Chapter 13

Digraphs and Networks

In this chapter we briefly discuss directed graphs (abbreviated as digraphs). As pointed out in the opening paragraphs of Chapter 11, digraphs are similar to graphs, the difference being that in digraphs, the edges have directions and are called arcs. Thus, digraphs model nonsymmetric relations, in the same sense that graphs model symmetric relations. Many of the results we prove are directed analogues of results already proved for graphs.

A network is a digraph with two distinguished vertices s and t, in which each arc has a nonnegative weight, called its *capacity.* If we think of each arc as a conduit over which flows some substance and think of the capacity of an arc as the amount that can flow through the conduit per unit time (say), one important problem is that of finding the maximum possible flow from the "source" s to the "target" t, subject to the given capacities. The answer to this problem, along with an efficient algorithm for constructing a maximum flow, is given by the so-called *max-flow min-cut theorem.* We then use the max-flow min-cut theorem to give another proof of the basic result, Theorem 12.5.3, about matchings in bipartite graphs.

13.1 Digraphs

A *digraph* $D = (V, A)$ has a set V of elements called *vertices* and a set A of *ordered* pairs of not necessarily distinct vertices called *arcs.* Each arc is of the form

$$\alpha = (a, b), \tag{13.1}$$

where a and b are vertices. We think of the arc α as *leaving* a and *entering* b, that is, pointed (or directed) from a to b.

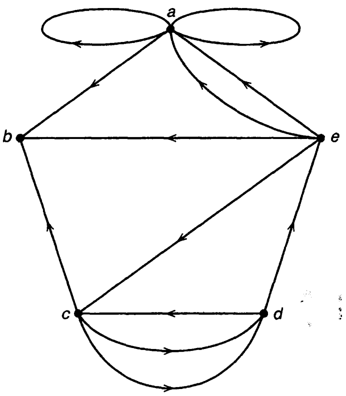

Figure 13.1

In contrast to graphs, (a, b) is not the same as (b, a). We shall use terminology which is similar to that used for graphs, but there are distinctions that apply to digraphs which don't apply to graphs. Thus, the arc α in (13.1) has *initial vertex* $\iota(\alpha) = a$ and *terminal vertex* $\tau(\alpha) = b$. A digraph may contain both of the arcs (a, b) and (b, a) as well as loops of the form (a, a). A loop (a, a) enters and exits the same vertex a. We may generalize a digraph to a *general digraph* in which multiple arcs are allowed.[1] We draw general digraphs as we draw graphs, but for digraphs we put an arrow on each edge in order to indicate its direction.

Example. A general digraph is shown in Figure 13.1. It is not a digraph, since some of the arcs have multiplicities greater than 1. □

A vertex x of a general digraph $D = (V, A)$ has two degrees. The *outdegree* of x is the number of arcs α of which x is the initial vertex:

$$|\{\alpha|\iota(\alpha) = x\}|.$$

The *indegree* of x is the number of arcs α of which x is the terminal vertex:

$$|\{\alpha|\tau(\alpha) = x\}|.$$

A loop (x, x) contributes 1 to both the indegree and outdegree of the vertex x. A proof similar to the one given for Theorem 11.1.1 establishes the next elementary result.

Theorem 13.1.1 *In a general digraph the sum of the indegrees of the vertices equals the sum of the outdegrees, and each is equal to the number of arcs.* □

[1]The number of arcs, including multiplicities, however, should always be finite.

Example. In the general digraph of Figure 13.1, the indegrees of the vertices a, b, c, d, e are

$$4, 3, 2, 2, 1;$$

the outdegrees are

$$3, 0, 3, 2, 4.$$

In each case the sum is 12, the number of arcs. □

With any general graph $G = (V, E)$, we can obtain a general digraph $D = (V, A)$ by giving each edge $\{a, b\}$ of E an orientation, that is, by replacing $\{a, b\}$ with either (a, b) or (b, a).[2] Such a digraph D is called an *orientation* of G. A general graph has many different orientations. Conversely, given a general digraph $D = (V, A)$, we can remove the directions of its arcs, thereby obtaining a general graph $G = (V, E)$. Such a graph is called the *underlying general graph of* G. A general digraph has exactly one underlying general graph.

Example. The underlying general graph of the general digraph in Figure 13.1 is shown in Figure 13.2. □

An orientation of a complete graph K_n with n vertices is called a *tournament.* It is a digraph such that each distinct pair of vertices is joined by exactly one arc. This arc may have either of the two possible directions. A tournament can be regarded as the record of who beat whom in a round-robin tournament in which each player plays every other player exactly once and there are no ties. The nicest kinds of tournaments[3] are those in which it is possible to order the players in a list

$$p_1, p_2, \ldots, p_n$$

so that each player beats all those further down on the list. Such tournaments are called *transitive tournaments.* The reason is that if p_i beats p_j and p_j beats p_k, then also p_i beats p_k. In a transitive tournament there is a consistent ranking of the players.

[2]If the multiplicity of $\{a, b\}$ is greater than 1, then some copies of $\{a, b\}$ can be replaced by (a, b), and others can be replaced by (b, a).

[3]From the point of view of ranking the players at the end.

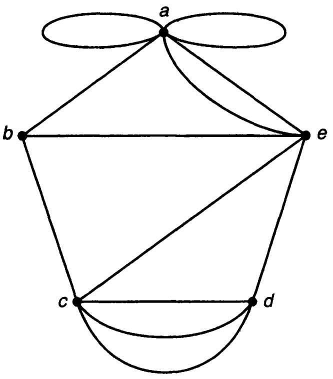

Figure 13.2

We modify our definitions of *walk*, *path*, and *cycle* in a general graph in order to obtain analogous concepts for general digraphs. Let $D = (V, A)$ be a general digraph. A sequence of m arcs of the form

$$(x_0, x_1), (x_1, x_2), \ldots, (x_{m-1}, x_m) \tag{13.2}$$

is called a *directed walk of length m from vertex x_0 to vertex x_m*. The *initial vertex* of the walk (13.2) is x_0 and the *terminal vertex* is x_m. The directed walk is *closed* if $x_0 = x_m$ and *open* otherwise. We also denote the walk (13.2) by

$$x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_m.$$

A directed walk with distinct arcs is a *directed trail*; a directed trail with distinct vertices (except possibly the initial and terminal vertices) is a *path*[4]; a closed path is a *directed cycle*.

Example. Consider the general digraph of order 5 in Figure 13.1. Then

(1) $d \rightarrow e \rightarrow c \rightarrow d \rightarrow e$ is a directed walk,

(2) $c \rightarrow d \rightarrow e \rightarrow c \rightarrow b$ is a directed trail,

(3) $c \rightarrow d \rightarrow e \rightarrow a \rightarrow b$ is a path, and

(4) each of $c \rightarrow d \rightarrow e \rightarrow c$, $c \rightarrow d \rightarrow c$ and $a \rightarrow a$ is a directed cycle. □

[4]In contrast to walks and cycles, we use *path* instead of *directed path*.

A general digraph is *connected*, provided that its underlying general graph is connected. A general digraph is *strongly connected*, provided that, for each pair of distinct vertices a and b, there is a directed walk[5] from a to b and a directed walk from b to a. Thinking of a general digraph as a network of one-way streets connecting the various parts of a city, we see that strong connectivity means that we can get from any part of the city to any other part, traveling along streets only in their given direction.

Example. The general digraph in Figure 13.1 is connected, but it is not strongly connected. The easiest way to see that it is not strongly connected is to observe that vertex b has outdegree equal to 0. Thus, it is not possible to leave b. □

A directed trail in a general digraph D is called *Eulerian*, provided that it contains every arc of D. A *Hamilton path* is a path that contains every vertex. A *directed Hamilton cycle* is a directed cycle that contains every vertex.

The next two theorems are the directed analogues of Theorems 11.2.2 and 11.2.3. Since their proofs are similar, we omit them.

Theorem 13.1.2 *Let D be a connected digraph. Then D has a closed Eulerian directed trail if and only if the indegree of each vertex equals the outdegree.*

Theorem 13.1.3 *Let D be a connected digraph and let x and y be distinct vertices of D. Then there is a directed Eulerian trail from x to y if and only if*

(*i*) *the outdegree of x exceeds its indegree by 1;*

(*ii*) *the indegree of y exceeds its outdegree by 1;*

(*iii*) *for each vertex $z \neq x, y$, the indegree of z equals its outdegree.*

There is also a directed analogue of Theorem 11.3.2 due to Ghouila-Houri[6] giving a sufficient condition for the existence of a directed Hamilton cycle, but it is much more difficult to prove. We shall be content simply to state the theorem. In the theorem, D is a digraph (and not a general digraph) without loops.[7]

Theorem 13.1.4 *Let D be a strongly connected digraph without any loops. If, for each vertex x, we have*

$$(\textit{outdegree of } x) + (\textit{indegree of } x) \geq n,$$

then D has a directed Hamilton cycle.

[5]And thus a path.

[6]A. Ghouila-Houri, Une condition suffisante d'existence d'un circuit hamiltonien, *C.R. Acad. Sci.*, 251 (1960), 494.

[7]More than one arc from one vertex to another is of no help in locating a Hamilton directed cycle, nor is a loop of any help.

We now show that a tournament always has a Hamilton path. This implies that it is always possible to rank the players in the order

$$p_1, p_2, \ldots, p_n, \tag{13.3}$$

so that p_1 beats p_2, p_2 beats $p_3, \ldots$, p_{n-1} beats p_n. This does not imply that we have a consistent ranking of the players, since we are not asserting that each player beats all players further down on the list. Indeed, a tournament may even have a directed Hamilton cycle, thereby implying that for each player there is a ranking (13.3) in which he or she is ranked first!

Theorem 13.1.5 *Every tournament has a Hamilton path.*

Proof. Let D be a tournament of order n. Let

$$\gamma : x_1 \to x_2 \to \cdots \to x_p \tag{13.4}$$

be a longest path in D. We show that a longest path (13.4) is a Hamilton path by showing that, if $p < n$, then we can find a longer path. Suppose that $p < n$ so that the set U of vertices not on the path (13.4) is nonempty. Let u be any vertex in U. If there is an arc from u to x_1 or an arc from x_p to u, then we can find a longer path. Thus, we can now assume that the arc between x_1 and u has u as its terminal vertex. Similarly, we can now assume that the arc between x_p and u has u as its initial vertex. So as we consider the arcs bewteen u and the vertices $x_1, x_2, \ldots, x_p$ in sequence, there must be consecutive vertices x_k and x_{k+1} on the path γ such that the arc between x_k and u has u as its terminal vertex, and the arc between x_{k+1} and u has u as its initial vertex. But then

$$x_1 \to \cdots \to x_k \to u \to x_{k+1} \to \cdots \to x_p$$

is a longer path than γ. We leave it as an exercise to use this proof to determine an algorithm for a Hamilton path in a tournament. □

We conclude this brief introduction to digraphs by proving two theorems of some practical importance. The first of these is a theorem of Robbins[8] which characterizes those general graphs that have a strongly connected orientation. Thus, this theorem will tell the traffic engineer of a city with no one-way streets whether it is possible (and how) to make all streets into one-way streets in such a way that one can get from any part of the city to any other part.[9]

Theorem 13.1.6 *Let $G = (V, E)$ be a connected graph. Then G has a strongly connected orientation if and only if G does not have any bridges.*

[8]H. E. Robbins A Theorem on Graphs, with an Application to a Problem in Traffic Control, *Amer. Math. Monthly*, 46 (1939), 281–283.

[9]The consequences to the traffic engineer if he or she fails to achieve this property are obvious.

Proof. First, assume that G has a bridge α. The removal of α from G results in a disconnected graph with two connected components $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. If we orient α from G_1 to G_2, then there is no directed walk from a vertex of G_2 to a vertex of G_1. If we orient α from G_2 to G_1, there is no directed walk from a vertex in G_1 to a vertex in G_2. Hence, G does not have a strongly connected orientation.

Now assume that G does not have any bridges. By Lemma 11.5.3, each edge of G is contained in some cycle. The next algorithm determines a strong orientation of G.

Algorithm for a strongly connected orientation of a bridgeless connected graph

Let $G = (V, E)$ be a connected graph without bridges.

```
(1) Put $U = \emptyset$.
(2) Locate a cycle $\gamma$ of $G$.
      (i) Orient the edges of $\gamma$ so that it becomes a directed cycle.
     (ii) Add the vertices of $\gamma$ to $U$.
(3) While $U \neq V$, do the following:
      (i) Locate an edge $\alpha = \{x, y\}$ joining a vertex $x$ in $U$ to a vertex $y$ not in $U$.
     (ii) Locate a cycle $\gamma$ containing the edge $\alpha$.
    (iii) Orient the edge $\alpha$ from $x$ to $y$ and continue to orient the edges of $\gamma$ as if to
          form a directed cycle until arriving at a vertex $z$ in $U$.
     (iv) Add to $U$ all the vertices of $\gamma$ from $x$ to $z$.
(4) Orient in either direction every edge that has not yet received an orientation.
```

We note that a cycle containing the edge $\alpha = \{x, y\}$ in (3)(ii) can be located by finding a path (for instance, a shortest path) joining x and y in the graph obtained by deleting the edge α. It should be clear that the digraph obtained by applying the preceding algorithm is a strongly connected orientation of G, provided that step (3) terminates—that is, provided that the set U does achieve the value V. But if $U \neq V$, then since G is connected, there must be an edge α joining a vertex in U to a vertex y not in U. Since each edge of G is contained in a cycle, the vertex y is, in fact, put in U. From this, it follows that the terminal value of U is V. □

Example. *A trading problem.*[10] There are n traders $t_1, t_2, \ldots, t_n$ who enter a market, each with an indivisible item[11] to offer in trade. We assume for simplicity that a trader

[10]This example and its subsequent analysis is partly based on the article "On Cores and Indivisibility" by L. Shapely and H. Scarf, in *Studies in Optimization* (MAA Studies in Mathematics, vol. 10), 1974, Mathematical Association of America, Washington, D.C., 104–123.

[11]For instance. a car or a house.

never has any use for more than one of the items, but except for this assumption, the items are freely transferable from one trader to another. Each trader ranks the n items brought to the market (including his own) according to his preference for them. There are no ties, and thus each trader ranks the items from 1 to n. The effect of the market activity is to redistribute (or permute) the ownership of the items among the n traders. Such a permutation is called an *allocation.* We regard an allocation as a one-to-one function

$$\rho : \{t_1, t_2, \ldots, t_n\} \to \{t_1, t_2, \ldots, t_n\},$$

where $\rho(t_i) = t_j$ means that trader t_i receives the item of trader t_j in the allocation. An allocation ρ is called a *core allocation*, provided that it has the following property: There does not exist a subset S of fewer than n traders such that, by trading among themselves, each receives an item that he or she ranks more highly than in the allocation ρ.[12] For example, suppose that $n = 5$ and the preferences of the traders are as given by the following table:

	t_1	t_2	t_3	t_4	t_5
t_1	4	3	1	2	5
t_2	4	3	1	2	5
t_3	4	3	5	1	2
t_4	1	4	3	5	2
t_5	4	5	2	1	3

(13.5)

The first row of this table gives t_1's ranking of the items. Thus, t_1 values the item of t_3 most highly, then the items of t_4, t_2, t_1, t_5 in this order. The interpretation of the other rows of the table is similar. One possible allocation ρ is

$$\rho(t_1) = t_2,\ \rho(t_2) = t_3,\ \rho(t_3) = t_1,\ \rho(t_4) = t_5,\ \rho(t_5) = t_4.$$

This allocation is not a core allocation since

$$\rho'(t_1) = t_4,\ \rho'(t_4) = t_1$$

defines an allocation for the two traders t_1, t_4 in which each gets an item he or she values more highly than he gets in ρ. A core allocation in this case is ρ^*:

$$\rho^*(t_1) = t_3,\ \rho^*(t_2) = t_2,\ \rho^*(t_3) = t_4,\ \rho^*(t_4) = t_1,\ \rho^*(t_5) = t_5.$$

Does every trading problem have a core allocation? In the remainder of this section we answer this question.[13] □

[12]Put another way, there does not exist a subset S of fewer than n traders and an allocation ρ' for them such that, for each trader t_i in S, t_i ranks the item of $\rho'(t_i)$ higher than that of $\rho(t_i)$.

[13]In the affirmative.

A digraph furnishes a convenient mathematical model for a trading problem. We consider a digraph $D = (V, A)$ in which the vertices are the n traders. We put an arc from each vertex to every other, including the vertex itself.[14] Each vertex has indegree equal to n and outdegree equal to n. The digraph D is a *complete digraph* of order n. For each vertex t_i, we label (or weight) the arcs leaving t_i with $1, 2, \ldots, n$ in accordance with the preferences of t_i. An allocation corresponds to a partition of the vertices into directed cycles. This is a consequence of the next lemma, which implies that a one-to-one function from a set to itself can be thought of as a digraph that consists of one or more directed cycles with no vertices in common.

Lemma 13.1.7 *Let D be a digraph in which each vertex has outdegree at least* 1. *Then there is a directed cycle in D.*

Proof. An algorithm that constructs a directed cycle in D is now given:

Algorithm for a directed cycle

```
Let $u$ be any vertex.
(1) Put $i = 1$ and $x_1 = u$.
(2) If $x_i$ is the same as one of the previously chosen vertices $x_j, (j < i)$, then go to
    (4). Else, go to (3).
(3) Do the following:
      (i) Choose an arc $(x_i, x_{i+1})$ leaving vertex $x_i$.
      (ii) Increase $i$ by 1.
      (iii) Go to (2).
(4) Output the directed cycle
      $x_j \to x_{j+1} \to \cdots \to x_{i-1} \to x_i = x_j$.
```

Since each vertex is the initial vertex of at least one arc and since we stop as soon as we obtain a repeated vertex, the algorithm does output a directed cycle as shown. □

Corollary 13.1.8 *Let X be a set of n elements and let $f : X \to X$ be a one-to-one function. Let $D_f = (X, A_f)$ be the digraph whose set of arcs is*

$$A_f = \{(x, f(x)) : x \text{ in } X\}.$$

Then the arcs of D_f can be partitioned into directed cycles with each vertex belonging to exactly one directed cycle.

[14]Thereby creating a loop at each vertex.

Proof. Since the function f is one-to-one, it is a consequence of the pigeonhole principle that f is also onto. It now follows from the definition of the set A_f of arcs that each vertex of D_f has its indegree and outdegree equal to 1. By Lemma 13.1.7, D_f has a directed cycle γ. Either each vertex is a vertex of γ, in which case our partition contains only γ, or, removing γ (its vertices and arcs), we are left with a digraph, each of whose vertices also has indegree and outdegree equal to 1. We continue to remove directed cycles until we exhaust all of the vertices, and this gives us the desired partition. □

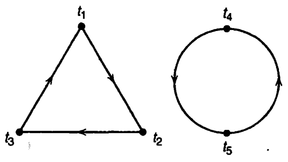

Figure 13.3

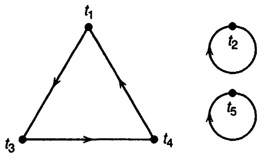

Figure 13.4

Example. The digraphs D_ρ and $D_{\rho'}$ corresponding to the allocations ρ and ρ^* defined in the example "*A trading problem*" give the directed cycle partitions shown in Figures 13.3 and 13.4, respectively. □

The problem of the existence of core allocations can be regarded as a directed version of the stable marriage problem described in Section 9.3. We now use the digraph model to answer our question about the existence of core allocations.

Theorem 13.1.9 *Every trading problem has a core allocation.*

Proof. The proof shows how successive use of the algorithm for directed cycles, given in the proof of Lemma 13.1.7, results in a core allocation.

Let the set of traders be $V = \{t_1, t_2, \ldots, t_n\}$. Consider the preference digraph $D^1 = (V, A^1)$, where there is an arc (t_i, t_j) from t_i to t_j if and only if t_i prefers the item of t_j over all other items. Each vertex has outdegree 1; hence, by Lemma 13.1.7, there is a directed cycle γ_1 in D^1. Let V^1 be the set of vertices of γ_1. Let $D^2 = (V - V^1, A^2)$

be the preference digraph[15] with vertex set $V - V^1$ in which there is an arc from t_i to t_j if and only if t_i prefers the item of t_j over all the other items of the traders in $V - V^1$. Each vertex of the digraph D^2 has outdegree 1 and again, by Lemma 13.1.7, we can find a directed cycle γ_2. We let V^2 be the set of vertices of γ_2, and we consider the preference digraph $D^3 = (V - (V^1 \cup V^2), A^3)$. Continuing in this way, we obtain $k \geq 1$ directed cycles $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$ with vertex sets $V^1, V^2, \ldots, V^k$, respectively, where $V^1, V^2, \ldots, V^k$ is a partition of V, the set of traders. The set Γ of cycles determines an allocation ρ: Each trader t_p is a vertex of exactly one of the directed cycles in Γ, and this directed cycle has an arc from t_p to some t_q. Defining $\rho(t_p) = t_q$, we obtain an allocation.

We now show that the allocation ρ is a core allocation. Let U be any subset of fewer than n traders. Let j be the smallest integer such that $U \cap V^j \neq \emptyset$. Then

$$U \subseteq V^j \cup \cdots \cup V^k = V - (V^1 \cup \cdots \cup V^{j-1}),$$

and U is a subset of the vertices of the digraph D^j. Let t_s be any trader in $U \cap V^j$. Then, in the allocation ρ, t_s gets the item he or she ranks the highest among all the items of traders in $V - (V^1 \cup \cdots \cup V^{j-1})$ and hence among all the traders in S. Thus, by trading among the members of U, t_s cannot obtain an item he or she ranks higher than the item he or she was assigned in ρ. Therefore, ρ is a core allocation. $\square$

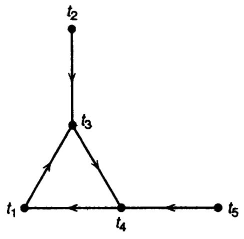

Figure 13.5

Example. Consider the trading problem determined by the table in (13.5). The preference digraph D^1, pictured in Figure 13.5, has exactly one directed cycle, namely,

$$t_1 \to t_3 \to t_4 \to t_1.$$

The preference digraph D^2, pictured in Figure 13.6, consists of the two disjoint directed cycles

$$t_2 \to t_2 \text{ and } t_5 \to t_5.$$

[15]Note well that the vertex set of D^2 is only a subset of the traders.

We can pick either of these directed cycles, and then the other is the preference digraph D^3.[16] A core allocation for our problem is given by

$$\rho(t_1) = t_3, \rho(t_3) = t_4, \rho(t_4) = t_1, \rho(t_2) = t_2, \rho(t_5) = t_5.$$

□

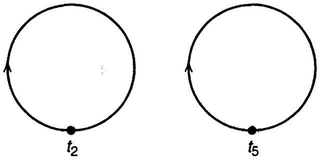

Figure 13.6

13.2 Networks

A *network* is a digraph (V, A) in which two vertices—the *source* s and the *target* t—are distinguished, where $s \neq t$, and in which each arc α has a nonnegative weight $c(\alpha)$, called its *capacity*. We denote a network by $N = (V, A, s, t, c)$.

The basic problem to be treated for networks is that of moving a substance from the source to the target, within the constraints provided by the arcs of the digraph and their capacities. Formally, a *flow* in the network N is defined to be a function f that assigns a real number $f(\alpha)$ to each arc α, subject to the following constraints:

(1) $0 \leq f(\alpha) \leq c(\alpha)$. (The flow through an arc is nonnegative and does not exceed its capacity.)

(2) $\sum_{\iota(\alpha)=x} f(\alpha) = \sum_{\tau(\alpha)=x} f(\alpha)$ for each vertex $x \neq s, t$. (For each vertex x other than the source and the target, the flow into x equals the flow out of x.)

In order to demonstrate that the net flow out of the source,

$$\sum_{\iota(\alpha)=s} f(\alpha) - \sum_{\tau(\alpha)=s} f(\alpha),$$

equals the net flow into the target,

$$\sum_{\tau(\alpha)=t} f(\alpha) - \sum_{\iota(\alpha)=t} f(\alpha)$$

[16]In general, when one of the preference digraphs consists of pairwise disjoint, directed cycles, then the core allocation ρ constructed in the proof of Theorem 13.1.9 is determined.

(where the common value is the amount moved from the source to the target), we prove the next result. For each set of vertices U, we let

$$\overrightarrow{U} = \{\alpha : \iota(\alpha) \text{ is in } U, \tau(\alpha) \text{ is not in } U\}$$

and

$$\overleftarrow{U} = \{\alpha : \iota(\alpha) \text{ is not in } U, \tau(\alpha) \text{ is in } U\}.$$

Lemma 13.2.1 *Let f be a flow in the network $N = (V, A, s, t, c)$ and let U be a set of vertices containing the source s but not the target t. Then*

$$\sum_{\alpha \in \overrightarrow{U}} f(\alpha) - \sum_{\alpha \in \overleftarrow{U}} f(\alpha) = \sum_{\iota(\alpha)=s} f(\alpha) - \sum_{\tau(\alpha)=s} f(\alpha).$$

Proof. We evaluate the sum

$$\sum_{x \in U} \left(\sum_{\iota(\alpha)=x} f(\alpha) - \sum_{\tau(\alpha)=x} f(\alpha) \right) \tag{13.6}$$

in two different ways. On the one hand, it follows from the definition of a flow that all terms in the outer sum are zero except for that one corresponding to the vertex s. Hence, the value is

$$\sum_{\iota(\alpha)=s} f(\alpha) - \sum_{\tau(\alpha)=s} f(\alpha). \tag{13.7}$$

On the other hand, we can rewrite the expression (13.6) as

$$\sum_{x \in U} \sum_{\iota(\alpha)=x} f(\alpha) - \sum_{x \in U} \sum_{\tau(\alpha)=x} f(\alpha),$$

or, equivalently,

$$\sum_{\iota(\alpha) \in U} f(\alpha) - \sum_{\tau(\alpha) \in U} f(\alpha). \tag{13.8}$$

Each arc α with both its initial and terminal vertex in U makes a net contribution of $f(\alpha) - f(\alpha) = 0$ to the sum (13.8); hence, the sum (13.8) equals

$$\sum_{\alpha \in \overrightarrow{U}} f(\alpha) - \sum_{\alpha \in \overleftarrow{U}} f(\alpha).$$

Thus, the equation in the statement of the lemma holds. □

In Lemma 13.2.1, take $U = V - \{t\}$. Then $\overrightarrow{U}$ is the set of all arcs whose terminal vertex is t, and $\overleftarrow{U}$ is the set of all arcs whose initial vertex is t. Hence,

$$\sum_{\iota(\alpha)=s} f(\alpha) - \sum_{\tau(\alpha)=s} f(\alpha) = \sum_{\tau(\alpha)=t} f(\alpha) - \sum_{\iota(\alpha)=t} f(\alpha). \tag{13.9}$$

The common value of the two expressions in (13.9) is called the *value of the flow f* and is denoted by val(f).

Given a network $N = (V, A, s, t, c)$, a flow in N is a *maximum flow*, provided that it has the largest value among all flows in N. The value of a maximum flow (the maximum value of a flow) equals the minimum value of another quantity associated with a network. We shall prove this important fact only in the case that the capacity function is integer-valued,[17] and in doing so, we obtain an algorithm for constructing a maximum flow.

A *cut* in a network $N = (V, A, s, t, c)$ is a set C of arcs such that each path from the source s to the target t contains at least one arc in C.[18] The *capacity* cap(C) of a cut C is the sum of the capacities of the arcs in C. A cut is a *minimum cut*, provided that it has the smallest capacity among all cuts in N.

A cut is a *minimal cut*, provided that each set obtained from C by the deletion of one of its arcs is not a cut.[19] (This means that, for each arc α in C, there is a path from s to t that contains α, but no other arc of C.)

We first show that any minimal cut is a cut of the form $\overrightarrow{U}$ for some set of vertices U containing s but not containing t. This implies that the smallest capacity of a cut is achieved by a cut of this form $\overrightarrow{U}$.

Lemma 13.2.2 *Let $N = (V, A, s, t, c)$ be a network with C a minimal cut. Let U be the set of all vertices x for which there exists a path from the source s to x that contains no arc in C. Then $\overrightarrow{U}$ is a cut and $C = \overrightarrow{U}$.*

Proof. Note that s is in U, since the trivial path consisting only of the vertex s contains no arc in C. Since C is a cut, the target t is not contained in U. Hence, $\overrightarrow{U}$ is a cut. Each arc (x, y) in $\overrightarrow{U}$ is in C, for otherwise there exists a path from s to y containing no arc in U, and y would be in U. Thus, $\overrightarrow{U} \subseteq C$.

Now let $\alpha = (a, b)$ be any arc in C. Since C is a minimal cut, there is a path γ from s to t that contains α, but no other arc of C. This implies that the initial vertex

[17]It then follows that it is also true for capacity functions, all of whose values are rational numbers, by choosing a common denominator for all the rational values. In case the values of the capacity function are not all rational, we must resort to a limiting process.

[18]So we cannot get from the s to t without going over one of the arcs in C.

[19]So a minimum cut is defined arithmetically, while a minimal cut is defined set theoretically. If all the arc capacities are positive, then a minimum cut is also a minimal cut.

a of α is in U. If there were a path γ' from s to b that contained no arc in C, then γ' followed by the part of γ from b to t would give a path from s to t containing no arc in C. It follows that the terminal vertex b of α is not in U. Thus, α is in $\overrightarrow{U}$, and we conclude that $C \subseteq \overrightarrow{U}$. Therefore $C = \overrightarrow{U}$. $\square$

We now prove the very important *max-flow min-cut theorem.*

Theorem 13.2.3 *Let $N = (V, A, s, t, c)$ be a network. Then the maximum value of a flow in N equals the minimum capacity of a cut in N. In other words, the value of a maximum flow equals the capacity of a minimum cut. If the capacities of all the arcs are integers, then there is a maximum flow all of whose values are integers as well.*

Proof. We prove the theorem only under the assumption that the capacity values are all integers. The full theorem can then be established by means of a limiting argument.

The first part of the proof does not use the integrality of the capacity function. We first show that, for each flow f and each cut C,

$$\text{val}(f) \leq \text{cap}(C). \tag{13.10}$$

By Lemma 13.2.2 it suffices to prove this inequality for cuts of the form $\overrightarrow{U}$, where U is a set of vertices with s in U and t not in U. By Lemma 13.2.1 and the fact that flow values are nonnegative, we have

$$\begin{aligned}
\text{val}(f) &= \sum_{\alpha \in \overrightarrow{U}} f(\alpha) - \sum_{\alpha \in \overleftarrow{U}} f(\alpha) \\
&\leq \sum_{\alpha \in \overrightarrow{U}} f(\alpha) \\
&\leq \sum_{\alpha \in \overrightarrow{U}} c(\alpha) \\
&= \text{cap}\, \overrightarrow{U}\, .
\end{aligned}$$

The remainder of the proof is devoted to showing that there is a flow $\hat{f}$ with only integer values and a cut $\hat{C}$ such that $\text{val}(\hat{f}) = \text{cap}(\hat{C})$. Since (13.10) holds, such a flow $\hat{f}$ is a maximum flow, and the cut $\hat{C}$ is a minimum cut.

We start with an arbitrary integer-valued flow f on N. The *zero flow*, in which all flow values equal zero, will suffice, although, in general, it is possible to find an integer-valued flow by trial and error which has a reasonable value for the problems at hand. We then describe an algorithm that results in one of the following two possibilities:

Breakthrough: An integer-valued flow f' has been found with $\text{val}(f') = \text{val}(f) + 1$. In this case, we repeat the algorithm with $f = f'$.

Nonbreakthrough: Breakthrough has not occurred. In this case, we exhibit a cut whose capacity equals the value of the flow f. The cut is our desired minimum cut $\hat{C}$, and the flow f is our desired maximum flow $\hat{f}$.

Basic flow algorithm

Begin with any integer-valued flow f on the network $N = (V, A, s, t, c)$.

(0) Set $U = \{s\}$.

(1) While there exists an arc $\alpha = (x, y)$ with either

  (a) x in U, y not in U, and $f(\alpha) < c(\alpha)$, or

  (b) x not in U, y in U, and $f(\alpha) > 0$,

  put y in U (in case of (a)) or put x in U (in case of (b)).

(2) Output U.

Thus, in the algorithm, we seek either (a) an arc in $\overrightarrow{U}$ (*flowing away from s and toward t*) whose flow value is less than capacity (and update U by putting its terminal vertex in U) or (b) an arc in $\overleftarrow{U}$ (*flowing toward s and away from t*) with a positive flow value (and update U by putting its initial vertex in U). The algorithm terminates when no such arcs remain, and we then output the current set U.

We consider two cases according to whether or not the target t is in U. As we shall see, these cases correspond to breakthrough and nonbreakthrough.

Case 1: The target t is in U.

It follows from the algorithm that, for some integer m, there is a sequence of distinct vertices

$$x_0 = s, x_1, x_2, \ldots, x_{m-1}, x_m = t$$

such that, for each $j = 0, 1, 2, \ldots, m-1$, either

(a) $\alpha_j = (x_j, x_{j+1})$ is an arc of the network with $f(\alpha_j) < c(\alpha_j)$,

or

(b) $\alpha_j = (x_{j+1}, x_j)$ is an arc of the network with $f(\alpha_j) > 0$.

We now define an integer-valued function f' on the set A of arcs by

$$f'(\alpha) = \begin{cases} f(\alpha)+1 & \text{if } \alpha \text{ is one of the arcs } \alpha_j \text{ in (a);} \\ f(\alpha)-1 & \text{if } \alpha \text{ is one of the arcs } \alpha_j \text{ in (b);} \\ f(\alpha) & \text{otherwise.} \end{cases}$$

It follows from the definition of f' and the assumption that all capacities and flow values of f are integers that $0 \leq f'(\alpha) \leq c(\alpha)$. The fact that f' is a flow can be checked by showing that, for each vertex x_j with $j = 1, 2, \ldots, m-1$, the total flow into x_j equals the total flow out of x_j (e.g., if (x_{j-1}, x_j) and (x_{j+1}, x_j) are both arcs, then the flow into x_i has a net change of $+1-1=0$). The value val(f') of the flow f' is val$(f)+1$, since either $(s, x_1) = (x_0, x_1)$ is an arc, in which case the flow out of s is increased by 1, or $(x_1, s) = (x_1, x_0)$ is an arc, in which case the flow into s is decreased by 1; in either case, there is a net increase of 1 in the flow out of s.

Case 2: The target t is not in U.

In this case, $\overrightarrow{U}$ is a cut, and it follows from the algorithm that

(a) $f(\alpha) = c(\alpha)$ for each arc α in $\overrightarrow{U}$ and

(b) $f(\alpha) = 0$ for each arc α in $\overleftarrow{U}$.

Hence,

$$\begin{aligned} \text{val}(f) &= \sum_{\alpha \in \overrightarrow{U}} f(\alpha) - \sum_{\alpha \in \overleftarrow{U}} f(\alpha) \\ &= \sum_{\alpha \in \overrightarrow{U}} c(\alpha) \\ &= \text{cap } \overrightarrow{U}\,. \end{aligned}$$

Hence, $\hat{f} = f$ is a maximum flow and $\hat{C} = \overrightarrow{U}$ is a minimum cut. □

We conclude this section by deducing from the max-flow min-cut theorem two important combinatorial results, including Theorem 12.5.3 from Chapter 12.

Example. Let $D = (V, A)$ be a digraph that models a communication network. The vertices represent junctions (relay points) in the network, and the arcs represent direct (one-way) lines of communication. Consider two junctions corresponding to vertices s and t in V. By putting together direct lines, we can hope to establish a communication path from s to t. Because communication lines may fail, for communication from s to t to be possible even in the presence of some failure, it is important to have redundancy in the digraph—that is, arcs whose failure does not prevent communication from s to t. Define an *st-separating set* to be a set S of arcs of D such that every path from s to t uses at least one arc in S. If the arcs of an st-separating set all fail, communication from s to t is impossible. Menger's theorem, stated next, characterizes the minimum number of arcs in an st-separating set. □

Theorem 13.2.4 *Let s and t be distinct vertices of a digraph $D = (V, A)$. Then the maximum number of pairwise arc-disjoint paths from s to t equals the minimum number of arcs in an st-separating set.*

Proof. Let $N = (V, A, s, t, c)$ be the network in which the capacity of each arc is 1. A cut in N is an st-separating set in D (and vice versa), and the capacity of a cut equals the number of its arcs.

Consider an integer-valued flow f in N, and let $\text{val}(f) = p$. Since all the capacity values equal 1, f takes on only the values 0 and 1: For each arc α, f either "chooses" α (if $f(\alpha) = 1$) or not (if $f(\alpha) = 0$). We prove by induction on p that there exist p pairwise arc-disjoint paths from s to t made up of arcs chosen by f. If $p = 0$, this is trivial. Assume $p \geq 1$. There exists a path γ from s to t; otherwise, if U is the set of vertices that can be reached from s by a path, then $\overrightarrow{U} = \emptyset$ is a cut in N with capacity equal to zero, contradicting $p \geq 1$. Let f' be the integer flow of value $p - 1$ obtained from f by reducing by 1 the value of the flow on the arcs of γ. By induction, there exist $p - 1$ pairwise arc-disjoint arcs from s to t made up of arcs chosen by f'. These $p - 1$ paths, together with γ, are p pairwise arc-disjoint paths made up of arcs chosen by f.

Conversely, if there are p pairwise arc-disjoint paths from s to t, then there is an integer flow in N with value p. The theorem now follows from Theorem 13.2.3. □

We recall some facts from Chapters 11 and 12. A bipartite graph G is a graph whose vertices can be partitioned into two sets X and Y so that each edge joins a vertex in X and a vertex in Y. The pair X, Y is a bipartition of G. A matching in G is a set of pairwise vertex-disjoint edges; a cover of G is a set C of vertices such that each edge of G has at least one of its vertices in C. The maximum number of edges in a matching in G is denoted by $\rho(G)$, and the minimum number of vertices in a cover is denoted by $c(G)$. We show how to deduce Theorem 12.5.3 of Chapter 12 from Theorem 13.2.4 of Menger.

Theorem 13.2.5 *Let G be a bipartite graph. Then $\rho(G) = c(G)$.*

Proof. Let X, Y be a bipartition of G. We first construct a digraph $D = (X \cup Y \cup \{s, t\}, A)$, where s and t are distinct elements not in $X \cup Y$. The arcs of D are those obtained as follows:

1. (s, x) for each x in X (arcs from the source s to each vertex in X;
2. (x, y) for each edge $\{x, y\}$ of G (thus, all arcs of N are directed from X to Y);
3. (y, t) for each y in Y (arcs from each vertex in Y to th etarget t).

Let $\gamma_1, \ldots, \gamma_p$ be a set of pairwise arc-disjoint paths of D from s to t. Each path γ_i is of the form $s \rightarrow x_i \rightarrow y_i \rightarrow t$ for some x_i in X and y_i in Y, and the edges $\{x_1, y_1\}, \ldots, \{x_p, y_p\}$ form a matching in G of size p. Conversely, from a matching in G of size p, we can construct in the natural way p pairwise arc-disjoint paths in D. Hence, $\rho(G)$ equals the maximum number of pairwise arc-disjoint paths from s to t in D.

Now let $C = X' \cup Y'$ be a cover of G, where $X' \subseteq X$ and $Y' \subseteq Y$. Since each path of D from s to t uses an arc of the form (x, y), where $\{x, y\}$ is an edge of G, it follows that

$$S = \{(s, x') | x' \text{ in } X'\} \cup \{(y', t) | y' \text{ in } Y'\} \tag{13.11}$$

is an st-separating set in D with $|C| = |S|$. Conversely, if S is an st-separating set in D of the form (13.11), then the set C defined by $C = X' \cup Y'$ is a cover of G. Now let T be *any* st-separating set in D. Then the set $\hat{T}$ obtained from T by replacing each arc in T of the form (x, y) (x in X and y in Y) with the arc (s, x) is also an st-separating set. Moreover, $\hat{T}$ has the form (13.11) for some $X' \subseteq X$ and $Y' \subseteq Y$, $|\hat{T}| \leq |T|$ (because, for instance, there may be several arcs in T of the form $(x, \cdot)$), and $X' \cup Y'$ is a cover of G. It now follows that $c(G)$ equals the the smallest number of arcs in an st-separating set in D. Therefore, the equality $\rho(G) = c(G)$ follows from Theorem 13.2.4. □

In the next section, we describe a specialization of the basic flow algorithm for finding a matching in a bipartite graph with the maximum number of edges.

13.3 Matchings in Bipartite Graphs Revisited

Let G be a bipartite graph with bipartition X, Y with matching number $\rho(G)$. Each matching M satisfies $|M| \leq \rho(G)$. A matching M with $|M| = \rho(G)$ is called a *max-matching.* If we know $\rho(G)$, we can determine whether any matching M is a max-matching by counting the number $|M|$ of edges in M and checking whether $|M| = \rho(G)$.

Example. Consider the bipartite graph G in Figure 13.7. The edges $\{x_1, y_2\}$ and $\{x_2, y_1\}$ are a matching of size 2 and hence, since clearly $\rho(G)$ cannot be more than 2, we have $\rho(G) = 2$. The edge $\{x_1, y_1\}$ determines a matching M with one edge. Moreover, there is no matching M' with $M \subseteq M'$ and $|M'| = 2$. Thus, we cannot conclude that a matching is a max-matching if we know it is impossible to enlarge the matching by including more edges. □

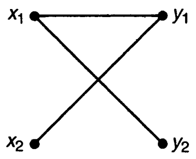

Figure 13.7

We now discuss how to recognize whether a matching is a max-matching without first knowing the value of $\rho(G)$. Once we have a max-matching M, then $\rho(G)$ is determined by $\rho(G) = |M|$.

Let M be a matching in the bipartite graph G. Let $\overline{M}$ be the complement of M in G, that is, the set of edges of G that do not belong to M. Let u and v be vertices, where one of u and v is in X and one is in Y. A path γ joining u and v is an *alternating path with respect to the matching M* (for brevity, an *M-alternating path*) provided that the following properties hold:

(1) The first, third, fifth, ... edges of γ do not belong to the matching M (and thus belong to $\overline{M}$).

(2) The second, fourth, sixth, ... edges of γ belong to the matching M.

(3) Neither u nor v meets an edge of the matching M.

Notice that the length of the M-alternating path γ is an odd number $2k + 1$ with $k \geq 0$, and that $k + 1$ of the edges of γ are edges of $\overline{M}$ while k of the edges of γ are edges of M. We introduce further notation as follows:

> M_γ denotes those edges of γ that belong to M, and $\overline{M}_\gamma$ denotes those edges of γ that do not belong to M.

We thus have $|\overline{M}_\gamma| = |M_\gamma| + 1$.

Example. Consider the bipartite graph G pictured in Figure 13.8. The set

$$M = \{\{x_1, y_1\}, \{x_2, y_3\}, \{x_3, y_4\}\}$$

is a matching of three edges. The path

$$\gamma : u = x_4, y_3, x_2, y_1, x_1, y_2 = v$$

is an M-alternating path. We have

$$M_\gamma = \{\{x_2, y_3\}, \{x_1, y_1\}\} \text{ and } \overline{M}_\gamma = \{\{x_4, y_3\}, \{x_2, y_1\}, \{x_1, y_2\}\}.$$

If we remove the edges of M_γ from M and replace them with the edges of $\overline{M}_\gamma$, we obtain a matching

$$M' = (M \setminus M_\gamma) \cup \overline{M}_\gamma = \{\{x_3, y_4\}, \{x_4, y_3\}, \{x_2, y_1\}, \{x_1, y_2\}\}$$

of four edges. □

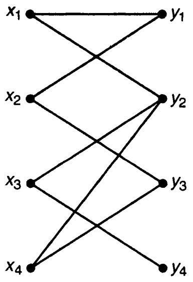

Figure 13.8

As illustrated in the previous example, if M is a matching and there is an M-alternating path γ, then

$$(M \setminus M_\gamma) \cup \overline{M}_\gamma$$

is a matching with one more edge than M, and hence M is not a max-matching. We now show that the converse holds as well; that is, the only way a matching M can fail to be a max-matching is for there to exist an M-alternating path.

Theorem 13.3.1 *Let M be a matching in the bipartite graph G. Then M is a max-matching if and only if there does not exist an M-alternating path.*

Proof. An M-alternating path would give rise to a matching with more edges than M. Thus if M is a max-matching, there cannot exist an M-alternating path.

To establish the converse, we now assume that M is not a max-matching and prove that there exists an M-alternating path. Let M' be a matching satisfying

$$|M'| > |M|.$$

We consider the bipartite graph G^* with the same bipartition as G whose edges are the edges in $(M \setminus M') \cup (M' \setminus M)$. Thus the edges of G^* are those edges which are in either M or M' but not in both. Since $|M'| > |M|$, we have

$$|M' \setminus M| > |M \setminus M'|. \tag{13.12}$$

The bipartite graph G^* has the property that the degree of each of its vertices is at most equal to 2 (each vertex meets at most one edge of $M \setminus M'$ and at most one edge of $M' \setminus M$). This implies that the set of edges of G^* can be partitioned into paths and cycles. In each of the paths and cycles of this partition, the edges alternate between $M \setminus M'$ and $M' \setminus M$. A path in the partition has the property that both its first and last vertices meet only one edge of G^*. These paths and cycles are of four types:

Type 1. A path whose first and last edges are both in $M' \setminus M$ (see Figure 13.9 where in this and the other figures the bold lines denote the edges of M). These paths have odd length and contain one more edge of M' than they do of M. Included among the Type 1 paths are paths with only one edge where this edge is an edge of $M' \setminus M$.

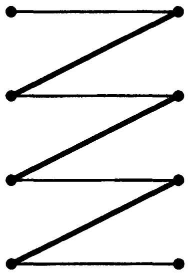

Figure 13.9. Type 1 path

Type 2. A path whose first and last edges are both in $M \setminus M'$ (see Figure 13.10). These paths also have odd length, but they contain one more edge of M than they do of M'.

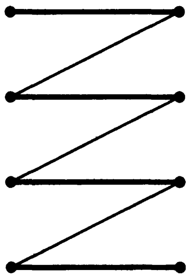

Figure 13.10. Type 2 path

Type 3. A path whose first edge is in $M \setminus M'$ and whose last edge is in $M' \setminus M$ (or vice versa) (see Figure 13.11). These paths have even length and contain as many edges of M as they do of M'.

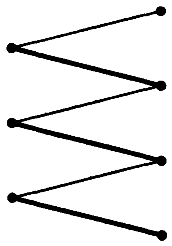

Figure 13.11 Type 3 path

Type 4. A cycle (see Figure 13.12). These cycles have even length and contain as many edges of M as they do of M'.

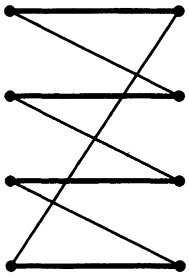

Figure 13.12 Type 4 cycle

There are more edges of $M \setminus M'$ than of $M' \setminus M$ in a path of Type 2, and the same number of edges of $M \setminus M'$ as of $M \setminus M'$ in a path of Type 3 and in a cycle of Type 4. In a path of Type 1, there are more edges of $M' \setminus M$ than of $M \setminus M'$. Since, by (13.12), $M' \setminus M$ has more edges than $M \setminus M'$, there must exist at least one path of Type 1. A path of Type 1 is by definition an M-alternating path. Thus, if a matching M is not a max-matching, there is an M-alternating path. □

Theorem 13.3.1 characterizes max-matchings among all the matchings in a bipartite graph. Its strength lies in the fact that, given a matching M, in order to determine whether M is a max-matching, we need only search for an M-alternating path γ. If we find such a path γ, then, by removing from M those edges of γ that belong to M and replacing them with the edges of γ that do not belong to M, we obtain a matching M' that has more edges than M. If we cannot find an M-alternating path γ, then, by Theorem 13.3.1, M is a max-matching.

The weakness of Theorem 13.3.1 lies in the preceding assertion. After searching for an M-alternating path and not finding one, we need to know that we didn't find one because there wasn't any to be found, not because we didn't look hard enough. We cannot expect to examine all possible paths in order to determine whether among them there is an M-alternating path. Such a task would require, in general, too much time and effort. What we seek is some way of easily certifying that a matching is a max-matching. In other words, we seek an easily verifiable *certification* that a matching is a max-matching. In fact, the covering number $c(G)$ gives such a certification. We call a cover S a *min-cover* provided that $|S| = c(G)$.

Suppose that, in whatever way, we have found a matching M in a bipartite graph G which we think might be a max-matching. If we can find a cover S such that $|M| = |S|$, then M is a max-matching and S is a min-cover. This fact is a consequence of

$$c(G) \leq |S| = |M| \leq \rho(G) \leq c(G), \tag{13.13}$$

implying that $|M| = \rho(G)$ (that is, M is a max-matching), and $|S| = c(G)$ (that is, S is a min-cover). Thus S acts as a certification that there is no matching with a larger number of edges than M.

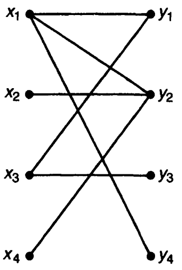

Figure 13.13

Example. Consider the bipartite graph in Figure 13.13. We see that

$$M = \{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}\}$$

is a matching of three edges. The set $S = \{x_1, x_3, y_2\}$ is a cover of three vertices. Hence,

$$3 = |M| \leq \rho(G) = c(G) \leq |S| = 3.$$

We have equality throughout, and hence M is a max-matching, S is a min-cover, and $\rho(G) = c(G) = 3$. □

We now describe our basic flow algorithm as it applies to the problem of determining a max-matching in a bipartite graph. Starting from any known matching M, the algorithm is a systematic search for an M-alternating path. Either (1) the algorithm produces an M-alternating path, and we use the proof of Theorem 13.3.1 to obtain a matching with one more edge than M, or (2) the algorithm fails to produce an M-alternating path but, as we shall see, produces a cover S with $|M| = |S|$, and we thus conclude that M is a max-matching and S is a certification for M (thus the algorithm didn't produce an M-alternating path because no such alternating path existed).

Matching algorithm

Let G be a bipartite graph with bipartition X, Y where $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$. Let M be any matching in G.

(0) Begin by labeling with (*) all vertices in X that do not meet any edge in M and call all such vertices *unscanned.* Go to (1).

(1) If in the previous step, no new label has been given to a vertex of X, then stop.[20] (This means that every vertex in X meets an edge of M. Thus $|X| \leq M$. Since $|M|$ cannot exceed $|X|$, this would mean that M is already a max-matching.) Otherwise go to (2).

(2) While there exists a labeled, but unscanned, vertex of X, select such a vertex, say, x_i, and label with the label (x_i) all vertices in Y joined to x_i by an edge *not belonging to M and not previously labeled.* The vertex x_i is now scanned. If there are no labeled but unscanned vertices, go to (3).

(3) If, in step (2), no new label has been given to a vertex of Y, then stop. Otherwise go to (4).

(4) While there exists a labeled, but unscanned vertex, of Y, select such a vertex, say, y_j, and label with the label (y_j) any vertex of X joined to y_j by an edge *belonging to M and not previously labeled.* The vertex y_j is now scanned. If there are no labeled but unscanned vertices, go to (1).

Since each vertex receives at most one label, and since each vertex is scanned, at most, once, the matching algorithm halts after a finite number of steps. There are two possibilities to consider:

Breakthrough: There is a labeled vertex of Y that does not meet an edge of M.

Nonbreakthrough: The algorithm has come to a halt, and breakthrough has not occurred; that is, each vertex of Y that is labeled also meets some edge of M.

[20]Initially, this can happen only if no vertex gets the label (*).

In the case of breakthrough, the matching algorithm has succeeded in finding an M-alternating path γ. One end vertex of γ is the vertex v of Y, which is labeled but does not meet any edge of M. The other end vertex of γ is a vertex u of X with label (*) (and which therefore does not meet any edge of M). The M-alternating path γ can be constructed by starting at v and working backward through the labels until a vertex u with label (*) is found. In this case, we can use γ to obtain (as in the proof of Theorem 13.3.1) a matching with one more edge than M.

If nonbreakthrough occurs, we shall show that it is because M is a max-matching; that is, according to Theorem 13.3.1, because there isn't any M-alternating path. Thus, breakthrough occurs exactly when M is not a max-matching, and when breakthrough occurs, we have a way of obtaining an M-alternating path and hence a matching with one more edge than M.

Theorem 13.3.2 *Assume that nonbreakthrough has occured in the matching algorithm. Let X^{un} consist of all the unlabeled vertices of X, let Y^{lab} consist of all the labeled vertices of Y, and let $S = X^{un} \cup Y^{lab}$. Then both of the following hold:*

(i) *S is a min-cover of the bipartite graph G;*

(ii) *$|M| = |S|$ and M is a max-matching.*

Proof. We first show that S is a cover by assuming that there is an edge $e = \{x, y\}$, neither of whose vertices belongs to S, and obtaining a contradiction.

Thus, assume that x is in $X \setminus X^{un}$ and y is in $Y \setminus Y^{lab}$ and $e = \{x, y\}$ is an edge. Since x is not in X^{un}, x is labeled; since y is not in Y^{lab}, y is unlabeled. Either e belongs to M or it does not. If e does not belong to M, then, in applying step (2) of the algorithm, y would receive the label (x), a contradiction. We now assume that e belongs to M. Since x meets the edge e of M, it follows from step (0) that the label of x is not (*). Since x is labeled, it follows from the algorithm that x has label (y). (See step (4).) By the algorithm again, vertex y can give label (y) to a vertex of X only if y is already labeled. Since y is not labeled, we have a contradiction again. Since both possibilities lead to a contradiction, we conclude that S is a cover.

We complete the proof of the theorem by showing that $|M| = |S|$. As we have already demonstrated, this equality also implies that S is a min-cover and M is a max-matching. We establish a one-to-one correspondence between the vertices in S and the edges in M, thereby proving $|M| = |S|$. Let y be a vertex in Y^{lab} so that y is labeled. Since Breakthrough has not occurred, y meets an edge of M, and hence exactly one edge of M, say, the edge $\{x, y\}$ of M. By step (4) of the algorithm, x gets the label (y) and hence x is not in X^{un}. Thus, each vertex of Y^{lab} meets an edge of M whose other vertex belongs to $X - X^{un}$. Now consider a vertex x' in X^{un}. Since x' is not labeled, it follows from step (0) that x' meets an edge of M (otherwise x' would have the label (*)), and hence exactly one edge of M, say $\{x', y'\}$ of M. The vertex y' cannot be in Y^{lab} since we previously showed that the unique edge of M meeting a vertex in Y^{lab} has its other vertex in $X - X^{un}$. Thus, we have shown that, for each vertex of $X^{un} \cup Y^{lab}$, there is a unique edge of M containing it and all these edges are distinct. Hence,

$$|S| = |X^{un} \cup Y^{lab}| \leq |M|,$$

and we conclude that $|S| = |M|$. □

We remark that the proof of Theorem 13.3.2 essentially contains another proof of the relation $\rho(G) = c(G)$.

The matching algorithm can be applied to obtain a max-matching in a bipartite graph as follows: We first choose a matching in a greedy fashion—we pick any edge e_1, then any edge e_2 that does not meet e_1, then any edge e_3 that does not meet e_1 or e_2, and continue like this until we run out of choices.[21] We call the resulting matching M^1 and apply the matching algorithm to it. If nonbreakthrough occurs, then, by Theorem 13.3.2, M^1 is a max-matching. If breakthrough occurs, then we obtain a matching M^2 with more edges than M^1. We now apply the matching algorithm to M^2. In this way, we obtain a sequence of matchings $M^1, M^2, M^3, \ldots$, each with more edges than the preceding one. After a finite number of applications of the matching algorithm, we obtain a matching M^k for which the matching algorithm results in nonbreakthrough, and hence M^k is a max-matching.

Example. We determine a max-matching in the bipartite graph G in Figure 13.14. We choose the edges $\{x_2, y_2\}, \{x_3, y_3\}$, and $\{x_4, y_4\}$ and obtain a matching M^1 of size 3. The edges of M^1 are in boldface in Figure 13.13. We now apply the matching algorithm to M^1, and the results as shown in Figure 13.13 are as follows:

(1) Step (0): The vertices x_1, x_5, and x_6, which do not meet an edge of M^1, are labeled (*).

(2) Step (2): We scan x_1, x_5 and x_6, in turn, and label y_3 with (x_1) and y_4 with (x_5). Since all vertices joined to x_6 already have a label, no vertex of Y gets labeled (x_6).

(3) Step (4): We scan the vertices y_3 and y_4, labeled in (ii), and label x_3 with (y_3) and x_4 with (y_4).

(4) Step (2): We scan the vertices x_3 and x_4, labeled in (iii), and label y_2 with (x_3).

[21]Or perhaps we stop because there are no more obvious choices.

(5) Step (4): We scan the vertex y_2, labeled in (iv), and label x_2 with (y_2).

(6) Step (2): We scan the vertex x_2, labeled in (v), and label y_1, y_5 and y_6 with (x_2).

(7) Step (4): We scan the vertices y_1, y_5, and y_6, labeled in (6), and find that no new labels are possible.

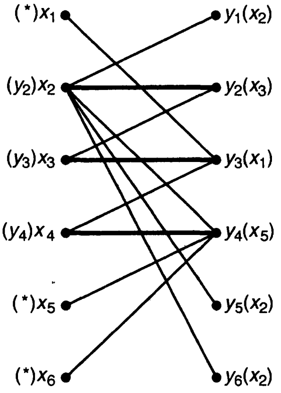

Figure 13.14

The first phase of the algorithm has now come to an end, and since we have labeled a vertex of Y that does not meet an edge of M^1 (in fact, the three vertices y_1, y_5, and y_6 have this property), we have achieved breakthrough.[22] If we trace backward from y_1, using the labels as a guide, we find the M^1-alternating path

$$\gamma : y_1, x_2, y_2, x_3, y_3, x_1.$$

We have

$$M^1_\gamma = \{\{x_2, y_2\}, \{x_3, y_3\}\}$$

and

$$\overline{M^1_\gamma} = \{\{y_1, x_2\}, \{y_2, x_3\}, \{y_3, x_1\}\}.$$

Then

$$\begin{aligned} M^2 &= (M^1 - M^1_\gamma) \cup (\overline{M}^1_\gamma) \\ &= \{\{x_4, y_4\}, \{y_1, x_2\}, \{y_2, x_3\}, \{y_3, x_1\}\} \end{aligned}$$

is a matching of four edges.

[22]The algorithm can be halted as soon as breakthrough is achieved.

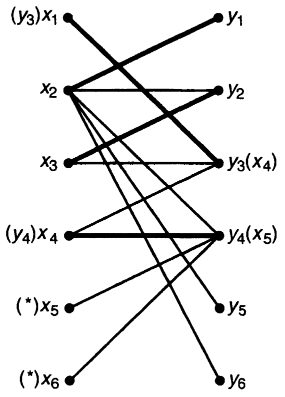

Figure 13.15

We now apply the matching algorithm to M^2. The resulting labeling of the vertices is shown in Figure 13.15. In this case, Breakthrough has not occurred. By Theorem 13.3.2, M^2 is a max-matching of size 4, and the set

$$S = \{x_2, x_3, y_3, y_4\},$$

of size 4, consisting of the unlabeled vertices of X and the labeled vertices of Y, is a min-cover. □

13.4 Exercises

1. Prove Theorem 13.1.2.

2. Prove Theorem 13.1.3.

3. Prove that an orientation of K_n is a transitive tournament if and only if it does not have any directed cycles of length 3.

4. Give an example of a digraph that does not have a closed Eulerian directed trail but whose underlying general graph has a closed Eulerian trail.

5. Prove that a digraph has no directed cycles if and only if its vertices can be labeled from 1 up to n so that the terminal vertex of each arc has a larger label than the initial vertex.

6. Prove that a digraph is strongly connected if and only if there is a closed, directed walk that contains each vertex at least once.

7. Let T be any tournament. Prove that it is possible to change the direction of at most one arc in order to obtain a tournament with a directed Hamilton cycle.

8. Use the proof of Theorem 13.1.5 in order to write an algorithm for determining a Hamilton path in a tournament.

9. Prove that a tournament is strongly connected if and only if it has a directed Hamilton cycle.

10. Prove that every tournament contains a vertex u such that, for every other vertex x, there is a path from u to x of length at most 2.

11. Prove that every graph has the property that it is possible to orient each of its edges so that, for each vertex x, the indegree and outdegree of x differ by at most 1.

12. $*$ Devise an algorithm for constructing a directed Hamilton cycle in a strongly connected tournament.

13. Apply the algorithm in Section 13.1 and determine a strongly connected orientation of the graphs in Figures 11.15 to 11.18.

14. Prove the following generalization of Theorem 13.1.6: Let G be a connected graph. Then, after replacing each bridge $\{a, b\}$ by the two arcs (a, b) and (b, a), one in each direction, it is possible to give the remaining edges of G an orientation so that the resulting digraph is strongly connected.

15. Modify the algorithm for constructing a strongly connected orientation of a bridgeless connected graph in order to accommodate the situation described in Exercise 14.

16. Consider a trader problem in which trader t_1 ranks his item number 1. Prove that, in every core allocation, t_1 gets to keep his own item.

17. Construct an example of a trading problem, with n traders, with the property that, in each core allocation, exactly one trader gets the item he ranks first.

18. Show that, for the trading problem in which the preferences are given by the table

	t_1	t_2	t_3
t_1	2	1	3
t_2	3	2	1
t_3	1	3	2

there are exactly two core allocations. Which of these results from applying the constructive proof of Theorem 13.1.9?

19. Suppose that, in a trading problem, some trader ranks his own item number k. Prove that, in each core allocation, that player obtains an item he ranks no lower than k. (Thus, a player never leaves with an item that he values less than the item he brought to trade.)

20. Prove that, in the core allocation obtained by applying the constructive proof of Theorem 13.1.9, at least one player gets an item he ranks number 1. Show by example that there may be core allocations in which no player gets his first choice.

21. Prove that, in a trading problem, there is a core allocation in which every trader gets the item he ranks number 1 if and only if the digraph D^1 constructed in the proof of Theorem 13.1.9 consists of directed cycles, no two of which have a vertex in common.

22. Construct a core allocation for the trading problem in which the preferences are given by the following table:

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
t_1	2	3	1	4	7	5	6
t_2	1	6	4	3	2	7	5
t_3	2	7	3	5	1	4	6
t_4	3	4	2	7	1	6	5
t_5	1	3	4	2	5	7	6
t_6	2	4	1	5	3	7	6
t_7	7	3	4	2	1	6	5

23. Explicitly write the algorithm for a core allocation that is implicit in the proof of Theorem 13.1.9.

24. Determine a maximum flow and a minimum cut in each of the networks $N = (V, A, s, t, c)$ in Figure 13.16. (The numbers near arcs are their capacities.)

25. Determine the maximum number of pairwise arc-disjoint paths from s to t in the digraphs of the networks in Exercise 24. Verify that the number is maximum by exhibiting an st-separating set with the same number of arcs (cf. Theorem 13.2.4).

26. Consider the network in Figure 13.17, where there are *three* sources s_1, s_2, and s_3 for a certain commodity and *three* targets t_1, t_2, and t_3. Each source has a certain supply of the commodity, and each target has a certain demand for the commodity. These supplies and demands are the numbers in brackets next to the sources and sinks. The supplies are to flow from the sources to the targets,

subject to the flow capacities on each arc. Determine whether all the demands can be met simultaneously with the available supplies. (One possible way to approach this problem is to introduce an auxiliary source s and an auxiliarly target t, arcs from s to each s_i with capacity equal to s_i's supply, and arcs from each t_j to t with capacity equal to t_j's demand, and then find a maximum flow from s to t in the augmented network and check whether all demands are met.)

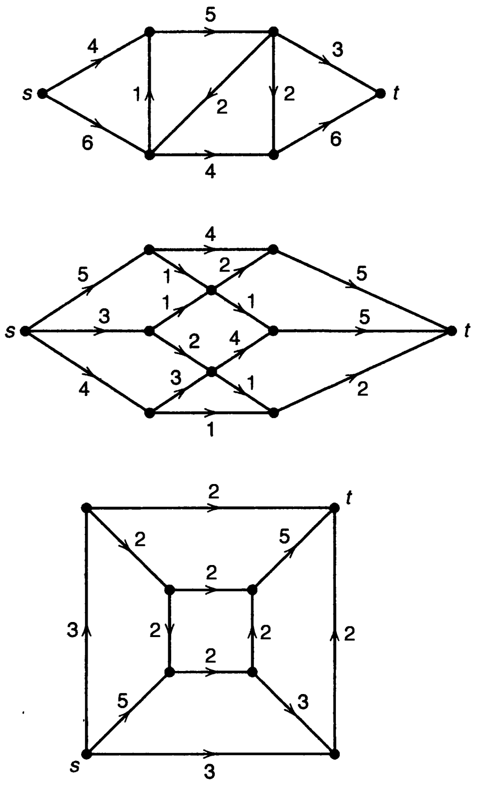

Figure 13.16

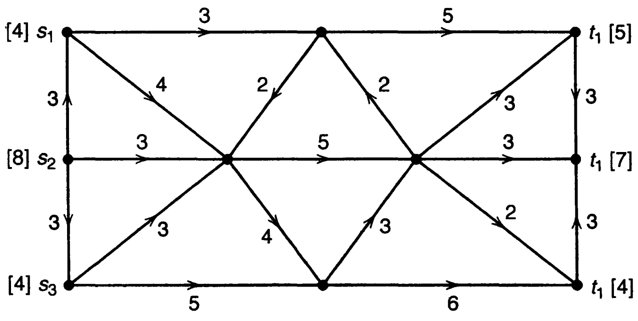

Figure 13.17

27. In Exercise 26, change the supplies at s_1, s_2, and s_3 to a, b, and c, respectively, and determine again whether all the demands can be met simultaneously with the available supplies.

28. * Formulate and prove a theorem that gives necessary and sufficient conditions so that a network with multiple sources and sinks has a flow that simultaneously meets all prescribed demands with available supplies.

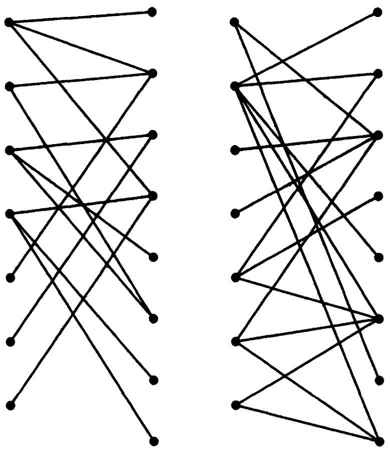

Figure 13.18

29. Use the matching algorithm to determine the largest number of edges in a matching M of the bipartite graphs in Figure 13.18. In each case, find a cover S with $|S| = |M|$.

30. Consider an m-by-n board, with squares alternately colored black and white, where some of the squares have been forbidden. In Chapter 9, we associated with each nonforbidden (free) white square the set of nonforbidden (free) black squares with which it shares an edge. This family of sets was called the domino family of the board. We can also associate with the board a bipartite graph G (the domino bipartite graph of the board) with bipartition X, Y, where X is the set of free white squares and Y is the set of free black squares. There is an edge joining a free white square to a free black square if and only if the two squares share an edge. A matching M of G corresponds to the placement of $|M|$ nonoverlapping dominoes on the board. Use the matching algorithm to determine the largest number of nonoverlapping dominoes that can be placed on the board shown here (that is, $\rho(G)$) and certify why you have the largest number by finding $c(G)$.

				×			
	×		×		×		×
			×	×	×		
	×					×	
×			×				×
	×				×		
			×	×			
		×			×		

31. Consider the set A of the 2^n binary sequences of length n. This exercise concerns the existence of a circular arrangement γ_n of 2^n 0s and 1s, so that the 2^n sequences of n consecutive bits of γ give all of A; that is, are all distinct. Such a circular arrangement is called a *de Bruijn cycle*. For example, if $n = 2$, the circular arrangement $0, 0, 1, 1$ (regarding the first 0 as following the last 1) gives $0, 0$; $0, 1$; $1, 1$; and $1, 0$. For $n = 3$, $0, 0, 0, 1, 0, 1, 1, 1$ (regarded cyclically) is a de Bruijn cycle. Define a digraph Γ_n whose vertices are the 2^{n-1} binary sequences of length $n - 1$. Given two such binary sequences x and y, we put an arc e from x to y, provided that the last $n - 2$ bits of x agree with the first $n - 2$ bits of y, and then we label the arc e with the first bit of x.

   (a) Prove that every vertex of Γ_n has indegree and outdegree equal to 2. Thus, Γ_n has a total of $2 \cdot 2^{n-1} = 2^n$ arcs.

   (b) Prove that Γ_n is strongly connected, and hence Γ_n has a closed Eulerian directed trail (of length 2^n).

   (c) Let $b_1, b_2, \ldots, b_{2^n}$ be the labels of the arcs (considered as a circular arrangement) as we traverse an Eulerian directed trail of Γ_n. Prove that $b_1, b_2, \ldots, b_{2^n}$ is a de Bruijn cycle.

(d) Prove that, given any two vertices x and y of the digraph Γ_n, there is a path from x to y of length at most $n - 1$.