13}	Balçova	Narlıdere	14
v_{12}	v_{18}	Balçova	Karabağlar	10
v_{13}	v_{17}	Narlıdere	Güzelbahçe	18
v_{13}	v_{18}	Narlıdere	Karabağlar	20
v_{14}	v_{15}	Gaziemir	Buca	14
v_{14}	v_{18}	Gaziemir	Karabağlar	16
v_{14}	v_{23}	Gaziemir	Menderes	12
v_{15}	v_{16}	Buca	Kemalpaşa	30
v_{15}	v_{23}	Buca	Menderes	26
v_{15}	v_{24}	Buca	Torbalı	52
v_{16}	v_{24}	Kemalpaşa	Torbalı	33
v_{16}	v_{25}	Kemalpaşa	Bayındır	54
v_{17}	v_{18}	Güzelbahçe	Karabağlar	35
v_{17}	v_{19}	Güzelbahçe	Urla	16
v_{17}	v_{22}	Güzelbahçe	Seferihisar	22
v_{17}	v_{23}	Güzelbahçe	Menderes	45
v_{18}	v_{23}	Karabağlar	Menderes	26
v_{19}	v_{20}	Urla	Karaburun	66
v_{19}	v_{21}	Urla	Çeşme	56
v_{19}	v_{22}	Urla	Seferihisar	19
v_{22}	v_{23}	Seferihisar	Menderes	39
v_{23}	v_{24}	Menderes	Torbalı	25
v_{23}	v_{30}	Menderes	Selçuk	60
v_{24}	v_{25}	Torbalı	Bayındır	43
v_{24}	v_{29}	Torbalı	Tire	42
v_{24}	v_{30}	Torbalı	Selçuk	41

Node	Node	City	City	Distance(km)
v_{25}	v_{26}	Bayındır	Ödemiş	38
v_{25}	v_{29}	Bayındır	Tire	23
v_{26}	v_{27}	Ödemiş	Kiraz	29
v_{26}	v_{28}	Ödemiş	Beydağ	30
v_{26}	v_{29}	Ödemiş	Tire	37
v_{27}	v_{28}	Kiraz	Beydağ	17
v_{29}	v_{30}	Tire	Selçuk	41

Figure 5.3: Distances of districts in İzmir

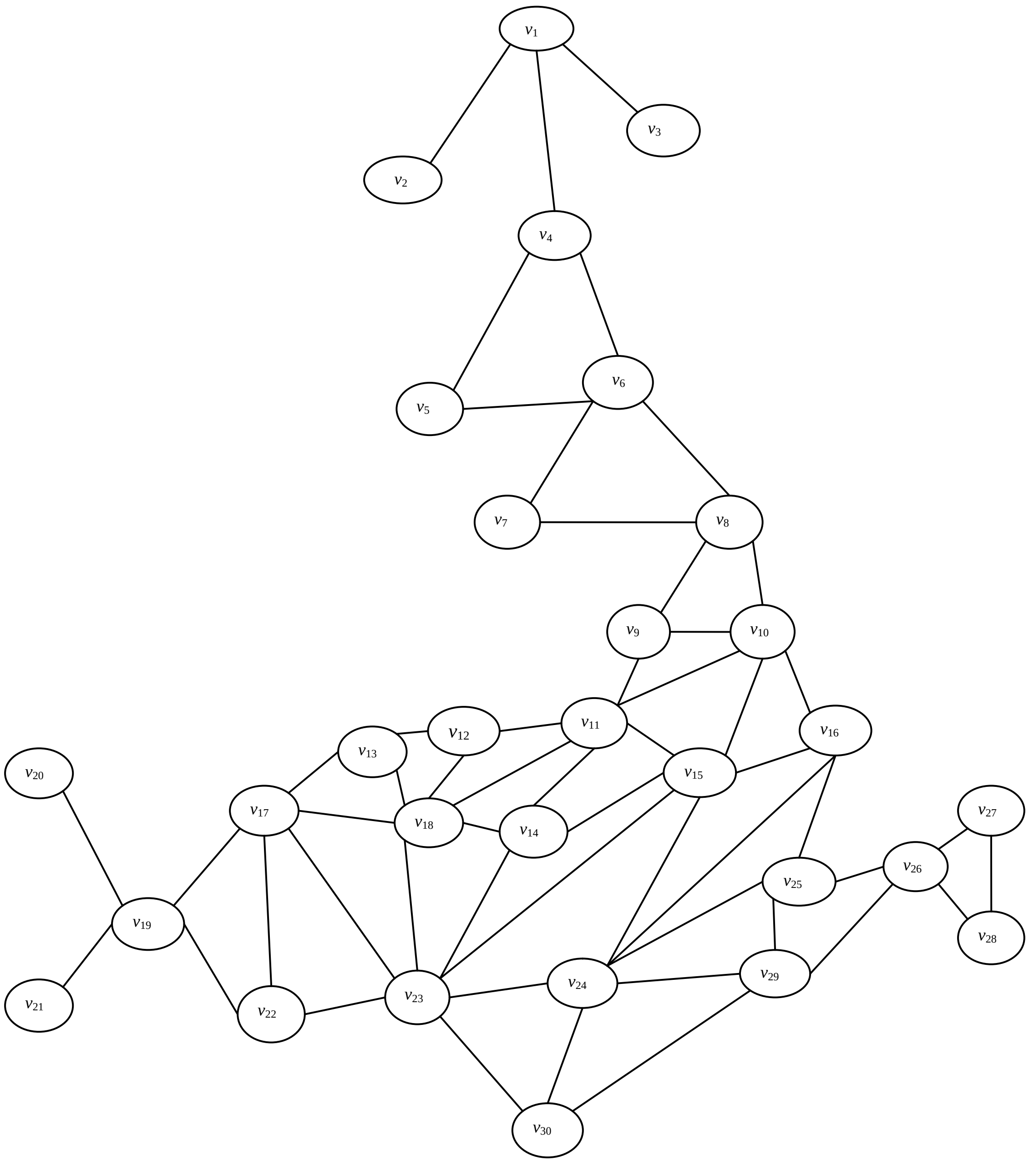


Figure 5.4: A map of İzmir is converted into a graph

Dijkstra's algorithm is used on this map for finding the shortest routes. The solution is:

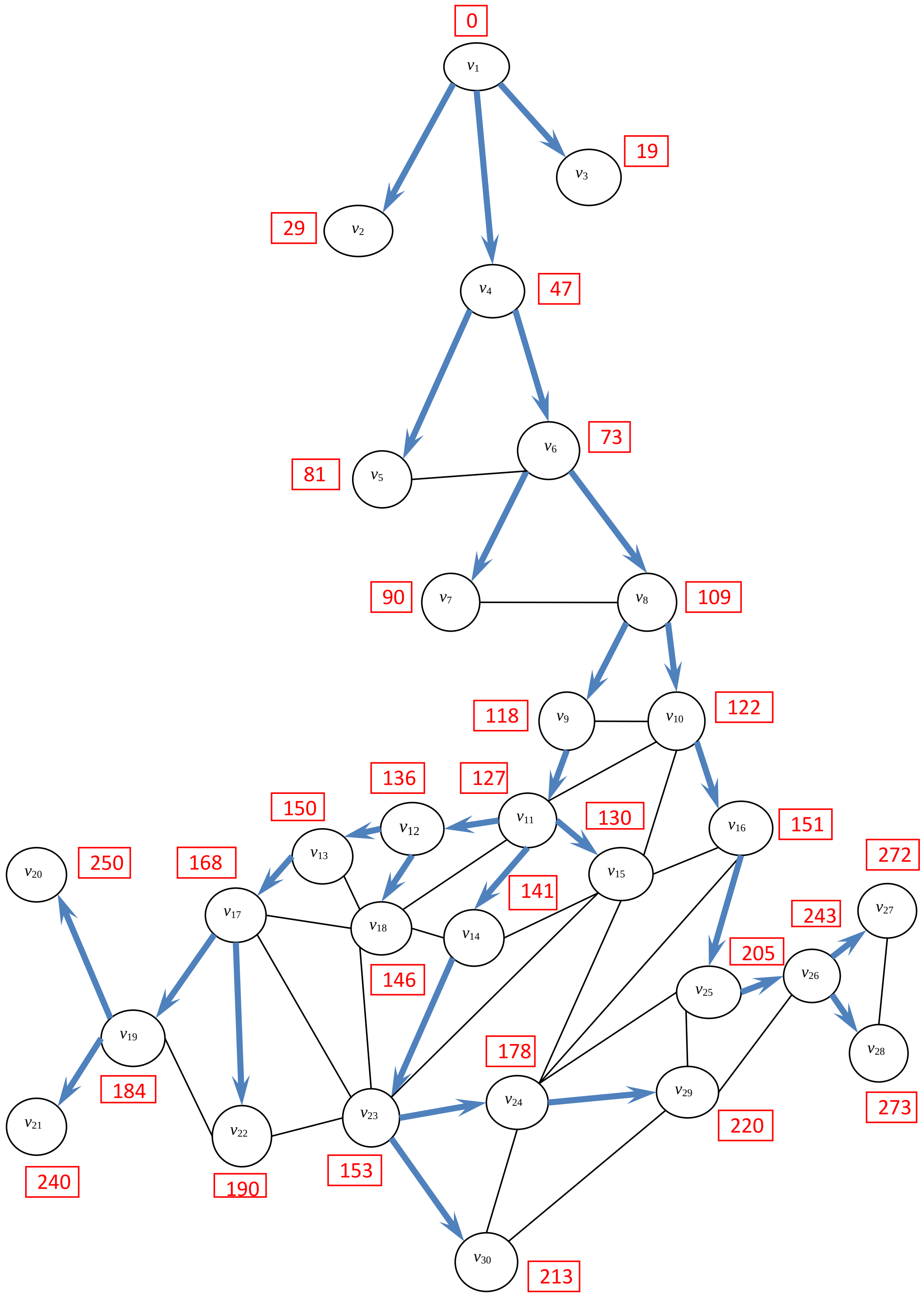


Figure 5.5: Application of the shortest path problem

From Bergama	Path	Distance(km)
to Dikili	Bergama-Dikili	29
to Kınık	Bergama-Kınık	19
to Aliağa	Bergama-Aliağa	47
to Foça	Bergama-Aliağa-Foça	81
to Menemen	Bergama-Aliağa-Menemen	73
to Çiğli	Bergama-Aliağa-Menemen-Çiğli	90
to Karşıyaka	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka	109
to Bayraklı	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı	118
to Bornova	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bornova	122
to Konak	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak	127
to Balçova	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Balçova	136
to Narlıdere	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Balçova-Narlıdere	150
to Gaziemir	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Gaziemir	141
to Buca	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Buca	130
to Kemalpaşa	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bornova-Kemalpaşa	151
to Güzelbahçe	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Balçova-Narlıdere-Güzelbahçe	168
to Karabağlar	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Balçova-Karabağlar	146
to Urla	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Balçova-Narlıdere-Güzelbahçe-Urla	184
to Karaburun	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Balçova-Narlıdere-Güzelbahçe-Urla-Karaburun	250
to Çeşme	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Balçova-Narlıdere-Güzelbahçe-Urla-Çeşme	240
to Seferihisar	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Balçova-Narlıdere-Güzelbahçe-Seferihisar	190
to Menderes	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Gaziemir-Menderes	153
to Torbalı	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Gaziemir-Menderes-Torbalı	178
to Bayındır	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bornova-Kemalpaşa-Bayındır	205
to Ödemiş	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bornova-Kemalpaşa-Bayındır-Ödemiş	243

From Bergama	Path	Distance (km)
to Kiraz	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bornova-Kemalpaşa-Bayındır-Ödemiş-Kiraz	272
to Beydağ	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bornova-Kemalpaşa-Bayındır-Ödemiş-Beydağ	273
to Tire	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Gaziemir-Menderes-Torbalı-Tire	220
to Selçuk	Bergama-Aliağa-Menemen-Çiğli-Karşıyaka-Bayraklı-Konak-Gaziemir-Menderes-Selçuk	213

Figure 5.6: Shortest Routes

Example 5.2:(Maximum Flow Problem) The BMZ Company is a European manufacturer of luxury automobiles. Its exports to the United States are particularly important. BMZ cars are becoming especially popular in California, so it is particularly important to keep the Los Angeles center well supplied with replacement parts for repairing these cars. BMZ needs to execute a plan quickly for shipping as much as possible from the main factory in Stuttgart, Germany to the distribution center in Los Angeles over the next month. The limiting factor on how much can be shipped is the limited capacity of the company's distribution network.How many units should be sent through each shipping lane to maximize the total units flowing from Stuttgart to Los Angeles?

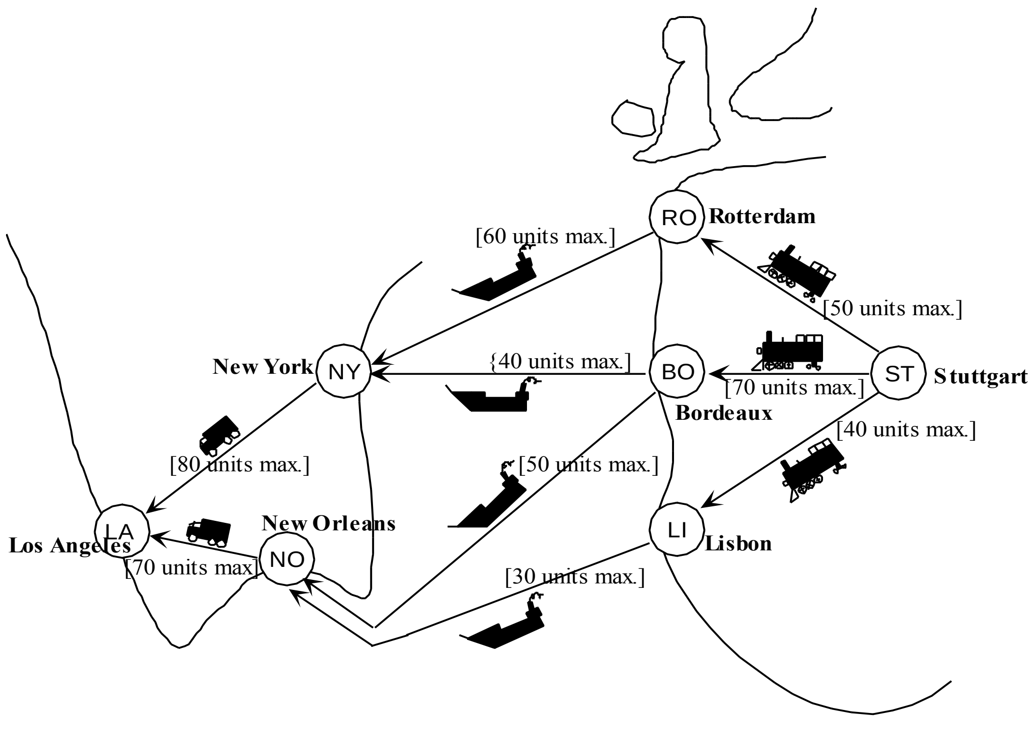


Figure 5.7: Application of Maximum Flow Problem (a)

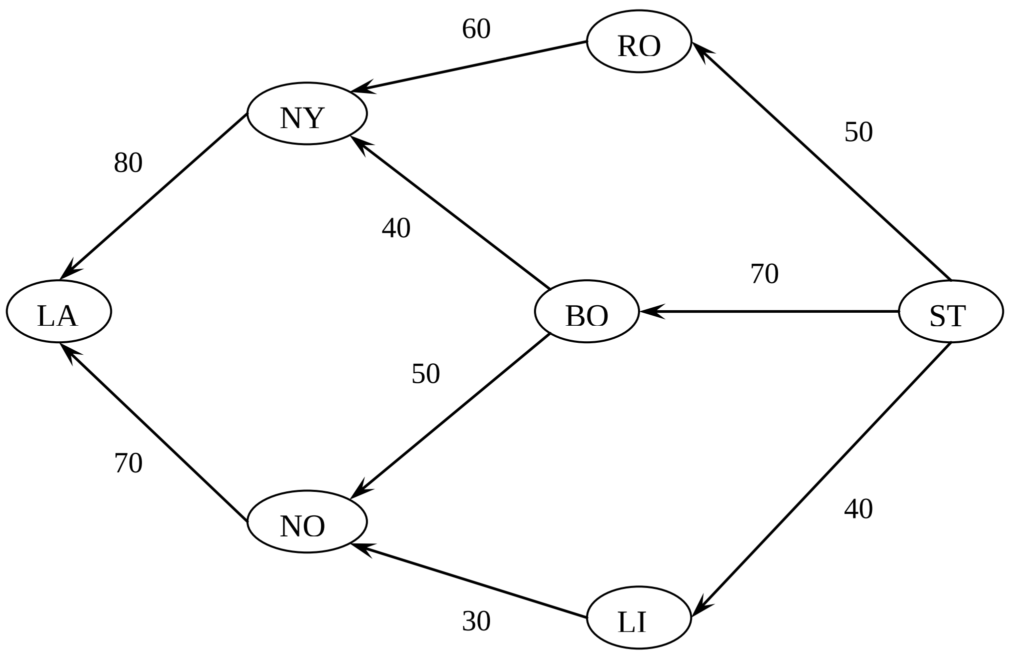


Figure 5.8: Application of Maximum Flow Problem (b)

The problem is solved by using Ford Fulkerson algorithm. First agumenting path is ST-RO-NY-LA, second is ST-BO-NY-LA, third is ST-BO-NO-LA, and final agumenting path is ST-LI-NO-LA. The solution is represented:

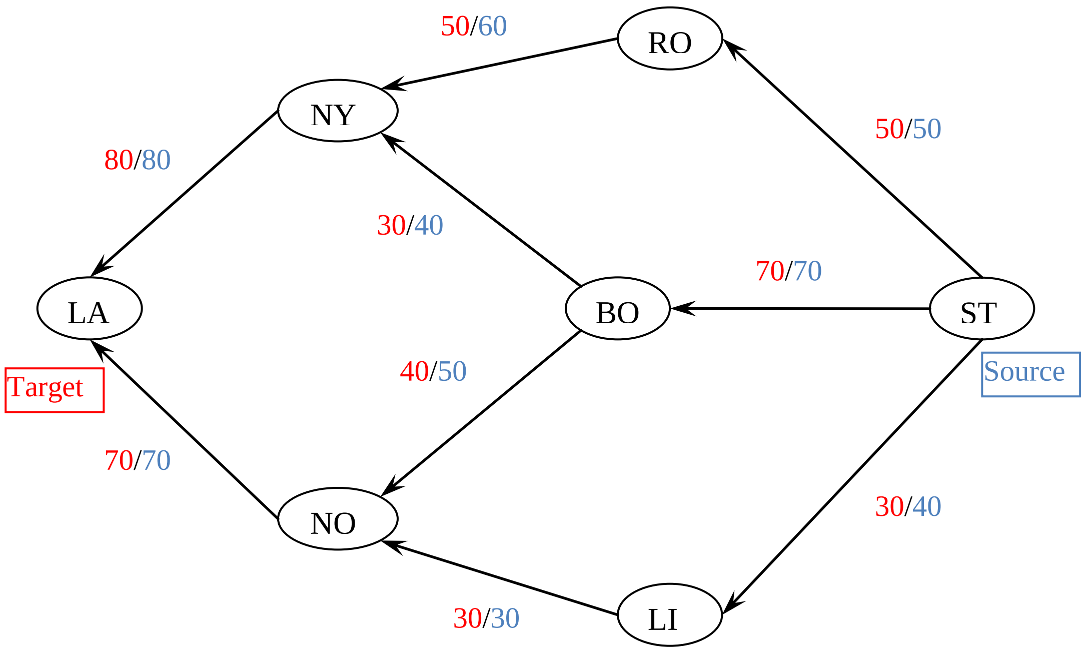


Figure 5.9: Application of Maximum Flow Problem (c)

From	To	Flow	Capacity	Node	Net Flow
Stuttgart	Rotterdam	50	50	Stuttgart	150
Stuttgart	Bordeaux	70	70	Rotterdam	0
Stuttgart	Lisbon	30	40	Bordeaux	0
Rotterdam	New York	50	60	Lisbon	0
Bordeaux	New York	30	40	New York	0
Bordeaux	New Orleans	40	50	New Orleans	0
Lisbon	New Orleans	30	30	Los Angeles	-150
New York	Los Angeles	80	80		
New Orleans	Los Angeles	70	70		
	Maximum Flow	150			

Figure 5.10: Application of Maximum Flow Problem (d)

<u>Example 5.3:</u> (Transportation Problem) A Medical Supply company produces catheters in packs at three production facilities. The company ships the packs from the production facilities to four warehouses. The packs are distributed directly to hospitals from the warehouses. The table is shown the costs per pack to truck to the four warehouses.

FROM PLANT	TO WAREHOUSE Adana	Kayseri	Konya	Trabzon
Ankara	19 TL	7 TL	3 TL	21 TL
İzmir	24	28	18	30
İstanbul	32	21	18	22

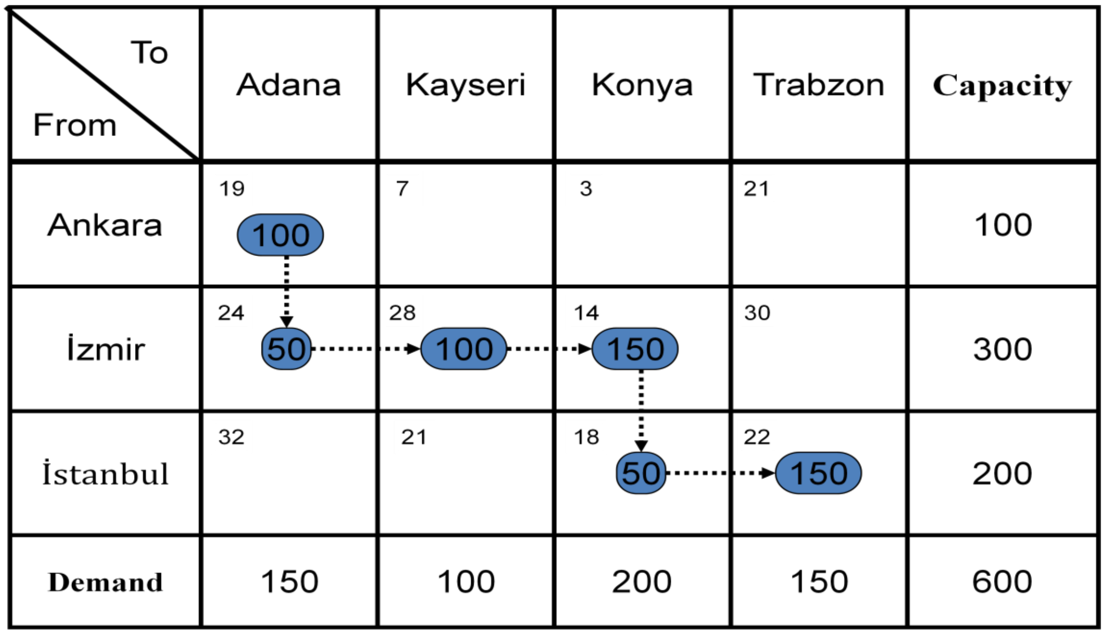
Figure 5.11: Application of Transportation Problem (a)

Capacity		Demand	
Ankara	100	Adana	150
İzmir	300	Kayseri	100
İstanbul	200	Konya	200
		Trabzon	150

The problem is solved by using North-West Corner method. The solution is represented:

To / From	Adana	Kayseri	Konya	Trabzon	**Capacity**
Ankara	19	7	3	21	100
İzmir	24	28	14	30	300
İstanbul	32	21	18	22	200
Demand	150	100	200	150	600

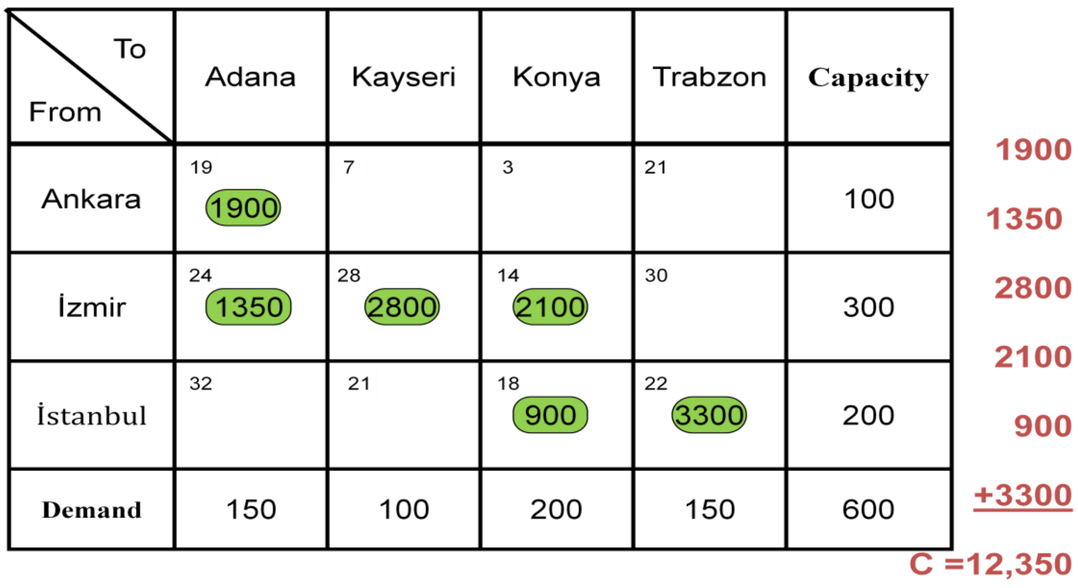
Figure 5.12: Application of Transportation Problem (b)



To / From	Adana	Kayseri	Konya	Trabzon	**Capacity**
Ankara	19 100	7	3	21	100
İzmir	24 50	28 100	14 150	30	300
İstanbul	32	21	18 50	22 150	200
Demand	150	100	200	150	600

Figure 5.13: Application of Transportation Problem (c)

The previous table show the process of satisfying all constraints and allows us to begin with a starting feasible solution. Multiply the quantity in each cell by the cost.



To / From	Adana	Kayseri	Konya	Trabzon	Capacity
Ankara	19 1900	7	3	21	100
İzmir	24 1350	28 2800	14 2100	30	300
İstanbul	32	21	18 900	22 3300	200
Demand	150	100	200	150	600

1900
1350
2800
2100
900
+3300
C =12,350

Figure 5.14: Application of Transportation Problem (d)

The solution is represented by a graph like this:

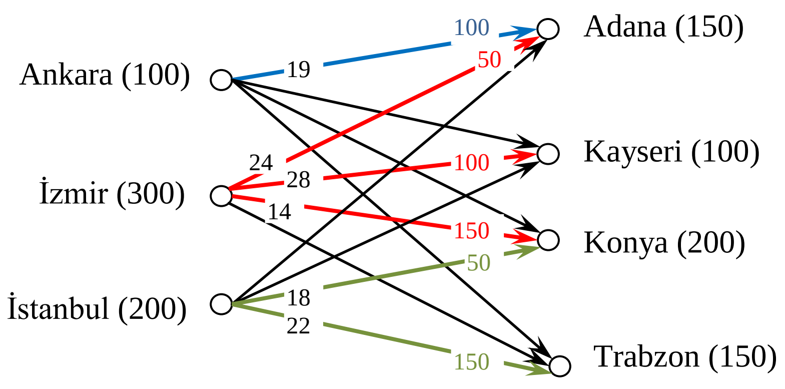


Figure 5.15: Application of Transportation Problem (e)

CHAPTER 6

CONCLUSION

In this thesis, we have solved three related problems in logistics. First, given a network where the speed of the connection between every two components is known, we can find the fastest (shortest) possible connection from one component to another by applying Dijkstra’s algorithm to corresponding graph. Second, given a network where there is a limit to how much traffic can pass over each connection between components, we can use the Ford- Fulkerson algorithm to determine a flow function that gives us the maximum possible traffic between one component and another. Third one is transportation problem. This type of problem is known as distribution or transportation problem in which the key idea is to minimize the cost or the time of transportation by using North-West Corner method. Finally, transportation in logistics is modeled and examined by applying network problems and their algorithms.

GLOSSARY

adjacent

Two vertices of a graph *G* are said to be adjacent if there exists an edge of *G* joining the two vertices.

arcs

The lines connecting the nodes in a network.

bipartite

A graph is **bipartite** if its vertices can be partitioned into two disjoint subsets *U* and *V* such that each edge connects a vertex from *U* to one from *V*. A bipartite graph is a **complete bipartite** graph if every vertex in *U* is connected to every vertex in *V*. If *U* has *n* elements and *V* has *m*, then we denote the resulting complete bipartite graph by $K_{n,m}$.

cardinality

The number of elements in a set is called its cardinality.

complete graph

A complete graph with *n* vertices (denoted K_n) is a graph with *n* vertices in which each vertex is connected to each of the others (with one edge between each pair of vertices). Here are the first five complete graphs:

complete bipartite

A **complete bipartite** graph is a bipartite graph in which each vertex in V_1 is joined to each vertex in V_2 by a unique edge. If V_1 has *r* vertices and V_2 has *s* vertices then the corresponding complete bipartite graph is denoted $K_{r,s}$.

component

A subgraph *H* of a graph *G* is called a component of *G* if *H* is a maximally connected subgraph of *G*.

connected

For vertices u and v of a graph G, u is said to be connected to v if G contains a u - v **path**. The graph G is called a connected graph if the vertices u and v are connected for any pair $u ,v \in V(G)$.

cycle

A cycle is a walk of length $n \geq 3$ in which the begin and end-vertices, are the same, but in which no other vertices repeat. A graph consisting of a single cycle of length n is so called and denoted C_n.

degree

The degree (or valence) of a vertex is the number of edge *ends* at that vertex. For example, in this graph all of the vertices have degree three.

In a digraph (directed graph) the degree is usually divided into the **in-degree** and the **out-degree** (whose sum is the degree of the vertex in the underlying undirected graph).

digraph

A digraph (or a **directed graph**) is a graph in which the edges are directed. (Formally: a digraph is a (usually finite) set of vertices V and set of *ordered* pairs (a,b) (where a, b are in V) called edges. The vertex a is the **initial vertex** of the edge and b the **terminal vertex**.

disconnected

A graph that is not connected is said to be disconnected.

edge

An edge is a 2 - element subset of the vertex set of a graph. Edges are indicated by inter-connecting lines between vertices in graphical representations of a graph.

edge set

The set $E(G)$, comprised of all the edges of a graph G, is called the edge set of the graph.

end- vertex

If the degree of a vertex is 1, then it is called an end-vertex.

Eulerian graph

A connected graph is Eulerian if it contains a closed trail that includes every edge; such a trail is an Eulerian trail.

flow capacity

The maximum flow for an arc of the network. The flow capacity in one direction may not equal the flow capacity in the reserve direction.

graph

Informally, a graph is a finite set of dots called **vertices** (or **nodes**) connected by links called **edges** (or **arcs**). More formally: a **simple graph** is a (usually finite) set of vertices V and set of unordered pairs of distinct elements of V called edges.

Not all graphs are simple. Sometimes a pair of vertices are connected by multiple edge yielding a **multigraph**. At times vertices are even connected to themselves by a edge called a **loop**. Finally, edges can also be given a direction yielding a directed graph (or **digraph**).

Hamiltonian graph

A connected graph is Hamiltonian if it contains a cycle that includes every vertex; such a cycle is a Hamiltonian cycle.

incident

A vertex *v* and edge *e* of a graph *G* is said to be incident, if *e* joins *v* to another vertex in *G*.

in-degree

The in-degree of a vertex *v* is the number of edges with *v* as their terminal vertex.

induced subgraph

A subgraph *H* of a graph *G* is said to be induced if, for any pair of vertices *x* and *y* of H, *xy* is an edge of *H* if and only if *xy* is an edge of G. In other words, *H* is an induced subgraph of *G* if it has the most edges that appear in *G* over the same vertex set. If *H* is chosen based on a vertex subset *S* of *V(G)*, then *H* can be written as *G*[*S*] and is said to be induced by *S*

isolated

A vertex of degree zero (with no edges connected) is isolated.

isomorphic

Two graphs *G* and *H* are said to be **isomorphic**, denoted by $G \sim H$, if there is a one-to-one correspondence, called an **isomorphism**, between the vertices of the graph such that two vertices are adjacent in *G* if and only if their corresponding vertices are adjacent in *H*.

length

For the length of a path see **path**.

loop

A loop is an edge that connects a vertex to itself.

maximal flow

The maximum amount of flow that can enter and exist a network system during a given period of time.

multiple

A set of arcs are **multiple**, or *parallel*, if they share the same head and the same tail. A pair of arcs are **anti-parallel** if one's head/tail is the other's tail/head.

multiple edge

An edge such that there is another edge with the same endvertices; antonyms: **simple edge**. The **multiplicity of an edge** is the number of multiple edges sharing the same endvertices; the **multiplicity of a graph**, the maximum multiplicity of its edges.

multigraph

A multigraph is a graph with multiple edges between the same vertices .

network

A weighted graph, possibly directed or undirected, possibly containing special vertices (nodes), such as **source** or **sink**.

node

A synonym for vertex.

oriented graph

A graph that contains only arcs. When stated without any qualification, a graph is almost always assumed to be undirected. Also, a digraph is usually assumed to contain no undirected edges.

out-degree

The out-degree of a vertex *v* is the number of edges with *v* as their initial vertex.

path

A walk in which no vertex is repeated is called a path. A graph solely consisting of a path of order n is so called and denoted Pn.

planar

A graph is planar if it can be drawn on a plane so that the edges intersect only at the vertices.

pseudograph

Informally, a pseudograph is a graph with multiple edges (or loops) between the same vertices (or the same vertex). Formally: a pseudograph is a set V of vertices along, a set E of edges, and a function *f* from E to {{*u*,*v*}|*u*,*v* in V}. (The function *f* shows which vertices are connected by which edge.) An edge is a loop if *f*(*e*) = {*u*} for some vertex *u* in V.

shortest route

Shortest path between two nodes in a network.

size

The cardinality of the edge set of a graph G is called the size of G.

source

A vertex of a network with in-degree of zero; see also target.

spanning tree

A spanning subgraph that is a tree. Every graph has a spanning forest. But only a connected graph has a spanning tree.

subgraph

A graph H is a subgraph of a graph G if $V(H) \subseteq V(G)$ and every edge of H is an edge of H. We write $H \le G$ to mean H is a subgraph of G.

target

A vertex of a network with out-degree of zero; see also source.

trail

A trail is a walk in which all the edges are distinct.

transportation problem

A network flow problem that often involves minimizig the cost of shipping goods from a set of orgins to a set of destinations; it can be formulated and solved as a linear program by including a variable for each arc and a constraint for each node.

tree

A connected graph with no cycles.

undirected

A graph in which each edge symbolizes an unordered, transitive relationship between two nodes. Such edges are rendered as plain lines or arcs.

vertex

A vertex is a combinatorial element in terms of which a graph is defined. Vertices are indicated by nodes in the graphical representation of a graph.

vertex set

The set comprised of all vertices of a graph *G,* is called the vertex set of *G*.

walk

A walk in a graph G is an alternating sequence of incident vertices and edges. The number of edges in the walk defines its length, while the number of vertices defines its order.

weighted

Weighted edges symbolize relationships between nodes which are considered to have some value, for instance, distance or lag time. Such edges are usually annotated by a number or letter placed beside the edge. Weighted nodes also have some value; this must be distinguished from identification.

weighted graph

A graph that associates a label (weight) with every edge in the graph. Weights are usually real numbers. They may be restricted to rational numbers or integers. Certain algorithms require further restrictions on weights; for instance, the Dijkstra algorithm works properly only for positive weights.

REFERENCES

Ahuja, R.K., Magnanti, T.L., Orlin, J.B. (1993). “Network Flows: Theory, Algorithms and Applications.” Prentice Hall.

Bang-Jensen, J and Gutin, G.(2007). Digraphs Theory, Algorithms and Applications.

Bollobas, B. (1998). Modern Graph Theory. New York: Springer-Verlag.

Brossard, E.(2010). Paper of Graph Theory: Network Flow, Washington.

Carroll, J. (2004) The magical reserve tracing system-RFID. Taiwan CNET, http://taiwan.cnet.com/enterprise/technology/0,2000062852,2008707 1,00.htm.

Chang, Y.H. (1998) Logistical Management. Hwa-Tai Bookstore Ltd., Taiwan.

Chartrand, G. and Lesniak, L. (2000). Graphs & Digraphs, (Chapman & Hall / CRC).

Ciupala, L. (2010). Bulletin of the Transilvania University of Bra¸sov • Vol 3(52) Series III: Mathematics, Informatics, Physics, 177-182.

Cooper, M.C., Lambert, D.M. and Pagh, J.D. (1997) Supply chain management: more than a new name for logistics, International Journal of Logistics Management, Vol. 8, No. 1, 1-13.

Cormen, T .H, Leiserson, C. E., Rivest, R.L., and _Stein, C. (2001) [1990]. "26". *Introduction to Algorithms* (2nd edition ed.). MIT Press and McGraw-Hill. pp. 696–697.

Council of Logistics Management (1991) Definition of Logistics. http://www.cscmp.org/.

Delaney, R. (1999). *A Look Back in Anger at Logistics Productivity*. 10th Annual “State of Logistics Report,” Cass Information Systems and ProLogis, Saint Louis, Mo.

Demaine, E. And Karger, D. (2003) Advanced Algorithms in Graph Theory.

Fair, M.L. and Williams, E.W.(1981). Transportation and Logistics. Business Publication Inc., USA.

Gibbons, A. (1985). *Algorithmic Graph Theory*. Cambridge: Cambridge University Press.

Goldberg, A.V, Tardos, E. And Tarjan E. (1990). *Algorithms and Combinatorics* Vol.9, Berlin.

Gross, L.J and Yellen, J. (2006). Graph theory and its applications.

Harju, T. (2007). Lecture Notes on Graph Theory, Finland.

Hiller, F.S. and Lieberman, G.J. (1995) *Introduction to Operations Research,* 6th ed. New York : McGraw-Hill, Inc. 998 pp.

Krumwiede, D.W. and Sheu, C.(2002) A model for reverse logistics entry by third-party providers, Science Direct, Vol. 30, 325-333.

Marcus, D.A. (2008). A Problem Oriented Approach Graph Theory, printed in the United States of America.

Proceedings of the Eastern Asia Society for Transportation Studies,(2005). Vol. 5, pp. 1657 – 1672.

Reeb, J. and Leavengood, S. (2002). Transportation Problem: A Special Case for Linear Programming Problems

Reynolds-Feighan, A.J. (2001). Air freight logistics. In A.M. Brewer, K.J. Button and D.A. Hensher (eds.), Handbook of Logistics and Supply-Chain Management. Elsevier Science Ltd., UK, 431-439.

Sokkalingam, P. T. , Ahuja, R.K. and Orlin, J.B. New Polynomial-Time Cycle-Canceling Algorithms for Minimum Cost Flows, India.

Tilanus, B. (1997) Information Systems in Logistics and Transportation. Elsevier Science Ltd., UK.

Tseng, Y., Yue, W.L. and Taylor, M. (2005). The Role of Transportation in Logistics Chain.

West, D.B.(2001). *Introduction to Graph Theory* (2ed). Upper Saddle River: Prentice Hall.

Weisstein, E.W,. "Graph." From MathWorld-AWolfram Web Resource., http://mathworld.wolfram.com/Graph.html.

Zaslavsky, T. Glossary of signed and gain graphs and allied areas. *Electronic Journal of Combinatorics*, Dynamic Surveys in Combinatorics.