A Course in Combinatorial Optimization

Alexander Schrijver

CWI,
Kruislaan 413,
1098 SJ Amsterdam,
The Netherlands

and

Department of Mathematics,
University of Amsterdam,
Plantage Muidergracht 24,
1018 TV Amsterdam,
The Netherlands.

February 7, 2003

Contents

1. Shortest paths and trees

1.1. Shortest paths with nonnegative lengths

Let $D = (V, A)$ be a directed graph, and let $s, t \in V$. A *path* is a sequence $P = (v_0, a_1, v_1, \dots, a_m, v_m)$ where a_i is an arc from v_{i-1} to v_i for $i = 1, \dots, m$. If $s = v_0$ and $t = v_m$, the vertices s and t are the *starting* and *end vertex* of P, respectively, and P is called an *$s - t$ path*. The *length* of P is m. The *distance* from s to t is the minimum length of any $s - t$ path. (If no $s - t$ path exists, we set the distance from s to t equal to ∞.)

It is not difficult to determine the distance from s to t: Let V_i denote the set of vertices of D at distance i from s. Note that for each i:

(1) V_{i+1} is equal to the set of vertices $v \in V \setminus (V_0 \cup V_1 \cup \dots \cup V_i)$ for which $(u, v) \in A$ for some $u \in V_i$.

This gives us directly an algorithm for determining the sets V_i: we set $V_0 := \{s\}$ and next we determine with rule (1) the sets $V_1, V_2, \dots$ successively, until $V_{i+1} = \emptyset$.

In fact, it gives a linear-time algorithm:

Theorem 1.1. *The algorithm has running time $O(|A|)$.*

Proof. Directly from the description. ▮

In fact the algorithm finds the distance from s to all vertices reachable from s. Moreover, it gives the shortest paths. These can be described by a rooted (directed) tree $T = (V', A')$, with root s, such that V' is the set of vertices reachable in D from s and such that for each $u, v \in V'$, each directed $u - v$ path in T is a shortest $u - v$ path in D.[1]

Indeed, when we reach a vertex t in the algorithm, we store the arc by which t is reached. Then at the end of the algorithm, all stored arcs form a rooted tree with this property.

There is also a trivial min-max relation characterizing the minimum length of an $s - t$ path. To this end, call a subset A' of A an *$s - t$ cut* if $A' = \delta^{\text{out}}(U)$ for some subset U of V satisfying $s \in U$ and $t \notin U$.[2] Then the following was observed by Robacker [1956]:

Theorem 1.2. *The minimum length of an $s - t$ path is equal to the maximum number of pairwise disjoint $s - t$ cuts.*

Proof. Trivially, the minimum is at least the maximum, since each $s - t$ path intersects each $s - t$ cut in an arc. The fact that the minimum is equal to the maximum follows by considering the $s - t$ cuts $\delta^{\text{out}}(U_i)$ for $i = 0, \dots, d - 1$, where d is the distance from s to t and where U_i is the set of vertices of distance at most i from s. ▮

This can be generalized to the case where arcs have a certain 'length'. For any 'length' function $l : A \to \mathbb{Q}_+$ and any path $P = (v_0, a_1, v_1, \dots, a_m, v_m)$, let $l(P)$ be the length of P. That is:

(2) $$l(P) := \sum_{i=1}^{m} l(a).$$

Now the *distance* from s to t (with respect to l) is equal to the minimum length of any $s - t$ path. If no $s - t$ path exists, the distance is $+\infty$.

[1]A *rooted tree*, with *root s*, is a directed graph such that the underlying undirected graph is a tree and such that each vertex $t \neq s$ has indegree 1. Thus each vertex t is reachable from s by a unique directed $s - t$ path.

[2]$\delta^{\text{out}}(U)$ and $\delta^{\text{in}}(U)$ denote the sets of arcs leaving and entering U, respectively.

Again there is an easy algorithm, due to Dijkstra [1959], to find a minimum-length $s-t$ path for all t. Start with $U := V$ and set $f(s) := 0$ and $f(v) = \infty$ if $v \neq s$. Next apply the following iteratively:

(3) Find $u \in U$ minimizing $f(u)$ over $u \in U$. For each $a = (u,v) \in A$ for which $f(v) > f(u) + l(a)$, reset $f(v) := f(u) + l(a)$. Reset $U := U \setminus \{u\}$.

We stop if $U = \emptyset$. Then:

Theorem 1.3. *The final function f gives the distances from s.*

Proof. Let $\text{dist}(v)$ denote the distance from s to v, for any vertex v. Trivially, $f(v) \geq \text{dist}(v)$ for all v, throughout the iterations. We prove that throughout the iterations, $f(v) = \text{dist}(v)$ for each $v \in V \setminus U$. At the start of the algorithm this is trivial (as $U = V$).

Consider any iteration (3). It suffices to show that $f(u) = \text{dist}(u)$ for the chosen $u \in U$. Suppose $f(u) > \text{dist}(u)$. Let $s = v_0, v_1, \ldots, v_k = u$ be a shortest $s-u$ path. Let i be the smallest index with $v_i \in U$.

Then $f(v_i) = \text{dist}(v_i)$. Indeed, if $i = 0$, then $f(v_i) = f(s) = 0 = \text{dist}(s) = \text{dist}(v_i)$. If $i > 0$, then (as $v_{i-1} \in V \setminus U$):

(4) $$f(v_i) \leq f(v_{i-1}) + l(v_{i-1}, v_i) = \text{dist}(v_{i-1}) + l(v_{i-1}, v_i) = \text{dist}(v_i).$$

This implies $f(v_i) \leq \text{dist}(v_i) \leq \text{dist}(u) < f(u)$, contradicting the choice of u. ∎

Clearly, the number of iterations is $|V|$, while each iteration takes $O(|V|)$ time. So the algorithm has a running time $O(|V|^2)$. In fact, by storing for each vertex v the last arc a for which (3) applied we find a rooted tree $T = (V', A')$ with root s such that V' is the set of vertices reachable from s and such that for each $u, v \in V'$, each directed $u-v$ path in T is a shortest $u-v$ path in D.

Thus we have:

Theorem 1.4. *Given a directed graph $D = (V, A)$, $s, t \in V$, and a length function $l : A \to \mathbb{Q}_+$, a shortest $s-t$ path can be found in time $O(|V|^2)$.*

Proof. See above. ∎

For an improvement, see Section 1.2.

A weighted version of Theorem 1.2 is as follows:

Theorem 1.5. *Let $D = (V, A)$ be a directed graph, $s, t \in V$, and let $l : A \to \mathbb{Z}_+$. Then the minimum length of an $s-t$ path is equal to the maximum number k of $s-t$ cuts $C_1, \ldots, C_k$ (repetition allowed) such that each arc a is in at most $l(a)$ of the cuts C_i.*

Proof. Again, the minimum is not smaller than the maximum, since if P is any $s-t$ path and $C_1, \ldots, C_k$ is any collection as described in the theorem:[3]

(5) $$l(P) = \sum_{a \in AP} l(a) \geq \sum_{a \in AP} (\text{ number of } i \text{ with } a \in C_i) \\ = \sum_{i=1}^{k} |C_i \cap AP| \geq \sum_{i=1}^{k} 1 = k.$$

To see equality, let d be the distance from s to t, and let U_i be the set of vertices at distance less than i from s, for $i = 1, \ldots, d$. Taking $C_i := \delta^{\text{out}}(U_i)$, we obtain a collection $C_1, \ldots, C_d$ as required.

[3] AP denotes the set of arcs traversed by P

▮

Application 1.1: Shortest path. Obviously, finding a shortest route between cities is an example of a shortest path problem. The length of a connection need not be the geographical distance. It might represent the time or energy needed to make the connection. It might cost more time or energy to go from A to B than from B to A. This might be the case, for instance, when we take differences of height into account (when routing trucks), or air and ocean currents (when routing airplanes or ships).

Moreover, a route for an airplane flight between two airports so that a minimum amount of fuel is used, taking weather, altitude, velocities, and air currents into account, can be found by a shortest path algorithm (if the problem is appropriately discretized — otherwise it is a problem of 'calculus of variations'). A similar problem occurs when finding the optimum route for boring say an underground railway tunnel.

Application 1.2: Dynamic programming. A company has to perform a job that will take 5 months. For this job a varying number of extra employees is needed:

(6)

month	number of extra employees needed
1	b_1=10
2	b_2=7
3	b_3=9
4	b_4=8
5	b_5=11

Recruiting and instruction costs DFL 800 per employee, while stopping engagement costs DFL 1200 per employee. Moreover, the company has costs of DFL 1600 per month for each employee that is engaged above the number of employees needed that month. The company now wants to decide what is the number of employees to be engaged so that the total costs will be as low as possible.

Clearly, in the example in any month i, the company should have at least b_i and at most 11 extra employees for this job. To solve the problem, make a directed graph $D = (V, A)$ with

(7)
$$\begin{aligned} V &:= \{(i,x) \mid i = 1,\ldots,5; b_i \leq x \leq 11\} \cup \{(0,0),(6,0)\}, \\ A &:= \{((i,x),(i+1,y)) \in V \times V \mid i = 0,\ldots,5\}. \end{aligned}$$

(Figure 1.1).

At the arc from (i,x) to $(i+1,y)$ we take as length the sum of

(8)
(i) the cost of starting or stopping engagement when passing from x to y employees (this is equal to $8(y-x)$ if $y \geq x$ and to $12(x-y)$ if $y < x$);
(ii) the cost of keeping the surplus of employees in month $i+1$ (that is, $16(y - b_{i+1})$)

(taking DFL 100 as unit).

Now the shortest path from $(0,0)$ to $(6,0)$ gives the number of employees for each month so that the total cost will be minimized. Finding a shortest path is thus a special case of *dynamic programming.*

Exercises

1.1. Solve the dynamic programming problem in Application 1.2 with Dijkstra's method.

1.2. Speeding up Dijkstra's algorithm with heaps

For dense graphs, a running time bound of $O(|V|^2)$ for a shortest path algorithm is best possible, since one must inspect each arc. But if $|A|$ is asymptotically smaller than $|V|^2$, one may expect faster methods.

In Dijkstra's algorithm, we spend $O(|A|)$ time on updating the values $f(u)$ and $O(|V|^2)$ time on finding a $u \in U$ minimizing $f(u)$. As $|A| \leq |V|^2$, a decrease in the running time bound requires a speed-up in finding a u minimizing $f(u)$.

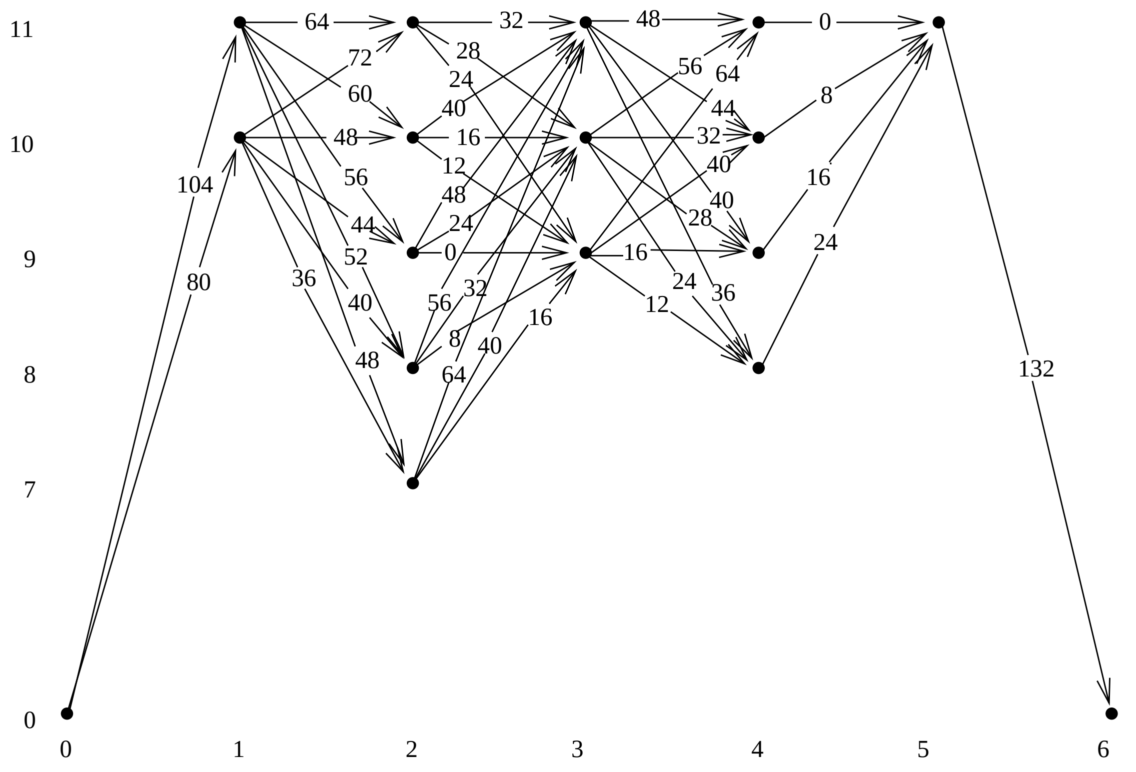


Figure 1.1

A way of doing this is based on storing the u in some order so that a u minimizing $f(u)$ can be found quickly and so that it does not take too much time to restore the order if we delete a minimizing u or if we decrease some $f(u)$.

This can be done by using a 'heap', which is a rooted forest (U, F) on U, with the property that if $(u, v) \in F$ then $f(u) \leq f(v)$.[4] So at least one of the roots minimizes $f(u)$.

Let us first consider the 2-*heap*. This can be described by an ordering $u_1, \ldots, u_n$ of the elements of U such that if $i = \lfloor \frac{j}{2} \rfloor$ then $f(u_i) \leq f(u_j)$. The underlying rooted forest is in fact a rooted tree: its arcs are the pairs (u_i, u_j) with $i = \lfloor \frac{j}{2} \rfloor$.

In a 2-heap, one easily finds a u minimizing $f(u)$: it is the root u_1. The following theorem is basic for estimating the time needed for updating the 2-heap:

Theorem 1.6. *If u_1 is deleted or if some $f(u_i)$ is decreased, the 2-heap can be restored in time $O(\log p)$, where p is the number of vertices.*

Proof. To remove u_1, perform the following 'sift-down' operation. Reset $u_1 := u_n$ and $n := n - 1$. Let $i = 1$. While there is a $j \leq n$ with $2i + 1 \leq j \leq 2i + 2$ and $f(u_j) < f(u_i)$, choose one with smallest $f(u_j)$, swap u_i and u_j, and reset $i := j$.

If $f(u_i)$ has decreased perform the following 'sift-up' operation. While $i > 0$ and $f(u_j) > f(u_i)$ for $j := \lfloor \frac{i-1}{2} \rfloor$, swap u_i and u_j, and reset $i := j$. The final 2-heap is as required.

Clearly, these operations give 2-heaps as required, and can be performed in time $O(\log |U|)$. ▮

[4]A *rooted forest* is an acyclic directed graph $D = (V, A)$ such that each vertex has indegree at most 1. The vertices of indegree 0 are called the *roots* of D. If $(u, v) \in A$, then u is called the *parent* of v and v is called a *child* of u.
If the rooted forest has only one root, it is a *rooted tree*.

This gives the result of Johnson [1977]:

Corollary 1.6a. *Given a directed graph $D = (V, A)$, $s, t \in V$ and a length function $l : A \to \mathbb{Q}_+$, a shortest $s - t$ path can be found in time $O(|A| \log |V|)$.*

Proof. Since the number of times a minimizing vertex u is deleted and the number of times a value $f(u)$ is decreased is at most $|A|$, the theorem follows from Theorem 1.6. ▮

Dijkstra's algorithm has running time $O(|V|^2)$, while Johnson's heap implementation gives a running time of $O(|A| \log |V|)$. So one is not uniformly better than the other.

If one inserts a 'Fibonacci heap' in Dijkstra's algorithm, one gets a shortest path algorithm with running time $O(|A| + |V| \log |V|)$, as was shown by Fredman and Tarjan [1984].

A *Fibonacci forest* is a rooted forest (V, A), so that for each $v \in V$ the children of v can be ordered in such a way that the ith child has at least $i - 2$ children. Then:[5]

Theorem 1.7. *In a Fibonacci forest (V, A), each vertex has at most $1 + 2 \log |V|$ children.*

Proof. For any $v \in V$, let $\sigma(v)$ be the number of vertices reachable from v. We show that $\sigma(v) \geq 2^{(d^{\text{out}}(v)-1)/2}$, which implies the theorem.[6]

Let $k := d^{\text{out}}(v)$ and let v_i be the ith child of v (for $i = 1, \ldots, k$). By induction, $\sigma(v_i) \geq 2^{(d^{\text{out}}(v_i)-1)/2} \geq 2^{(i-3)/2}$, as $d^{\text{out}}(v_i) \geq i - 2$. Hence $\sigma(v) = 1 + \sum_{i=1}^{k} \sigma(v_i) \geq 1 + \sum_{i=1}^{k} 2^{(i-3)/2} = 2^{(k-1)/2} + 2^{(k-2)/2} + \frac{1}{2} - \frac{1}{2}\sqrt{2} \geq 2^{(k-1)/2}$. ▮

Now a *Fibonacci heap* consists of a Fibonacci forest (U, F), where for each $v \in U$ the children of v are ordered so that the ith child has at least $i - 2$ children, and a subset T of U with the following properties:

(9)
(i) if $(u, v) \in F$ then $f(u) \leq f(v)$;
(ii) if v is the ith child of u and $v \notin T$ then v has at least $i - 1$ children;
(iii) if u and v are two distinct roots, then $d^{\text{out}}(u) \neq d^{\text{out}}(v)$.

So by Theorem 1.7, (9)(iii) implies that there exist at most $2 + 2 \log |U|$ roots.

The Fibonacci heap will be described by the following data structure:

(10)
(i) for each $u \in U$, a doubly linked list C_u of children of u (in order);
(ii) a function $p : U \to U$, where $p(u)$ is the parent of u if it has one, and $p(u) = u$ otherwise;
(iii) the function $d^{\text{out}} : U \to \mathbb{Z}_+$;
(iv) a function $b : \{0, \ldots, t\} \to U$ (with $t := 1 + \lfloor 2 \log |V| \rfloor$) such that $b(d^{\text{out}}(u)) = u$ for each root u;
(v) a function $l : U \to \{0, 1\}$ such that $l(u) = 1$ if and only if $u \in T$.

Theorem 1.8. *When finding and deleting n times a u minimizing $f(u)$ and decreasing m times the value $f(u)$, the structure can be updated in time $O(m + p + n \log p)$, where p is the number of vertices in the initial forest.*

[5] $d^{\text{out}}(v)$ and $d^{\text{in}}(v)$ denote the outdegree and indegree of v.

[6] In fact, $\sigma(v) \geq F(d^{\text{out}}(v))$, where $F(k)$ is the kth Fibonacci number, thus explaining the name Fibonacci forest.

Proof. Indeed, a u minimizing $f(u)$ can be identified in time $O(\log p)$, since we can scan $f(b(i))$ for $i = 0, \ldots, t$. It can be deleted as follows. Let $v_1, \ldots, v_k$ be the children of u. First delete u and all arcs leaving u from the forest. In this way, $v_1, \ldots, v_k$ have become roots, of a Fibonacci forest, and conditions (9)(i) and (ii) are maintained. To repair condition (9)(iii), do for each $r = v_1, \ldots, v_k$ the following:

(11)
```
*repair*$(r)$:
if $d^{\text{out}}(r) = d^{\text{out}}(s)$ for some root $s \neq r$, then:
{if $f(r) \leq f(s)$, add $s$ as last child of $r$ and repair$(r)$;
otherwise, add $r$ as last child of $s$ and repair$(s)$}.
```

Note that conditions (9)(i) and (ii) are maintained, and that the existence of a root $s \neq r$ with $d^{\text{out}}(r) = d^{\text{out}}(s)$ can be checked with the functions b, d^{out}, and p. (During the process we update the data structure.)

If we decrease the value $f(u)$ for some $u \in U$ we apply the following to u:

(12)
```
*make root*$(u)$:
if $u$ has a parent, $v$ say, then:
{delete arc $(v, u)$ and repair$(u)$;
if $v \notin T$, add $v$ to $T$; otherwise, remove $v$ from $T$ and make root$(v)$}.
```

Now denote by incr(..) and decr(..) the number of times we increase and decrease .. , respectively. Then:

(13)
$$\begin{aligned}&\text{number of calls of make root} = \text{decr}(f(u)) + \text{decr}(T)\\ &\leq \text{decr}(f(u)) + \text{incr}(T) + p \leq 2\text{decr}(f(u)) + p = 2m + p,\end{aligned}$$

since we increase T at most once after we have decreased some $f(u)$.

This also gives, where R denotes the set of roots:

(14)
$$\begin{aligned}&\text{number of calls of repair} = \text{decr}(F) + \text{decr}(R)\\ &\leq \text{decr}(F) + \text{incr}(R) + p = 2\text{decr}(F) + p\\ &\leq 2(n \log p + \text{number of calls of make root}) + p \leq 2(n \log p + 2m + p) + p.\end{aligned}$$

Since deciding calling make root or repair takes time $O(1)$ (by the data structure), we have that the algorithm takes time $O(m + p + n \log p)$. ∎

As a consequence one has:

Corollary 1.8a. *Given a directed graph $D = (V, A)$, $s, t \in V$ and a length function $l : A \to \mathbb{Q}_+$, a shortest $s - t$ path can be found in time $O(|A| + |V| \log |V|)$.*

Proof. Directly from the description of the algorithm. ∎

1.3. Shortest paths with arbitrary lengths

If lengths of arcs may take negative values, it is not always the case that a shortest path exists. If the graph has a directed circuit of negative length, then we can obtain $r - s$ paths of arbitrary small negative length (for appropriate r and s).

However, it can be shown that if there are no directed circuits of negative length, then for each choice of r and s there exists a shortest $r - s$ path (if there exists at least one $r - s$ path).

Theorem 1.9. *Let each directed circuit have nonnegative length. Then for each pair r, s of vertices for which there exists at least one $r - s$ path, there exists a shortest $r - s$ path. In fact, there exists a shortest $r - s$ path that is simple.*

Proof. Clearly, if there exists an $r - s$ path, there exists a simple $r - s$ path. Hence there exists also a shortest simple path P, that is, a simple $r - s$ path that has minimum length *among all simple $r - s$ paths.* This follows from the fact that there exist only finitely many simple paths.

We show that P is shortest among *all* $r - s$ paths. Let P have length L. Suppose there exists an $r - s$ path Q of length less than L. Choose such a Q with a minimum number of arcs. Since Q is not simple (as it has length less than L), Q contains a directed circuit C. Let Q' be the path obtained from Q by removing C. As $l(C) \geq 0$, $l(Q') = l(Q) - l(C) \leq l(Q) < L$. So Q' is another $r - s$ path of length less than L, however with a smaller number of arcs than Q. This contradicts the assumption that Q has a minimum number of arcs. ∎

In particular, it follows that,

(15) if there are no directed circuits of negative length, there is a shortest path traversing at most $|V| - 1$ arcs.

Also in this case there is an easy algorithm, the *Bellman-Ford method* (Bellman [1958], Ford [1956]), determining a shortest $r - s$ path.

Let $n := |V|$. The algorithm calculates functions $f_0, f_1, f_2, \ldots, f_n : V \to \mathbb{R} \cup \{\infty\}$ successively by the following rule:

(16) (i) Put $f_0(r) := 0$ and $f_0(v) := \infty$ for all $v \in V \setminus \{r\}$.

(ii) For $k < n$, if f_k has been found, put

$$f_{k+1}(v) := \min\{f_k(v), \min_{(u,v)\in A} (f_k(u) + l(u, v))\}$$

for all $v \in V$.

Then $f_n(v)$ is equal to the length of a shortest $r - v$ path, for each $v \in V$. (If there is no $r - v$ path at all, $f_n(v) = \infty$.)

This follows directly from the following theorem:

Theorem 1.10. *For each $k = 0, \ldots, n$ and for each $v \in V$,*

(17) $f_k(v) = \min\{l(P) \ |P \text{ is an } r - v \text{ path traversing at most } k \text{ arcs}\}$.

Proof. By induction on k from (16). ∎

So the above method gives us the length of a shortest $r - s$ path. It is not difficult to derive a method finding an explicit shortest $r - s$ path. To this end, determine parallel to the functions $f_0, \ldots, f_n$, functions

(18) $g_0, \ldots, g_n : V \to \{P \mid P \text{ path}\} \cup \{\infty\}$

as follows:

(19) (i) Put $g_0(r) := (r)$ and $g_0(v) := \infty$ for all $v \in V \setminus \{r\}$.

(ii) For $k < n$, if g_k has been found, put $g_{k+1}(v) := g_k(v)$ if $f_{k+1}(v) = f_k(v)$, and put $g_{k+1}(v) := (g_k(u), (u, v), v)$ if $f_{k+1}(v) = f_k(u) + l(u, v)$ for some arc (u, v).

Then $g_n(v)$ is a shortest $r-v$ path.

This implies:

Corollary 1.10a. *Given a directed graph $D=(V,A)$, $s,t\in V$ and a length function $l:A\to\mathbb{Q}$, such that D has no negative-length directed circuit, a shortest $s-t$ path can be found in time $O(|V||A|)$.*

Proof. Directly from the description of the algorithm. ∎

Application 1.3: Knapsack problem. Suppose we have a knapsack with a volume of 8 liter and a number of articles $1,2,3,4,5$. Each of the articles has a certain volume and a certain value:

(20)

article	volume	value
1	5	4
2	3	7
3	2	3
4	2	5
5	1	4

So we cannot take all articles in the knapsack and we have to make a selection. We want to do this so that the total value of articles taken into the knapsack is as large as possible.

We can describe this problem as one of finding x_1,x_2,x_3,x_4,x_5 such that:

(21) $x_1,x_2,x_3,x_4,x_5\in\{0,1\}$,
$5x_1+3x_2+2x_3+2x_4+x_5\leq 8$,
$4x_1+7x_2+3x_3+5x_4+4x_5$ is as large as possible.

We can solve this problem with the shortest path method as follows. Make a directed graph in the following way:

There are vertices (i,x) for $0\leq i\leq 6$ and $0\leq x\leq 8$ and there is an arc from $(i-1,x)$ to (i,y) if $y=x$ or $y=x+a_i$ (where a_i is the volume of article i) if $i\leq 5$ and there are arcs from each $(5,x)$ to $(6,8)$. We have deleted in the picture all vertices and arcs that do not belong to any directed path from $(0,0)$.

The length of arc $((i-1,x),(i,y))$ is equal to 0 if $y=x$ and to $-c_i$ if $y=x+a_i$ (where c_i denotes the value of i). Moreover, all arcs ending in $(6,8)$ have length 0.

Now a shortest path from $(0,0)$ to $(6,8)$ gives us the optimal selection.

Application 1.4: PERT-CPM. For building a house certain activities have to be executed. Certain activities have to be done before other and every activity takes a certain number of days:

(22)

	activity	days needed	to be done before activity #
1.	groundwork	2	2
2.	foundation	4	3
3.	building walls	10	4,6,7
4.	exterior plumbing	4	5,9
5.	interior plumbing	5	10
6.	electricity	7	10
7.	roof	6	8
8.	finishing off outer walls	7	9
9.	exterior painting	9	14
10.	panelling	8	11,12
11.	floors	4	13
12.	interior painting	5	13
13.	finishing off interior	6	
14.	finishing off exterior	2	

We introduce two dummy activities 0 (start) and 15 (completion), each taking 0 days, where activity 0 has to be performed before all other activities and 15 after all other activities.

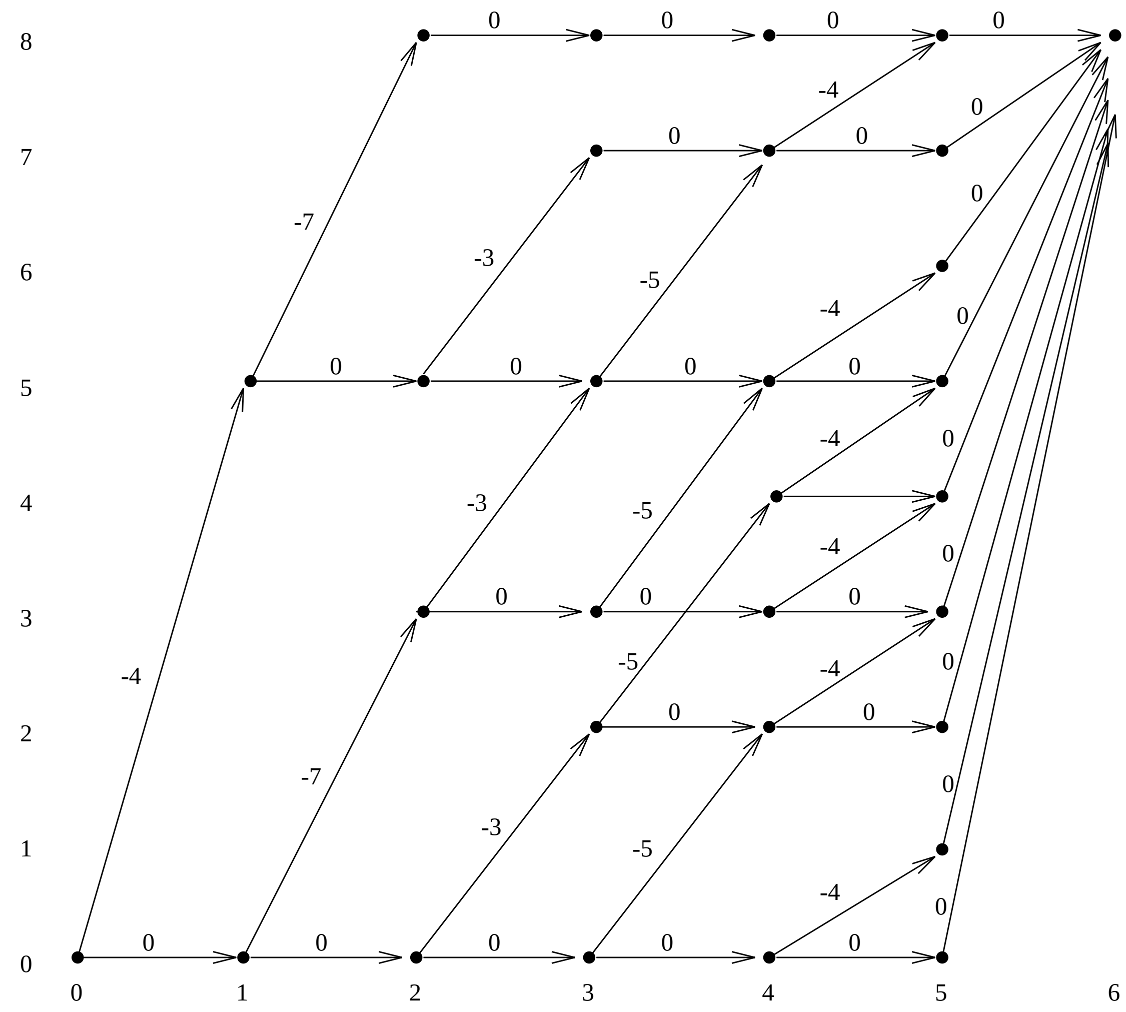


Figure 1.2

The project can be represented by a directed graph D with vertices $0, 1, \ldots, 14, 15$, where there is an arc from i to j if i has to be performed before j. The length of arc (i, j) will be the number t_i of days needed to perform activity i. This graph with length function is called the *project network*.

Now a *longest* path from 0 to 15 gives the minimum number of days needed to build the house. Indeed, if l_i denotes the length of a longest path from 0 to i, we can start activity i on day l_i. If activity j has been done after activity i, then $l_j \geq l_i + t_i$ by definition of longest path. So there is sufficient time for completing activity i and the schedule is practically feasible. That is, there is the following min-max relation:

(23) the minimum number of days needed to finish the project is equal to the maximum length of a path in the project network.

A longest path can be found with the Bellman-Ford method, as it is equivalent to a shortest path when we replace each length by its opposite. Note that D should not have any directed circuits since otherwise the whole project would be infeasible.

So the project network helps in planning the project and is the basis of the so-called 'Program Evaluation and Review Technique' (PERT). (Actually, one often represents activities by arcs instead of vertices, giving a more complicated way of defining the graph.)

Any longest path from 0 to 15 gives the minimum number of days needed to complete the project. Such a path is called a *critical path* and gives us the bottlenecks in the project. It tells us which activities should

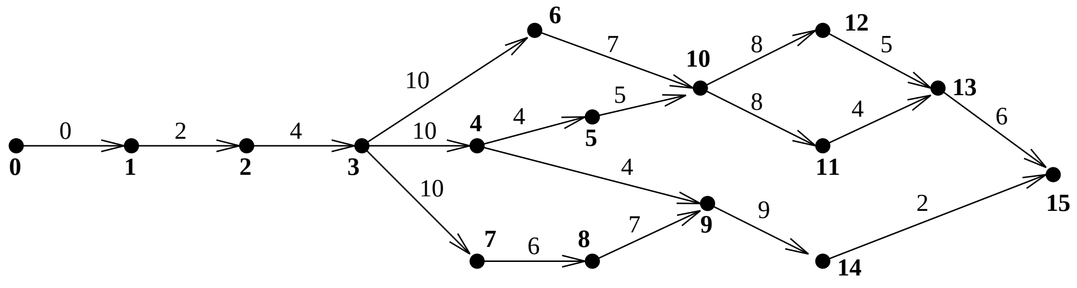


Figure 1.3

be controlled carefully in order to meet a deadline. At least one of these activities should be sped up if we wish to complete the project faster. This is the basis of the 'Critical Path Method' (CPM).

Application 1.5: Price equilibrium. A small example of an economical application is as follows. Consider a number of remote villages, say B, C, D, E and F. Certain pairs of villages are connected by routes (like in Figure 1.4).

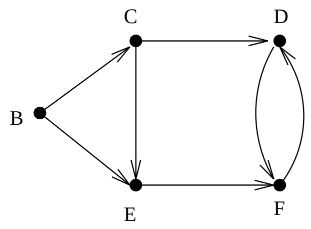


Figure 1.4

If villages X and Y are connected by a route, let $k_{X,Y}$ be the cost of transporting one liter of oil from X to Y.

At a certain day, one detects an oil well in village B, and it makes oil freely available in village B. Now one can follow how the oil price will develop, assuming that no other oil than that from the well in B is available and that only once a week there is contact between adjacent villages.

It will turn out that the oil prices in the different villages will follow the iterations in the Bellman-Ford algorithm. Indeed in week 0 (the week in which the well was detected) the price in B equals 0, while in all other villages the price is ∞, since there is simply no oil available yet.

In week 1, the price in B equals 0, the price in any village Y adjacent to B is equal to $k_{B,Y}$ per liter and in all other villages it is still ∞.

In week $i+1$ the liter price $p_{i+1,Y}$ in any village Y is equal to the minimum value of $p_{i,Y}$ and all $p_{i,X} + k_{X,Y}$ for which there is a connection from X to Y.

There will be price equilibrium if for each village Y one has:

(24) it is not cheaper for the inhabitants of Y to go to an adjacent village X and to transport the oil from X to Y.

Moreover, one has the min-max relation for each village Y:

(25) the maximum liter price in village Y is equal to the the minimum length of a path in the graph from B to Y

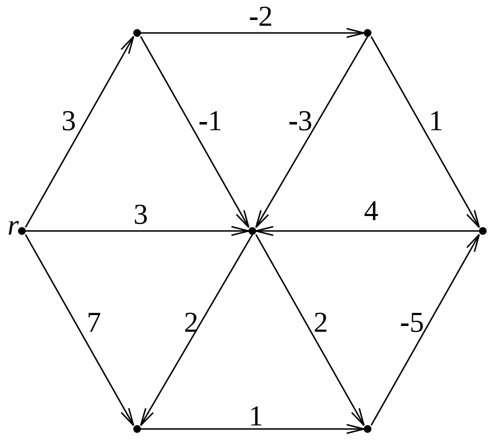
taking $k_{X,Y}$ as length function.

A comparable, but less spatial example is: the vertices of the graph represent oil products (instead of villages) and $k_{X,Y}$ denotes the cost per unit of transforming oil product X to oil product Y. If oil product B is free, one can determine the costs of the other products in the same way as above.

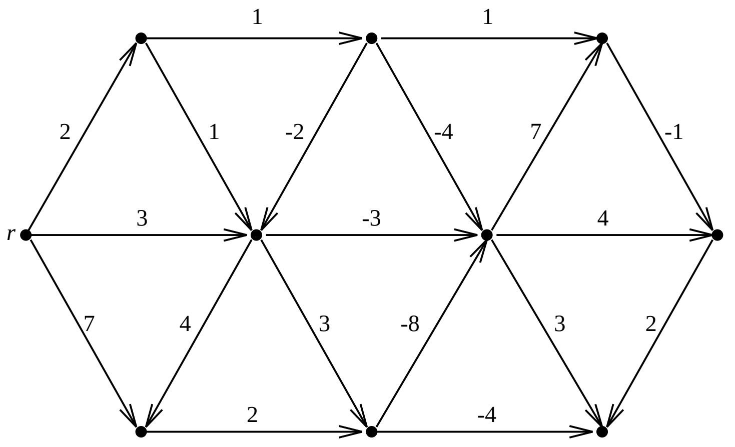
Exercises

1.2. Find with the Bellman-Ford method shortest paths from r to each of the other vertices in the following graphs (where the numbers at the arcs give the length):

(i)



(ii)



1.3. Let be given the distance table:

	to:	A	B	C	D	E	F	G
from:	A	0	1	∞	∞	∞	2	12
	B	∞	0	∞	∞	∞	∞	∞
	C	∞	-15	0	4	8	∞	∞
	D	∞	∞	4	0	∞	∞	-2
	E	∞	∞	∞	4	0	∞	∞
	F	∞	∞	∞	9	3	0	12
	G	∞	-12	2	3	-1	-4	0

A distance ∞ from X to Y should be interpreted as no direct route existing from X to Y. Determine with the Bellman-Ford method the distance from A to each of the other cities.

1.4. Solve the knapsack problem of Application 1.3 with the Bellman-Ford method.

1.5. Describe an algorithm that tests if a given directed graph with length function contains a directed circuit of negative length.

1.6. Let $D = (V, A)$ be a directed graph and let r and s be vertices of D. Show that the minimum number of arcs in an $r-s$ path is equal to the maximum value of $\phi(s) - \phi(r)$, where ϕ ranges over all functions $\phi : V \to \mathbb{Z}$ such that $\phi(w) - \phi(v) \leq 1$ for each arc (v, w).

1.4. Minimum spanning trees

Let $G = (V, E)$ be a connected graph and let $l : E \to \mathbb{R}$ be a function, called the *length* function. For any subset F of E, the *length* $l(F)$ of F is, by definition:

$$l(F) := \sum_{e \in F} l(e). \tag{26}$$

In this section we consider the problem of finding a spanning tree in G of minimum length. There is an easy algorithm for finding a minimum-length spanning tree, essentially due to Borůvka [1926]. There are a few variants. The first one we discuss is sometimes called the *Dijkstra-Prim method* (after Prim [1957] and Dijkstra [1959]).

Choose a vertex $v_1 \in V$ arbitrarily. Determine edges $e_1, e_2 \ldots$ successively as follows. Let $U_1 := \{v_1\}$. Suppose that, for some $k \geq 0$, edges $e_1, \ldots, e_k$ have been chosen, spanning a tree on the set U_k. Choose an edge $e_{k+1} \in \delta(U_k)$ that has minimum length among all edges in $\delta(U_k)$.[7] Let $U_{k+1} := U_k \cup e_{k+1}$.

By the connectedness of G we know that we can continue choosing such an edge until $U_k = V$. In that case the selected edges form a spanning tree T in G. This tree has minimum length, which can be seen as follows.

Call a forest F *greedy* if there exists a minimum-length spanning tree T of G that contains F.

Theorem 1.11. *Let F be a greedy forest, let U be one of its components, and let $e \in \delta(U)$. If e has minimum length among all edges in $\delta(U)$, then $F \cup \{e\}$ is again a greedy forest.*

Proof. Let T be a minimum-length spanning tree containing F. Let P be the unique path in T between the end vertices of e. Then P contains at least one edge f that belongs to $\delta(U)$. So $T' := (T \setminus \{f\}) \cup \{e\}$ is a tree again. By assumption, $l(e) \leq l(f)$ and hence $l(T') \leq l(T)$. Therefore, T' is a minimum-length spanning tree. As $F \cup \{e\} \subseteq T'$, it follows that $F \cup \{e\}$ is greedy. ▮

Corollary 1.11a. *The Dijkstra-Prim method yields a spanning tree of minimum length.*

Proof. It follows inductively with Theorem 1.11 that at each stage of the algorithm we have a greedy forest. Hence the final tree is greedy — equivalently, it has minimum length. ▮

In fact one may show:

Theorem 1.12. *Implementing the Dijkstra-Prim method using Fibonacci heaps gives a running time of $O(|E| + |V| \log |V|)$.*

Proof. The Dijkstra-Prim method is similar to Dijkstra's method for finding a shortest path. Throughout the algorithm, we store at each vertex $v \in V \setminus U_k$, the length $f(v)$ of a shortest edge $\{u, v\}$ with $u \in U_k$, organized as a Fibonacci heap. A vertex u_{k+1} to be added to U_k to form U_{k+1} should be identified and removed from the Fibonacci heap. Moreover, for each edge e connecting u_{k+1} and some $v \in V \setminus U_{k+1}$, we should update $f(v)$ if the length of $u_{k+1}v$ is smaller than $f(v)$.

Thus we find and delete $\leq |V|$ times a u minimizing $f(u)$ and we decrease $\leq |E|$ times a value $f(v)$. Hence by Theorem 1.8 the algorithm can be performed in time $O(|E| + |V| \log |V|)$. ▮

The Dijkstra-Prim method is an example of a so-called *greedy* algorithm. We construct a spanning tree by throughout choosing an edge that seems the best at the moment. Finally we get a minimum-length spanning tree. Once an edge has been chosen, we never have to replace it by another edge (no 'back-tracking').

[7] $\delta(U)$ is the set of edges e satisfying $|e \cap U| = 1$.

There is a slightly different method of finding a minimum-length spanning tree, *Kruskal's* method (Kruskal [1956]). It is again a greedy algorithm, and again iteratively edges $e_1, e_2, \ldots$ are chosen, but by some different rule.

Suppose that, for some $k \geq 0$, edges $e_1, \ldots, e_k$ have been chosen. Choose an edge e_{k+1} such that $\{e_1, \ldots, e_k, e_{k+1}\}$ forms a forest, with $l(e_{k+1})$ as small as possible. By the connectedness of G we can (starting with $k = 0$) iterate this until the selected edges form a spanning tree of G.

Corollary 1.12a. *Kruskal's method yields a spanning tree of minimum length.*

Proof. Again directly from Theorem 1.11. ▮

In a similar way one finds a *maximum*-length spanning tree.

Application 1.6: Minimum connections. There are several obvious practical situations where finding a minimum-length spanning tree is important, for instance, when designing a road system, electrical power lines, telephone lines, pipe lines, wire connections on a chip. Also when clustering data say in taxonomy, archeology, or zoology, finding a minimum spanning tree can be helpful.

Application 1.7: The maximum reliability problem. Often in designing a network one is not primarily interested in minimizing length, but rather in maximizing 'reliability' (for instance when designing energy or communication networks). Certain cases of this problem can be seen as finding a *maximum* length spanning tree, as was observed by Hu [1961]. We give a mathematical description.

Let $G = (V, E)$ be a graph and let $s : E \to \mathbb{R}_+$ be a function. Let us call $s(e)$ the *strength* of edge e. For any path P in G, the *reliability* of P is, by definition, the minimum strength of the edges occurring in P. The *reliability* $r_G(u, v)$ of two vertices u and v is equal to the maximum reliability of P, where P ranges over all paths from u to v.

Let T be a spanning tree of maximum strength, i.e., with $\sum_{e \in ET} s(e)$ as large as possible. (Here ET is the set of edges of T.) So T can be found with any maximum spanning tree algorithm.

Now T has the same reliability as G, for each pair of vertices u, v. That is:

$$r_T(u, v) = r_G(u, v) \text{ for each } u, v \in V. \tag{27}$$

We leave the proof as an exercise (Exercise 1.11).

Exercises

1.7. Find, both with the Dijkstra-Prim algorithm and with Kruskal's algorithm, a spanning tree of minimum length in the graph in Figure 1.5.

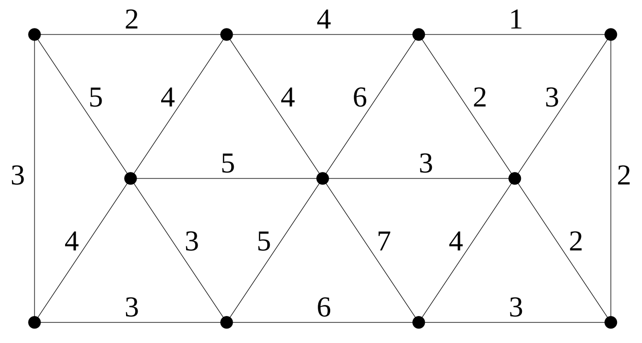


Figure 1.5

1.8. Find a spanning tree of minimum length between the cities given in the following distance table:

	Ame	Ams	Ape	Arn	Ass	BoZ	Bre	Ein	Ens	s-G	Gro	Haa	DH	s-H	Hil	Lee	Maa	Mid	Nij	Roe	Rot	Utr	Win	Zut	Zwo
Amersfoort	0	47	47	46	139	123	86	111	114	81	164	67	126	73	18	147	190	176	63	141	78	20	109	65	70
Amsterdam	47	0	89	92	162	134	100	125	156	57	184	20	79	87	30	132	207	175	109	168	77	40	151	107	103
Apeldoorn	47	89	0	25	108	167	130	103	71	128	133	109	154	88	65	129	176	222	42	127	125	67	66	22	41
Arnhem	46	92	25	0	132	145	108	78	85	116	157	112	171	63	64	154	151	200	17	102	113	59	64	31	66
Assen	139	162	108	132	0	262	225	210	110	214	25	182	149	195	156	68	283	315	149	234	217	159	143	108	69
Bergen op Zoom	123	134	167	145	262	0	37	94	230	83	287	124	197	82	119	265	183	59	128	144	57	103	209	176	193
Breda	86	100	130	108	225	37	0	57	193	75	250	111	179	45	82	228	147	96	91	107	49	66	172	139	156
Eindhoven	111	125	103	78	210	94	57	0	163	127	235	141	204	38	107	232	100	153	61	50	101	91	142	109	144
Enschede	114	156	71	85	110	230	193	163	0	195	135	176	215	148	132	155	236	285	102	187	192	134	40	54	71
's-Gravenhage	81	57	128	116	214	83	75	127	195	0	236	41	114	104	72	182	162	124	133	177	26	61	180	146	151
Groningen	164	184	133	157	25	287	250	235	135	236	0	199	147	220	178	58	308	340	174	259	242	184	168	133	94
Haarlem	67	20	109	112	182	124	111	141	176	41	199	0	73	103	49	141	203	165	129	184	67	56	171	127	123
Den Helder	126	79	154	171	149	197	179	204	215	114	147	73	0	166	109	89	276	238	188	247	140	119	220	176	144
's-Hertogenbosch	73	87	88	63	195	82	45	38	148	104	220	103	166	0	69	215	123	141	46	81	79	53	127	94	129
Hilversum	18	30	65	64	156	119	82	107	132	72	178	49	109	69	0	146	192	172	81	150	74	16	127	83	88
Leeuwarden	147	132	129	154	68	265	228	232	155	182	58	141	89	215	146	0	306	306	171	256	208	162	183	139	91
Maastricht	190	207	176	151	283	183	147	100	236	162	308	203	276	123	192	305	0	242	135	50	188	176	213	182	217
Middelburg	176	175	222	200	315	59	96	153	285	124	340	165	238	141	172	306	242	0	187	203	98	156	264	231	246
Nijmegen	63	109	42	17	149	128	91	61	102	133	174	129	188	46	81	171	135	187	0	85	111	76	81	48	83
Roermond	141	168	127	102	234	144	107	50	187	177	259	184	247	81	150	256	50	203	85	0	151	134	166	133	168
Rotterdam	78	77	125	113	217	57	49	101	192	26	242	67	140	79	74	208	188	98	111	151	0	58	177	143	148
Utrecht	20	40	67	59	159	103	66	91	134	61	184	56	119	53	16	162	176	156	76	134	58	0	123	85	90
Winterswijk	109	151	66	64	143	209	172	142	40	180	168	171	220	127	127	183	213	264	81	166	177	123	0	44	92
Zutphen	65	107	22	31	108	176	139	109	54	146	133	127	176	94	83	139	182	231	48	133	143	85	44	0	48
Zwolle	70	103	41	66	69	193	156	144	71	151	94	123	144	129	88	91	217	246	83	168	148	90	92	48	0

1.9. Let $G = (V, E)$ be a graph and let $l : E \to \mathbb{R}$ be a 'length' function. Call a forest T *good* if $l(ET') \geq l(ET)$ for each forest T' satisfying $|ET'| = |ET|$. (Again, ET is the set of edges of T.)

Let T be a good forest and e be an edge not in T such that $T \cup \{e\}$ is a forest and such that $l(e)$ is as small as possible. Show that $T \cup \{e\}$ is good again.

1.10. Let $G = (V, E)$ be a complete graph and let $l : E \to \mathbb{R}_+$ be a length function satisfying $l(uw) \geq \min\{l(uv), l(vw)\}$ for all distinct $u, v, w \in V$. Let T be a longest spanning tree in G.

Show that for all $u, w \in V$, $l(uw)$ is equal to the minimum length of the edges in the unique $u - w$ path in T.

1.11. Prove (27).

2. Polytopes, polyhedra, Farkas' lemma, and linear programming

2.1. Convex sets

A subset C of $\mathbb{R}^n$ is called *convex* if for all x, y in C and any $0 \leq \lambda \leq 1$ also $\lambda x + (1-\lambda)y$ belongs to C. So C is convex if with any two points in C, the whole line segment connecting x and y belongs to C.

Clearly, the intersection of any number of convex sets is again a convex set. So, for any subset X of $\mathbb{R}^n$, the smallest convex set containing X exists. This set is called the *convex hull* of X and is denoted by conv.hull(X). One easily proves:

(1)
$$\begin{aligned}&\text{conv.hull}(X) = \\ &\{x \mid \exists t \in \mathbb{N}, \exists x_1, \ldots, x_t \in X, \exists \lambda_1, \ldots, \lambda_t \geq 0 : \\ &x = \lambda_1 x_1 + \cdots + \lambda_t x_t, \lambda_1 + \cdots + \lambda_t = 1\}.\end{aligned}$$

A basic property of closed convex sets is that any point not in C can be separated from C by a 'hyperplane'. Here a subset H of $\mathbb{R}^n$ is called a *hyperplane* (or an *affine hyperplane*) if there exist a vector $c \in \mathbb{R}^n$ with $c \neq 0$ and a $\delta \in \mathbb{R}$ such that:

(2)
$$H = \{x \in \mathbb{R}^n \mid c^T x = \delta\}.$$

We say that H *separates* z and C if z and C are in different components of $\mathbb{R}^n \setminus H$.

Theorem 2.1. *Let C be a closed convex set in $\mathbb{R}^n$ and let $z \notin C$. Then there exists a hyperplane separating z and C.*

Proof. Since the theorem is trivial if $C = \emptyset$, we assume $C \neq \emptyset$. Then there exists a vector y in C that is nearest to z, i.e., that minimizes $\|z - y\|$.

(The fact that such a y exists, can be seen as follows. Since $C \neq \emptyset$, there exists an $r > 0$ such that $B(z, r) \cap C \neq \emptyset$. Here $B(z, r)$ denotes the closed ball with center z and radius r. So y minimizes the continuous function $\|z - y\|$ over the compact set $B(z, r) \cap C$.)

Now define:

(3)
$$c := z - y, \delta := \frac{1}{2}(\|z\|^2 - \|y\|^2).$$

We show

(4)
$$\begin{aligned}&\text{(i)} \;\; c^T z > \delta, \\ &\text{(ii)} \;\; c^T x < \delta \text{ for each } x \in C.\end{aligned}$$

Indeed, $c^T z = (z - y)^T z > (z - y)^T z - \frac{1}{2}\|z - y\|^2 = \delta$. This shows (4)(i).

If (4)(ii) would not hold, there exists an x in C such that $c^T x \geq \delta$. Since $c^T y < c^T y + \frac{1}{2}\|c\|^2 = \delta$, we know $c^T(x - y) > 0$. Hence there exists a λ with $0 < \lambda \leq 1$ and

(5)
$$\lambda < \frac{2c^T(x - y)}{\|x - y\|^2}.$$

Define

(6)
$$w := \lambda x + (1 - \lambda)y.$$

So w belongs to C. Moreover,

$$\begin{aligned}(7)\qquad \|w-z\|^2 &= \|\lambda(x-y)+(y-z)\|^2 = \|\lambda(x-y)-c\|^2 \\ &= \lambda^2\|x-y\|^2 - 2\lambda c^T(x-y) + \|c\|^2 < \|c\|^2 = \|y-z\|^2.\end{aligned}$$

Here $<$ follows from (5).

However, (7) contradicts the fact that y is a point in C nearest to z. ∎

Theorem 2.1 implies a characterization of closed convex sets – see Exercise 2.1. Call a subset H of $\mathbb{R}^n$ a *halfspace* (or an *affine halfspace*) if there exist a vector $c \in \mathbb{R}^n$ with $c \neq 0$ and a $\delta \in \mathbb{R}$ such that

$$(8)\qquad H = \{x \in \mathbb{R}^n \mid c^T x \leq \delta\}.$$

Clearly, each affine halfspace is a closed convex set.

Theorem 2.1 implies that if C is a closed convex set and $z \notin C$, then there exists an affine halfspace H so that $C \subseteq H$ and $z \notin H$.

Exercises

2.1. Let $C \subseteq \mathbb{R}^n$. Then C is a closed convex set, if and only if $C = \bigcap \mathcal{F}$ for some collection $\mathcal{F}$ of affine halfspaces.

2.2. Let $C \subseteq \mathbb{R}^n$ be a convex set and let A be an $m \times n$ matrix. Show that the set $\{Ax \mid x \in C\}$ is again convex.

2.3. Let X be a finite set of vectors in $\mathbb{R}^n$. Show that conv.hull(X) is compact.
(*Hint:* Show that conv.hull(X) is the image under a continuous function of a compact set.)

2.4. Show that if $z \in$ conv.hull(X), then there exist affinely independent vectors $x_1, \ldots, x_m$ in X such that $z \in$ conv.hull$\{x_1, \ldots, x_m\}$. (This is the affine form of 'Carathéodory's theorem' (Carathéodory [1911]).)
(Vectors $x_1, \ldots, x_m$ are called *affinely independent* if there are no reals $\lambda_1, \ldots, \lambda_m$, such that $\lambda_1 x_1 + \cdots + \lambda_m x_m = 0$ and $\lambda_1 + \cdots + \lambda_m = 0$ and such that $\lambda_1, \ldots, \lambda_m$ are not all equal to 0.)

2.5. (i) Let C and D be two nonempty, bounded, closed, convex subsets of $\mathbb{R}^n$ such that $C \cap D = \emptyset$. Derive from Theorem 2.1 that there exists an affine hyperplane separating C and D.
(*Hint:* Consider the set $C - D := \{x - y \mid x \in C, y \in D\}$.)
(ii) Show that in (i) we cannot delete the boundedness condition.

2.2. Polytopes and polyhedra

Special classes of closed convex sets are formed by the polytopes and the polyhedra. In the previous section we saw that each closed convex set is the intersection of affine halfspaces, possibly infinitely many. If it is the intersection of a *finite* number of affine halfspaces, the convex set is called a *polyhedron*.

So a subset P of $\mathbb{R}^n$ is a polyhedron if and only if there exists an $m \times n$ matrix A and a vector $b \in \mathbb{R}^m$ such that

$$(9)\qquad P = \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

Here $Ax \leq b$ means:

$$(10)\qquad a_1 x \leq b_1, \ldots, a_m x \leq b_m,$$

where $a_1, \ldots, a_m$ are the rows of A.

The matrix A may have zero rows, i.e. $m = 0$. In that case, $P = \mathbb{R}^n$.

Related is the notion of 'polytope'. A subset P of $\mathbb{R}^n$ is called a *polytope* if P is the convex hull of a finite number of vectors. That is, there exist vectors $x_1, \ldots, x_t$ in $\mathbb{R}^n$ such that

(11) $$P = \text{conv.hull}\{x_1, \ldots, x_t\}.$$

We will show that a subset P of $\mathbb{R}^n$ is a polytope, if and only if it is a bounded polyhedron. This might be intuitively clear, but a strictly mathematical proof requires some work.

We first give a definition. Let P be a convex set. A point $z \in P$ is called a *vertex* of P if z is *not* a convex combination of two other points in P. That is, there do not exist points x, y in P and a λ with $0 < \lambda < 1$ such that $x \neq z, y \neq z$ and $z = \lambda x + (1-\lambda)y$.

To characterize vertices we introduce the following notation. Let $P = \{x \mid Ax \leq b\}$ be a polyhedron and let $z \in P$. Then A_z is the submatrix of A consisting of those rows a_i of A for which $a_i z = b_i$.

Then we can show:

Theorem 2.2. *Let $P = \{x \mid Ax \leq b\}$ be a polyhedron in $\mathbb{R}^n$ and let $z \in P$. Then z is a vertex of P, if and only if* $\text{rank}(A_z) = n$.

Proof. *Necessity.* Let z be a vertex of P and suppose $\text{rank}(A_z) < n$. Then there exists a vector $c \neq 0$ such that $A_z c = 0$. Since $a_i z < b_i$ for every a_i that does not occur in A_z, there exists a $\delta > 0$ such that:

(12) $$a_i(z + \delta c) \leq b_i \text{ and } a_i(z - \delta c) \leq b_i$$

for every row a_i of A not occurring in A_z. Since $A_z c = 0$ and $Az \leq b$ it follows that

(13) $$A(z + \delta c) \leq b \text{ and } A(z - \delta c) \leq b.$$

So $z+\delta c$ and $z-\delta c$ belong to P. Since z is a convex combination of these two vectors, this contradicts the fact that z is a vertex of P.

Sufficiency. Suppose $\text{rank}(A_z) = n$ while z is not a vertex of P. Then there exist points x and y in P such that $x \neq z \neq y$ and $z = \frac{1}{2}(x + y)$. Then for every row a_i of A_z:

(14) $$\begin{aligned} a_i x \leq b_i = a_i z &\implies a_i(x - z) \leq 0, \text{ and} \\ a_i y \leq b_i = a_i z &\implies a_i(y - z) \leq 0. \end{aligned}$$

Since $y - z = -(x - z)$, this implies that $a_i(x - z) = 0$. Hence $A_z(x - z) = 0$. Since $x - z \neq 0$, this contradicts the fact that $\text{rank}(A_z) = n$. ∎

Theorem 2.2 implies that a polyhedron has only a finite number of vertices: For each two different vertices z and z' one has $A_z \neq A_{z'}$, since $A_z x = b_z$ has only one solution, namely $x = z$ (where b_z denotes the part of b corresponding to A_z). Since there exist at most 2^m collections of subrows of A, P has at most 2^m vertices.

From Theorem 2.2 we derive:

Theorem 2.3. *Let P be a bounded polyhedron, with vertices $x_1, \ldots, x_t$. Then $P =$* $\text{conv.hull}\{x_1, \ldots, x_t\}$.

Proof. Clearly

(15) $$\text{conv.hull}\{x_1, \ldots, x_t\} \subseteq P,$$

since $x_1, \ldots, x_t$ belong to P and since P is convex.

The reverse inclusion amounts to:

(16) $$\text{if } z \in P \text{ then } z \in \text{conv.hull}\{x_1, \ldots, x_t\}.$$

We show (16) by induction on $n - \text{rank}(A_z)$.

If $n - \text{rank}(A_z) = 0$, then $\text{rank}(A_z) = n$, and hence, by Theorem 2.2, z itself is a vertex of P. So $z \in \text{conv.hull}\{x_1, \ldots, x_t\}$.

If $n - \text{rank}(A_z) > 0$, then there exists a vector $c \neq 0$ such that $A_z c = 0$. Define

(17) $$\begin{aligned} \mu_0 &:= \max\{\mu \mid z + \mu c \in P\}, \\ \nu_0 &:= \max\{\nu \mid z - \nu c \in P\}. \end{aligned}$$

These numbers exist since P is compact. Let $x := z + \mu_0 c$ and $y := z - \nu_0 c$.

Now

(18) $$\mu_0 = \min\{\frac{b_i - a_i z}{a_i c} \mid a_i \text{ is a row of } A;\ a_i c > 0\}.$$

This follows from the fact that μ_0 is the largest μ such that $a_i(z + \mu c) \leq b_i$ for each $i = 1, \ldots, m$. That is, it is the largest μ such that

(19) $$\mu \leq \frac{b_i - a_i z}{a_i c}$$

for every i with $a_i c > 0$.

Let the minimum (18) be attained by i_0. So for i_0 we have equality in (18). Therefore

(20) $$\begin{aligned} &\text{(i)} \quad A_z x = A_z z + \mu_0 A_z c = A_z z, \\ &\text{(ii)} \quad a_{i_0} x = a_{i_0}(z + \mu_0 c) = b_{i_0}. \end{aligned}$$

So A_x contains all rows in A_z, and moreover it contains row a_{i_0}. Now $A_z c = 0$ while $a_{i_0} c \neq 0$. This implies $\text{rank}(A_x) > \text{rank}(A_z)$. So by our induction hypothesis, x belongs to $\text{conv.hull}\{x_1, \ldots, x_t\}$. Similarly, y belongs to $\text{conv.hull}\{x_1, \ldots, x_t\}$. Therefore, as z is a convex combination of x and y, z belongs to $\text{conv.hull}\{x_1, \ldots, x_t\}$. ∎

As a direct consequence we have:

Corollary 2.3a. *Each bounded polyhedron is a polytope.*

Proof. Directly from Theorem 2.3. ∎

Conversely:

Theorem 2.4. *Each polytope is a bounded polyhedron.*

Proof. Let P be a polytope in $\mathbb{R}^n$, say

(21) $$P = \text{conv.hull}\{x_1, \ldots, x_t\}.$$

We may assume that $t \geq 1$. We prove the theorem by induction on n. Clearly, P is bounded.

If P is contained in some affine hyperplane, the theorem follows from the induction hypothesis.

So we may assume that P is not contained in any affine hyperplane. It implies that the vectors $x_2 - x_1, \ldots, x_t - x_1$ span $\mathbb{R}^n$. It follows that there exist a vector x_0 in P and a real $r > 0$ such that the ball $B(x_0, r)$ is contained in P.

Without loss of generality, $x_0 = 0$. Define P^* by

(22) $$P^* := \{y \in \mathbb{R}^n \mid x^T y \leq 1 \text{ for each } x \in P\}.$$

Then P^* is a polyhedron, as

(23) $$P^* = \{y \in \mathbb{R}^n \mid x_j^T y \leq 1 \text{ for } j = 1, \ldots, t\}.$$

This follows from the fact that if y belongs to the right hand set in (23) and $x \in P$ then $x = \lambda_1 x_1 + \cdots + \lambda_t x_t$ for certain $\lambda_1, \ldots, \lambda_t \geq 0$ with $\lambda_1 + \cdots + \lambda_t = 1$, implying

(24) $$x^T y = \sum_{j=1}^{t} \lambda_j x_j^T y \leq \sum_{j=1}^{t} \lambda_j = 1.$$

So y belongs to P^*.

Moreover, P^* is bounded, since for each $y \neq 0$ in P^* one has that $x := r \cdot \|y\|^{-1} \cdot y$ belongs to $B(0, r)$ and hence to P. Therefore, $x^T y \leq 1$, and hence

(25) $$\|y\| = (x^T y)/r \leq 1/r.$$

So $P^* \subseteq B(0, 1/r)$.

This proves that P^* is a bounded polyhedron. By Corollary 2.3a, P^* is a polytope. So there exist vectors $y_1, \ldots, y_s$ in $\mathbb{R}^n$ such that

(26) $$P^* = \text{conv.hull}\{y_1, \ldots, y_s\}.$$

We show:

(27) $$P = \{x \in \mathbb{R}^n \mid y_j^T x \leq 1 \text{ for all } j = 1, \ldots, s\}.$$

This implies that P is a polyhedron.

To see the inclusion $\subseteq$ in (27), it suffices to show that each of the vectors x_i belongs to the right hand side in (27). This follows directly from the fact that for each $j = 1, \ldots, s$, $y_j^T x_i = x_i^T y_j \leq 1$, since y_j belongs to P^*.

To see the inclusion $\supseteq$ in (25), let $x \in \mathbb{R}^n$ be such that $y_j^T x \leq 1$ for all $j = 1, \ldots, s$. Suppose $x \notin P$. Then there exists a hyperplane separating z and P. That is, there exist a vector $c \neq 0$ in $\mathbb{R}^n$ and a $\delta \in \mathbb{R}$ such that $c^T x' < \delta$ for each $x' \in P$, while $c^T x > \delta$. As $0 \in P$, $\delta > 0$. So we may assume $\delta = 1$. Hence $c \in P^*$. So there exist $\mu_1, \ldots, \mu_s \geq 0$ such that $c = \mu_1 y_1 + \cdots \mu_s y_s$ and $\mu_1 + \cdots + \mu_s = 1$. This gives the contradiction:

(28) $$1 < c^T x = \sum_{j=1}^{s} \mu_j y_j^T x \leq \sum_{j=1}^{s} \mu_j = 1.$$ ∎

Convex cones

Convex cones are special cases of convex sets. A subset C of $\mathbb{R}^n$ is called a *convex cone* if for any $x, y \in C$ and any $\lambda, \mu \geq 0$ one has $\lambda x + \mu y \in C$.

For any $X \subseteq \mathbb{R}^n$, cone(X) is the smallest cone containing X. One easily checks:

(29) $$\text{cone}(X) = \{\lambda_1 x_1 + \cdots \lambda_t x_t \mid x_1, \ldots, x_t \in X; \lambda_1, \ldots, \lambda_t \geq 0\}.$$

A cone C is called *finitely generated* if $C = \text{cone}(X)$ for some finite set X.

Exercises

2.6. Determine the vertices of the following polyhedra:

(i) $P = \{(x, y) \mid x \geq 0, y \geq 0, y - x \leq 2, x + y \leq 8, x + 2y \leq 10, x \leq 4\}$.

(ii) $P = \{(x, y, z) \mid x + y \leq 2, y + z \leq 4, x + z \leq 3, -2x - y \leq 3, -y - 2z \leq 3, -2x - z \leq 2\}$.

(iii) $P = \{(x, y) \mid x + y \leq 1, x - y \leq 2\}$.

(iv) $P = \{(x,y) \mid x+y=1, x \geq 3\}$.

(v) $P = \{(x,y,z) \mid x \geq 0, y \geq 0, x+y \leq 1\}$.

(vi) $P = \{(x,y,z) \mid x+y \geq 1, x+z \geq 1, y-z \geq 0\}$.

(vii) $P = \{(x,y) \mid 3x+2y \leq 18, x-y \geq -6, 5x+2y \leq 20, x \geq 0, y \geq 0\}$.

2.7. Let $C \subseteq \mathbb{R}^n$. Then C is a closed convex cone, if and only if $C = \bigcap \mathcal{F}$ for some collection $\mathcal{F}$ of linear halfspaces.

(A subset H of $\mathbb{R}^n$ is called a *linear halfspace* if $H = \{x \in \mathbb{R}^n \mid c^T x \leq 0\}$ for some nonzero vector c.)

2.8. Show that if $z \in \text{cone}(X)$, then there exist linearly independent vectors $x_1, \ldots, x_m$ in X such that $z \in \text{cone}\{x_1, \ldots, x_m\}$. (This is the linear form of 'Carathéodory's theorem'.)

2.9. Let A be an $m \times n$ matrix of rank m and let $b \in \mathbb{R}^m$. Derive from Exercise 2.8 that the system $Ax = b$ has a nonnegative solution x, if and only if it has a nonnegative basic solution.

(A submatrix B of A is called a *basis* of A if B is a nonsingular $m \times m$ submatrix of A. A solution x of $Ax = b$ is a *basic solution* if A has a basis B so that x is 0 in those coordinates *not* corresponding to columns in B.)

2.10. Prove that every finitely generated convex cone is a closed set. (This can be derived from Exercise 2.3 and Corollary 2.3a.)

2.11. Prove that a convex cone is finitely generated, if and only if it is the intersection of finitely many linear halfspaces.

(*Hint:* Use Corollary 2.3a and Theorem 2.4.)

2.12. Let P be a subset of $\mathbb{R}^n$. Show that P is a polyhedron, if and only if $P = Q + C$ for some polytope Q and some finitely generated convex cone C.

(*Hint:* Apply Exercise 2.11 to cone(X) in $\mathbb{R}^{n+1}$, where X is the set of vectors $\begin{pmatrix} 1 \\ x \end{pmatrix}$ in $\mathbb{R}^{n+1}$ with $x \in P$.)

2.13. For any subset X of $\mathbb{R}^n$, define

(30) $$X^* := \{y \in \mathbb{R}^n \mid x^T y \leq 1 \text{ for each } x \in X\}.$$

(i) Show that for each convex cone C, C^* is a closed convex cone.

(ii) Show that for each closed convex cone C, $(C^*)^* = C$.

2.14. Let P be a polyhedron.

(i) Show that P^* is again a polyhedron.
(*Hint:* Use previous exercises.)

(ii) Show that P contains the origin, if and only if $(P^*)^* = P$.

(iii) Show that the origin is an internal point of P, if and only if P^* is bounded.

2.3. Farkas' lemma

Let A be an $m \times n$ matrix and let $b \in \mathbb{R}^m$. With the Gaussian elimination method one can prove that

(31) $$Ax = b$$

has a solution x, if and only if there is no solution y for the following system of linear equations:

(32) $$y^T A = 0, y^T b = -1.$$

Farkas' lemma (Farkas [1894,1896,1898]) gives an analogous characterization for the existence of a *nonnegative* solution x for (31).

Theorem 2.5 (Farkas' lemma)**.** *The system $Ax = b$ has a nonnegative solution, if and only if there is no vector y satisfying $y^TA \geq 0$ and $y^Tb < 0$.*

Proof. *Necessity.* Suppose $Ax = b$ has a solution $x_0 \geq 0$ and suppose there exists a vector y_0 satisfying $y_0^TA \geq 0$ and $y_0^Tb < 0$. Then we obtain the contradiction

(33) $$0 > y_0^Tb = y_0^T(Ax_0) = (y_0^TA)x_0 \geq 0.$$

Sufficiency. Suppose $Ax = b$ has no solution $x \geq 0$. Let $a_1, \ldots, a_n$ be the columns of A. So

(34) $$b \notin C := \text{cone}\{a_1, \ldots, a_n\}.$$

So by Exercise 2.7 there exists a linear halfspace H containing C and not containing b. That is, there exists a vector c such that $c^Tb < 0$ while $c^Tx \geq 0$ for each x in C. In particular, $c^Ta_j \geq 0$ for $j = 1, \ldots, n$. So $y := c$ satisfies $y^TA \geq 0$ and $y^Tb < 0$. ▮

So Farkas' lemma states that exactly one of the following two assertions is true:

(35) $$\begin{aligned} &\text{(i) } \exists x \geq 0 : Ax = b, \\ &\text{(ii) } \exists y : y^TA \geq 0 \text{ and } y^Tb < 0. \end{aligned}$$

There exist several variants of Farkas' lemma, that can be easily derived from Theorem 2.5.

Corollary 2.5a. *The system $Ax \leq b$ has a solution x, if and only if there is no vector y satisfying $y \geq 0, y^TA = 0$ and $y^Tb < 0$.*

Proof. Let A' be the matrix

(36) $$A' := [A \quad -A \quad I],$$

where I denotes the identity matrix.

Then $Ax \leq b$ has a solution x, if and only if the system $A'x' = b$ has a nonnegative solution x'. Applying Theorem 2.5 to $A'x' = b$ gives the corollary. ▮

Another consequence is:

Corollary 2.5b. *Suppose the system $Ax \leq b$ has at least one solution. Then for every solution x of $Ax \leq b$ one has $c^Tx \leq \delta$, if and only if there exists a vector $y \geq 0$ such that $y^TA = c^T$ and $y^Tb \leq \delta$.*

Proof. *Sufficiency.* If such a vector y exists, then for every vector x one has

(37) $$Ax \leq b \Longrightarrow y^TAx \leq y^Tb \Longrightarrow c^Tx \leq y^Tb \Longrightarrow c^Tx \leq \delta.$$

Necessity. Suppose such a vector y does not exist. It means that the following system of linear inequalities in the variables y and λ has no solution $(y^T \ \lambda) \geq (0 \ 0)$:

(38) $$(y^T \quad \lambda)\begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} = (c^T \quad \delta).$$

According to Farkas' lemma this implies that there exists a vector $\begin{pmatrix} z \\ \mu \end{pmatrix}$ so that

(39) $$\begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix}\begin{pmatrix} z \\ \mu \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ and } (c^T \quad \delta)\begin{pmatrix} z \\ \mu \end{pmatrix} < 0.$$

We distinguish two cases.

Case 1: $\mu = 0$. Then $Az \geq 0$ and $c^T z < 0$. However, by assumption, $Ax \leq b$ has a solution x_0. Then, for τ large enough:

(40) $$A(x_0 - \tau z) \leq b \text{ and } c^T(x_0 - \tau z) > \delta.$$

This contradicts the fact that $Ax \leq b$ implies $c^T x \leq \delta$.

Case 2: $\mu > 0$. As (39) is homogeneous, we may assume that $\mu = 1$. Then for $x := -z$ one has:

(41) $$Ax \leq b \text{ and } c^T x > \delta.$$

Again this contradicts the fact that $Ax \leq b$ implies $c^T x \leq \delta$. ∎

Exercises

2.15. Prove that there exists a vector $x \geq 0$ such that $Ax \leq b$, if and only if for each $y \geq 0$ satisfying $y^T A \geq 0$ one has $y^T b \geq 0$.

2.16. Prove that there exists a vector $x > 0$ such that $Ax = 0$, if and only if for each y satisfying $y^T A \geq 0$ one has $y^T A = 0$. (Stiemke's theorem (Stiemke [1915]).)

2.17. Prove that there exists a vector $x \neq 0$ satisfying $x \geq 0$ and $Ax = 0$, if and only if there is no vector y satisfying $y^T A > 0$. (Gordan's theorem (Gordan [1873]).)

2.18. Prove that there exists a vector x satisfying $Ax < b$, if and only if $y = 0$ is the only solution for $y \geq 0, y^T A = 0, y^T b \leq 0$.

2.19. Prove that there exists a vector x satisfying $Ax < b$ and $A'x \leq b'$, if and only if for all vectors $y, y' \geq 0$ one has:

(i) if $y^T A + y'^T A' = 0$ then $y^T b + y'^T b' \geq 0$, and

(ii) if $y^T A + y'^T A' = 0$ and $y \neq 0$ then $y^T b + y'^T b' > 0$.

(Motzkin's theorem (Motzkin [1936]).)

2.20. Let A be an $m \times n$ matrix and let $b \in \mathbb{R}^m$, with $m \geq n + 1$. Suppose that $Ax \leq b$ has no solution x. Prove that there exist indices $i_0, \ldots, i_n$ so that the system $a_{i_0} x \leq b_{i_0}, \ldots, a_{i_n} x \leq b_{i_n}$ has no solution x. Here a_i is the ith row of A and b_i is the ith component of b.

(*Hint:* Combine Farkas' lemma with Carathéodory's theorem.)

2.4. Linear programming

One of the standard forms of a linear programming (LP) problem is:

(42) $$\begin{aligned}&\text{maximize } c^T x,\\ &\text{subject to } Ax \leq b.\end{aligned}$$

So linear programming can be considered as maximizing a 'linear function' $c^T x$ over a polyhedron $P = \{x \mid Ax \leq b\}$. Geometrically, this can be seen as shifting a hyperplane to its 'highest' level, under the condition that it intersects P.

Problem (42) corresponds to determining the following maximum:

(43) $$\max\{c^T x \mid Ax \leq b\}.$$

This is the form in which we will denote an LP-problem.

If $P = \{x \mid Ax \leq b\}$ is a nonempty polytope, then it is clear that $\max\{c^T x \mid Ax \leq b\}$ is attained by a *vertex* of P (cf. Exercise 2.21).

Clearly, also any *minimization* problem can be transformed to form (43), since

(44) $$\min\{c^Tx \mid Ax \leq b\} = -\max\{-c^Tx \mid Ax \leq b\}.$$

One says that x is a *feasible solution* of (43) if x satisfies $Ax \leq b$. If x moreover attains the maximum, x is called an *optimum solution.*

The famous method to solve linear programming problems is the *simplex method*, designed by Dantzig [1951b]. The first polynomial-time method for LP-problems is due to Khachiyan [1979, 1980], based on the *ellipsoid method.* In 1984, Karmarkar [1984] published another polynomial-time method for linear programming, the *interior point method*, which turns out to be competitive in practice with the simplex method.

The Duality theorem of linear programming, due to von Neumann [1947], states that if the maximum (43) is finite, then the maximum value is equal to the minimum value of another, so-called *dual* LP-problem:

(45) $$\min\{y^Tb \mid y \geq 0; y^TA = c^T\}.$$

In order to show this, we first prove:

Lemma 2.1. *Let P be a polyhedron in $\mathbb{R}^n$ and let $c \in \mathbb{R}^n$. If* $\sup\{c^Tx \mid x \in P\}$ *is finite, then* $\max\{c^Tx \mid x \in P\}$ *is attained.*

Proof. Let $\delta := \sup\{c^Tx \mid x \in P\}$. Choose matrix A and vector b so that $P = \{x \mid Ax \leq b\}$. We must show that there exists an $x \in \mathbb{R}^n$ such that $Ax \leq b$ and $c^Tx \geq \delta$.

Suppose such an x does not exist. Then by Farkas' lemma, in the form of Corollary 2.5a, there exists a vector $y \geq 0$ and a real number $\lambda \geq 0$ such that:

(46) $$y^TA - \lambda c^T = 0, y^Tb - \lambda\delta < 0.$$

Since $Ax \leq b$ has a solution x_0 (as the supremum is finite), we know $\lambda > 0$ (since if $\lambda = 0$ then $0 = y^TAx_0 \leq y^Tb < 0$).

As (46) is homogeneous, we may assume $\lambda = 1$. Then $y^TA = c^T$ and $y^Tb < \delta$. So for each x satisfying $Ax \leq b$ we have $c^Tx = y^TAx \leq y^Tb$. This implies $\delta = \sup\{c^Tx \mid Ax \leq b\} \leq y^Tb < \delta$, a contradiction. ∎

From this we derive:

Theorem 2.6 (Duality theorem of linear programming)**.** *Let A be an $m \times n$ matrix, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$. Then*

(47) $$\max\{c^Tx \mid Ax \leq b\} = \min\{y^Tb \mid y \geq 0; y^TA = c^T\},$$

provided that both sets are nonempty.

Proof. First note that

(48) $$\sup\{c^Tx \mid Ax \leq b\} \leq \inf\{y^Tb \mid y \geq 0; y^TA = c^T\},$$

because if $Ax \leq b, y \geq 0, y^TA = c^T$, then

(49) $$c^Tx = (y^TA)x = y^T(Ax) \leq y^Tb.$$

As both sets are nonempty,the supremum and the infimum are finite. By Lemma 2.1 it suffices to show that we have equality in (48).

Let $\delta := \sup\{c^T x \mid Ax \leq b\}$. Hence:

(50) $$\text{if } Ax \leq b \text{ then } c^T x \leq \delta.$$

So by Corollary 2.5b there exists a vector y such that

(51) $$y \geq 0, y^T A = c^T, y^T b \leq \delta.$$

This implies that the infimum in (48) is at most δ. ▮

The Duality theorem can be interpreted geometrically as follows. Let

(52) $$\max\{c^T x \mid Ax \leq b\} =: \delta$$

be attained at a point x^*. Without loss of generality we may assume that the first k rows of A belong to the matrix A_{x^*}. So $a_1 x \leq b_1, \ldots, a_k x \leq b_k$ are those inequalities in $Ax \leq b$ for which $a_i x^* = b_i$ holds. Elementary geometric insight (cf. Figure 2.1) gives that $c^T x = \delta$ must be a nonnegative linear combination of the equations $a_1 x = b_1, \ldots, a_k x = b_k$.

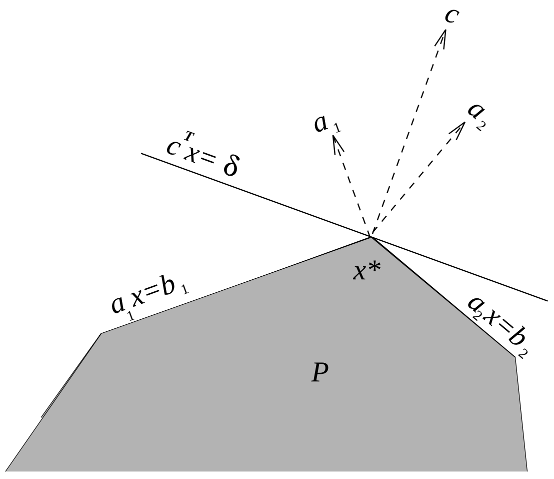


Figure 2.1

That is, there exist $\lambda_1, \ldots, \lambda_k \geq 0$ such that:

(53) $$\begin{aligned} &\lambda_1 a_1 + \cdots + \lambda_k a_k = c^T, \\ &\lambda_1 b_1 + \cdots + \lambda_k b_k = \delta. \end{aligned}$$

Define

(54) $$y^* := (\lambda_1, \ldots, \lambda_k, 0, \ldots, 0)^T.$$

Then y^* is a feasible solution for the dual problem $\min\{y^T b \mid y \geq 0; y^T A = c^T\}$. Therefore,

(55) $$\max\{c^T x \mid Ax \leq b\} = \delta = \lambda_1 b_1 + \cdots + \lambda_k b_k \geq \min\{y^T b \mid y \geq 0; y^T A = c^T\}.$$

Since trivially the converse inequality holds:

(56) $$\max\{c^T x \mid Ax \leq b\} \leq \min\{y^T b \mid y \geq 0; y^T A = c^T\}$$

(cf. (49)), y^* is an optimum solution of the dual problem.

There exist several variants of the Duality theorem.

Corollary 2.6a. *Let A be an $m \times n$ matrix, $b \in \mathbb{R}^m, c \in \mathbb{R}^n$. Then*

$$\max\{c^Tx \mid x \geq 0; Ax = b\} = \min\{y^Tb \mid y^TA \geq c^T\}, \tag{57}$$

provided that both sets are nonempty.

Proof. Define

$$\tilde{A} := \begin{pmatrix} A \\ -A \\ -I \end{pmatrix}, \tilde{b} := \begin{pmatrix} b \\ -b \\ 0 \end{pmatrix}. \tag{58}$$

Then

$$\begin{aligned} &\max\{c^Tx \mid x \geq 0; Ax = b\} = \max\{c^Tx \mid \tilde{A}x \leq \tilde{b}\} = \\ &\min\{z^T\tilde{b} \mid z \geq 0; z^T\tilde{A} = c^T\} = \\ &\min\{u^Tb - v^Tb + w^T0 \mid u, v, w \geq 0; u^TA - v^TA - w^T = c^T\} = \\ &\min\{y^Tb \mid y^TA \geq c^T\}. \end{aligned} \tag{59}$$

The last equality follows by taking $y := u - v$. ∎

Exercises

2.21. Let $P = \{x \mid Ax \leq b\}$ be a nonempty polytope. Prove that $\max\{c^Tx \mid Ax \leq b\}$ is attained by a vertex of P.

2.22. Let $P = \{x \mid Ax \leq b\}$ be a (not necessarily bounded) polyhedron, such that P has at least one vertex. Prove that if $\max\{c^Tx \mid Ax \leq b\}$ is finite, it is attained by a vertex of P.

2.23. Prove the following variant of the Duality theorem:

$$\max\{c^Tx \mid x \geq 0; Ax \leq b\} = \min\{y^Tb \mid y \geq 0; y^TA \geq c^T\} \tag{60}$$

(assuming both sets are nonempty).

2.24. Prove the following variant of the Duality theorem:

$$\max\{c^Tx \mid Ax \geq b\} = \min\{y^Tb \mid y \leq 0; y^TA = c^T\} \tag{61}$$

(assuming both sets are nonempty).

2.25. Let a matrix, a column vector, and a row vector be given:

$$\begin{pmatrix} A & B & C \\ D & E & F \\ G & H & K \end{pmatrix}, \begin{pmatrix} a \\ b \\ c \end{pmatrix}, (d \quad e \quad f), \tag{62}$$

where $A, B, C, D, E, F, G, H, K$ are matrices, a, b, c are column vectors, and d, e, f are row vectors (of appropriate dimensions). Then

$$\begin{aligned} &\max\{dx + ey + fz \mid && x \geq 0; z \leq 0; \\ & && Ax + By + Cz \leq a; \\ & && Dx + Ey + Fz = b; \\ & && Gx + Hy + Kz \geq c\} \\ = &\min\{ua + vb + wc \mid && u \geq 0; w \leq 0; \\ & && uA + vD + wG \geq d; \\ & && uB + vE + wH = e; \\ & && uC + vF + wK \leq f\}, \end{aligned} \tag{63}$$

assuming that both sets are nonempty.

2.26. Give an example of a matrix A and vectors b and c for which both $\{x \mid Ax \leq b\}$ and $\{y \mid y \geq 0; y^TA = c^T\}$ are empty.

2.27. Let $\tilde{x}$ be a feasible solution of $\max\{c^T x \mid Ax \leq b\}$ and let $\tilde{y}$ be a feasible solution of $\min\{y^T b \mid y \geq 0; y^T A = c^T\}$. Prove that $\tilde{x}$ and $\tilde{y}$ are optimum solutions of the maximum and minimum, respectively, if and only if for each $i = 1, \ldots, m$ one has: $\tilde{y}_i = 0$ or $a_i\tilde{x} = b_i$.

(Here A has m rows and a_i denotes the ith row of A.)

2.28. Let A be an $m \times n$ matrix and let $b \in \mathbb{R}^m$. Let $\{x \mid Ax \leq b\}$ be nonempty and let C be the convex cone $\{x \mid Ax \leq 0\}$. Prove that the set of all vectors c for which $\max\{c^T x \mid Ax \leq b\}$ is finite, is equal to C^*.

3. Matchings and covers in bipartite graphs

3.1. Matchings, covers, and Gallai's theorem

Let $G = (V, E)$ be a graph. A *coclique* is a subset C of V such that $e \not\subseteq C$ for each edge e of G. A *vertex cover* is a subset W of V such that $e \cap W \neq \emptyset$ for each edge e of G. It is not difficult to show that for each $U \subseteq V$:

(1) $$U \text{ is a coclique} \iff V \setminus U \text{ is a vertex cover.}$$

A *matching* is a subset M of E such that $e \cap e' = \emptyset$ for all $e, e' \in M$ with $e \neq e'$. A matching is called *perfect* if it covers all vertices (that is, has size $\frac{1}{2}|V|$). An *edge cover* is a subset F of E such that for each vertex v there exists $e \in F$ satisfying $v \in e$. Note that an edge cover can exist only if G has no isolated vertices.

Define:

(2) $$\begin{aligned} \alpha(G) &:= \max\{|C| \mid C \text{ is a coclique}\}, \\ \rho(G) &:= \min\{|F| \mid F \text{ is an edge cover}\}, \\ \tau(G) &:= \min\{|W| \mid W \text{ is a vertex cover}\}, \\ \nu(G) &:= \max\{|M| \mid M \text{ is a matching}\}. \end{aligned}$$

These numbers are called the *coclique number*, the *edge cover number*, the *vertex cover number*, and the *matching number* of G, respectively.

It is not difficult to show that:

(3) $$\alpha(G) \leq \rho(G) \text{ and } \nu(G) \leq \tau(G).$$

The triangle K_3 shows that strict inequalities are possible. In fact, equality in one of the relations (3) implies equality in the other, as Gallai [1958,1959] proved:

Theorem 3.1 (Gallai's theorem)**.** *For any graph $G = (V, E)$ without isolated vertices one has*

(4) $$\alpha(G) + \tau(G) = |V| = \nu(G) + \rho(G).$$

Proof. The first equality follows directly from (1).

To see the second equality, first let M be a matching of size $\nu(G)$. For each of the $|V| - 2|M|$ vertices v missed by M, add to M an edge covering v. We obtain an edge cover of size $|M| + (|V| - 2|M|) = |V| - |M|$. Hence $\rho(G) \leq |V| - \nu(G)$.

Second, let F be an edge cover of size $\rho(G)$. For each $v \in V$ delete from F, $d_F(v) - 1$ edges incident with v. We obtain a matching of size at least $|F| - \sum_{v\in V}(d_F(v) - 1) = |F| - (2|F| - |V|) = |V| - |F|$. Hence $\nu(G) \geq |V| - \rho(G)$. ▮

This proof also shows that if we have a matching of maximum cardinality in any graph G, then we can derive from it a minimum cardinality edge cover, and conversely.

Exercises

3.1. Let $G = (V, E)$ be a graph without isolated vertices. Define:

(5)

$\alpha_2(G) :=$	the maximum number of vertices such that no edge contains more than two of these vertices;
$\rho_2(G) :=$	the minimum number of edges such that each vertex is contained in at least two of these edges;
$\tau_2(G) :=$	the minimum number of vertices such that each edge contains at least two of these vertices
$\nu_2(G) :=$	the maximum number of edges such that no vertex is contained in more than two of these edges;

possibly taking vertices (edges, respectively) more than once.

(i) Show that $\alpha_2(G) \leq \rho_2(G)$ and that $\nu_2(G) \leq \tau_2(G)$.

(ii) Show that $\alpha_2(G) + \tau_2(G) = 2|V|$.

(iii) Show that $\nu_2(G) + \rho_2(G) = 2|V|$.

3.2. Kőnig's theorems

A classical min-max relation due to Kőnig [1931] (extending a result of Frobenius [1917]) characterizes the maximum size of a matching in a bipartite graph:

Theorem 3.2 (Kőnig's matching theorem)**.** *For any bipartite graph $G = (V, E)$ one has*

(6) $$\nu(G) = \tau(G).$$

That is, the maximum cardinality of a matching in a bipartite graph is equal to the minimum cardinality of a vertex cover.

Proof. Let $G = (V, E)$ be a bipartite graph, with colour classes U and W, say. By (3) it suffices to show that $\nu(G) \geq \tau(G)$, which we do by induction on $|V|$. We distinguish two cases.

Case 1: *There exists a vertex cover C with $|C| = \tau(G)$ intersecting both U and W.*

Let $U' := U \cap C$, $U'' := U \setminus C$, $W' := W \setminus C$ and $W'' := W \cap C$. Let G' and G'' be the subgraphs of G induced by $U' \cup W'$ and $U'' \cup W''$ respectively.

We show that $\tau(G') \geq |U'|$. Let K be a vertex cover of G' of size $\tau(G')$. Then $K \cup W''$ is a vertex cover of G, since K intersects all edges of G that are contained in $U' \cup W'$ and W'' intersects all edges of G that are not contained in $U' \cup W'$ (since each edge intersects $C = U' \cup W''$). So $|K \cup W''| \geq \tau(G) = |U'| + |W''|$ and hence $|K| \geq |U'|$.

So $\tau(G') \geq |U'|$. It follows by our induction hypothesis that G' contains a matching of size $|U'|$. Similarly, G'' contains a matching of size $|W''|$. Combining the two matchings we obtain a matching of G of size $|U'| + |W''| = \tau(G)$.

Case 2: *There exists no such vertex cover C.*

Let $e = uw$ be any edge of G. Let G' be the subgraph of G induced by $V \setminus \{u, w\}$. We show that $\tau(G') \geq \tau(G) - 1$. Suppose to the contrary that G' contains a vertex cover K of size $\tau(G) - 2$. Then $C := K \cup \{u, w\}$ would be a vertex cover of G of size $\tau(G)$ intersecting both U and W, a contradiction.

So $\tau(G') \geq \tau(G) - 1$, implying by our induction hypothesis that G' contains a matching M of size $\tau(G) - 1$. Hence $M \cup \{e\}$ is a matching of G of size $\tau(G)$. ▮

Combination of Theorems 3.1 and 3.2 yields the following result of Kőnig [1932].

Corollary 3.2a (Kőnig's edge cover theorem)**.** *For any bipartite graph $G = (V, E)$, without isolated vertices, one has*

(7) $$\alpha(G) = \rho(G).$$

That is, the maximum cardinality of a coclique in a bipartite graph is equal to the minimum cardinality of an edge cover.

Proof. Directly from Theorems 3.1 and 3.2, as $\alpha(G) = |V| - \tau(G) = |V| - \nu(G) = \rho(G)$. ▮

Exercises

3.2. (i) Prove that a k-regular bipartite graph has a perfect matching (if $k \geq 1$).

(ii) Derive that a k-regular bipartite graph has k disjoint perfect matchings.

(iii) Give for each $k > 1$ an example of a k-regular graph not having a perfect matching.

3.3. Prove that in a matrix, the maximum number of nonzero entries with no two in the same line (=row or column), is equal to the minimum number of lines that include all nonzero entries.

3.4. Let $\mathcal{A} = (A_1, \ldots, A_n)$ be a family of subsets of some finite set X. A subset Y of X is called a *transversal* or a *system of distinct representatives* (*SDR*) of $\mathcal{A}$ if there exists a bijection $\pi : \{1, \ldots, n\} \to Y$ such that $\pi(i) \in A_i$ for each $i = 1, \ldots, n$.

Decide if the following collections have an SDR:

(i) $\{3,4,5\}, \{2,5,6\}, \{1,2,5\}, \{1,2,3\}, \{1,3,6\}$,

(ii) $\{1,2,3,4,5,6\}, \{1,3,4\}, \{1,4,7\}, \{2,3,5,6\}, \{3,4,7\}, \{1,3,4,7\}, \{1,3,7\}$.

3.5. Let $\mathcal{A} = (A_1, \ldots, A_n)$ be a family of subsets of some finite set X. Prove that $\mathcal{A}$ has an SDR if and only if

$$(8) \qquad |\bigcup_{i \in I} A_i| \geq |I|$$

for each subset I of $\{1, \ldots, n\}$.

[Hall's 'marriage' theorem (Hall [1935]).]

3.6. Let $\mathcal{A} = (A_1, \ldots, A_n)$ be subsets of the finite set X. A subset Y of X is called a *partial transversal* or a *partial system of distinct representatives* (*partial SDR*) if it is a transversal of some subcollection $(A_{i_1}, \ldots, A_{i_k})$ of $(A_1, \ldots, A_n)$.

Show that the maximum cardinality of a partial SDR of $\mathcal{A}$ is equal to the minimum value of

$$(9) \qquad |X \setminus Z| + |\{i \mid A_i \cap Z \neq \emptyset\}|,$$

where Z ranges over all subsets of X.

3.7. Let $\mathcal{A} = (A_1, \ldots, A_n)$ be a family of finite sets and let k be a natural number. Show that $\mathcal{A}$ has k pairwise disjoint SDR's of $\mathcal{A}$, if and only if

$$(10) \qquad |\bigcup_{i \in I} A_i| \geq k|I|$$

for each subset I of $\{1, \ldots, n\}$.

3.8. Let $\mathcal{A} = (A_1, \ldots, A_n)$ be a family of subsets of a finite set X and let k be a natural number. Show that X can be partitioned into k partial SDR's, if and only if

$$(11) \qquad k \cdot |\{i \mid A_i \cap Y \neq \emptyset\}| \geq |Y|$$

for each subset Y of X.

(*Hint:* Replace each A_i by k copies of A_i and use Exercise 3.6 above.)

3.9. Let $(A_1, \ldots, A_n)$ and $(B_1, \ldots, B_n)$ be two partitions of the finite set X.

(i) Show that $(A_1, \ldots, A_n)$ and $(B_1, \ldots, B_n)$ have a *common* SDR if and only if for each subset I of $\{1, \ldots, n\}$, the set $\bigcup_{i \in I} A_i$ intersects at least $|I|$ sets among $B_1, \ldots, B_n$.

(ii) Suppose that $|A_1| = \cdots = |A_n| = |B_1| = \cdots = |B_n|$. Show that the two partitions have a common SDR.

3.10. Let $(A_1, \ldots, A_n)$ and $(B_1, \ldots, B_n)$ be two partitions of the finite set X. Show that the minimum cardinality of a subset of X intersecting each set among $A_1, \ldots, A_n, B_1, \ldots, B_n$ is equal to the maximum number of pairwise disjoint sets in $A_1, \ldots, A_n, B_1, \ldots, B_n$.

3.11. A matrix is called *doubly stochastic* if it is nonnegative and each row sum and each column sum is equal to 1. A matrix is called a *permutation matrix* if each entry is 0 or 1 and each row and each column contains exactly one 1. Show that each doubly stochastic matrix is a convex linear combination of permutation matrices.

[Birkhoff-von Neumann theorem (Birkhoff [1946], von Neumann [1953]).]

3.12. Let $G = (V, E)$ be a bipartite graph with colour classes U and W. Let $b : V \to \mathbb{Z}_+$ be so that $\sum_{v \in U} b(v) = \sum_{v \in W} b(v) =: t$.
A *b-matching* is a function $c : E \to \mathbb{Z}_+$ so that for each vertex v of G:

$$\sum_{e \in E, v \in e} c(e) = b(v) \tag{12}$$

Show that there exists a b-matching if and only if

$$\sum_{v \in X} b(v) \geq t \tag{13}$$

for each vertex cover X.

3.13. Let $G = (V, E)$ be a bipartite graph with colour classes U and W. Let $b : V \to \mathbb{Z}_+$ be so that $\sum_{v \in U} b(v) = \sum_{v \in W} b(v) = t$.
Show that there exists a subset F of E so that each vertex v of G is incident with exactly $b(v)$ of the edges in F, if and only if

$$t + |E(X)| \geq \sum_{v \in X} b(v) \tag{14}$$

for each subset X of V, where $E(X)$ denotes the set of edges contained in X.

3.14. Let $G = (V, E)$ be a bipartite graph and let $b : V \to \mathbb{Z}_+$. Show that the maximum number of edges in a subset F of E so that each vertex v of G is incident with at most $b(v)$ of the edges in F, is equal to

$$\min_{X \subseteq V} \sum_{v \in X} b(v) + |E(V \setminus X)|. \tag{15}$$

3.15. Let G be a bipartite graph with colour classes U and W satisfying $|U| = |W| = t$. Prove that G has k disjoint perfect matchings if and only if for all $U' \subseteq U$ and $W' \subseteq W$ there are at least $k(|U'| + |W'| - t)$ edges connecting U' and W'.

3.16. Show that each $2k$-regular graph contains a set F of edges so that each vertex is incident with exactly two edges in F.

3.3. Cardinality bipartite matching algorithm

We now focus on the problem of finding a maximum-sized matching in a bipartite graph algorithmically. Basis is finding an 'augmenting path'.

Let M be a matching in a graph $G = (V, E)$. A path $P = (v_0, v_1, \ldots, v_t)$ in G is called *M-augmenting* if

(16)
(i) t is odd and $v_0, v_1, \ldots, v_t$ are all distinct;
(ii) $v_1v_2, v_3v_4, \ldots, v_{t-2}v_{t-1} \in M$;
(iii) $v_0, v_t \notin \bigcup M$.

Note that this implies that $v_0v_1, v_2v_3, \ldots, v_{t-1}v_t$ do not belong to M.

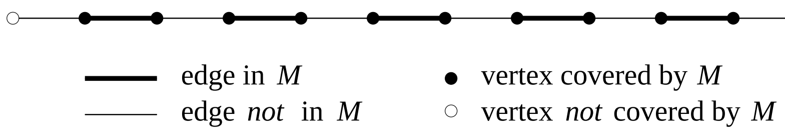


Figure 3.1

Clearly, if $P = (v_0, v_1, \ldots, v_t)$ is an M-augmenting path, then

$$M' := M \triangle EP \tag{17}$$

is a matching satisfying $|M'| = |M| + 1$.[8]

In fact, it is not difficult to show that:

Theorem 3.3. *Let $G = (V, E)$ be a graph and let M be a matching in G. Then either M is a matching of maximum cardinality, or there exists an M-augmenting path.*

Proof. If M is a maximum-cardinality matching, there cannot exist an M-augmenting path P, since otherwise $M \triangle EP$ would be a larger matching.

If M' is a matching larger than M, consider the components of the graph $G' := (V, M \cup M')$. As G' has maximum valency two, each component of G' is either a path (possibly of length 0) or a circuit. Since $|M'| > |M|$, at least one of these components should contain more edges of M' than of M. Such a component forms an M-augmenting path. ▮

So in any graph, if we have an algorithm finding an M-augmenting path for any matching M, then we can find a maximum cardinality matching: we iteratively find matchings $M_0, M_1, \ldots$, with $|M_i| = i$, until we have a matching M_k such that there does not exist any M_k-augmenting path.

We now describe how to find such an augmenting path in a bipartite graph.

Matching augmenting algorithm for bipartite graphs

input: a bipartite graph $G = (V, E)$ and a matching M,
output: a matching M' satisfying $|M'| > |M|$ (if there is one).
description of the algorithm: Let G have colour classes U and W. Orient each edge $e = \{u, w\}$ of G (with $u \in U, w \in W$) as follows:

(18) if $e \in M$ then orient e from w to u,
if $e \notin M$ then orient e from u to w.

Let D be the directed graph thus arising. Consider the sets

(19) $U' := U \setminus \bigcup M$ and $W' := W \setminus \bigcup M$.

Now an M-augmenting path (if it exists) can be found by finding a directed path in D from any vertex in U' to any vertex in W'. Hence in this way we can find a matching larger than M. ▮

The correctness of this algorithm is immediate. Since a directed path can be found in time $O(|E|)$, we can find an augmenting path in time $O(|E|)$. Hence a maximum cardinality matching in a bipartite graph can be found in time $O(|V||E|)$ (as we do at most $|V|$ iterations). Hopcroft and Karp [1973] gave an $O(|V|^{1/2}|E|)$ algorithm — see Section 4.2.

Application 3.1: Assignment problem. Suppose we have k machines at our disposal: $m_1, \ldots, m_k$. On a certain day we have to carry out n jobs: $j_1, \ldots, j_n$. Each machines is capable of performing some jobs, but can do only one job a day. E.g., we could have five machines $m_1, \ldots, m_5$ and five jobs $j_1, \ldots, j_5$ and the capabilities of the machines are indicated by crosses in the following table:

	j_1	j_2	j_3	j_4	j_5
m_1	X	X			X
m_2	X	X	X	X	
m_3	X	X			
m_4		X			
m_5		X			

[8] EP denotes the set of edges in P. $\triangle$ denotes symmetric difference.

We want to assign the machines to the jobs in such a way that every machine performs at most one job and that a largest number of jobs is carried out.

In order to solve this problem we represent the machines and jobs by vertices $m_1, \ldots, m_k$ and $j_1, \ldots, j_n$ of a bipartite graph $G = (V, E)$, and we make an edge from m_i to j_j if job j can be performed by machine i. Thus the example gives Figure 3.2. Then a maximum matching in G corresponds to a maximum assignment of jobs.

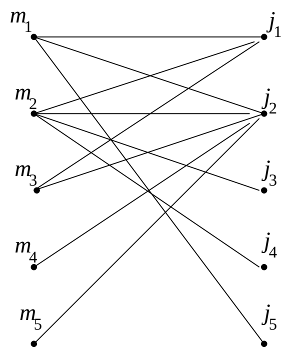


Figure 3.2

Exercises

3.17. Find a maximum matching and a minimum vertex cover in the bipartite graph in Figure 3.3.

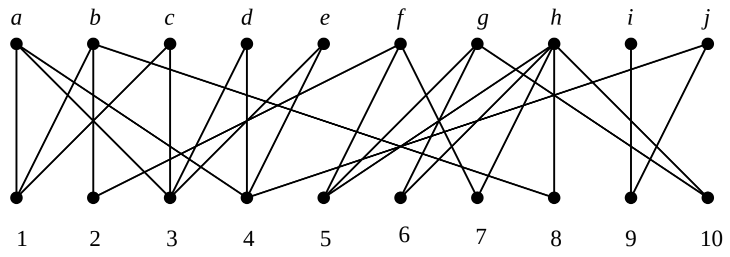


Figure 3.3

3.18. Solve the assignment problem given in Application 3.1.

3.19. Derive Kőnig's matching theorem from the cardinality matching algorithm for bipartite graphs.

3.20. Show that a minimum-size vertex cover in a bipartite graph can be found in polynomial time.

3.21. Show that, given a family of sets, a system of distinct representatives can be found in polynomial time (if it exists).

3.4. Weighted bipartite matching

We now consider the problem of finding a matching of maximum weight for which we describe the so-called *Hungarian method* developed by Kuhn [1955], using work of Egerváry [1931] (see Corollary 3.5b below).

Let $G = (V, E)$ be a graph and let $w : E \to \mathbb{R}$ be a 'weight' function. For any subset M of E define the *weight* $w(M)$ of M by

$$(20) \qquad w(M) := \sum_{e \in M} w(e).$$

The maximum-weight matching problem consists of finding a matching of maximum weight.

Again, augmenting paths are of help at this problem. Call a matching M *extreme* if it has maximum weight among all matchings of cardinality $|M|$.

Let M be an extreme matching. Define a 'length' function $l : E \to \mathbb{R}$ as follows:

(21) $$l(e) := w(e) \text{ if } e \in M, \quad l(e) := -w(e) \text{ if } e \notin M.$$

Then the following holds:

(22) Let P be an M-augmenting path of minimum length. Then $M' := M \triangle EP$ is extreme again

(Exercise 3.22).

This implies that if we are able to find a minimum-length M-augmenting path in polynomial time, we can find a maximum-weight matching in polynomial time: find iteratively extreme matchings $M_0, M_1, \ldots$ such that $|M_k| = k$ for each k. Then the matching among $M_0, M_1, \ldots$ of maximum weight is a maximum-weight matching.

If G is bipartite, we can find a minimum-length M-augmenting path as follows. Let G have colour classes U and W. Orient the edges of G as in (18), making the directed graph D, and let U' and W' as in (19). Then a minimum-length M-augmenting path can be found by finding a minimum-length path in D from any vertex in U' to any vertex in W'. This can be done in polynomial time since:

Theorem 3.4. *Let M be an extreme matching. Then D has no directed circuit of negative length.*

Proof. Suppose C is a directed circuit in D with length $l(C) < 0$. We may assume $C = (u_0, w_1, u_1, \ldots, w_t, u_t)$ with $u_0 = u_t$ and $u_1, \ldots, u_t \in U$ and $w_1, \ldots, w_t \in W$. Then the edges $w_1u_1, \ldots, w_tu_t$ belong to M and the edges $u_0w_1, u_1w_2, \ldots, u_{t-1}w_t$ do not belong to M. Then $M'' := M \triangle EC$ is a matching of cardinality k of weight $w(M'') = w(M) - l(C) > w(M)$, contradicting the fact that M is extreme. ∎

This gives a polynomial-time algorithm to find a maximum-weight matching in a bipartite graph. The description above yields an $O(|V|^2|E|)$ algorithm, since we do $O(|V|)$ iterations, each consisting of finding a shortest path (in a graph without negative-length directed circuits), which can be done in $O(|V||E|)$ time (with the Bellman-Ford algorithm — see Corollary 1.10a).

In fact, a sharpening of this method (by transmitting a 'potential' $p : V \to \mathbb{Q}$ throughout the matching augmenting iterations, making the length function l nonnegative, so that Dijkstra's method can be used) gives an $O(|V|(|E| + |V| \log |V|))$ algorithm.

Application 3.2: Optimal assignment. Suppose that we have n jobs and m machines and that each job can be done on each machine. Moreover, let a cost function (or cost matrix) $k_{i,j}$ be given, specifying the cost of performing job j by machine i. We want to perform the jobs with a minimum of total costs.

This can be solved with the maximum-weight bipartite matching algorithm. To this end, we make a complete bipartite graph G with colour classes of cardinality m and n. Let K be the maximum of $k_{i,j}$ over all i, j. Define the weight of the edge connecting machine i and job j to be equal to $K - k_{i,j}$. Then a maximum-weight matching in G corresponds to an optimum assignment of machines to jobs.

So the algorithm for solving the assignment problem counters the remarks made by Thorndike [1950] in an Address delivered on September 9, 1949 at a meeting of the American Psychological Association at Denver, Colorado:

> There are, as has been indicated, a finite number of permutations in the assignment of men to jobs. When the classification problem as formulated above was presented to a mathematician, he pointed to this fact and said that from the point of view of the mathematician there was no problem. Since the number of permutations was finite, one had only to try them all and choose the best. He dismissed the problem at that point. This is rather cold comfort to the

psychologist, however, when one considers that only ten men and ten jobs mean over three and a half million permutations. Trying out all the permutations may be a mathematical solution to the problem, it is not a practical solution.

Application 3.3: Transporting earth. Monge [1784] was one of the first to consider the assignment problem, in the role of the problem of transporting earth from one area to another, which he considered as the discontinuous, combinatorial problem of transporting molecules:

> Lorsqu'on doit transporter des terres d'un lieu dans un autre, on a coutime de donner le nom de *Déblai* au volume des terres que l'on doit transporter, & le nom de *Remblai* à l'espace qu'elles doivent occuper après le transport.
>
> Le prix du transport d'une molécule étant, toutes choses d'ailleurs égales, proportionnel à son poids & à l'espace qu'on lui fait parcourir, & par conséquent le prix du transport total devant être proportionnel à la somme des produits des molécules multipliées chacune par l'espace parcouru, il s'ensuit que le déblai & le remblai étant donné de figure & de position, il n'est pas indifférent que telle molécule du déblai soit transportée dans tel ou tel autre endroit du remblai, mais qu'il y a une certaine distribution à faire des molécules du premier dans le second, daprès laquelle la somme de ces produits sera la moindre possible, & le prix du transport total sera *minimum.*[9]

Monge describes an interesting geometric method to solve the assignment problem in this case: let l be a line touching the two areas from one side; then transport the earth molecule touched in one area to the position touched in the other area. Then repeat, until all molecules are transported.

Exercises

3.22. Prove (22).

3.23. Five mechanics, stationed in the cities A, B, C, D, E, have to perform jobs in the cities F, G, H, I, J. The jobs must be assigned in such a way to the mechanics that everyone gets one job and that the total distance traveled by them is as small as possible. The distances are given in the tables below. Solve these assignment problems with the weighted matching algorithm.

(i)

	F	G	H	I	J
A	6	17	10	1	3
B	9	23	21	4	5
C	2	8	5	0	1
D	19	31	19	20	9
E	21	25	22	3	9

(ii)

	F	G	H	I	J
A	11	5	21	7	18
B	17	4	20	9	25
C	4	1	3	2	4
D	6	2	19	3	9
E	19	7	23	18	26

3.24. Derive from the weighted matching algorithm for bipartite graphs an algorithm for finding a minimum-weight perfect matching in a bipartite graph $G = (V, E)$. (A matching M is *perfect* if $\bigcup M = V$.)

[9]When one must transport earth from one place to another, one usually gives the name of *Déblai* to the volume of earth that one must transport, & the name of *Remblai* to the space that they should occupy after the transport.

The price of the transport of one molecule being, if all the rest is equal, proportional to its weight & to the distance that one makes it covering, & hence the price of the total transport having to be proportional to the sum of the products of the molecules each multiplied by the distance covered, it follows that, the déblai & the remblai being given by figure and position, it makes difference if a certain molecule of the déblai is transported to one or to another place of the remblai, but that there is a certain distribution to make of the molcules from the first to the second, after which the sum of these products will be as little as possible, & the price of the total transport will be a *minimum.*

3.25. Let $A_1, \ldots, A_n$ be subsets of the finite set X and let $w : X \to \mathbb{R}_+$ be a 'weight' function. Derive from the weighted matching algorithm a polynomial-time algorithm to find a minimum-weight SDR.

3.5. The matching polytope

The weighted matching problem is related to the 'matching polytope'. Let $G = (V, E)$ be a graph. For each matching M let the *incidence vector* $\chi^M : E \to \mathbb{R}$ of M be defined by:

(23)
$$\begin{aligned}\chi^M(e) &:= 1 \text{ if } e \in M,\\ \chi^M(e) &:= 0 \text{ if } e \notin M,\end{aligned}$$

for $e \in E$.

It is important to realize that the set of functions $f : E \to \mathbb{R}$ can be considered as a vector space and each such function as a vector. Thus we can denote $f(e)$ by f_e. The function χ^M can be considered alternatively as a vector in the vector space $\mathbb{R}^E$. Similarly for functions $g : V \to \mathbb{R}$.

The *matching polytope* of G is defined as:

(24)
$$P_{\text{matching}}(G) := \text{conv.hull}\{\chi^M \mid M \text{ is a matching in } G\}.$$

So $P_{\text{matching}}(G)$ is a polytope in $\mathbb{R}^E$.

The matching polytope is a polyhedron, and hence can be described by linear inequalities. For bipartite graphs, these inequalities are quite simple. To this end it is convenient first to consider *perfect* matchings. A matching M is *perfect* if $\bigcup M = V$. Now the *perfect matching polytope* of G is defined by:

(25)
$$P_{\text{perfect matching}}(G) := \text{conv.hull}\{\chi^M \mid M \text{ is a perfect matching in } G\}.$$

Again, $P_{\text{perfect matching}}(G)$ is a polytope in $\mathbb{R}^E$. Now the following can be derived quite directly from Exercise 3.11:

Theorem 3.5. *Let $G = (V, E)$ be a bipartite graph. Then the perfect matching polytope $P_{\textit{perfect matching}}(G)$ is equal to the set of vectors $x \in \mathbb{R}^E$ satisfying:*

(26)
$$\begin{aligned} x_e &\geq 0 && \text{for each } e \in E;\\ \textstyle\sum_{e \ni v} x_e &= 1 && \text{for each } v \in V.\end{aligned}$$

Proof. Left to the reader (Exercise 3.26). ∎

Clearly, each vector x in $P_{\text{perfect matching}}(G)$ should satisfy (26), since each vector χ^M satisfies (26). The essence of the theorem is that the inequalities (26) are enough to define the polytope $P_{\text{perfect matching}}(G)$.

(An alternative way of proving Theorem 3.5 is using the 'total unimodularity' of the incidence matrix of a bipartite graph, together with the Hoffman-Kruskal theorem on integer solutions to linear programming problems with integer data and totally unimodular constraint matrix — see Section 8.3.)

From Theorem 3.5 one can derive the linear inequalities describing the matching polytope of a bipartite graph:

Corollary 3.5a. *Let $G = (V, E)$ be a bipartite graph. Then the matching polytope $P_{\textit{matching}}(G)$ is equal to the set of vectors $x \in \mathbb{R}^E$ satisfying:*

(27)
$$\begin{aligned} x_e &\geq 0 && \text{for each } e \in E;\\ \textstyle\sum_{e \ni v} x_e &\leq 1 && \text{for each } v \in V.\end{aligned}$$

Proof. Left to the reader (Exercise 3.27). ▮

Clearly, one cannot delete the bipartiteness condition: if G is the triangle K_3 then the function x defined by $x_e := 1/2$ for each edge e satisfies (27), but does not belong to the matching polytope.

Corollary 3.5a asserts that the weighted matching problem can be formulated as a linear programming problem:

(28)
$$\begin{array}{lrll} \text{maximize} & w^T x, & & \\ \text{subject to} & x_e & \geq 0 & \text{for each } e \in E; \\ & \sum_{e \ni v} x_e & \leq 1 & \text{for each } v \in V. \end{array}$$

With linear programming duality one can derive from this a 'weighted' extension of Kőnig's matching theorem, due to Egerváry [1931]:

Corollary 3.5b. *Let $G = (V, E)$ be a bipartite graph and let $w : E \to \mathbb{R}_+$ be a 'weight' function. Then the maximum weight of a matching is equal to the minimum value of $\sum_{v \in V} y(v)$, where y ranges over all functions $y : V \to \mathbb{R}_+$ satisfying $y(u) + y(v) \geq w(e)$ for each edge $e = uv$ of G.*

Proof. The maximum weight of a matching in G is equal to

(29) $$\max\{w^T \chi^M \mid M \text{ is a matching in } G\}.$$

Since $P_{\text{matching}}(G)$ is the convex hull of the χ^M, (29) is equal to

(30) $$\max\{w^T x \mid x \in P_{\text{matching}}(G)\}.$$

By Corollary 3.5a this is equal to

(31)
$$\begin{array}{lrll} \max\{w^T x \mid & x_e & \geq 0 & \text{for each } e \in E; \\ & \sum_{e \ni v} x_e & \leq 1 & \text{for each } v \in V\}. \end{array}$$

By linear programming duality this is equal to

(32)
$$\begin{array}{lrll} \min\{\sum_{v \in V} y_v \mid & y_v & \geq 0 & \text{for each } v \in V; \\ & y_u + y_v & \geq w_e & \text{for each edge } e = uv\}. \end{array}$$

This is exactly the minimum described in the Corollary. ▮

An extension of this corollary gives a further extension of Kőnig's matching theorem (Theorem 3.2):

Theorem 3.6. *In Corollary* 3.5b*, if w is integer-valued, then we can take also y integer-valued.*

Proof. Let $y \in \mathbb{R}_+^V$ attain the minimum, and assume that we have chosen y so that the number of vertices v with $y_v \notin \mathbb{Z}$ is as small as possible. Let U and W be the two colour classes of G and let X be the set of vertices v of G with $y_v \notin \mathbb{Z}$. If $X = \emptyset$ we are done, so assume that $X \neq \emptyset$. Without loss of generality, $|X \cap U| \geq |X \cap W|$. Let u be a vertex in $X \cap U$ with $y_u - \lfloor y_u \rfloor$ as small as possible. Let $\varepsilon := y_u - \lfloor y_u \rfloor$. Reset

(33)
$$\begin{array}{ll} \tilde{y}_v := y_v - \varepsilon & \text{if } v \in X \cap U, \\ \tilde{y}_v := y_v + \varepsilon & \text{if } v \in X \cap W, \\ \tilde{y}_v := y_v & \text{if } v \notin X. \end{array}$$

One easily checks that again $\tilde{y}_v + \tilde{y}_{v'} \geq w(e)$ for each edge $e = vv'$ of G (using the fact that w is integer-valued). Moreover, $\sum_{v \in V} \tilde{y}_v = \sum_{v \in V} y_v - \varepsilon|X \cap U| + \varepsilon|X \cap W| \leq \sum_{v \in V} y_v$. So $\tilde{y}$ also attains the minimum. However, $\tilde{y}$ has fewer noninteger-valued components than y (as $\tilde{y}_u \in \mathbb{Z}$), contradicting our assumption. ▮

Exercises

3.26. Derive Theorem 3.5 from Exercise 3.11.

3.27. Derive Corollary 3.5a from Theorem 3.5.

4. Menger's theorem, flows, and circulations

4.1. Menger's theorem

Let $D = (V, A)$ be a directed graph and let s and t be vertices of D. A path is called an $s-t$ *path* if it begins in s and ends in t.

In this section we study the maximum number k of pairwise disjoint $s-t$ paths in D. Here 'disjoint' can mean: *internally vertex-disjoint* (= having no vertex in common except for s and t) or *arc-disjoint* (= having no arc in common).

A basic result is Menger's theorem (Menger [1927], Kőnig [1931]) on the maximum number of pairwise internally vertex-disjoint $s-t$ paths. We first show however the arc-disjoint version (which can be shown to be equivalent to the vertex-disjoint version).

To formulate this theorem, we say that a set A' of arcs is an $s-t$ *cut* if $A' = \delta^{\text{out}}(U)$ for some subset U of V with $s \in U$ and $t \notin U$.[10]

Theorem 4.1 (Menger's theorem (directed arc-disjoint form))**.** *Let $D = (V, A)$ be a directed graph and let $s, t \in V$. Then the maximum number of pairwise arc-disjoint $s-t$ paths is equal to the minimum cardinality of any $s-t$ cut.*

Proof. Clearly, the maximum is not more than the minimum. To see the reverse inequality, let $P_1, \ldots, P_k$ be a maximum number of pairwise arc-disjoint $s-t$ paths in D. We may assume that they are simple. Let A' be the set of arcs occurring in these paths. So, in $D' = (V, A')$, the indegree of any vertex $v \neq s, t$ is equal to its outdegree, while s has indegree 0 and outdegree k, and t has indegree k and outdegree 0. Let $\tilde{D}$ be the directed graph arising from D by reversing the orientation of each of the arcs in A'.

Then $\tilde{D}$ has no $s-t$ path. For suppose Q is an $s-t$ path in $\tilde{D}$. Let A_1 be the set of arcs in Q that belong to $A \setminus A'$, and let $A_2 := AQ \setminus A_1$. Let $A'' := (A' \cup A_1) \setminus A_2^{-1}$. Then in $D'' = (V, A'')$, the indegree of any vertex $v \neq s, t$ is equal to its outdegree, while s has indegree 0 and outdegree $k+1$, and t has indegree $k+1$ and outdegree 0. This however implies that A'' contains $k+1$ pairwise arc-disjoint $s-t$ paths, contradicting the maximality of k.

So $\tilde{D}$ has no $s-t$ path. Let U be the set of vertices that are reachable in $\tilde{D}$ from s. So $s \in U$ and $t \notin U$. Since no arc of $\tilde{D}$ leaves U, we have that no arc in A' enters U, and hence $|\delta^{\text{out}}_{A'}(U)| = k$. Moreover, no arc in $A \setminus A'$ leaves U, and hence $\delta^{\text{out}}_{A}(U) = \delta^{\text{out}}_{A'}(U)$. Therefore we have $|\delta^{\text{out}}_{A}(U)| = |\delta^{\text{out}}_{A'}(U)| = k$. ∎

The proof also directly yields a polynomial-time algorithm for finding a maximum number of arc-disjoint $s-t$ paths. We treat this in further detail in Section 4.2.

The following are direct corollaries of Theorem 4.1. Let $D = (V, A)$ be a directed graph and let $s, t \in V$. A subset W of V is called $s-t$ *disconnecting* if each $s-t$ path has at least one vertex in common with W.

Corollary 4.1a (Menger's theorem (directed vertex-disjoint form))**.** *Let $D = (V, A)$ be a directed graph and let s and t be two nonadjacent vertices of D. Then the maximum number of internally vertex-disjoint $s-t$ paths is equal to the minimum cardinality of any $s-t$ disconnecting subset of $V \setminus \{s, t\}$.*

Proof. Make an auxiliary directed graph D' as follows. Replace each vertex $v \neq s, t$ by two vertices v' and v'', and redirect each arc with head v to v' and redirect each arc with tail v from v''; moreover add an arc from v' to v''.

[10] $\delta^{\text{out}}(U)$ and $\delta^{\text{in}}(U)$ denote the sets of arcs leaving U and entering U, respectively.

Then the maximum number of internally vertex-disjoint $s-t$ paths in D is equal to the maximum number of arc-disjoint $s-t$ paths in D'. Similarly, the minimum size of an $s-t$ disconnecting subset of $V \setminus \{s,t\}$ in D is not more than the minimum size of an $s-t$ cut in D'. Hence Theorem 4.1 implies the Corollary. ▮

Similarly as before this gives a polynomial-time algorithm to find a maximum number of pairwise internally vertex-disjoint $s-t$ paths.

Note that the arc-disjoint version of Menger's theorem can be derived in turn from the vertex-disjoint version. Similar theorems hold for *undirected* graphs. They can be derived from the directed case by replacing each undirected edge uw by two opposite arcs (u,w) and (w,u).

Application 4.1: Routing airplanes. An airline company carries out a certain number of flights according to some fixed timetable, in a weekly cycle. The timetable is basically given by a flight number i (for instance 562), a departure city dc_i (for instance Vancouver), a departure time dt_i (for instance Monday 23.15h), an arrival city ac_i (for instance Tokyo), and an arrival time at_i (for instance Tuesday 7.20h). All times include boarding and disembarking and preparing the plane for a next flight. Thus a plane with arrival time Tuesday 7.20h at city c, can be used for any flight from c with departure time from Tuesday 7.20h on.

The flights are carried out by n airplanes of one type, denoted by $a_1, \ldots, a_n$. At each weekday there should be an airplane for maintenance at the home basis, from 6.00h till 18.00h. Legal rules prescribe which of the airplanes $a_1, \ldots, a_n$ should be at the home basis during one day the coming week, but it is not prescribed which airplane should be at the home basis at which day (see Application 9.4 for an extension where this *is* prescribed).

The timetable is made in such a way that at each city the number of incoming flights is equal to the number of outgoing flights. Here 'maintenance' is also considered as a flight. However, there is flexibility in assigning the airplanes to the flights: if at a certain moment at a certain city two or more airplanes are available for a flight, in principle any of them can be used for that flight. Which of the available airplanes will be used, is decided by the main office of the company. This decision is made at 18.00h on the Saturday before. At that time the company makes the exact routing of the planes for the coming week.

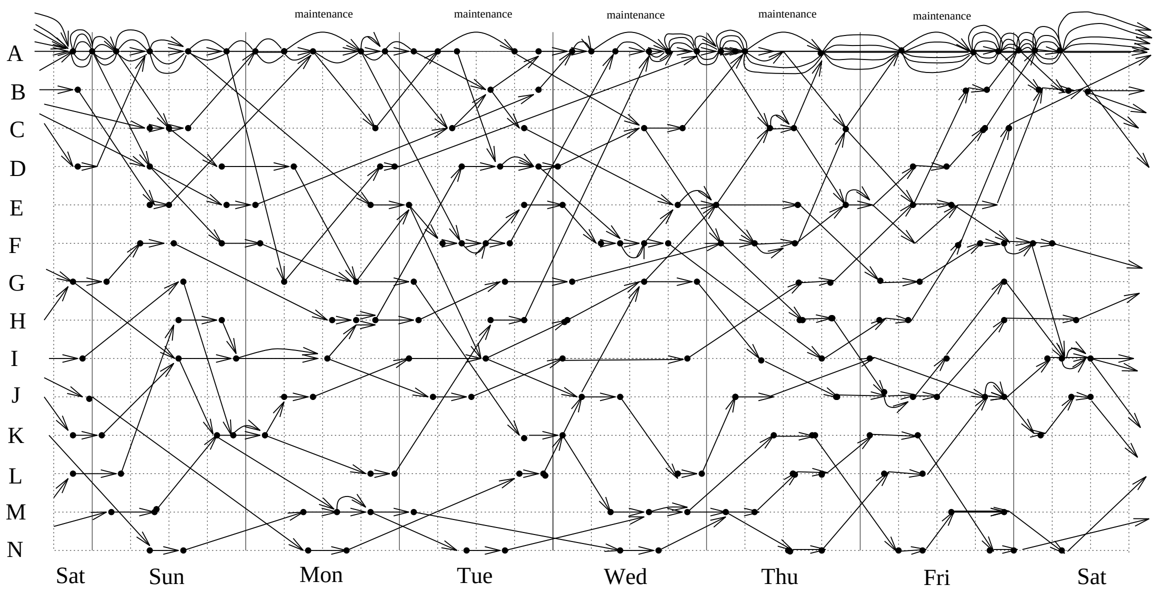


Figure 4.1

At that moment, certain planes are performing certain flights, while other planes are grounded at certain cities. Routing the airplanes is easy as the timetable is set up in such a way that at each moment and each city enough airplanes are available.

Indeed, one can make a directed graph D (Figure 4.1) with vertex set all pairs (dc_i, dt_i) and (ac_i, at_i) for all flight numbers i. For each flight i that is not in the air at Saturday 18.00h, one makes an arc from (dc_i, dt_i) to (ac_i, at_i). We also do this for the "flights" representing maintenance.

Moreover, for each city c and each two consecutive times t, t' at which any flight departs or arrives at c, one makes m parallel arcs from (c, t) to (c, t'), where m is the number of airplanes that will be in city c during the period t–t'.

In this way we obtain a directed graph such that at each vertex the indegree is equal to the outdegree, except at any (c, t_c) where t_c is the earliest time after Saturday 18.00h at which any flight arrives at or leaves city c. Hence we can find in D arc-disjoint paths $P_1, \ldots, P_n$ (where n is the number of airplanes) in D such that each arc is in exactly one of the P_i. This would give a routing for the airplanes.

However, the restriction that some prescribed airplanes must undergo maintenance the coming week gives some complications. It means for instance that if a certain airplane a_i (say) is now on the ground at city c and should be home for maintenance the coming week, then the path P_i should start at (c, t_c) and should traverse one of the arcs representing maintenance. If a_i is now in the air, then path P_i should start at (c, t) where t is the first-coming arrival time of a_i and should traverse a maintenance arc. So the company first finds arc-disjoint paths $P_{i_1}, \ldots, P_{i_k}$, where $a_{i_1}, \ldots, a_{i_k}$ are the airplanes that should undergo maintenance the coming week. These paths can be extended to paths $P_1, \ldots, P_n$ such that each arc is traversed exactly once.

So the problem can be solved by finding arc-disjoint paths starting in a given set of vertices and ending in a given set of vertices (by slightly extending the graph D).

Exercises

4.1. Let $D = (V, A)$ be a directed graph and let $r, s_1, \ldots, s_k$ be vertices of D. Prove that there exist pairwise arc-disjoint paths $P_1, \ldots, P_k$ such that P_i is an $r - s_i$ path ($i = 1, \ldots, k$), if and only if for each $U \subseteq V$ with $r \in U$ one has

(1) $$|\delta^{\text{out}}(U)| \geq |\{i \mid s_i \notin U\}|.$$

4.2. Let $\mathcal{A} = (A_1, \ldots, A_n)$ and $\mathcal{B} = (B_1, \ldots, B_n)$ be families of subsets of a finite set. Show that $\mathcal{A}$ and $\mathcal{B}$ have a common SDR, if and only if for all $I, J \subseteq \{1, \ldots, n\}$ one has

(2) $$|\bigcup_{i \in I} A_i \cap \bigcup_{j \in J} B_j| \geq |I| + |J| - n.$$

4.3. Let $G = (V, E)$ be a bipartite graph, with colour classes V_1 and V_2, such that $|V_1| = |V_2|$. Show that G has k pairwise disjoint perfect matchings, if and only if for each subset U of V_1:

(3) $$\sum_{v \in V_2} \min\{k, |E(v) \cap U|\} \geq k|U|,$$

where $E(v)$ denotes the set of vertices adjacent to v.

4.4. Let $D = (V, A)$ be a simple directed graph and let $s, t \in V$. Let α be the minimum length of an $s - t$ path. Show that the maximum number of pairwise arc-disjont $s - t$ paths is at most $(|V|/\alpha)^2$.
(*Hint*: Let U_k denote the set of vertices at distance at most k from s. Show that $|\delta^{\text{out}}(U_k)| \leq (|V|/\alpha)^2$ for some $k < \alpha$.)

4.2. Path packing algorithmically

Let $D = (V, A)$ be a directed graph, and let $s, t \in V$. The proof of Theorem 4.1 gives directly a polynomial-time algorithm to find a maximum number of pairwise arc-disjoint $s - t$ paths. To this end, let, for any directed graph D and any path P in D, the graph D/P arise from D by reversing the orientation of each arc occurring in P.

We determine $D_0, D_1, \ldots$ as follows. Set $D_0 := D$. If D_k has been found and contains an $s - t$ path P, set $D_{k+1} := D_k/P$. If D_k does not contain any $s - t$ path we stop.

Now, as in the proof of Theorem 4.1, the set of arcs of D that are reversed in the final D_k forms a maximum number of arc-disjoint $s-t$ paths in D. For the discussion below it is important to observe that it follows similarly, that for any i, the set of arcs of D_i that are reversed in the final D_k (compared with D_i) forms a maximum number of arc-disjoint $s-t$ paths in D_i.

Since an $s-t$ path in D_k can be found in time $O(|A|)$, and since the maximum number of arc-disjoint $s-t$ paths is at most $|A|$, the algorithm described has running time $O(|A|^2)$.

The process might be fastened by selecting, at each iteration, not just *one* path P, but several arc-disjoint paths $P_1, \ldots, P_l$ in D_k at one blow, and setting

$$(4) \qquad D_{k+1} := D_k/P_1/\cdots/P_l.$$

This might give a reduction of the number of iterations — but of course this should be weighed against the increase in complexity of each iteration.

Such a fastening is obtained by a method of Dinits [1970] as follows. For any directed graph $D = (V, A)$ and $s, t \in V$, let $\mu(D)$ denote the minimum length of an $s-t$ path in D. (If no such path exists, set $\mu(D) = \infty$.) If we choose the paths $P_1, \ldots, P_l$ in such a way that $\mu(D_{k+1}) > \mu(D_k)$, then the number of iterations clearly is not larger than $|V|$ (as $\mu(D_k) < |V|$ for each k). In fact, as Even and Tarjan [1975] noted, in that case there are the following better bounds on the total number N of iterations:

Theorem 4.2. *If $\mu(D_{k+1}) > \mu(D_k)$ for each $k < N$, then $N \leq 2|A|^{1/2}$. If moreover D is simple, then $N \leq 2|V|^{2/3}$.*

Proof. Let $k := \lfloor |A|^{1/2} \rfloor$. So each $s-t$ path in D_k has length at least $|A|^{1/2}$. Hence D_k contains at most $|A|/|A|^{1/2} = |A|^{1/2}$ pairwise arc-disjoint $s-t$ paths. Therefore $N - k \leq |A|^{1/2}$, and hence $N \leq 2|A|^{1/2}$.

If D is simple, then let $k := \lfloor |V|^{2/3} \rfloor$. So each $s-t$ path in D_k has length at least $|V|^{2/3}$. From Exercise 4.4 we know that D_k contains at most $(|V|/|V|^{2/3})^2 = |V|^{2/3}$ pairwise arc-disjoint paths. Therefore $N - k \leq |V|^{2/3}$, and hence $N \leq 2|V|^{2/3}$. ∎

We show that a collection $P_1, \ldots, P_l$ with the property that $\mu(D/P_1/\cdots/P_l) > \mu(D)$ indeed can be found quickly, namely in linear time.

To that end, call a collection of arc-disjoint $s-t$ paths $P_1, \ldots, P_l$ *blocking* if deleting in D all arcs occurring in the P_i gives a directed graph with no $s-t$ path. This is weaker than a maximum number of arc-disjoint paths, but can be found in linear time. (This gives a fast 'heuristic' for finding a large number of arc-disjoint paths. Such heuristics go back to the 'flooding technique' of Boldyreff [1955], while Dinits [1970] and Karzanov [1974] gave fast implementations.)

Theorem 4.3. *Given a directed graph $D = (V, A)$ and $s, t \in V$, a blocking collection of arc-disjoint $s-t$ paths can be found in time $O(|A|)$.*

Proof. With depth-first search we can find in time $O(|A'|)$ a subset A' of A and an $s-t$ path P_1 in A' such that each $s-t$ path in D intersecting A' also intersects AP_1.[11]

Next we find (recursively) a blocking collection $P_2, \ldots, P_k$ of arc-disjoint $s-t$ paths in the graph $D' := (V, A \setminus A')$. Then $P_1, \ldots, P_k$ is blocking in D. For suppose not. Then D contains an $s-t$ path Q that is arc-disjoint from $P_1, \ldots, P_k$. Then $AQ \cap A' \neq \emptyset$, since $P_2, \ldots, P_k$ is blocking in D'.

[11] To this end, define the operation of *scanning* a vertex v recursively by:

(5) For each arc $a = (v, w) \in \delta^{\text{out}}(v)$: reset $A' := A' \cup \{a\}$; if $w = t$ stop; otherwise scan w.

Now starting with $A' = \emptyset$, scan s, until we get the stop signal. This gives the required A' and $s-t$ path P in A', in time $O(|A'|)$.

So AQ intersects AP_1, a contradiction. ∎

This implies:

Corollary 4.3a. *Given a directed graph $D = (V, A)$ and $s, t \in V$, a collection of arc-disjoint $s - t$ paths $P_1, \ldots, P_l$ such that $\mu(D/P_1/\cdots/P_l) > \mu(D)$ can be found in time $O(|A|)$.*

Proof. Let $\tilde{D}$ be the subgraph of D consisting of all arcs of D that occur in at least one shortest $s - t$ path. These arcs can be identified in time $O(|A|)$.

By Theorem 4.3 we can find in time $O(|A|)$ a blocking collection $P_1, \ldots, P_l$ in $\tilde{D}$. Then $\mu(D/P_1/\cdots/P_l) > \mu(D)$. For suppose $\mu(D/P_1/\cdots/P_l) \leq \mu(D)$. Let for each $v \in V$, $d(v)$ be the minimum length of an $s - v$ path in D. Let $v_0, a_1, v_1, \ldots, a_m, v_m$ be an $s - t$ path in $D/P_1/\cdots/P_l$ with $m \leq d(t)$.

Then for each $i = 1, \ldots, m$, if a_i is an arc of D, then $d(v_i) \leq d(v_{i-1}) + 1$; if a_i^{-1} is an arc of D, then $d(v_{i-1}) = d(v_i) + 1$, since a_i^{-1} belongs to one of the P_j.

Now at least one of the a_i is not an arc of D (as $P_1, \ldots, P_l$ is blocking in $\tilde{D}$). Hence $m > d(v_m) = d(t)$, a contradiction. ∎

This gives the following result of Even and Tarjan [1975]:

Theorem 4.4. *Given a directed graph $D = (V, A)$ and $s, t \in V$, a maximum number of pairwise arc-disjoint $s - t$ paths can be found in time $O(|A|^{3/2})$. If D is simple, the paths can be found also in time $O(|V|^{2/3}|A|)$.*

Proof. Directly from Corollary 4.3a and Theorem 4.2. ∎

The vertex-disjoint case. If we are interested in *vertex*-disjoint paths, the results can be sharpened. Note that if $D = (V, A)$ is a directed graph and $s, t \in V$, then the problem of finding a maximum number of pairwise internally vertex-disjoint $s - t$ paths can be reduced to the arc-disjoint case by replacing each vertex $v \neq s, t$ by two vertices v', v'', while each arc with head v is redirected to v' and each arc with tail v is redirected from v''; moreover, an arc (v', v'') is added.

By Theorem 4.4, this construction directly yields algorithms with running time $O(|A|^{3/2})$ and $O(|V|^{2/3}|A|)$. But one can do better. Note that, with this construction, each of the directed graphs D_k has the property that each vertex has indegree at most 1 or outdegree at most 1. Under this condition, the bound in Theorem 4.2 can be improved to $2|V|^{1/2}$. Hence we have similarly to Theorem 4.4 another result of Even and Tarjan [1975]:

Theorem 4.5. *Given a directed graph $D = (V, A)$ and $s, t \in V$, a maximum number of pairwise internally vertex-disjoint $s - t$ paths can be found in time $O(|V|^{1/2}|A|)$.*

Proof. Similarly to Theorem 4.4. ∎

As a corollary one has the result of Hopcroft and Karp [1973]:

Corollary 4.5a. *In a bipartite graph $G = (V, E)$, a maximum matching can be found in time $O(|V|^{1/2}|E|)$.*

Proof. Make a directed graph $D = (V, A)$ as follows. Let U and W be the colour classes of G. Orient all edges from U to W. Moreover, add a new vertex s, with arcs (s, u) for all $u \in U$, and a new vertex t, with arcs (w, t) for all $w \in W$. Then the maximum number of pairwise internally vertex-disjoint $s - t$ paths in D is equal to the maximum size of a matching in G. The result follows now from Theorem 4.5. ∎

Exercises

4.5. Show that in a bipartite graph $G = (V, E)$ with colour classes V_1 and V_2, a maximum matching can be found in time $O(|V_1|^{1/2}|E|)$.

4.3. Flows in networks

Other consequences of Menger's theorem concern 'flows in networks'. Let $D = (V, A)$ be a directed graph and let $r, s \in V$. A function $f : A \to \mathbb{R}$ is called an $r - s$ *flow* if:[12]

$$(6)\qquad \begin{array}{lrcll} \text{(i)} & f(a) & \geq & 0 & \text{for each } a \in A; \\ \text{(ii)} & \displaystyle\sum_{a\in\delta^{\text{in}}(v)} f(a) & = & \displaystyle\sum_{a\in\delta^{\text{out}}(v)} f(a) & \text{for each } v \in V \setminus \{r, s\}. \end{array}$$

Condition (6)(ii) is called the *flow conservation law*: the amount of flow entering a vertex $v \neq r, s$ should be equal to the amount of flow leaving v.

The *value* of an $r - s$ flow f is, by definition:

$$(7)\qquad \text{value}(f) := \sum_{a\in\delta^{\text{out}}(r)} f(a) - \sum_{a\in\delta^{\text{in}}(r)} f(a).$$

So the value is the net amount of flow leaving r. It can be shown that it is equal to the net amount of flow entering s.

Let $c : A \to \mathbb{R}_+$, called a *capacity function*. We say that a flow f is *under* c (or *subject to* c) if

$$(8)\qquad f(a) \leq c(a) \text{ for each } a \in A.$$

The *maximum flow problem* now is to find an $r - s$ flow under c, of maximum value.

To formulate a min-max relation, define the *capacity* of a cut $\delta^{\text{out}}(U)$ by:

$$(9)\qquad c(\delta^{\text{out}}(U)) := \sum_{a\in\delta^{\text{out}}(U)} c(a).$$

Then:

Proposition 1. *For every flow f and every cut $\delta^{\text{out}}(W)$ one has:*

$$(10)\qquad \text{value}(f) \leq c(\delta^{\text{out}}(W)).$$

Proof.

$$(11)\qquad \begin{aligned} \text{value}(f) &= \sum_{a\in\delta^{\text{out}}(r)} f(a) - \sum_{a\in\delta^{\text{in}}(r)} f(a) \\ &= \sum_{a\in\delta^{\text{out}}(r)} f(a) - \sum_{a\in\delta^{\text{in}}(r)} f(a) + \sum_{v\in W\setminus\{r\}} \Big(\sum_{a\in\delta^{\text{out}}(v)} f(a) - \sum_{a\in\delta^{\text{in}}(v)} f(a)\Big) \\ &= \sum_{v\in W}\Big(\sum_{a\in\delta^{\text{out}}(v)} f(a) - \sum_{a\in\delta^{\text{in}}(v)} f(a)\Big) = \sum_{a\in\delta^{\text{out}}(W)} f(a) - \sum_{a\in\delta^{\text{in}}(W)} f(a) \\ &\overset{\star}{\leq} \sum_{a\in\delta^{\text{out}}(W)} f(a) \overset{\star\star}{\leq} \sum_{a\in\delta^{\text{out}}(W)} c(a) = c(\delta^{\text{out}}(W)). \end{aligned}$$

∎

[12] $\delta^{\text{out}}(v)$ and $\delta^{\text{in}}(v)$ denote the sets of arcs leaving v and entering v, respectively.

It is convenient to note the following:

(12) $$\text{equality holds in (10)} \iff \begin{array}{l}\forall a \in \delta^{\text{in}}(W) : f(a) = 0 \text{ and} \\ \forall a \in \delta^{\text{out}}(W) : f(a) = c(a).\end{array}$$

This follows directly from the inequalities $\star$ and $\star\star$ in (11).

Now from Menger's theorem one can derive that equality can be attained in (10), which is a theorem of Ford and Fulkerson [1956]:

Corollary 4.5b (max-flow min-cut theorem)**.** *For any directed graph $D = (V, A)$, $r, s \in V$, and $c : A \to \mathbb{R}_+$, the maximum value of an $r - s$ flow under c is equal to the minimum capacity of an $r - s$ cut. In formula:*

(13) $$\max_{f \text{ } r\text{-}s \text{ flow}} \text{value}(f) = \min_{\delta^{\text{out}}(U) \text{ } r\text{-}s \text{ cut}} c(\delta^{\text{out}}(U)).$$

Proof. If c is integer-valued, the corollary follows from Menger's theorem by replacing each arc a by $c(a)$ parallel arcs. If c is rational-valued, there exists a natural number N such that $Nc(a)$ is integer for each $a \in A$. This resetting multiplies both the maximum and the minimum by N. So the equality follows from the case where c is integer-valued.

If c is real-valued, we can derive the corollary from the case where c is rational-valued, by continuity and compactness arguments. ▮

Moreover, one has (Dantzig [1951a]):

Corollary 4.5c (Integrity theorem)**.** *If c is integer-valued, there exists an integer-valued maximum flow.*

Proof. Directly from Menger's theorem. ▮

Exercises

4.6. Let $D = (V, A)$ be a directed graph and let $r, s \in V$. Let $f : A \to \mathbb{R}_+$ be an $r - s$ flow of value β. Show that there exists an $r - s$ flow $f' : A \to \mathbb{Z}_+$ of value $\lceil \beta \rceil$ such that $\lfloor f(a) \rfloor \leq f'(a) \leq \lceil f(a) \rceil$ for each arc a. (Integer flow theorem (Dantzig [1951a]).)

4.7. Let $G = (V, E)$ be a graph and let $c : E \to \mathbb{R}_+$ be a 'capacity' function. Let K be the complete graph on V. For each edge rs of K, let $w(rs)$ be the minimum capacity of any $r - s$ cut in G. [An $r - s$ cut is any subset $\delta(W)$ with $r \in W, s \notin W$.]

Let T be a spanning tree in K of maximum total weight with respect to the function w. Prove that for all $r, s \in V$, $w(rs)$ is equal to the minimum weight of the edges of T in the unique $r - s$ path in T. (*Hint:* Use Exercise 1.10.)

4.4. Finding a maximum flow

Let $D = (V, A)$ be a directed graph, let $r, s \in V$, and let $c : A \to \mathbb{Q}_+$ be a 'capacity' function. We now describe the algorithm of Ford and Fulkerson [1956] to find an $r - s$ flow of maximum value under c.

In this section, by *flow* we will mean an $r - s$ flow under c, and by *cut* an $r - s$ cut. A *maximum flow* is a flow of maximum value.

We now describe the algorithm of Ford and Fulkerson [1957] to determine a maximum flow. We assume that $c(a) > 0$ for each arc a. First we give an important subroutine:

Flow augmenting algorithm

input: a flow f.
output: either (i) a flow f' with value$(f') >$ value(f),
or (ii) a cut $\delta^{\text{out}}(W)$ with $c(\delta^{\text{out}}(W)) =$ value(f).
description of the algorithm: For any pair $a = (v, w)$ define $a^{-1} := (w, v)$. Make an auxiliary graph $D_f = (V, A_f)$ by the following rule: for any arc $a \in A$,

(14) $$\begin{array}{l} \text{if } f(a) < c(a) \text{ then } a \in A_f, \\ \text{if } f(a) > 0 \text{ then } a^{-1} \in A_f. \end{array}$$

So if $0 < f(a) < c(a)$ then both a and a^{-1} are arcs of A_f.

Now there are two possibilities:

(15) **Case 1:** *There exists an $r - s$ path in D_f,*

Case 2: *There is no $r - s$ path in D_f.*

Case 1: *There exists an $r - s$ path $P = (v_0, a_1, v_1, \ldots, a_t, v_t)$ in $D_f = (V, A_f)$.*
So $v_0 = r$ and $v_t = s$. We may assume that P is a simple path. As $a_1, \ldots, a_t$ belong to A_f, we know by (14) that for each $i = 1, \ldots, t$:

(16) $$\begin{array}{rl} \text{either (i)} & a_i \in A \text{ and } \sigma_i := c(a_i) - f(a_i) > 0 \\ \text{or (ii)} & a_i^{-1} \in A \text{ and } \sigma_i := f(a_i^{-1}) > 0. \end{array}$$

Define $\varepsilon := \min\{\sigma_1, \ldots, \sigma_t\}$. So $\varepsilon > 0$. Let $f' : A \to \mathbb{R}_+$ be defined by, for $a \in A$:

(17) $$\begin{array}{rcll} f'(a) & := & f(a) + \varepsilon, & \text{if } a = a_i \text{ for some } i = 1, \ldots, t; \\ & := & f(a) - \varepsilon, & \text{if } a = a_i^{-1} \text{ for some } i = 1, \ldots, t; \\ & := & f(a), & \text{for all other } a. \end{array}$$

Then f' again is an $r - s$ flow under c. The inequalities $0 \leq f'(a) \leq c(a)$ hold because of our choice of ε. It is easy to check that also the flow conservation law (6)(ii) is maintained.

Moreover,

(18) $$\text{value}(f') = \text{value}(f) + \varepsilon,$$

since either $(v_0, v_1) \in A$, in which case the outgoing flow in r is increased by ε, or $(v_1, v_0) \in A$, in which case the ingoing flow in r is decreased by ε.

Path P is called a *flow augmenting path.*

Case 2: *There is no path in $D_f = (V, A_f)$ from r to s.*
Now define:

(19) $$W := \{w \in V \mid \text{there exists a path in } D_f \text{ from } r \text{ to } w\}.$$

Then $r \in W$ while $s \notin W$, and so $\delta^{\text{out}}(W)$ is an $r - s$ cut.

By definition of W, if $u \in W$ and $v \notin W$, then $(u, v) \notin A_f$ (as otherwise also v would belong to W). Therefore:

(20) $$\begin{array}{l} \text{if } (u, v) \in \delta^{\text{out}}(W), \text{ then } (u, v) \notin A_f, \text{ and so (by (14)): } f(u, v) = c(u, v), \\ \text{if } (u, v) \in \delta^{\text{in}}(W), \text{ then } (v, u) \notin A_f, \text{ and so (by (14)): } f(u, v) = 0. \end{array}$$

Then (12) gives:

(21) $$c(\delta^{\text{out}}(W)) = \text{value}(f).$$ ▮

This finishes the description of the flow augmenting algorithm. The description of the *(Ford-Fulkerson) maximum flow algorithm* is now simple:

Maximum flow algorithm

input: directed graph $D = (V, A), r, s \in V, c : A \to \mathbb{R}_+$.
output: a maximum flow f and a cut $\delta^{\text{out}}(W)$ of minimum capacity, with value$(f) = c(\delta^{\text{out}}(W))$.
description of the algorithm: Let f_0 be the 'null flow' (that is, $f_0(a) = 0$ for each arc a). Determine with the flow augmenting algorithm flows $f_1, f_2, \ldots, f_N$ such that $f_{i+1} = f_i'$, until, in the Nth iteration, say, we obtain output (ii) of the flow augmenting algorithm. Then we have flow f_N and a cut $\delta^{\text{out}}(W)$ with the given properties. ▌

We show that the algorithm terminates, provided that all capacities are rational.

Theorem 4.6. *If all capacities $c(a)$ are rational, the algorithm terminates.*

Proof. If all capacities are rational, there exists a natural number K so that $Kc(a)$ is an integer for each $a \in A$. (We can take for K the l.c.m. of the denominators of the $c(a)$.)

Then in the flow augmenting iterations, every flow $f_i(a)$ and every ε is a multiple of $1/K$. So at each iteration, the flow value increases by at least $1/K$. Since the flow value cannot exceed $c(\delta^{\text{out}}(\{r\}))$, we can have only finitely many iterations. ▌

We should note here that this theorem is not true if we allow general real-valued capacities.

In Section 4.5 we shall see that if we choose always a shortest path as flow augmenting path, then the algorithm has polynomially bounded running time.

Note that the algorithm also implies the max-flow min-cut theorem (Theorem 4.5b). Note moreover that in the maximum flow algorithm, if all capacities are integer, then the maximum flow will also be integer-valued. So it also implies the integrity theorem (Corollary 4.5c).

Application 4.2: Transportation problem. Suppose there are m factories, that all produce the same product, and n customers that use the product. Each month, factory i can produce s_i tons of the product. Customer j needs every month d_j tons of the product. From factory i to customer j we can transport every month at most $c_{i,j}$ tons of the product. The problem is: can the needs of the customers be fulfilled?

In order to solve the problem with the maximum-flow algorithm, we make the graph as in Figure 4.2 (for $m = 3, n = 5$):

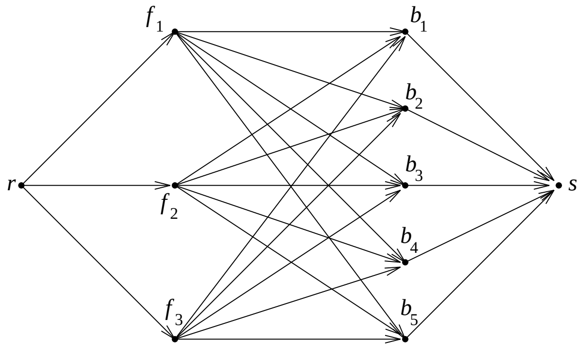


Figure 4.2

We define a capacity function c on the arcs as follows:

$$\begin{array}{llll} (22) \quad & c(r, f_i) & := s_i & \text{for } i = 1, \ldots, m, \\ & c(f_i, b_j) & := c_{i,j} & \text{for } i = 1, \ldots, m; j = 1, \ldots, n, \\ & c(b_j, s) & := d_j & \text{for } j = 1, \ldots, n. \end{array}$$

Now we have:

(23) the needs of the customers can be fulfilled $\Longleftrightarrow$ there is an $r-s$ flow under c with value $d_1+\cdots+d_n$.

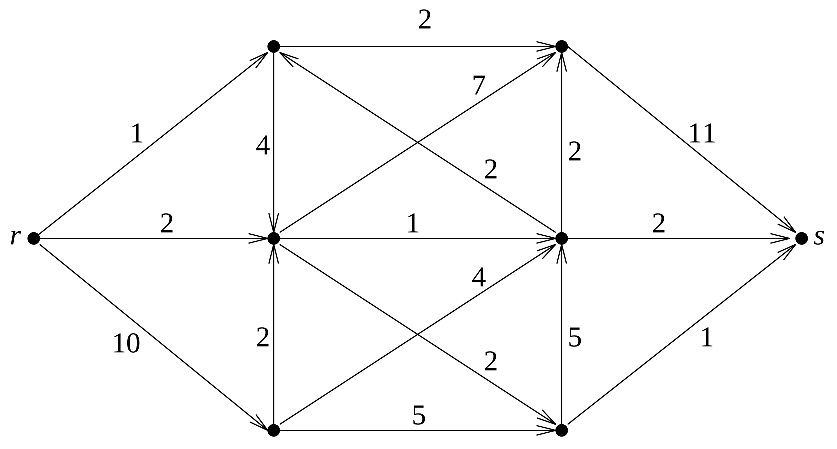
Since there cannot exist an $r-s$ flow under c of value larger than $d_1+\cdots+d_n$ (since $c(\delta^{\text{out}}(s))=d_1+\cdots+d_n$), the problem can be solved with the maximum-flow algorithm.

If there exists a flow of value $d_1+\cdots+d_n$, then the flow on arc (f_i,b_j) gives the amount that should be transported each month from factory i to customer j. The flow on arc (r,f_i) gives the amount to be produced each month by factory f_i.

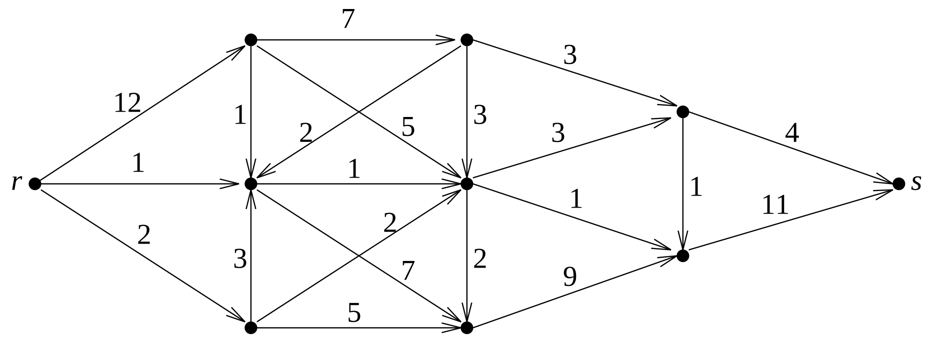
Exercises

4.8. Determine with the maximum flow algorithm an $r-s$ flow of maximum value and an $r-s$ cut of minimum capacity in the following graphs (where the numbers at the arcs give the capacities):

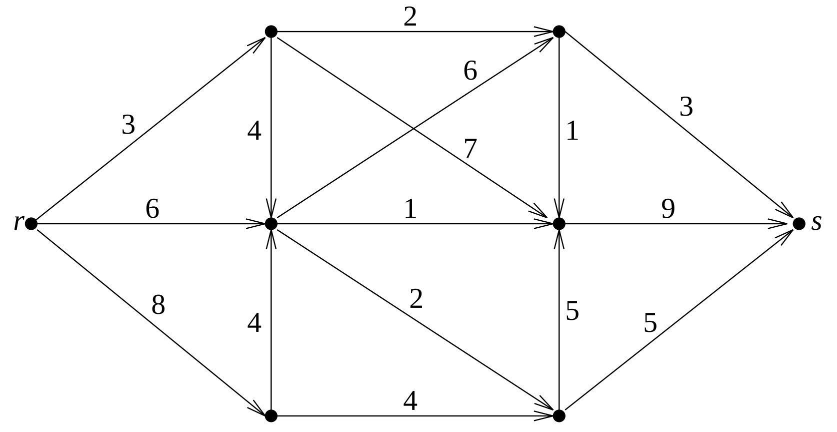
(i)



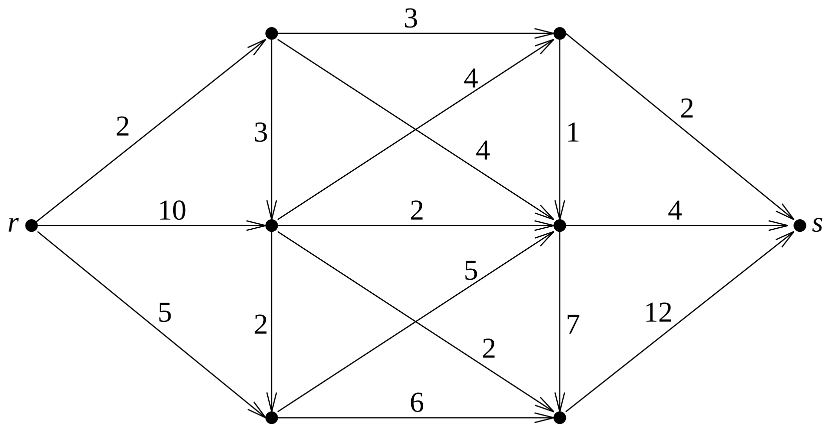
(ii)



(iii)



(iv)



4.9. Solve the transportation problem with the maximum-flow algorithm for the following data: $m = n = 3, s_1 = 13, s_2 = 9, s_3 = 4, d_1 = 3, d_2 = 7, d_3 = 12$,

$c_{i,j}$	$j = 1$	$j = 2$	$j = 3$
$i = 1$	2	0	8
$i = 2$	3	8	3
$i = 3$	0	1	3

4.10. Describe the problem of finding a maximum matching in a bipartite graph as a maximum flow problem.

4.11. Determine with the maximum-flow algorithm if there exists a 3×3 matrix $A = (a_{i,j})$ satisfying:[13]

$$a_{i,j} \geq 0 \text{ for all } i, j = 1, 2, 3;$$

$$A\mathbf{1} \leq \begin{pmatrix} 13 \\ 9 \\ 4 \end{pmatrix};$$

$$\mathbf{1}^T A = (3, 7, 12);$$

$$A \leq \begin{pmatrix} 2 & 0 & 8 \\ 3 & 8 & 3 \\ 0 & 1 & 3 \end{pmatrix}.$$

4.12. Give an example of a directed graph with irrational capacities, in which, at a bad choice of flow augmenting paths, the maximum flow algorithm does not terminate.

4.13. Let $D = (V, A)$ be a directed graph, let $r, s \in V$ and let $f : A \to \mathbb{Q}_+$ be an $r - s$ flow of value b. Show that for each $W \subseteq V$ with $r \in W, s \notin W$ one has:

$$\sum_{a \in \delta^{\text{out}}(W)} f(a) - \sum_{a \in \delta^{\text{in}}(W)} f(a) = b. \tag{24}$$

4.5. Speeding up the maximum flow algorithm

We saw that the number of iterations in the maximum flow algorithm is finite, if all capacities are rational. If we choose as our flow augmenting path P in the auxiliary graph D_f an *arbitrary* $s - t$ path, the number of iterations yet can get quite large. For instance, in the graph in Figure 4.3 the number of iterations, at a bad choice of paths, can become 2000.

However, if we choose always a *shortest* $s - t$ path in D_f as our flow augmenting path P (that is, with a minimum number of arcs), then the number of iterations is at most $|V| \cdot |A|$. This was shown by Dinits [1970] and Edmonds and Karp [1972].

[13] $\mathbf{1}$ denotes the vector $(1, 1, 1)^T$.

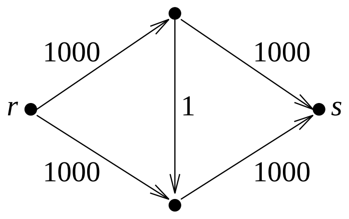


Figure 4.3

Again, for any directed graph $D = (V, A)$ and $r, s \in V$, let $\mu(D)$ denote the minimum length of an $r - s$ path. Moreover, let $\alpha(D)$ denote the set of arcs contained in at least one shortest $r - s$ path. Then one has:

Proposition 2. *Let $D = (V, A)$ and $r, s \in V$. Let $D' := (V, A \cup \alpha(D)^{-1})$. Then $\mu(D') = \mu(D)$ and $\alpha(D') = \alpha(D)$.*

Proof. It suffices to show that $\mu(D)$ and $\alpha(D)$ are invariant if we add a^{-1} to D for one arc $a \in \alpha(D)$. Suppose not. Then there is an $r - s$ path P traversing a^{-1}, of length at most $\mu(D)$. As $a \in \alpha(D)$, there is an $r - s$ path Q traversing a, of length $\mu(D)$. Hence $AP \cup AQ \setminus \{a, a^{-1}\}$ contains an $r - s$ path of length less than $\mu(D)$, a contradiction. ▌

This implies the result of Dinits [1970] and Edmonds and Karp [1972]:

Theorem 4.7. *If we choose in each iteration a shortest $r - s$ path as flow augmenting path, the number of iterations is at most $|V||A|$.*

Proof. If we augment flow f along a shortest path P, obtaining flow f', then $D_{f'}$ is a subgraph of $D' := (V, A_f \cup \alpha(D_f)^{-1})$. Hence $\mu(D_{f'}) \geq \mu(D') = \mu(D_f)$ (by Proposition 2). Moreover, if $\mu(D_{f'}) = \mu(D_f)$, then $\alpha(D_{f'}) \subseteq \alpha(D') = \alpha(D_f)$ (again by Proposition 2). As at least one arc in P belongs to D_f but not to $D_{f'}$, we have a strict inclusion. ▌

Since a shortest path can be found in time $O(|A|)$, this gives:

Corollary 4.7a. *The maximum flow problem can be solved in time $O(|V||A|^2)$.*

Proof. Directly from Theorem 4.7. ▌

This algorithm can be improved, as was shown by Karzanov [1974]. In each iteration we find a shortest path in $O(|A|)$ time. But as long as the distance from r to s does not increase, we could use the data-structure of the previous shortest path search so as to find the next shortest path.

This can be described as follows. Call an $r - s$ flow f *blocking* if for each $r - s$ flow $g \geq f$ one has $g = f$. Now Karzanov [1974] showed the following (we give the short proof of Malhotra, Kumar, and Maheshwari [1978]; see also Tarjan [1984]):

Theorem 4.8. *Given an acyclic directed graph $D = (V, A)$, $r, s \in V$, and a capacity function $c : A \to \mathbb{Q}_+$, a blocking flow can be found in time $O(|V|^2)$.*

Proof. First order the vertices reachable from s as $s = v_1, v_2, \ldots, v_{n-1}, v_n$ *topologically*; that is, if

$(v_i, v_j) \in A$ then $i < j$. This can be done in time $O(|A|)$.[14]

We describe the procedure recursively. Consider the minimum of the values $c(\delta^{\text{in}}(v))$ for all $v \in V \setminus \{s\}$ and $c(\delta^{\text{out}}(v))$ for all $v \in V \setminus \{t\}$. Let the minimum be attained by v_i and $c(\delta^{\text{out}}(v_i))$ (without loss of generality). Define $f(a) := c(a)$ for each $a \in \delta^{\text{out}}(v_i)$ and $f(a) := 0$ for all other a.

Next for $j = i+1, \ldots, n-1$, redefine $f(a)$ for each $a \in \delta^{\text{out}}(v_j)$ so that $f(a) \leq c(a)$ and so that $f(\delta^{\text{out}}(v_j)) = f(\delta^{\text{in}}(v_j))$. By the minimality of v_i and $c(\delta^{\text{in}}(v))$, we can always do this, as initially $f(\delta^{\text{in}}(v_j)) \leq c(\delta^{\text{out}}(v_i)) \leq c(\delta^{\text{in}}(v_j))$. We do this in such a way that finally $f(a) \in \{0, c(a)\}$ for all but at most one a in $\delta^{\text{out}}(v_j)$.

After that, for $j = i, i-1, \ldots, 2$, redefine similarly $f(a)$ for $a \in \delta^{\text{in}}(v_j)$ so that $f(a) \leq c(a)$ and so that $f(\delta^{\text{in}}(v_j)) = f(\delta^{\text{out}}(v_j))$.

If $v_i \in \{r, s\}$ we stop, and f is a blocking flow.

If $v_i \notin \{r, s\}$, set $c'(a) := c(a) - f(a)$ for each $a \in A$, and delete all arcs a with $c'(a) = 0$ and delete v_i and all arcs incident with v_i, thus obtaining the directed graph $D' = (V', A')$. Obtain (recursively) a blocking flow f' in D' subject to the capacity function c'. Define $f''(a) := f(a) + f'(a)$ for $a \in A'$ and $f''(a) = f(a)$ for $a \in A \setminus A'$. Then f'' is a blocking flow in D.

This describes the algorithm. The correctness can be seen as follows. If $v_i \in \{r, s\}$ the correctness is immediate. If $v_i \notin \{r, s\}$, suppose f'' is not a blocking flow in D, and let P be an $r - s$ path in D such that $f''(a) < c(a)$ for each arc a in P. Then each arc of P belongs to A', since $f''(a) = f(a) = c(a)$ for each $a \in A \setminus (A' \cup \delta^{\text{in}}(v_i))$. So for each arc a of P one has $c'(a) = c(a) - f(a) > f''(a) - f(a) = f'(a)$. This contradicts the fact that f' is a blocking flow in D'.

The running time of the algorithm is $O(|V|^2)$, since the running time of the iteration is $O(|V| + |A \setminus A'|)$, and since there are at most $|V|$ iterations. (Note that we determine the topological ordering only once, at the preprocessing.) ∎

Theorem 4.8 has the following consequence:

Corollary 4.8a. *Given a directed graph $D = (V, A)$, $r, s \in V$, and a capacity function $c : A \to \mathbb{Q}$, a flow f satisfying $\mu(D_f) > \mu(D)$ can be found in time $O(|V|^2)$.*

Proof. Let $\tilde{D}$ be the subgraph of D consisting of all arcs that are contained in a shortest $r - s$ path in D. Find a blocking flow in $\tilde{D}$. Then $\mu(D_f) > \mu(D)$ (by Proposition 2). ∎

Hence we have:

Corollary 4.8b. *Given a directed graph $D = (V, A)$, $r, s \in V$, and a capacity function $c : A \to \mathbb{Q}$, a maximum $r - s$ flow can be found in time $O(|V|^3)$.*

Proof. Directly from the foregoing. ∎

Goldberg and Tarjan [1990] gave an $O(|A| \log(|V|^2/|A|))$ algorithm for finding a blocking flow in an acyclic directed graph, implying an $O(|V||A| \log(|V|^2/|A|))$ algorithm for finding a maximum flow in any directed graph. An alternative approach finding a maximum flow in time $O(|V||A| \log(|V|^2/|A|))$ was described in Goldberg and Tarjan [1988].

For surveys on maximum flow algorithms, see Goldberg, Tardos, and Tarjan [1990] and Ahuja, Magnanti, and Orlin [1993].

[14]This can be done recursively as follows (cf. Knuth [1968], Tarjan [1974]). If $\delta^{\text{out}}(s) = \emptyset$, then the ordering is trivial. If $\delta^{\text{out}}(s) \neq \emptyset$, choose $(s, v) \in \delta^{\text{out}}(s)$, and order the vertices reachable from v topologically, as $w_1, \ldots, w_m$, delete them from D, and order the remaining vertices reachable from s topologically as $v_1, \ldots, v_k$; then $v_1, \ldots, v_k, w_1, \ldots, w_m$ gives a required topological ordering.

4.6. Circulations

A theorem related to the max-flow min-cut theorem is due to Hoffman [1960] and concerns circulations. Let $D = (V, A)$ be a directed graph. A function $f : A \to \mathbb{R}$ is called a *circulation* if for each vertex $v \in V$ one has:

(25) $$\sum_{a\in\delta^{\text{in}}(v)} f(a) = \sum_{a\in\delta^{\text{out}}(v)} f(a).$$

So now the flow conservation law holds at *each* vertex v.

Hoffman [1960] proved the following theorem (which can also be derived from the max-flow min-cut theorem, but a direct proof seems shorter). For any directed graph $D = (V, A)$, and any $d, c, f : A \to \mathbb{R}$ with $d(a) \le f(a) \le c(a)$ for each $a \in A$, we define

(26) $$A_f := \{a \mid f(a) < c(a)\} \cup \{a^{-1} \mid d(a) < f(a)\},$$

and $D_f := (V, A_f)$.

Theorem 4.9 (Hoffman's circulation theorem)**.** *Let $D = (V, A)$ be a directed graph and let $d, c : A \to \mathbb{R}$ be such that $d(a) \le c(a)$ for each arc a. Then there exists a circulation f such that*

(27) $$d(a) \le f(a) \le c(a)$$

for each arc a, if and only if

(28) $$\sum_{a\in\delta^{\text{in}}(U)} d(a) \le \sum_{a\in\delta^{\text{out}}(U)} c(a)$$

for each subset U of V.

Proof. To see necessity of (28), suppose that a circulation f satisfying (27) exists. Then

(29) $$d(\delta^{\text{in}}(U)) \le f(\delta^{\text{in}}(U)) = f(\delta^{\text{out}}(U)) \le c(\delta^{\text{out}}(U)).$$

To see sufficiency, define for any $f : A \to \mathbb{R}$ and any $v \in V$, $\text{loss}_f(v) := f(\delta^{\text{out}}(v)) - f(\delta^{\text{in}}(v))$. Choose a function f satisfying $d \le f \le c$ and minimizing $\|\text{loss}_f\|_1$. Let $S := \{v \in V \mid \text{loss}_f(v) < 0\}$ and $T := \{v \in V \mid \text{loss}_f(v) > 0\}$. Suppose $S \neq \emptyset$. If D_f contains an $S-T$ path, we can modify f so as to reduce $\|\text{loss}_f\|_1$. So D_f does not contain any $S-T$ path. Let U be the set of vertices reachable in D_f from S. Then for each $a \in \delta^{\text{out}}(U)$ we have $a \notin A_f$ and hence $f(a) = c(a)$. Similarly, for each $a \in \delta^{\text{in}}(U)$ we have $a^{-1} \notin A_f$ and hence $f(a) = d(a)$. Therefore

(30) $$c(\delta^{\text{out}}(U)) - d(\delta^{\text{in}}(U)) = f(\delta^{\text{out}}(U)) - f(\delta^{\text{in}}(U)) = \text{loss}_f(U) = \text{loss}_f(S) < 0,$$

contradicting (28). ∎

One has moreover:

Theorem 4.10. *In Theorem 4.9, if c and d are integer and there exists a circulation f satisfying $d \le f \le c$, then there exists an integer-valued circulation f satisfying $d \le f \le c$.*

Proof. Directly from the proof above. ∎

Exercises

4.14. Let $D = (V, A)$ be a directed graph and let $f : A \to \mathbb{R}$ be a circulation. Show that there exists a circulation f' such that f' is integer-valued and such that $\lfloor f(a) \rfloor \le f'(a) \le \lceil f(a) \rceil$ for each arc a.

4.15. Let $D = (V, A)$ be a directed graph and let $d, c : A \to \mathbb{R}$.

Derive an algorithm finding a circulation f satisfying $d \leq f \leq c$ from the maximum flow algorithm.

4.16. Let $\mathcal{A} = (A_1, \ldots, A_n)$ and $\mathcal{B} = (B_1, \ldots, B_n)$ be partitions of a finite set X and let k be a natural number. Prove that X can be covered by k common SDR's of $\mathcal{A}$ and $\mathcal{B}$, if and only if

$$|(\bigcup_{i \in I} A_i \cup \bigcup_{j \in J} B_j)| \geq |X| + k(|I| + |J| - n) \tag{31}$$

for all $I, J \subseteq \{1, \ldots, n\}$ with $\bigcup_{i \in I} A_i \cap \bigcup_{j \in J} B_j = \emptyset$.

4.17. Let $D = (V, A)$ be a directed graph, and let $f : A \to \mathbb{R}_+$. Let $\mathcal{C}$ be the collection of directed circuits in D. For each directed circuit C in D let χ^C be the incidence vector of C. That is, $\chi^C : A \to \{0, 1\}$, with $\chi^C(a) = 1$ if C traverses a and $\chi^C(a) = 0$ otherwise.

Show that f is a nonnegative circulation, if and only if there exists a function $\lambda : \mathcal{C} \to \mathbb{R}_+$ such that

$$f = \sum_{C \in \mathcal{C}} \lambda(C) \chi^C. \tag{32}$$

That is, the circulations form the cone generated by $\{\chi^C \mid C \in \mathcal{C}\}$.

4.7. Minimum-cost flows

In the previous sections we were searching for flows of maximum value. In this section we consider the problem of finding a flow of maximum value with the additional property that it has 'minimum cost'.

Let be given again a directed graph $D = (V, A)$, vertices r and s of D, and a capacity function $c : A \to \mathbb{R}_+$. Let moreover be given a function $k : A \to \mathbb{R}_+$, called the *cost function*.

Define for any function $f : A \to \mathbb{R}_+$ the *cost* of f as:

$$\text{cost}(f) := \sum_{a \in A} k(a) f(a). \tag{33}$$

The following is the *minimum-cost flow problem* (or *min-cost flow problem*):

(34) given: a directed graph $D = (V, A)$, $r, s \in V$, a capacity function $c : A \to \mathbb{R}_+$ and a cost function $k : A \to \mathbb{R}_+$;

find: an $r - s$ flow subject to c of maximum value, such that f has minimum cost among all $r - s$ flows subject to c of maximum value.

This problem can be solved with an adaptation of the algorithm described in Section 4.4. Let us define an $r - s$ flow $f \leq c$ to be an *extreme flow* if f has minimum cost among all $r - s$ flows $g \leq c$ with value(g) = value(f).

So an extreme flow does not need to have maximum value. An extreme flow is a flow f that has minimum cost among all flows with the same value as f.

Let f be a flow and let $D_f = (V, A_f)$ be the auxiliary graph corresponding to f (in the sense of the flow augmenting algorithm). Define a length function $l : A_f \to \mathbb{R}$ on A_f by:

$$\begin{aligned} l(a) \quad &:= k(a) && \text{if } a \in A, \\ &:= -k(a^{-1}) && \text{if } a^{-1} \in A \end{aligned} \tag{35}$$

for each $a \in A_f$.

Given this the following can be shown:

Proposition 3. *f is an extreme flow, if and only if D_f has no directed circuits of negative length (with respect to l).*

Proof. *Necessity.* Suppose that $C = (a_1, \ldots, a_t)$ is a directed circuit in D_f of negative length; that is,

(36) $$\text{length}(C) = l(a_1) + l(a_2) + \cdots + l(a_t) < 0.$$

So $a_1, \ldots, a_t$ are arcs in D_f. Define for $i = 1, \ldots, t$:

(37) $$\begin{aligned} \sigma_i &:= c(a_i) - f(a_i) && \text{if } a_i \in A; \\ &:= f(a_i^{-1}) && \text{if } a_i^{-1} \in A. \end{aligned}$$

Note that by definition of D_f, $\sigma_i > 0$ for each $i = 1, \ldots, t$. Let $\varepsilon := \min\{\sigma_1, \ldots, \sigma_t\}$ and define for each $a \in A$:

(38) $$\begin{aligned} g(a) &:= f(a) + \varepsilon && \text{if } a \in C, \\ &:= f(a) - \varepsilon && \text{if } a^{-1} \in C, \\ &:= f(a) && \text{otherwise.} \end{aligned}$$

Then g is again an $r - s$ flow subject to c, with value(g) = value(f). Moreover one has

(39) $$\text{cost}(g) = \text{cost}(f) + \varepsilon \cdot \text{length}(C) < \text{cost}(f).$$

So f is not an extreme flow.

Sufficiency. Let g be any flow with value(g) =value(f). Define $h : A_f \to \mathbb{R}_+$ by:

(40) $$\begin{aligned} h(a) &:= g(a) - f(a) && \text{if } g(a) > f(a), \text{ and} \\ h(a^{-1}) &:= f(a) - g(a) && \text{if } g(a) < f(a), \end{aligned}$$

for $a \in A$, while $h(a) = 0$ for all other arcs a of A_f. Then h is a circulation in D_f.

Hence, by Exercise 4.17, there exists a function $\lambda : \mathcal{C} \to \mathbb{R}_+$ such that $h = \sum_{C \in \mathcal{C}} \lambda(C)\chi^C$. Hence cost$(g)$ − cost$(f) = \sum_{C \in \mathcal{C}} \lambda(C)\text{length}(C)$. Assuming D_f has no directed circuits of negative length, it follows that cost$(g) \geq$ cost(f). So f is an extreme flow. ∎

With this we can show:

Proposition 4. *Let f be an extreme flow. Let f' arise by choosing in the flow augmenting algorithm a path in D_f of minimum length with respect to l. Then f' is an extreme flow again.*

Proof. Suppose $D_{f'}$ has a directed circuit C of negative length with respect to l. As C does not occur in D_f, part of C occurs in the flow augmenting path chosen. But then we could have chosen a shorter flow augmenting path. ∎

This implies that the min-cost flow problem can be solved by choosing in the flow augmenting algorithm a shortest path in the auxiliary graph throughout. The first flow, the all-zero flow f_0, is trivially a min-cost flow. Hence also all further flows $f_1, f_2, f_3, \ldots$ will be min-cost flows by Proposition 4. Therefore, also the last flow, which is of maximum value, is a min-cost flow. So we have a solution to the min-cost flow problem. (Here we assume that all capacities are rational.)

In this process, we should be able to find a shortest $r - s$ path in the auxiliary graphs D_f. This is indeed possible with the Bellman-Ford method, since D_f does not have directed circuits of negative length as we saw in Proposition 3.

One can show that the running time of this algorithm is $O(M \cdot (m + n \log n))$, where M is the value of a maximum flow (assuming all capacities to be integer). So it is not polynomial-time. At the moment of writing, the asymptotically fastest method for finding a minimum-cost maximum flow was designed by Orlin [1988,1993] and runs in $O(m \log n(m + n \log n))$ time.

In a similar way one can describe a *minimum-cost circulation* algorithm.

For more about network flows we refer to the books of Ford and Fulkerson [1962] and Ahuja, Magnanti, and Orlin [1993].

Application 4.3: Minimum-cost transportation problem. Beside the data in Application 4.2 one may also have a cost function $k_{i,j}$, giving the cost of transporting 1 ton from factory i to costumer j. Moreover, there is given a cost k_i of producing 1 ton by factory i (for each i). We want to make a production and transportation plan that minimizes the total cost.

This problem can be solved by assigning also costs to the arcs in Application 4.2. We can take the costs on the arcs from b_j to s equal to 0.

Application 4.4: Routing empty freighters. Historically, in his paper “Optimum utilization of the transportation system”, Koopmans [1948] was one of the first studying the minimum-cost transportation problem, in particular with application to the problem of routing empty freighters. Koopmans considered the surplus and need of register ton of ship capacity at harbours all over the world, as given by the following table (data are aggregated to main harbours):

Net receipt of dry cargo in overseas trade, 1925
Unit: Millions of metric tons per annum

Harbour	Received	Dispatched	Net receipts
New York	23.5	32.7	−9.2
San Francisco	7.2	9.7	−2.5
St. Thomas	10.3	11.5	−1.2
Buenos Aires	7.0	9.6	−2.6
Antofagasta	1.4	4.6	−3.2
Rotterdam	126.4	130.5	− 4.1
Lisbon	37.5	17.0	20.5
Athens	28.3	14.4	13.9
Odessa	0.5	4.7	−4.2
Lagos	2.0	2.4	−0.4
Durban	2.1	4.3	−2.2
Bombay	5.0	8.9	−3.9
Singapore	3.6	6.8	−3.2
Yokohama	9.2	3.0	6.2
Sydney	2.8	6.7	−3.9
Total	266.8	266.8	0.0

Given is moreover a distance table between these harbours. Koopmans wondered how ships should be routed between harbours so as to minimize the total amount of ton kilometers made by empty ships.

This problem is a special case of the min-cost flow problem. Make a graph with vertex set all harbours, together with two dummy harbours r and s. From any harbour u with a surplus (positive net receipt) to any harbour w with a need (negative net receipt) make an arc with cost equal to the distance between u and w, and with capacity ∞. Moreover, from r to any harbour u with a surplus σ, make an arc with cost 0 and capacity equal to σ. Similarly, from any harbour w with a need ν, make an arc to s, with cost 0 and capacity equal to ν.

Now a maximum flow of minimum cost corresponds to an optimum routing of ships between harbours.

A similar model applies to the problem of routing empty box cars in a railway network (Feeney [1957], cf. Norman and Dowling [1968], White and Bomberault [1969]).

Application 4.5: Routing of railway stock. NS (Nederlandse Spoorwegen = Dutch Railways) performs a daily schedule on its line Amsterdam–Vlissingen, with the following (weekday) timetable:

ride number		2123	2127	2131	2135	2139	2143	2147	2151	2155	2159	2163	2167	2171	2175	2179	2183	2187	2191
Amsterdam	d		6.48	7.55	8.56	9.56	10.56	11.56	12.56	13.56	14.56	15.56	16.56	17.56	18.56	19.56	20.56	21.56	22.56
Rotterdam	a		7.55	8.58	9.58	10.58	11.58	12.58	13.58	14.58	15.58	16.58	17.58	18.58	19.58	20.58	21.58	22.58	23.58
Rotterdam	d	7.00	8.01	9.02	10.03	11.02	12.03	13.02	14.02	15.02	16.00	17.01	18.01	19.02	20.02	21.02	22.02	23.02	
Roosendaal	a	7.40	8.41	9.41	10.43	11.41	12.41	13.41	14.41	15.41	16.43	17.43	18.42	19.41	20.41	21.41	22.41	23.54	
Roosendaal	d	7.43	8.43	9.43	10.45	11.43	12.43	13.43	14.43	15.43	16.45	17.45	18.44	19.43	20.43	21.43			
Vlissingen	a	8.38	9.38	10.38	11.38	12.38	13.38	14.38	15.38	16.38	17.40	18.40	19.39	20.38	21.38	22.38			

ride number		2108	2112	2116	2120	2124	2128	2132	2136	2140	2144	2148	2152	2156	2160	2164	2168	2172	2176
Vlissingen	d			5.30	6.54	7.56	8.56	9.56	10.56	11.56	12.56	13.56	14.56	15.56	16.56	17.56	18.56	19.55	
Roosendaal	a			6.35	7.48	8.50	9.50	10.50	11.50	12.50	13.50	14.50	15.50	16.50	17.50	18.50	19.50	20.49	
Roosendaal	d		5.29	6.43	7.52	8.53	9.53	10.53	11.53	12.53	13.53	14.53	15.53	16.53	17.53	18.53	19.53	20.52	21.53
Rotterdam	a		6.28	7.26	8.32	9.32	10.32	11.32	12.32	13.32	14.32	15.32	16.32	17.33	18.32	19.32	20.32	21.30	22.32
Rotterdam	d	5.31	6.29	7.32	8.35	9.34	10.34	11.34	12.34	13.35	14.35	15.34	16.34	17.35	18.34	19.34	20.35	21.32	22.34
Amsterdam	a	6.39	7.38	8.38	9.40	10.38	11.38	12.38	13.38	14.38	15.38	16.40	17.38	18.38	19.38	20.38	21.38	22.38	23.38

The rides are carried out by one type of stock, that consists of two-way units that can be coupled with each other. The length of the trains can be changed at the end stations and at two intermediate stations: Rotterdam and Roosendaal. So in this example, each train ride consists of three ride 'segments'.

Based on the expected number of passengers, NS determines for each ride segment a minimum number of units that should be deployed for that segment:

ride number	2123	2127	2131	2135	2139	2143	2147	2151	2155	2159	2163	2167	2171	2175	2179	2183	2187	2191
Amsterdam-Rotterdam		3	5	4	3	3	3	3	3	3	4	5	5	3	2	2	2	1
Rotterdam-Roosendaal	2	3	4	4	2	3	3	3	3	4	5	5	4	2	2	2	1	
Roosendaal-Vlissingen	3	2	2	2	2	3	2	3	3	3	4	4	3	2	1			

ride number	2108	2112	2116	2120	2124	2128	2132	2136	2140	2144	2148	2152	2156	2160	2164	2168	2172	2176
Vlissingen-Roosendaal			2	4	4	4	2	2	3	2	2	2	3	3	2	2	2	
Roosendaal-Rotterdam		2	4	5	4	5	3	3	3	2	3	3	4	3	2	2	2	2
Rotterdam-Amsterdam	1	3	5	4	4	5	3	3	3	3	3	4	5	3	2	2	2	2

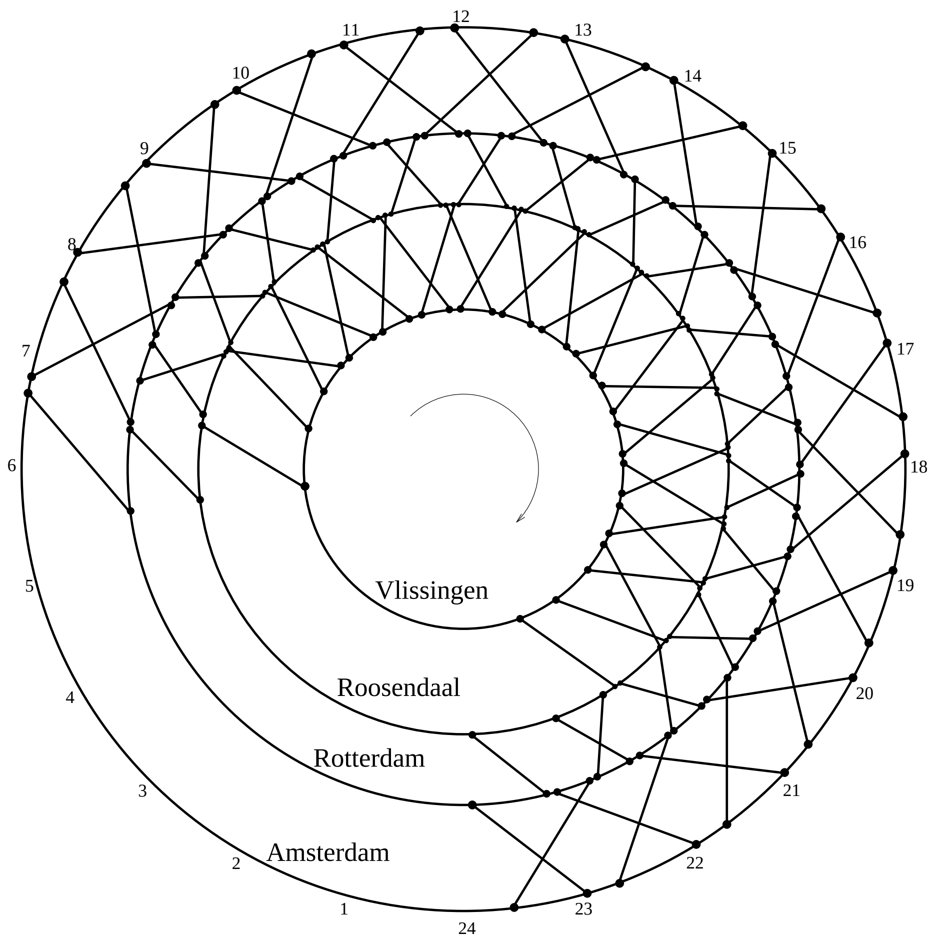


Figure 4.4

A unit uncoupled from a train at a station can be coupled at any other later train, in the same direction or the other. Moreover, for each segment there is a maximum number of units given that can be used for that segment (depending for instance on the length of station platforms).

The company now wishes to find the minimum number of units that should be used to run the schedule (excluding maintenance).

As was observed by Bartlett [1957] (cf. van Rees [1965]) this problem can be considered as a minimum-cost circulation problem (cf. Figure 4.4). Make a directed graph D with vertex set all pairs (s, t) where s

is any station where the train composition can be changed (in our example: the end stations and the two intermediate stations) and t is any time at which there is a train arriving at or leaving s. For each ride segment make an arc from (s, t) to (s', t') if the segment leaves s at time t and arrives at s' at time t'.

Moreover, for each station s and each two consecutive times t, t' at which segments arrive or leave, one makes an arc from (s, t) to (s, t'). One also does this overnight.

Now for each arc a coming from a segment assign a lower bound $d(a)$ equal to the number given in the table above for the segment. Moreover, define an upper bound $c(a)$ equal to the maximum number of units that can be used for that segment. For each arc a from (s, t) to (s, t') let $d(a) := 0$ and $c(a) := \infty$.

For each arc a define a cost $k(a) := 0$, except if a corresponds to an overnight stay at one of cities, when $k(a) := 1$. Then a minimum-cost circulation corresponds to a routing of the stock using a minimum number of units.

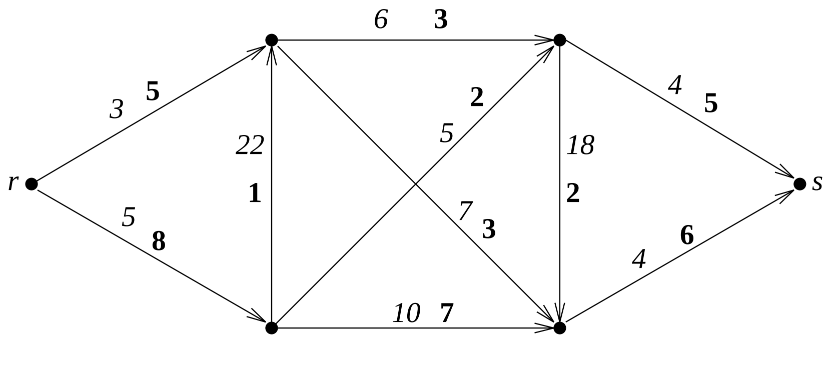
There are several variations possible. Instead of an upper bound $c(a) = \infty$ for the arcs a from (c, t) to (s, t') one can take $c(a)$ equal to the capacity of the storage area at s. Instead of a cost $k(a) = 0$ at each segment one can take $k(a)$ equal to the cost of riding one unit of stock over that segment. This can be weighed against the cost of buying extra units.

A similar model for routing airplanes was considered by Ferguson and Dantzig [1955].

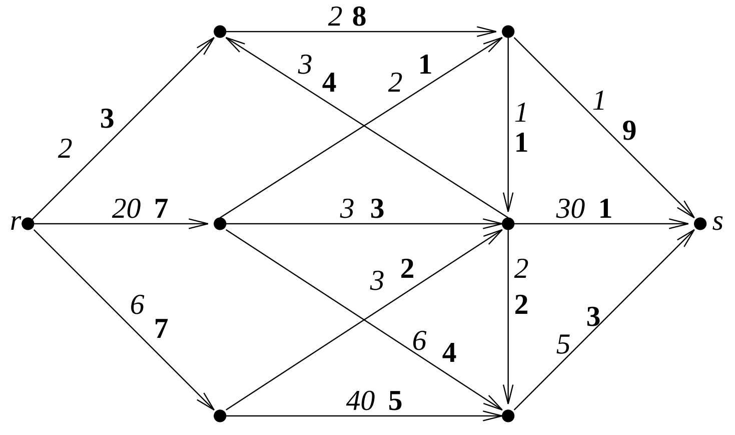
Exercises

4.18. Determine in the following networks a maximum $r - s$ flow of minimum-cost (*cost* in *italics*, **capacity** in **bold**):

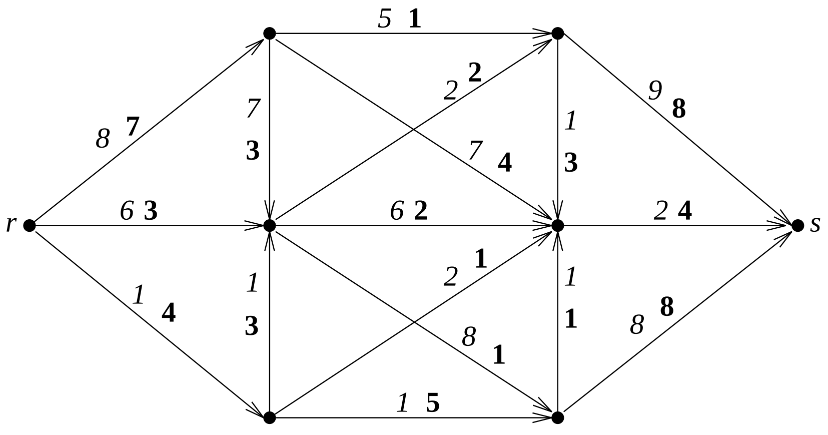
(i)



(ii)



(iii)



4.19. Solve the minimum-cost transportation problem for the following data sets:

(i) $m = n = 3, s_1 = 9, s_2 = 15, s_3 = 7, d_1 = 5, d_2 = 13, d_3 = 7, k_1 = 2, k_2 = 3, k_3 = 2,$

$c_{i,j}$	$j=1$	$j=2$	$j=3$
$i=1$	6	4	0
$i=2$	3	9	4
$i=3$	0	2	6

$k_{i,j}$	$j=1$	$j=2$	$j=3$
$i=1$	8	3	5
$i=2$	2	7	1
$i=3$	2	5	9

(ii) $m = n = 3, s_1 = 11, s_2 = 7, s_3 = 6, d_1 = 9, d_2 = 7, d_3 = 5, k_1 = 4, k_2 = 3, k_3 = 3,$

$c_{i,j}$	$j=1$	$j=2$	$j=3$
$i=1$	7	4	0
$i=2$	3	3	2
$i=3$	0	2	4

$k_{i,j}$	$j=1$	$j=2$	$j=3$
$i=1$	3	2	4
$i=2$	2	8	4
$i=3$	1	3	2

4.20. Describe the problem of finding a maximum-weight matching in a bipartite graph as a minimum-cost flow problem.

4.21. Reduce the problem of finding a min-cost flow of given value, to the min-cost flow problem as described above.

5. Nonbipartite matching

5.1. Tutte's 1-factor theorem and the Tutte-Berge formula

A basic result on matchings in arbitrary (not necessarily bipartite) graphs was found by Tutte [1947]. It characterizes graphs that have a perfect matching. A *perfect matching* (or a $1-$*factor*) is a matching M that covers all vertices of the graph. (So M partitions the vertex set of G.)

Berge [1958] observed that Tutte's theorem implies a min-max formula for the maximum size of a matching in a graph, the Tutte-Berge formula, which we prove first.

Call a component of a graph *odd* if it has an odd number of vertices. For any graph G, let

(1) $$o(G) := \text{number of odd components of } G.$$

Let $\nu(G)$ denotes the maximum size of a matching. Then:

Theorem 5.1 (Tutte-Berge formula)**.** *For each graph* $G = (V, E)$,

(2) $$\nu(G) = \min_{U \subseteq V} \tfrac{1}{2}(|V| + |U| - o(G - U)).$$

Proof. To see $\leq$, we have for each $U \subseteq V$:

(3) $$\nu(G) \leq |U| + \nu(G - U) \leq |U| + \tfrac{1}{2}(|V \setminus U| - o(G - U)) = \tfrac{1}{2}(|V| + |U| - o(G - U)).$$

We prove the reverse inequality by induction on $|V|$, the case $V = \emptyset$ being trivial. We can assume that G is connected, since otherwise we can apply induction to the components of G.

First assume that there exists a vertex v covered by all maximum-size matchings. Then $\nu(G-v) = \nu(G) - 1$, and by induction there exists a subset U' of $V \setminus \{v\}$ with

(4) $$\nu(G - v) = \tfrac{1}{2}(|V \setminus \{v\}| + |U'| - o(G - v - U')).$$

Then $U := U' \cup \{v\}$ gives equality in (2), since

(5) $$\begin{aligned}\nu(G) &= \nu(G - v) + 1 = \tfrac{1}{2}(|V \setminus \{v\}| + |U'| - o(G - v - U')) + 1 \\ &= \tfrac{1}{2}(|V| + |U| - o(G - U)).\end{aligned}$$

So we can assume that there is no such v. In particular, $\nu(G) < \frac{1}{2}|V|$. We show that there exists a matching of size $\frac{1}{2}(|V| - 1)$, which implies the theorem (taking $U := \emptyset$).

Indeed, suppose to the contrary that each maximum-size matching M misses at least two distinct vertices u and v. Among all such M, u, v, choose them such that the distance $\text{dist}(u, v)$ of u and v in G is as small as possible.

If $\text{dist}(u, v) = 1$, then u and v are adjacent, and hence we can augment M by the edge uv, contradicting the maximality of $|M|$. So $\text{dist}(u, v) \geq 2$, and hence we can choose an intermediate vertex t on a shortest $u - v$ path. By assumption, there exists a maximum-size matching N missing t. Choose such an N with $|M \cap N|$ maximal.

By the minimality of $\text{dist}(u, v)$, N covers both u and v. Hence, as M and N cover the same number of vertices, there exists a vertex $x \neq t$ covered by M but not by N. Let $x \in e = xy \in M$. Then y is covered by some edge $f \in N$, since otherwise $N \cup \{e\}$ would be a matching larger than N. Replacing N by $(N \setminus \{f\}) \cup \{e\}$ would increase its intersection with M, contradicting the choice of N. ∎

(This proof is based on the proof of Lovász [1979] of Edmonds' matching polytope theorem.)

The Tutte-Berge formula immediately implies Tutte's 1-factor theorem.

Corollary 5.1a (Tutte's 1-factor theorem). *A graph $G = (V, E)$ has a perfect matching if and only if $G - U$ has at most $|U|$ odd components, for each $U \subseteq V$.*

Proof. Directly from the Tutte-Berge formula (Theorem 5.1), since G has a perfect matching if and only if $\nu(G) \geq \frac{1}{2}|V|$. ∎

In the following sections we will show how to find a maximum matching algorithmically.

With Gallai's theorem, the Tutte-Berge formula implies a formula for the edge cover number $\rho(G)$:

Corollary 5.1b. *Let $G = (V, E)$ be a graph without isolated vertices. Then*

$$\rho(G) = \max_{U \subseteq V} \frac{|U| + o(U)}{2}. \tag{6}$$

Proof. By Gallai's theorem (Theorem 3.1) and the Tutte-Berge formula (Theorem 5.1),

$$\rho(G) = |V| - \nu(G) = |V| - \min_{W \subseteq V} \frac{|V| + |W| - o(V \setminus W)}{2} = \max_{U \subseteq V} \frac{|U| + o(U)}{2}. \tag{7}$$

∎

Exercises

5.1. (i) Show that a tree has at most one perfect matching.

(ii) Show (not using Tutte's 1-factor theorem) that a tree $G = (V, E)$ has a perfect matching if and only if the subgraph $G - v$ has exactly one odd component, for each $v \in V$.

5.2. Let G be a 3-regular graph without any isthmus. Show that G has a perfect matching.

5.3. Let $A_1, \ldots, A_n$ be a collection of nonempty subsets of the finite set X so that each element in X is in exactly two sets among $A_1, \ldots, A_n$. Show that there exists a set Y intersecting all sets $A_1, \ldots, A_n$, and satisfying $|Y| \leq t$ if and only if for each subset I of $\{1, \ldots, n\}$ the number of components of $(A_i \mid i \in I)$ containing an odd number of sets in $(A_i \mid i \in I)$ is at most $2t - |I|$.

(Here a subset Y of X is called a *component* of $(A_i \mid i \in I)$ if it is a minimal nonempty subset of X with the property that for each $i \in I$: $A_i \cap Y \neq \emptyset$ or $A_i \subseteq Y$.)

5.4. Let $G = (V, E)$ be a graph and let T be a subset of V. Then G has a matching covering T, if and only if the number of odd components of $G - W$ contained in T is at most $|W|$, for each $W \subseteq V$.

5.5. Let $G = (V, E)$ be a graph and let $b : V \to \mathbb{Z}_+$. Show that there exists a function $f : E \to \mathbb{Z}_+$ so that for each $v \in V$:

$$\sum_{e \in E, v \in e} f(e) = b(v), \tag{8}$$

if and only if for each subset W of V the number $\beta(W)$ is at most $b(V \setminus W)$.

(Here for any subset W of V, $b(W) := \sum_{v \in W} b(v)$. Moreover, $\beta(W)$ denotes the following. Let U be the set of isolated vertices in the graph $G|W$ induced by W and let t denote the number of components C of the graph $G|W \setminus U$ with $b(C)$ odd. Then $\beta(W) := b(U) + t$.)

5.6. Let $G = (V, E)$ be a graph and let $b : V \to \mathbb{Z}_+$. Show that there exists a subset F of E so that each vertex v is incident with exactly $b(v)$ edges in F, if and only if for each two disjoint subsets U and W of V one has

$$\sum_{v \in U} b(v) \geq q(U, W) + \sum_{v \in W} (b(v) - d_{G-U}(v)). \tag{9}$$

(Here $q(U, W)$ denotes the number of components K of $G - (U \cup W)$ for which $b(K)$ plus the number of edges connecting K and W, is odd. Moreover, $d_{G-U}(v)$ is the degree of v in the subgraph induced by $V \setminus U$.)

5.2. Cardinality matching algorithm

We now investigate the problem of finding a maximum-cardinality matching algorithmically. Like in the bipartite case, the key is to find an augmenting path. However, the idea for bipartite graphs to orient the edges using the two colour classes, does not apply to nonbipartite graphs.

Yet one could try to find an M-augmenting path by finding a so-called M-alternating path, but such a path can run into a loop that cannot immediately be deleted. It was J. Edmonds who found the trick to resolve this problem, namely by 'shrinking ' the loop (which he called a 'blossom'). Then applying recursion to a smaller graph solves the problem.

We first describe the operation of *shrinking*. Let X and Y be sets. Then we define X/Y as follows:

(10) $$X/Y := X \text{ if } X \cap Y = \emptyset,$$
$$X/Y := (X \setminus Y) \cup \{Y\} \text{ if } X \cap Y \neq \emptyset.$$

So if $G = (V, E)$ is a graph and $C \subseteq V$, then V/C arises from V by deleting all vertices in C, and adding one new vertex called C. For any edge e of G, $e/C = e$ if e is disjoint from C, while $e/C = uC$ if $e = uv$ with $u \notin C$, $v \in C$. (If $e = uv$ with $u, v \in C$, then e/C is a loop CC; they can be neglected in the context of matchings.) Then for any $F \subseteq E$:

(11) $$F/C := \{f/C \mid f \in F\}.$$

So $G/C := (V/C, E/C)$ is again a graph. We say that G/C arises from G by *shrinking* C.

Let $G = (V, E)$ be a graph and let M be a matching in G, and let W be the set of vertices missed by M. A path $P = (v_0, v_1, \ldots, v_t)$ is called *M-alternating* if for each $i = 1, \ldots, t-1$ exactly one of $v_{i-1}v_i$ and v_iv_{i+1} belongs to M. Note that one can find a shortest M-alternating $W - W$ path, by considering the auxiliary directed graph $D = (V, A)$ with

(12) $$A := \{(w, w') \mid \exists x \in V : \{w, x\} \in E, \{x, w'\} \in M\}.$$

Then M-alternating $W - W$ paths correspond to directed paths in D from a vertex in W to a vertex that is adjacent to at least one vertex in W.

As before, we call an M-alternating path $P = (v_0, v_1, \ldots, v_t)$ *M-augmenting* if $v_0, \ldots, v_t$ are distinct and v_0 and v_t are missed by M. (Hence t is odd.) So by Theorem 3.3, a matching M has maximum size if and only if there is no M-augmenting path. We call an M-alternating path P an *M-blossom* if $v_0, \ldots, v_{t-1}$ are distinct, v_0 is missed by M, and $v_t = v_0$.

The core of the algorithm is the following observation.

Theorem 5.2. *Let C be an M-blossom in G. Then M has maximum size in G if and only if M/C has maximum size in G/C.*

Proof. Let $C = (v_0, v_1, \ldots, v_t)$, $G' := G/C$ and $M' := M/C$.

First let P be an M-augmenting path in G. We may assume that P does not start in v_0 (otherwise we can inverse P). If P does not traverse any vertex in C, then P is also M'-augmenting in G'. If P does traverse a vertex in C, we can decompose P as $P = QR$, where Q ends in a vertex in C, and no other vertex on Q belongs to C. Then by replacing the last vertex of Q by C makes Q to an M'-augmenting path in G'.

Conversely, let P' be an M'-augmenting path in G'. If P' does not traverse vertex C of G', then P' is also an M-augmenting path in G. If P' traverses vertex C of G', we may assume it ends in C (as C is missed by M'). So we can replace C in P' by some vertex $v_i \in C$ to obtain a path Q in G ending in v_i. If i is odd, extending Q by $v_{i+1}, \ldots, v_{t-1}, v_t$ gives an M-augmenting path in G. If i is even, extending Q by $v_{i-1}, \ldots, v_1, v_0$ gives an M-augmenting path in G. ▮

Another useful observation is:

Theorem 5.3. *Let $P = (v_0, v_1, \ldots, v_t)$ be a shortest even-length M-alternating $W - v$ path. Then either P is simple or there exist $i < j$ such that $v_i = v_j$, i is even, j is odd, and $v_0, \ldots, v_{j-1}$ are all distinct.*

Proof. Assume P is not simple. Choose $i < j$ such that $v_j = v_i$ and such that j is as small as possible. If $j - i$ is even, we can delete $v_{i+1}, \ldots, v_j$ from P so as to obtain a shorter M-alternating $W - v$ path. So $j - i$ is odd. If j is even and i is odd, then $v_{i+1} = v_{j-1}$ (as it is the vertex matched to $v_i = v_j$), contradicting the minimality of j. ▮

We now describe an algorithm for the following problem:

(13) given: a matching M;
find: a matching N with $|N| = |M| + 1$ or conclude that M is a maximum-size matching.

Let W be the set of vertices missed by M.

(14) **Case 1.** *There is no M-alternating $W - W$ path.* Then M has maximum size (as there is no M-augmenting path).

Case 2. *There is an M-alternating $W - W$ path.* Let $P = (v_0, v_1, \ldots, v_t)$ be a shortest such path.

Case 2a. *P is M-augmenting.* Then output $N := M \triangle EP$.

Case 2b. *P is not M-augmenting.* Choose $i < j$ such that $v_i = v_j$ with j as small as possible. Reset $M := M \triangle \{v_0v_1, v_1v_2, \ldots, v_{i-1}v_i\}$. Then $C := (v_i, , v_{i+1}, \ldots, v_j)$ is an M-blossom. Apply the algorithm (recursively) to $G' = G/C$ and $M' := M/C$.

- If it gives an M'-augmenting path P' in G', transform P' to an M-augmenting path in G (as in the proof of Theorem 5.2).
- If it concludes that M' has maximum size in G', then M has maximum size in G (by Theorem 5.2).

This gives a polynomial-time algorithm to find a maximum matching, which is a basic result of Edmonds [1965c].

Theorem 5.4. *Given an undirected graph, a maximum matching can be found in time $O(|V|^2|E|)$.*

Proof. The algorithm directly follows from algorithm (14), since one can iteratively apply it, starting with $M = \emptyset$, until a maximum-size matching is attained.

By using (12), a shortest M-alternating $W - W$ path can be found in time $O(m)$. Moreover, the graph G/C can be constructed in time $O(m)$. Since the recursion has depth at most n, each application of algorithm (14) takes $O(nm)$ time. Since the number of applications is at most n, we have the time bound given in the theorem. ▮

In fact, the method can be sharpened to $O(n^3)$ (Balinski [1969]), $O(n^{5/2})$ (Even and Kariv [1975]) and even to $O(n^{1/2}m)$ (Micali and Vazirani [1980]). For surveys, see Lawler [1976] Ch. 6 and Christofides [1975] Ch. 12.

Application 5.1: Pairing. If a certain group of people has to be split into pairs, where certain pairs fit and other pairs do not fit (for instance, when assigning hotel rooms or bus seats to a touring group), we have an example of a (perfect) matching problem.

Application 5.2: Two-processor scheduling. Suppose we have to carry out certain jobs, where some of the jobs have to be done before other. We can represent this by a partially ordered set $(X, \leq)$ where X is the set of jobs and $x < y$ indicates that job x has to be done before job y. Each job takes one time-unit, say one hour.

Suppose now that there are two workers, each of which can do one job at a time. Alternatively, suppose that you have one machine, that can do at each moment two jobs simultaneously (such a machine is called a *two-processor*).

We wish to do all jobs within a minimum total time span. This problem can be solved with the matching algorithm as follows. Make a graph $G = (X, E)$, with vertex set X (the set of jobs) and with edge set

$$E := \{\{u, v\} \mid u \not\leq v \text{ and } v \not\leq u\}. \tag{15}$$

(So (X, E) is the complementary graph of the 'comparability graph' associated with $(X, \leq)$.)

Consider now a possible schedule of the jobs. That is, we have a sequence $p_1, \ldots, p_t$, where each p_i is either a singleton vertex or an edge of G so that $p_1, \ldots, p_t$ partition X and so that if $x \in p_i$ and $y \in p_j$ and $x < y$ then $i < j$.[15]

Now the pairs in this list should form a matching M in G. Hence $t = |X| - |M|$. In particular, t cannot be smaller than $|X| - \nu(G)$, where $\nu(G)$ is the matching number of G.

Now it can be shown that in fact one can always make a schedule with $t = |X| - \nu(G)$. To this end, let Q be a minimum partition of V into vertices and edges of G, and let Y be the set of minimal elements of X. If $q \subseteq Y$ for some $q \in Q$, we can replace X by $X \setminus q$ and Q by $Q \setminus \{q\}$, and apply induction.

So we may assume that each $y \in Y$ is contained in an edge $yz \in Q$ with $z \notin Y$. Choose an edge $yz \in Q$ such that $y \in Y$ and such that the height of z is as small as possible. (The *height* of an element z is the maximum size of a chain in $(X, \leq)$ with maximum element z.) As $z \notin Y$ there exists an $y'z' \in Q$ with $y' \in Y$ and $y' < z$.

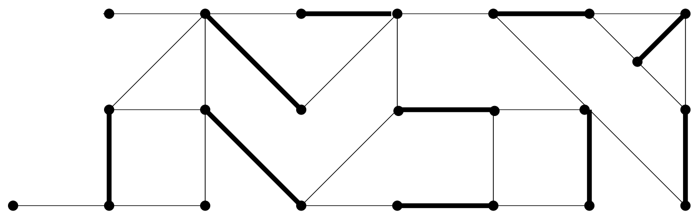
Now clearly yy' is an edge of G, as y and y' are minimal elements. Moreover, zz' is an edge of G. For if $z < z'$ then $y' < z < z'$, contradicting the fact that $y'z' \in EG$; and if $z' < z$ than z' would have smaller height than z.

So replacing yz and $y'z'$ in Q by yy' and zz', we have $yy' \subseteq Y$, and we can apply induction as before.

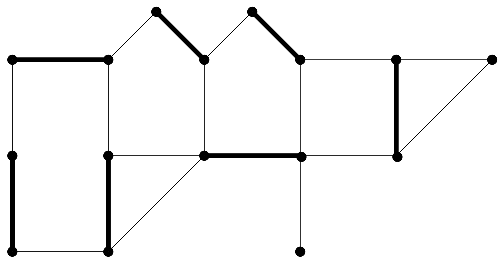
Exercises

5.7. Apply the matching augmenting algorithm to the matchings in the following graphs:

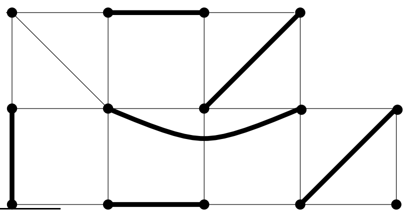
(i)



(ii)



(iii)



[15]Here we identify a vertex v with the set $\{v\}$.

5.3. Weighted matching algorithm

Edmonds [1965a] proved that also the maximum-weight matching problem can be solved in polynomial time. Equivalently, the minimum-weight perfect matching problem can be solved in polynomial time. It is the problem:

(16) given: a graph $G = (V, E)$ and a 'weight' function $w : E \to \mathbb{Q}$;
find: a perfect matching M minimizing $\sum_{e \in M} w(e)$.

We describe the algorithm, assuming without loss of generality that G has at least one perfect matching and that $w(e) \geq 0$ for each edge e (we can add a constant to all edge weights without changing the problem).

Like the cardinality matching algorithm, the weighted matching algorithm is based on shrinking sets of vertices. Unlike the cardinality matching algorithm however, for weighted matchings one has to 'deshrink' sets of vertices (the reverse operation of shrinking). Thus we have to keep track of the shrinking history throughout the iterations.

The algorithm is 'primal-dual'. The 'vehicle' carrying us to a minimum-weight perfect matching is a pair of a nested[16] collection Ω of odd-size subsets of V, and a function $\pi : \Omega \to \mathbb{Q}$ satisfying:

(17)
$$\begin{array}{lll}\text{(i)} & \pi(U) \geq 0 & \text{if } U \in \Omega \text{ with } |U| \geq 3,\\ \text{(ii)} & \displaystyle\sum_{\substack{U \in \Omega\\ e \in \delta(U)}} \pi(U) \leq w(e) & \text{for each } e \in E.\end{array}$$

This implies that for each perfect matching N in G one has $w(N) \geq \sum_{U \in \Omega} \pi(U)$, since

(18)
$$w(N) = \sum_{e \in N} w(e) \geq \sum_{e \in N} \sum_{\substack{U \in \Omega\\ e \in \delta(U)}} \pi(U) = \sum_{U \in \Omega} \pi(U) |N \cap \delta(U)| \geq \sum_{U \in \Omega} \pi(U).$$

Notation and assumptions. Let be given Ω and $\pi : \Omega \to \mathbb{Q}$. Define

(19)
$$w_\pi(e) := w(e) - \sum_{\substack{U \in \Omega\\ e \in \delta(U)}} \pi(U)$$

for any edge $e \in E$. (So (17)(ii) implies $w_\pi(e) \geq 0$.)

G/Ω denotes the graph obtained from G by shrinking all sets in $\Omega^{\max}$, the set of inclusionwise maximal sets in Ω. We will assume throughout that $\{v\} \in \Omega$ for each $v \in V$. Hence, as Ω is nested and covers V, $\Omega^{\max}$ is a partition of V.

When shrinking a set $U \in \Omega$, we denote the new vertex representing the shrunk set U just by U. So G/Ω has vertices the sets in $\Omega^{\max}$, with two distinct elements $U, U' \in \Omega^{\max}$ adjacent if and only if G has an edge connecting U and U'. We denote any edge of G/Ω by the original edge in G.

Throughout we restrict ourselves to Ω and π satisfying:

(20) for each $U \in \Omega$ with $|U| \geq 3$, the graph obtained from $G|U$ by shrinking all inclusionwise maximal proper subsets of U that are in Ω, has a Hamiltonian circuit C_U of edges e with $w_\pi(e) = 0$.

Hungarian forests. An important role in the algorithm is played by a so-called 'Hungarian forest' relative to a matching M.

Let M be a matching in a graph $G = (V, E)$ and let W be the set of vertices missed by M. Then a subset F of E is an M-*Hungarian forest* in G if F is a forest containing M such that each

[16]A collection Ω of subsets of a set V is called *nested* if $U \cap W = \emptyset$ or $U \subseteq W$ or $W \subseteq U$ for any $U, W \in \Omega$.

component of (V, F) consists either of an edge in M or contains exactly one vertex in W and such that each simple path in F starting in W is M-alternating.

The set of vertices $v \in V$ for which there exists an even-length (odd-length, respectively) $W - v$ path in F is denoted by even(F) (odd(F), respectively).

The algorithm. We iterate with Ω and $\pi : \Omega \to \mathbb{Q}$ satisfying (17) and (20), a matching M in G/Ω and an M-Hungarian forest F in G/Ω with $w_\pi(F) = 0$.

Initially, we set $M := \emptyset$, $F := \emptyset$, $\Omega := \{\{v\} \mid v \in V\}$, and $\pi(\{v\}) := 0$ for each $v \in V$. Then, as long as M is not a perfect matching in G/Ω, we perform the following iteratively:

(21) Reset $\pi(U) := \pi(U) - \varepsilon$ for $U \in$ odd(F) and $\pi(U) := \pi(U) + \varepsilon$ for $U \in$ even(F), where ε is the largest value such that (17) is maintained. After that

(i) there exists an edge e of G/Ω with $w_\pi(e) = 0$ such that e intersects even(F) but not odd(F),
or (ii) there exists a $U \in$ odd(F) with $|U| \geq 3$ and $\pi(U) = 0$.

First assume (i) holds. If only one end of e belongs to even(F), extend F by e. If both ends of e belong to even(F) and $F \cup \{e\}$ contains an M-blossom U, add U to Ω (defining $\pi(U) := 0$), replace F by F/U and M by M/U. If both ends of e belong to even(F) and $F \cup \{e\}$ contains an M-augmenting path, augment M and reset $F := M$.
Next assume (ii) holds. Delete U from Ω, replace F by $F \cup P \cup N$ and M by $M \cup N$, where P is the even-length path in C_U connecting the two edges of F incident with U and where N is the matching in C_U covering all vertices in U that are not covered by M.

(Note that in this iteration ε is bounded, since $\sum_{U \in \Omega} \pi(U)$ is bounded (by (18), as there is at least one perfect matching), and since $|\text{even}(F)| > |\text{odd}(F)|$ (as M is not perfect).)

If M is a perfect matching in G/Ω, we are done: by (20) we can expand M to a perfect matching N in G with $w_\pi(N) = 0$ and $|N \cap \delta(U)| = 1$ for each $U \in \Omega$; then N has equality throughout in (18), and hence it is a minimum-weight perfect matching.

Theorem 5.5. *There are at most $|V|^2$ iterations.*

Proof. In any iteration where we augment M, the value of $|V(G/\Omega)| - 2|M|$ decreases by 2. If there is no matching augmentation, this value remains invariant. So there are at most $\frac{1}{2}|V|$ matching augmentations.

Let V_{even} be the set of vertices $v \in V$ that are shrunk to a vertex in even(F). Let Ω_0 be the set of vertices of G/Ω that do not belong to even(F). Then in any iteration with no matching augmentation, $2|V_{\text{even}}| + |\Omega_0|$ increases. Since this value cannot exceed $2|V|$, between any two matching augmentations there are at most $2|V|$ iterations. ∎

This gives the theorem of Edmonds [1965a]:

Corollary 5.5a. *A minimum-weight perfect matching can be found in polynomial time.*

Proof. The nestedness of Ω implies that $|\Omega| \leq 2|V|$ (which is an easy exercise — see Exercise 5.10). Hence each iteration can be performed in polynomial time. With any $U \in \Omega$ with $|U| \geq 3$ we should keep the Hamiltonian circuit C_U of (20) — which we had obtained earlier when shrinking U. ∎

As a consequence one can derive:

Corollary 5.5b. *In any graph with weight function on the edges, a maximum-weight matching can be found in polynomial time.*

Proof. Left to the reader. (Exercise 5.9.) ▮

The above algorithm can be implemented in time $O(|V|^3)$, which is a result of Gabow [1973] and Lawler [1976]. Faster algorithms were given by Galil, Micali, and Gabow [1986] $(O(|E||V|\log|V|))$ and Gabow [1990] $(O(|V||E|+|V|^2\log|V|))$.

For more about matchings we refer to the book of Lovász and Plummer [1986].

Application 5.3: Optimal pairing. In several practical situations one has to find an 'optimal pairing', for example, when scheduling crews for airplanes. Also if one has to assign bus seats optimally to the participants of an organized tour, or to accommodate the participants most satisfactorily in two-bed hotel rooms, one has to solve a maximum-weight perfect matching problem.

Application 5.4: Airline timetabling. A European airline company has for its European flights a number of airplanes available. Each plane can make on any day two return flights to European destinations (not necessarily the same destinations). The profit one makes on any flight depends on the departure and arrival times of the flight (also due to intercontinental connections). The company wants to make a timetable so that it can be performed by the available fleet and so that the total profit is maximized. Assume that the number of destinations to be reached is equal to twice the number of airplanes available.

To solve this problem, consider the complete graph with vertex set all possible destinations. For each edge of this graph, connecting destinations B and C say, one calculates the profit that will be made if one and the same air plane will make its flights to B and C (in one order or the other). So one determines the optimum schedule for the flights to B and C so that the two return flights can be done by the same airplane and so that the total profit on the two flights is maximized.

Now a timetable yielding maximum profit is found by determining a maximum-weight perfect matching in this graph.

Application 5.5: Chinese postman problem. The *Chinese postman problem*, first studied by Guan [1960], consists of the following. Given a graph $G=(V,E)$ and a length function $l:E\to\mathbb{Q}_+$, find a minimum-length tour T that traverses each edge *at least* once.

It is not difficult to see that if each vertex of G has an even degree, then the optimal tour traverses each edge *exactly* once. But if the graph has vertices of odd degree, certain edges have to be traversed more than once. To find such edges we can proceed as follows.

First determine the set U of vertices of odd degree. Note that $|U|$ is even. For each pair u,u' of vertices in U determine the distance $d(u,u')$ between u and u' in the graph G (taking l as length). Consider the complete graph $H=(U,E')$ on U. Determine a minimum-weight perfect matching M in H, taking d as weight function. For each edge uu' in M we can determine a path $P_{u,u'}$ in G of length $d(u,u')$. It can be shown that any two different such paths do not have any edge in common (assuming that each edge has positive length) — see Exercise 5.13. Let $\tilde{E}$ be the set of edges occurring in the $P_{u,u'}$ $(uu'\in M)$. Then there exists a tour T so that each edge $e\in E\setminus\tilde{E}$ is traversed exactly once and each edge $e\in\tilde{E}$ is traversed exactly twice. This tour T is a shortest 'Chinese postman tour'.

Application 5.6: Christofides' approximative algorithm for the traveling salesman problem. Christofides [1976] designed the following algorithm to find a short traveling salesman tour in a graph (generally not the shortest however). The *traveling salesman problem* is the problem, given a finite set V and a 'length' function $l:V\times V\to\mathbb{Q}_+$, to find a shortest traveling salesman tour. A *traveling salesman tour* (or *Hamiltonian circuit*) is a circuit in the complete graph on V traversing each vertex exactly once.

Suppose that the length function satisfies the triangle inequality:

(22) $$l(u,w)\le l(u,v)+l(v,w)$$

for all $u,v,w\in V$. Then a reasonably short traveling salesman tour can be found as follows.

First determine a shortest spanning tree S (with the greedy algorithm). Next, let U be the set of vertices that have odd degree in S. Find a shortest perfect matching M on U, taking l as weight function. Now

$ES \cup M$ forms a set of edges such that each vertex has even degree. (If an edge occurs both in ES and in M, we take it as two parallel edges.) So we can make a cycle T such that each edge in $ES \cup M$ is traversed exactly once. Then T traverses each vertex at least once. By inserting shortcuts we obtain a traveling salesman tour T' with length$(T') \leq$length(T).

How far away is the length of T' from the length of a shortest traveling salesman tour? Let ρ be the length of a shortest traveling salesman tour. It is not difficult to show that:

(23)
(i) length$(S) \leq \rho$;
(ii) length$(M) \leq \frac{1}{2}\rho$.

(Exercise 5.17.) Hence

(24) $$\text{length}(T') \leq \text{length}(T) = \text{length}(S) + \text{length}(M) \leq \tfrac{3}{2}\rho.$$

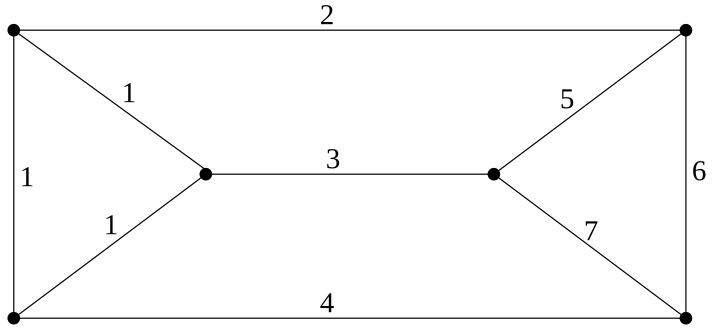
So the tour obtained with Christofides' algorithm is not longer than $\frac{3}{2}$ times the optimal tour.

The factor $\frac{3}{2}$ seems quite large, but it is the smallest factor for which a polynomial-time method is known. Don't forget moreover that it is a *worst-case* bound, and that in practice (or in average) the algorithm might have a much better performance.

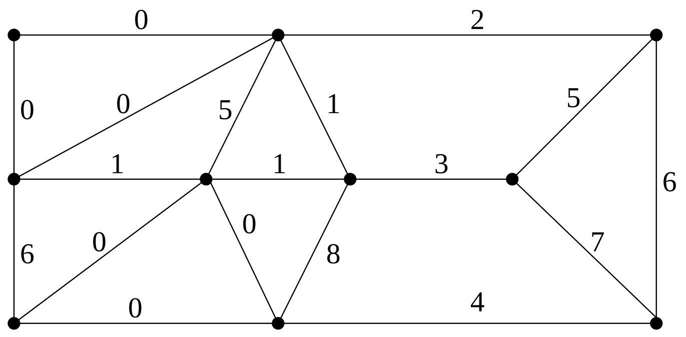
Exercises

5.8. Find with the weighted matching algorithm a minimum-weight perfect matching in the following weighted graphs:

(i)



(ii)



5.9. Derive Corollary 5.5b from Corollary 5.5a.

5.10. A collection Ω of subsets of a finite set V is called *cross-free* if:

(25) if $X, Y \in \Omega$, then $X \subseteq Y$, or $Y \subseteq X$, or $X \cap Y = \emptyset$, or $X \cup Y = V$.

Show that if Ω is cross-free, then $|\Omega| \leq 4|V|$.

5.11. Find a shortest Chinese postman route in the graph in Figure 5.1.

5.12. Find a shortest Chinese postman route in the map of Figure 5.2.

5.13. Show that the paths found in the algorithm for the Chinese postman problem pairwise do not have any edge in common (if each edge has positive length).

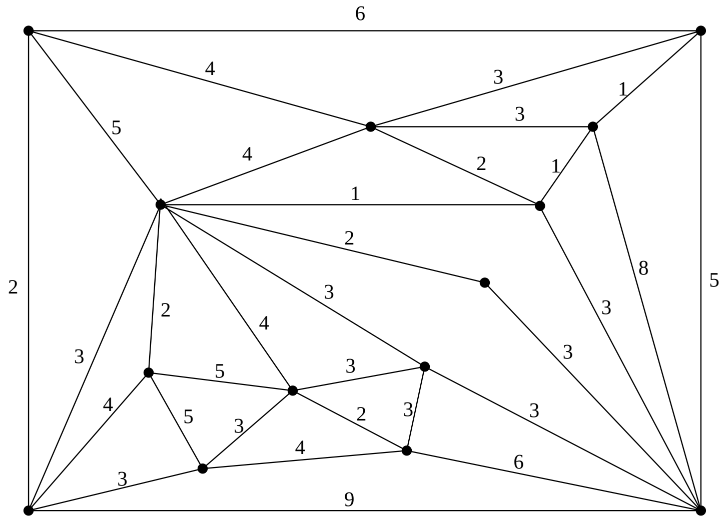


Figure 5.1

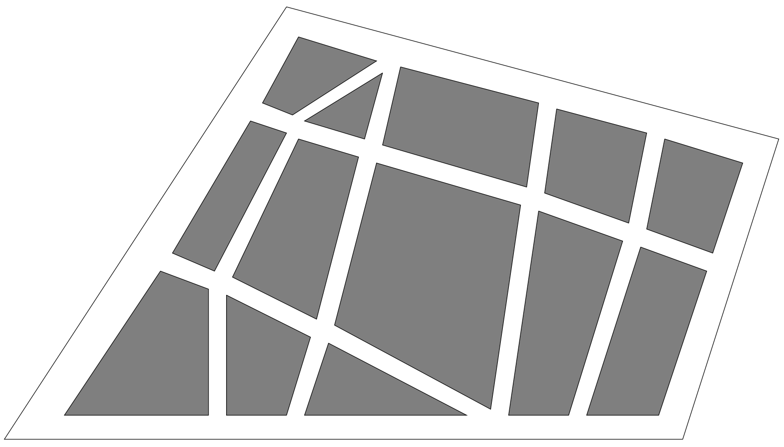


Figure 5.2 Part of the Xuhui district of Shanghai

5.14. Apply Christofides' algorithm to the table in Exercise 1.8.

5.15. Let $G = (V, E)$ be a graph and let $T \subseteq V$ with $|T|$ even. Call a subset F of E a *T-join* if T is equal to the set of vertices of odd degree in the graph (V, F).

Derive from Corollary 5.5a that a minimum-weight T-join can be found in polynomial time.

5.16. Let $G = (V, E)$ be a graph and let $l : E \to \mathbb{Q}$ be a length function such that each circuit has nonnegative length. Let $r, s \in V$.

Derive from the minimum-weight perfect matching algorithm an algorithm to find a minimum-length (simple) $r - s$ path in G.

5.17. Show (23).

5.4. The matching polytope

The weighted matching algorithm of Edmonds [1965a] gives as a side result a characterization of the perfect matching polytope $P_{\text{perfect matching}}(G)$ of any graph G. This is Edmonds' matching polytope theorem.

The *perfect matching polytope* of a graph $G = (V, E)$, denoted by $P_{\text{perfect matching}}(G)$, is the convex hull of the incidence vectors of the perfect matchings in G.[17] That is,

(26) $$P_{\text{perfect matching}}(G) = \text{conv.hull}\{\chi^M \mid M \text{ perfect matching in } G\}.$$

So $P_{\text{perfect matching}}(G)$ is a polytope in $\mathbb{R}^E$.

In Section 3.5 we saw that for a bipartite graph $G = (V, E)$, the perfect matching polytope is fully determined by the following set of inequalities:

(27) $$\begin{array}{llll} \text{(i)} & x_e & \geq 0 & \text{for each } e \in E; \\ \text{(ii)} & \sum_{e \ni v} x_e & = 1 & \text{for each } v \in V. \end{array}$$

These inequalities are not enough for, say, K_3: taking $x(e) := \frac{1}{2}$ for each edge e of K_3 gives a vector x satisfying (27) but not belonging to the perfect matching polytope of K_3.

Edmonds [1965a] showed that it is enough to add the following set of inequalities:

(28) $$\sum_{e \in \delta(U)} x_e \geq 1 \text{ for each odd subset } U \text{ of } V.$$

It is clear that for any perfect matching M in G the incidence vector χ^M satisfies (28). So clearly, $P_{\text{perfect matching}}(G)$ is contained in the polyhedron Q defined by (27) and (28). The essence of Edmonds' theorem is that one does not need more.

In order to show Edmonds' theorem, we derive from Edmonds' algorithm the following theorem, where $\mathcal{P}_{\text{odd}}(V)$ denotes the collection of odd subsets of V:

Theorem 5.6. *Let $G = (V, E)$ be a graph and let $w : E \to \mathbb{Q}$ be a 'weight' function. Then the minimum weight of a perfect matching is equal to the maximum value of $\sum_{X \in \mathcal{P}_{\text{odd}}(V)} \pi(X)$ where π ranges over all functions $\pi : \mathcal{P}_{\text{odd}}(V) \to \mathbb{Q}$ satisfying* (17).

Proof. We may assume that w is nonnegative: if μ is the minimum value of $w(e)$ over all edges e, decreasing each $w(e)$ by μ decreases both the maximum and the minimum by $\frac{1}{2}|V|\mu$.

The fact that the minimum is not smaller than the maximum follows from (18). Equality follows from the fact that in the algorithm the final perfect matching and the final function π have equality throughout in (18). ▮

This implies:

Corollary 5.6a (Edmonds' perfect matching polytope theorem)**.** *The perfect matching polytope of any graph $G = (V, E)$ is determined by* (27) *and* (28).

Proof. By Theorem 5.6 and LP-duality, for any weight function $w \in \mathbb{Q}^E$, the minimum weight of a perfect matching is equal to the minimum of $w^T x$ taken over the polytope determined by (27) and (28). Hence the two polytopes coincide. ▮

From this one can derive Edmonds' matching polytope theorem, characterizing the *matching polytope* of a graph $G = (V, E)$, denoted by $P_{\text{matching}}(G)$, which is the convex hull of the incidence vectors of the matchings in G. That is,

(29) $$P_{\text{matching}}(G) = \text{conv.hull}\{\chi^M \mid M \text{ matching in } G\}.$$

[17] For any finite set X and any subset Y of X, the *incidence vector* or *incidence function* of a subset Y of X is the vector $\chi^Y \in \mathbb{R}^X$ defined by: $\chi^Y_x := 1$ if $x \in Y$ and $\chi^Y_x := 0$ otherwise.

Again, $P_{\text{matching}}(G)$ is a polytope in $\mathbb{R}^E$.

Corollary 5.6b (Edmonds' matching polytope theorem)**.** *For any graph $G = (V, E)$ the matching polytope is determined by:*

$$\text{(30)}\qquad \begin{array}{rrll} \text{(i)} & x_e & \geq 0 & \text{for each } e \in E; \\ \text{(ii)} & \sum_{e \ni v} x_e & \leq 1 & \text{for each } v \in V; \\ \text{(iii)} & \sum_{e \subseteq U} x_e & \leq \lfloor \frac{1}{2}|U| \rfloor & \text{for each } U \subseteq V \text{ with } |U| \text{ odd.} \end{array}$$

Proof. Left to the reader (Exercise 5.20). ∎

This in turn has the following consequence:

Corollary 5.6c. *Let $G = (V, E)$ be a graph and let $w : E \to \mathbb{Q}_+$. Then the maximum weight of a matching is equal to the minimum value of*

$$\text{(31)}\qquad \sum_{v \in V} y_v + \sum_{U \subseteq V} z_U \lfloor \frac{1}{2}|U| \rfloor,$$

where $y \in \mathbb{Q}_+^V$ and $z \in \mathbb{Q}_+^{\mathcal{P}_{\text{odd}}(V)}$ satisfy $\sum_{v \in e} y_v + \sum_{U \in \mathcal{P}_{\text{odd}}(V), e \subseteq U} z_U \geq w(e)$ for each edge e.

Proof. Directly with LP-duality from Corollary 5.6b. ∎

In fact, Cunningham and Marsh' theorem shows that if w is integer-valued, we can restrict y and z to integer vectors — see Section 5.5.

Exercises

5.18. Show that for any graph $G = (V, E)$, if the inequalities (30)(i)(ii) fully determine the matching polytope, then G is bipartite.

5.19. Show that the perfect matching polytope of a graph $G = (V, E)$ is also determined by the following inequalities:

$$\text{(32)}\qquad \begin{array}{rll} x_e & \geq 0 & \text{for each } e \in E; \\ \sum_{e \in \delta(U)} x_e & \geq 1 & \text{for each odd subset } U \text{ of } V; \\ \sum_{e \in E} x_e & = \frac{1}{2}|V|. & \end{array}$$

5.20. Derive Edmonds' matching polytope theorem from Edmonds' perfect matching polytope theorem.

5.21. Derive from Edmonds matching polytope theorem the linear inequalities determining the convex hull of all *symmetric* permutation matrices.

5.22. Let $G = (V, E)$ be a graph. Show that the convex hull of the incidence vectors of matchings of size k is equal to the intersection of the matching polytope of G with the hyperplane $\{x \mid 1^T x = k\}$.

5.23. Let $G = (V, E)$ be a graph. Show that the convex hull of the incidence vectors of matchings of size at least k and at most l is equal to the intersection of the matching polytope of G with the set $\{x \mid k \leq 1^T x \leq l\}$.

5.5. The Cunningham-Marsh formula

Cunningham and Marsh [1978] showed a more general result, which generalizes both Edmonds' matching polytope theorem and the Tutte-Berge formula. We give a direct proof.

Theorem 5.7 (Cunningham-Marsh formula)**.** *In Corollary* 5.6c*, if w is integer, we can take y and z integer.*

Proof. We must give a matching M and integer values y_v, z_U as required with $w(M)$ equal to (31).

Let T be equal to the maximum weight of a matching and let $\mathcal{M}$ be the set of matchings M of weight T. We prove the theorem by induction on T. We may assume that G is the complete graph on V. Let G, w be a counterexample to the theorem with (fixing V and T) $\sum_{e\in E} w(e)$ as large as possible.

First assume that there exists a vertex u of G covered by every matching $M \in \mathcal{M}$. Let w' be obtained from w by decreasing $w(e)$ by 1 for each edge e incident with u with $w(e) \geq 1$. Then the maximum of $w'(M)$ over all matchings M is equal to $T-1$, since each $M \in \mathcal{M}$ contains an edge e incident with u with $w(e) \geq 1$. Hence, by induction, there exist y'_v, z'_U as required for w'. Now increasing y'_u by 1 and leaving all other values of y'_v, z'_U invariant, gives y_v, z_U as required for w.

So we may assume that for each vertex v there exists a matching $M \in \mathcal{M}$ not covering v. We show that for each three distinct vertices $a, b, c \in V$ one has

$$w(ac) \geq \min\{w(ab), w(bc)\}. \tag{33}$$

Indeed, by the maximality of $\sum_{e\in E} w(e)$ there exists a matching $M \in \mathcal{M}$ containing ac. (Otherwise we could increase the weight of ac without increasing T, contradicting the maximality of $\sum_{e\in E} w(e)$.) Moreover, there exists a matching $M' \in \mathcal{M}$ not covering b. Let P be the component of $M \cup M'$ containing ac. At least one component, Q say, of $P \setminus \{ac\}$ does not contain b. By symmetry of a and c we may assume that Q contains a. Then $M\triangle(Q \cup \{ac\})$ and $M'\triangle(Q \cup \{ab\})$ are matchings again. Now $w(M\triangle(Q\cup\{ac\})) \leq T = w(M)$, and so $w(Q \cap M') \leq w(Q \cap M) + w(ac)$. Moreover, $w(M'\triangle(Q \cup \{ab\})) \leq T = w(M')$, and so $w(Q\cap M) + w(ab) \leq w(Q \cap M')$. Hence $w(ab) \leq w(ac)$, proving (33).

For each natural number $n \geq 1$ let G_n be the graph on V with as edges all $e \in E$ with $w(e) \geq n$, and let $\mathcal{K}_n$ be the set of components of G_n. Consider some n and some $U \in \mathcal{K}_n$.

By (33), $G|U$ is a complete graph. We show that each $M \in \mathcal{M}$ contains exactly $\lfloor \frac{1}{2}|U|\rfloor$ edges that are in EU (= set of edges contained in U).

Suppose to the contrary that U contains two vertices a and b such that a and b are not covered by any edge in $M \cap EU$. If a or b is not covered by M we could replace the edge in M incident with a or b (if any) by the edge ab, thereby increasing the weight — a contradiction. So we may assume that $ac, bd \in M$ for some $c, d \notin U$. By (33), $w(cd) \geq \min\{w(ac), w(ad)\} \geq \min\{w(ac), w(ab), w(bd)\} = \min\{w(ac), w(bd)\}$. Since $w(ab) > \max\{w(ac), w(bd)\}$ this implies $w(ab) + w(cd) > w(ac) + w(bd)$. Therefore, replacing ac and bd in M by ab and cd would increase the weight — a contradiction. So $|M \cap EU| = \lfloor \frac{1}{2}|U|\rfloor$.

For each $U \subseteq V$ with $|U| > 1$, define z_U as the number of natural numbers $n \geq 1$ for which $U \in \mathcal{K}_n$. Then $\sum_{U\supseteq e} z_U \geq w(e)$ for each edge e (since e is in $w(e)$ graphs G_n). Moreover, choose $M \in \mathcal{M}$ arbitrarily. Then

$$\begin{aligned}\sum_{U\subseteq V} z_U \lfloor\frac{1}{2}|U|\rfloor &= \sum_{n=1}^{\infty}\sum_{U\in\mathcal{K}_n}\lfloor\frac{1}{2}|U|\rfloor = \sum_{n=1}^{\infty}\sum_{U\in\mathcal{K}_n}|M\cap EU| \\ &= \sum_{e\in M}(\text{number of } n, U \text{ with } e \subseteq U \in \mathcal{K}_n) = \sum_{e\in M} w(e).\end{aligned} \tag{34}$$

∎

Exercises

5.24. Derive the Tutte-Berge formula from the Cunningham-Marsh formula (Theorem 5.7).

5.25. Derive Edmonds' matching polytope theorem from the Cunningham-Marsh formula (Theorem 5.7).

6. Problems, algorithms, and running time

6.1. Introduction

Probably most of the readers will have some intuitive idea about what is a problem and what is an algorithm, and what is meant by the running time of an algorithm. Although for the greater part of this course this intuition will be sufficient to understand the substance of the matter, in some cases it is important to formalize this intuition. This is particularly the case when we deal with concepts like NP and NP-complete.

The class of problems solvable in polynomial time is usually denoted by P. The class NP, that will be described more precisely below, is a class of problems that might be larger (and many people believe it *is* larger). It includes most combinatorial optimization problems, including all problems that are in P. That is: P⊆NP. In particular, NP does **not** mean: "non-polynomial time". The letters NP stand for "nondeterministic polynomial-time". The class NP consists, roughly speaking, of all those questions with the property that for any input that has a positive answer, there is a 'certificate' from which the correctness of this answer can be derived in polynomial time.

For instance, the question:

(1) 'Given a graph G, is G Hamiltonian?'

belongs to NP. If the answer is 'yes', we can convince anyone that this answer is correct by just giving a Hamiltonian circuit in G as a certificate. With this certificate, the answer 'yes' can be checked in polynomial time — in fact: trivially. Here it is not required that we are able to *find* the certificate in polynomial time. The only requirement is that there *exists* a certificate which can be checked in polynomial time.

Checking the certificate in polynomial time means: checking it in time bounded by a polynomial in the original input. In particular, it implies that the certificate itself has size bounded by a polynomial in the original input.

To elucidate the meaning of NP, it is not known if for any graph G for which question (1) has a *negative* answer, there is a certificate from which the correctness of this answer can be derived in polynomial time. So there is an easy way of convincing 'your boss' that a certain graph is Hamiltonian (just by exhibiting a Hamiltonian circuit), but no easy way is known for convincing this person that a certain graph is non-Hamiltonian.

Within the class NP there are the "NP-complete" problems. These are by definition the hardest problems in the class NP: a problem Π in NP is NP-*complete* if every problem in NP can be reduced to Π, in polynomial time. It implies that if one NP-complete problem can be proved to be solvable in polynomial time, then *each* problem in NP can be solved in polynomial time. In other words: then P=NP would follow.

Surprisingly, there are several prominent combinatorial optimization problems that are NP-complete, like the traveling salesman problem and the problem of finding a maximum clique in a graph. This pioneering eye-opener was given by Cook [1971] and Karp [1972].

Since that time one generally sets the polynomially solvable problems against the NP-complete problems, although there is no proof that these two concepts really are distinct. For almost every combinatorial optimization problem one has been able either to prove that it is solvable in polynomial time, or that it is NP-complete. But theoretically it is still a possibility that these two concepts are just the same! Thus it is unknown which of the two diagrams in Figure 6.1 applies.

Below we make some of the notions more precise. We will not elaborate all technical details fully, but hope that the reader will be able to see the details with not too much effort. For precise discussions we refer to the books by Aho, Hopcroft, and Ullman [1974], Garey and Johnson [1979], and Papadimitriou [1994].

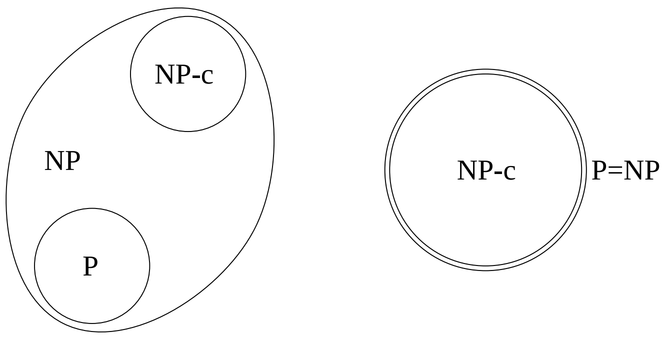


Figure 6.1

6.2. Words

If we use the computer to solve a certain graph problem, we usually do not put a picture of the graph in the computer. (We are not working with analog computers, but with digital computers.) Rather we put some appropriate encoding of the problem in the computer, by describing it by a sequence of symbols taken from some fixed finite 'alphabet' Σ. We can take for Σ for instance the ASCII set of symbols or the set $\{0, 1\}$. It is convenient to have symbols like (,) , { , } and the comma in Σ, and moreover some symbol like ␣ meaning: 'blank'. Let us fix one alphabet Σ.

We call any ordered finite sequence of elements from Σ a *word*. The set of all words is denoted by Σ^*.

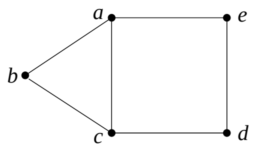


Figure 6.2

It is not difficult to encode objects like rational numbers, vectors, matrices, graphs, and so on, as words. For instance, the graph given in Figure 6.2 can be encoded, as usual, by the word:

(2) $$(\{a, b, c, d, e\}, \{\{a, b\}, \{a, c\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, a\}\}).$$

A function f defined on a finite set X can be encoded by giving the set of pairs $(x, f(x))$ with $x \in X$. For instance, the following describes a function defined on the edges of the graph above:

(3) $$\{(\{a, b\}, 32), (\{a, c\}, -17), (\{b, c\}, 5/7), (\{c, d\}, 6), (\{d, e\}, -1), (\{e, a\}, -9)\}.$$

A pair of a graph and a function can be described by the word (w, v), where w is the encoding of the graph and v is the encoding of the function.

The *size* of a word w is the number of symbols used in w, counting multiplicities. (So the word $abaa32bc$ has size 8.) The size is important when we make estimates on the running time of algorithms.

Note that in encoding numbers (integers or rational numbers), the size depends on the number of symbols necessary to encode these numbers. Thus if we encounter a problem on a graph with numbers defined on the edges, then the size of the input is the total number of bits necessary to represent this structure. It might be much larger than just the number of nodes and edges of the graph, and much smaller than the sum of all numbers occurring in the input.

Although there are several ways of choosing an alphabet and encoding objects by words over this alphabet, any way chosen is quite arbitrary. We will be dealing with solvability in polynomial time in this chapter, and for that purpose most encodings are equivalent. Below we will sometimes exploit this flexibility.

6.3. Problems

What is a problem? Informally, it is a question or a task, for instance, "Does this given graph have a perfect matching?" or "Find a shortest traveling salesman tour in this graph!". In fact there are two types of problems: problems that can be answered by 'yes' or 'no' and those that ask you to find an object with certain prescribed properties. We here restrict ourselves to the first type of problems. From a complexity point of view this is not that much of a restriction. For instance, the problem of finding a shortest traveling salesman tour in a graph can be studied by the related problem: Given a graph, a length function on the edges, and a rational number r, does there exist a traveling salesman tour of length at most r? If we can answer this question in polynomial time, we can find the length of a shortest tour in polynomial time, for instance, by binary search.

So we study problems of the form: Given a certain object (or sequence of objects), does it have a certain property? For instance, given a graph G, does it have a perfect matching?

As we encode objects by words, a problem is nothing but: given a word w, does it have a certain property? Thus the problem is fully described by describing the "certain property". This, in turn, is fully described by just the set of all words that have the property. Therefore we have the following mathematical definition: a *problem* is any subset Π of Σ^*.

If we consider any problem $\Pi \subseteq \Sigma^*$, the corresponding 'informal' problem is:

(4) Given word w, does w belong to Π?

In this context, the word w is called an *instance* or the *input*.

6.4. Algorithms and running time

An algorithm is a list of instructions to solve a problem. The classical mathematical formalization of an algorithm is the *Turing machine*. In this section we will describe a slightly different concept of an algorithm (the 'Thue system') that is useful for our purposes (explaining NP-completeness). In Section 6.10 below we will show that it is equivalent to the notion of a Turing machine.

A basic step in an algorithm is: replace subword u by u'. It means that if word w is equal to tuv, where t and v are words, we replace w by the word $tu'v$. Now by definition, an *algorithm* is a finite list of instructions of this type. It thus is fully described by a sequence

(5) $((u_1, u_1'), \ldots, (u_n, u_n')),$

where $u_1, u_1', \ldots, u_n, u_n'$ are words. We say that word w' *follows from* word w if there exists a $j \in \{1, \ldots, n\}$ such that $w = tu_jv$ and $w' = tu_j'v$ for certain words t and v, in such a way that j is the smallest index for which this is possible and the size of t is as small as possible. The algorithm *stops at* word w if w has no subword equal to one of $u_1, \ldots, u_n$. So for any word w, either there is a unique word w' that follows from w, or the algorithm stops at w. A (finite or infinite) sequence of words $w_0, w_1, w_2, \ldots$ is called *allowed* if each w_{i+1} follows from w_i and, if the sequence is finite, the algorithm stops at the last word of the sequence. So for each word w there is a unique allowed sequence starting with w. We say that A *accepts* w if this sequence is finite.

For reasons of consistency it is important to have the 'empty space' at both sides of a word as part of the word. Thus instead of starting with a word w, we start with $\textvisiblespace w \textvisiblespace$, where $\textvisiblespace$ is a symbol indicating space.

Let A be an algorithm and let $\Pi \subseteq \Sigma^*$ be a problem. We say that A *solves* Π if Π equals the set of words accepted by A. Moreover, A solves π *in polynomial-time* if there exists a polynomial $p(x)$

such that for any word $w \in \Sigma^*$: if A accepts w, then the allowed sequence starting with w contains at most $p(\text{size}(w))$ words.

This definition enables us indeed to decide in polynomial time if a given word w belongs to Π. We just take $w_0 := w$, and next, for $i = 0, 1, 2, \ldots$, we choose 'the first' subword u_j in w_i and replace it by u'_j (for some $j \in \{1, \ldots, n\}$) thus obtaining w_{i+1}. If within $p(\text{size}(w))$ iterations we stop, we know that w belongs to Π, and otherwise we know that w does not belong to Π.

Then P denotes the set of all problems that can be solved by a polynomial-time algorithm.

6.5. The class NP

We mentioned above that NP denotes the class of problems for which a positive answer has a 'certificate' from which the correctness of the positive answer can be derived in polynomial time. We will now make this more precise.

The class NP consists of those problems $\Pi \subseteq \Sigma^*$ for which there exist a problem $\Pi' \in$P and a polynomial $p(x)$ such that for any $w \in \Sigma^*$:

(6) $w \in \Pi$ if and only if there exists a word v such that $(w, v) \in \Pi'$ and such that $\text{size}(v) \leq p(\text{size}(w))$.

So the word v acts as a certificate showing that w belongs to Π. With the polynomial-time algorithm solving Π', the certificate proves in polynomial time that w belongs to Π.

As examples, the problems

(7) $\Pi_1 := \{G \mid G \text{ is a graph having a perfect matching}\}$ and
$\Pi_2 := \{G \mid G \text{ is a Hamiltonian graph}\}$

(encoding G as above) belong to NP, since the problems

(8)
$$\begin{aligned} \Pi'_1 &:= \{(G, M) \mid G \text{ is a graph and } M \text{ is a perfect matching in } G\} \text{ and} \\ \Pi'_2 &:= \{(G, H) \mid G \text{ is a graph and } H \text{ is a Hamiltonian circuit in } G\} \end{aligned}$$

belong to P.

Similarly, the problem

(9) TSP $:=$ $\{(G, l, r) \mid G$ is a graph, l is a 'length' function on the edges of G and r is a rational number such that G has a Hamiltonian tour of length at most $r\}$

('the traveling salesman problem') belongs to NP, since the problem

(10) TSP$'$ $:=$ $\{(G, l, r, H) \mid G$ is a graph, l is a 'length' function on the edges of G, r is a rational number, and H is a Hamiltonian tour in G of length at most $r\}$

belongs to P.

Clearly, P$\subseteq$NP, since if Π belongs to P, then we can just take the empty string as certificate for any word w to show that it belongs to Π. That is, we can take $\Pi' := \{(w,) \mid w \in \Pi\}$. As $\Pi \in$P, also $\Pi' \in$P.

The class NP is apparently much larger than the class P, and there might be not much reason to believe that the two classes are the same. But, as yet, nobody has been able to show that they really are different! This is an intriguing mathematical question, but besides, answering the question might also have practical significance. If P=NP can be shown, the proof might contain a revolutionary new algorithm, or alternatively, it might imply that the concept of 'polynomial-time' is completely useless. If P$\neq$NP can be shown, the proof might give us more insight in the reasons why certain problems are more difficult than other, and might guide us to detect and attack the kernel of the difficulties.

6.6. The class co-NP

By definition, a problem $\Pi \subseteq \Sigma^*$ belongs to the class co-NP if the 'complementary' problem $\overline{\Pi} := \Sigma^* \setminus \Pi$ belongs to NP.

For instance, the problem Π_1 defined in (7) belongs to co-NP, since the problem

(11) $$\Pi_1'' := \{(G,W) \mid G \text{ is a graph and } W \text{ is a subset of the vertex set of } G \text{ such that the graph } G - W \text{ has more than } |W| \text{ odd components}\}$$

belongs to P. This follows from Tutte's '1-factor theorem' (Corollary 5.1a): a graph G has no perfect matching, if and only if there is a subset W of the vertex set of G with the properties described in (11). (Here, strictly speaking, the complementary problem $\overline{\Pi_1}$ of Π_1 consists of all words w that either do not represent a graph, or represent a graph having no perfect matching. We assume however that there is an easy way of deciding if a given word represents a graph. Therefore, we might assume that the complementary problem is just $\{G \mid G$ is a graph having no perfect matching$\}$.)

It is not known if the problems Π_2 and TSP belong to co-NP.

Since for any problem Π in P also the complementary problem $\overline{\Pi}$ belongs to P, we know that P⊆co-NP. So P⊆NP∩co-NP. The problems in NP∩co-NP are those for which there exist certificates both in case the answer is positive and in case the answer is negative. As we saw above, the perfect matching problem Π_1 is such a problem. Tutte's theorem gives us the certificates. Therefore, Tutte's theorem is called a *good characterization.*

In fact, there are very few problems known that are proved to belong to NP∩co-NP, but that are not known to belong to P. Most problems having a good characterization, have been proved to be solvable in polynomial time. The notable exception for which this is not yet proved is *primality testing* (testing if a given natural number is a prime number).

6.7. NP-completeness

The NP-complete problems are by definition the hardest problems in NP. To be more precise, we first define the concept of a polynomial-time reduction. Let Π and Π' be two problems and let A be an algorithm. We say that A is a *polynomial-time reduction* of Π' to Π if A is a polynomial-time algorithm ('solving' Σ^*), so that for any allowed sequence starting with w and ending with v one has: $w \in \Pi'$ if and only if $v \in \Pi$. A problem Π is called NP-*complete*, if $\Pi \in$NP and for each problem Π' in NP there exists a polynomial-time reduction of Π' to Π.

It is not difficult to see that if Π belongs to P and there exists a polynomial-time reduction of Π' to Π, then also Π' belongs to P. It implies that if one NP-complete problem can be solved in polynomial time, then each problem in NP can be solved in polynomial time. Moreover, if Π belongs to NP, Π' is NP-complete and there exists a polynomial-time reduction of Π' to Π, then also Π is NP-complete.

6.8. NP-completeness of the satisfiability problem

We now first show that in fact there exist NP-complete problems. In fact we show that the so-called *satisfiability problem*, denoted by SAT, is NP-complete.

To define SAT, we need the notion of a *boolean expression.* Examples are:

(12) $$((x_2 \wedge x_3) \vee \neg(x_3 \vee x_5) \wedge x_2),\ ((\neg x_{47} \wedge x_2) \wedge x_{47}),\ \neg(x_7 \wedge \neg x_7).$$

Boolean expressions can be defined inductively. First, for each natural number n, the 'word' x_n is a boolean expression (using some appropriate encoding of natural numbers and of subscripts). Next, if v and w are boolean expressions, then also $(v \wedge w)$, $(v \vee w)$ and $\neg v$ are boolean expressions. These rules give us all boolean expressions. (If necessary, we may use other subscripts than the natural numbers.)

Now SAT is a subcollection of all boolean expressions, namely it consists of those boolean expressions that are satisfiable. A boolean expression $f(x_1, x_2, x_3, \ldots)$ is called *satisfiable* if there exist $\alpha_1, \alpha_2, \alpha_3, \ldots \in \{0, 1\}$ such that $f(\alpha_1, \alpha_2, \alpha_3, \ldots) = 1$, using the well-known identities

$$\begin{aligned} &0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0, 1 \wedge 1 = 1, \\ &0 \vee 0 = 0, 0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1, \\ &\neg 0 = 1, \neg 1 = 0, (0) = 0, (1) = 1. \end{aligned} \tag{13}$$

Exercise. Let $n \geq 1$ be a natural number and let W be a collection of words in $\{0, 1\}^*$ all of length n. Prove that there exists a boolean expression $f(x_1, \ldots, x_n)$ in the variables $x_1, \ldots, x_n$ such that for each word $w = \alpha_1 \ldots \alpha_n$ in the symbols 0 and 1 one has: $w \in W$ if and only if $f(\alpha_1, \ldots, \alpha_n) = 1$. ▮

The satisfiability problem SAT trivially belongs to NP: we can take as certificate for a certain $f(x_1, x_2, x_3, \ldots)$ to belong to SAT, the equations $x_i = \alpha_i$ that give f the value 1. (We only give those equations for which x_i occurs in f.)

To show that SAT is NP-complete, it is convenient to assume that $\Sigma = \{0, 1\}$. This is not that much a restriction: we can fix some order of the symbols in Σ, and encode the first symbol by 10, the second one by 100, the third one by 1000, and so on. There is an easy (certainly polynomial-time) way of obtaining one encoding from the other.

The following result is basic for the further proofs:

Theorem 6.1. *Let $\Pi \subseteq \{0, 1\}^*$ be in* P*. Then there exist a polynomial $p(x)$ and an algorithm that finds for each natural number n in time $p(n)$ a boolean expression $f(x_1, x_2, x_3, \ldots)$ with the property:*

(14) *any word $\alpha_1 \alpha_2 \ldots \alpha_n$ in $\{0, 1\}^*$ belongs to Π, if and only if the boolean expression $f(\alpha_1, \ldots, \alpha_n, x_{n+1}, x_{n+2}, \ldots)$ is satisfiable.*

Proof. Since Π belongs to P, there exists a polynomial-time algorithm A solving Π. So there exists a polynomial $p(x)$ such that a word w belongs to Π if and only if the allowed sequence for w contains at most $p(\text{size}(w))$ words. It implies that there exists a polynomial $q(x)$ such that any word in the allowed sequence for w has size less than $q(\text{size}(w))$.

We describe the algorithm meant in the theorem. Choose a natural number n. Introduce variables $x_{i,j}$ and $y_{i,j}$ for $i = 0, 1, \ldots, p(n)$, $j = 1, \ldots, q(n)$. Now there exists (cf. the Exercise above) a boolean expression f in these variables with the following properties. Any assignment $x_{i,j} := \alpha_{i,j} \in \{0, 1\}$ and $y_{i,j} := \beta_{i,j} \in \{0, 1\}$ makes f equal to 1, if and only if the allowed sequence starting with the word $w_0 := \alpha_{0,1} \alpha_{0,2} \ldots \alpha_{0,n}$ is a finite sequence $w_0, \ldots, w_k$, so that:

(15)
(i) $\alpha_{i,j}$ is equal to the jth symbol in the word w_i, for each $i \leq k$ and each $j \leq \text{size}(w_i)$;
(ii) $\beta_{i,j} = 1$ if and only if $i > k$ or $j \leq \text{size}(w_i)$.

The important point is that f can be found in time bounded by a polynomial in n. To see this, we can encode the fact that word w_{i+1} should follow from word w_i by a boolean expression in the 'variables' $x_{i,j}$ and $x_{i+1,j}$, representing the different positions in w_i and w_{i+1}. (The extra variables $y_{i,j}$ and $y_{i+1,j}$ are introduced to indicate the sizes of w_i and w_{i+1}.) Moreover, the fact that the algorithm stops at a word w also can be encoded by a boolean expression. Taking the 'conjunction' of all these boolean expressions, will give us the boolean expression f. ▮

As a direct consequence we have:

Corollary 6.1a. *Theorem* 6.1 *also holds if we replace* P *by* NP *in the first sentence.*

Proof. Let $\Pi \subseteq \{0,1\}^*$ belong to NP. Then, by definition of NP, there exists a problem Π' in P and a polynomial $r(x)$ such that any word w belongs to Π if and only if (w,v) belongs to Π' for some word v with $\text{size}(v) \leq r(\text{size}(w))$. By properly re-encoding, we may assume that for each $n \in \mathbb{N}$, any word $w \in \{0,1\}^*$ belongs to Π if and only if wv belongs to Π' for some word v of size $r(\text{size}(w))$. Applying Theorem 6.1 to Π' gives the corollary. ▌

Now the main result of Cook [1971] follows:

Corollary 6.1b (Cook's theorem)**.** *The satisfiability problem* SAT *is* NP*-complete.*

Proof. Let Π belong to NP. We describe a polynomial-time reduction of Π to SAT. Let $w = \alpha_1 \ldots \alpha_n \in \{0,1\}^*$. By Corollary 6.1a we can find in time bounded by a polynomial in n a boolean expression f such that w belongs to Π if and only if $f(\alpha_1, \ldots, \alpha_n, x_{n+1}, \ldots)$ is satisfiable. This is the required reduction to SAT. ▌

6.9. NP-completeness of some other problems

We next derive from Cook's theorem some of the results of Karp [1972]. First we show that the 3-*satisfiability problem* 3-SAT is NP-complete. Let B_1 be the set of all words $x_1, \neg x_1, x_2, \neg x_2, \ldots$. Let B_2 be the set of all words $(w_1 \vee \cdots \vee w_k)$, where $w_1, \cdots, w_k$ are words in B_1 and $1 \leq k \leq 3$. Let B_3 be the set of all words $w_1 \wedge \ldots \wedge w_k$, where $w_1, \ldots, w_k$ are words in B_2. Again, we say that a word $f(x_1, x_2, \ldots) \in B_3$ is *satisfiable* if there exists an assignment $x_i := \alpha_i \in \{0,1\}$ $(i = 1, 2, \ldots)$ such that $f(\alpha_1, \alpha_2, \ldots) = 1$ (using the identities (13)).

Now the 3-satisfiability problem 3-SAT is: Given a word $f \in B_3$, decide if it is satisfiable.

Corollary 6.1c. *The 3-satisfiability problem* 3-SAT *is* NP*-complete.*

Proof. We give a polynomial-time reduction of SAT to 3-SAT. Let $f(x_1, x_2, \ldots)$ be a boolean expression. Introduce a variable y_g for each subword g of f that is a boolean expression.

Now f is satisfiable if and only if the following system is satisfiable:

$$(16) \qquad \begin{aligned} &y_g = y_{g'} \vee y_{g''} && (\text{if } g = g' \vee g''), \\ &y_g = y_{g'} \wedge y_{g''} && (\text{if } g = g' \wedge g''), \\ &y_g = \neg y_{g'} && (\text{if } g = \neg g'), \\ &y_f = 1. \end{aligned}$$

Now $y_g = y_{g'} \vee y_{g''}$ can be equivalently expressed by: $y_g \vee \neg y_{g'} = 1, y_g \vee \neg y_{g''} = 1, \neg y_g \vee y_{g'} \vee y_{g''} = 1$. Similarly, $y_g = y_{g'} \wedge y_{g''}$ can be equivalently expressed by: $\neg y_g \vee y_{g'} = 1, \neg y_g \vee y_{g''} = 1, y_g \vee \neg y_{g'} \vee \neg y_{g''} = 1$. The expression $y_g = \neg y_{g'}$ is equivalent to: $y_g \vee y_{g'} = 1, \neg y_g \vee \neg y_{g'} = 1$.

By renaming variables, we thus obtain words $w_1, \ldots, w_k$ in B_2, so that f is satisfiable if and only if the word $w_1 \wedge \ldots \wedge w_k$ is satisfiable. ▌

We next derive that the *partition problem* PARTITION is NP-complete. This is the problem: Given a collection $\mathcal{C}$ of subsets of a finite set X, is there a subcollection of $\mathcal{C}$ that forms a partition of X?

Corollary 6.1d. *The partition problem* PARTITION *is* NP*-complete.*

Proof. We give a polynomial-time reduction of 3-SAT to PARTITION. Let $f = w_1 \wedge \ldots \wedge w_k$ be a word in B_3, where $w_1, \ldots, w_k$ are words in B_2. Let $x_1, \ldots, x_m$ be the variables occurring in f. Make a bipartite graph G with colour classes $\{w_1, \ldots, w_k\}$ and $\{x_1, \ldots, x_m\}$, by joining w_i and x_j by an edge if and only if x_j or $\neg x_j$ occurs in w_i. Let X be the set of all vertices and edges of G.

Let $\mathcal{C}'$ be the collection of all sets $\{w_i\} \cup E'$, where E' is a nonempty subset of the edge set incident with w_i. Let $\mathcal{C}''$ be the collection of all sets $\{x_j\} \cup E'_j$ and $\{x_j\} \cup E''_j$, where E'_j is the set of all edges $\{w_i, x_j\}$ so that x_j occurs in w_i and where E''_j is the set of all edges $\{w_i, x_j\}$ so that $\neg x_j$ occurs in w_i.

Now f is satisfiable, if and only if the collection $\mathcal{C}' \cup \mathcal{C}''$ contains a subcollection that partitions X. Thus we have a reduction of 3-SAT to PARTITION. ▌

We derive the NP-completeness of the *directed Hamiltonian cycle problem* DIRECTED HAMILTONIAN CYCLE: Given a directed graph, does it have a directed Hamiltonian cycle?

Corollary 6.1e. DIRECTED HAMILTONIAN CYCLE *is* NP-*complete.*

Proof. We give a polynomial-time reduction of PARTITION to DIRECTED HAMILTONIAN CYCLE. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a collection of subsets of the set $X = \{x_1, \ldots, x_k\}$. Introduce 'vertices' $r_0, r_1, \ldots, r_m, s_0, s_1, \ldots, s_k$.

For each $i = 1, \ldots, m$ we do the following. Let $C_i = \{x_{j_1}, \ldots, x_{j_t}\}$. We construct a directed graph on the vertices r_{i-1}, r_i, $s_{j_h - 1}, s_{j_h}$ (for $h = 1, \ldots, t$) and $3t$ new vertices, as in Figure 6.3. Moreover, we make arcs from r_m to s_0 and from s_k to r_0.

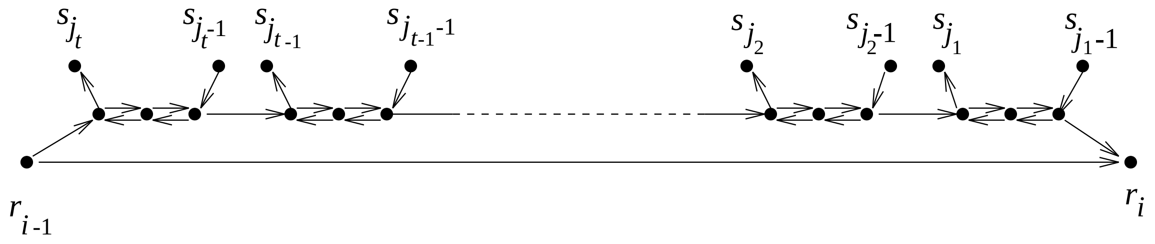


Figure 6.3

Let D be the directed graph arising. Then it is not difficult to check that there exists a subcollection $\mathcal{C}'$ of $\mathcal{C}$ that partitions X, if and only if D has a directed Hamiltonian cycle C. (Take: $(r_{i-1}, r_i) \in C \Longleftrightarrow C_i \in \mathcal{C}'$.) ▌

From this we derive the NP-completeness of the *undirected Hamiltonian cycle problem* UNDIRECTED HAMILTONIAN CYCLE: Given a graph, does it have a Hamiltonian cycle?

Corollary 6.1f. UNDIRECTED HAMILTONIAN CYCLE *is* NP-*complete.*

Proof. We give a polynomial-time reduction of DIRECTED HAMILTONIAN CYCLE to UNDIRECTED HAMILTONIAN CYCLE. Let D be a directed graph. Replace each vertex v by three vertices v', v'', v''', and make edges $\{v', v''\}$ and $\{v'', v'''\}$. Moreover, for each arc (v_1, v_2) of D, make an edge $\{v'_1, v'''_2\}$. This makes the undirected graph G. One easily checks that D has a directed Hamiltonian cycle, if and only if G has an (undirected) Hamiltonian cycle. ▌

This trivially implies the NP-completeness of the *traveling salesman problem* TSP: Given a complete graph $G = (V, E)$, a 'length' function l on E, and a rational r, does there exist a Hamiltonian cycle of length at most r?

Corollary 6.1g. *The traveling salesman problem* TSP *is* NP*-complete.*

Proof. We give a polynomial-time reduction of UNDIRECTED HAMILTONIAN CYCLE to TSP. Let G be a graph. Let G' be the complete graph on V. Let $l(e) := 0$ for each edge e of G and let $l(e) := 1$ for each edge of G' that is not an edge of G. Then G has a Hamiltonian cycle, if and only if G' has a Hamiltonian cycle of length at most 0. ∎

6.10. Turing machines

In Section 6.4 we gave a definition of 'algorithm'. How adequate is this definition? Can any computer program be modelled after that definition?

To study this question, we need to know what we understand by a 'computer'. Turing [1937] gave the following computer model, now called a *Turing machine* or a *one-tape Turing machine.*

A Turing machine consists of a 'processor' that can be in a finite number of 'states' and of a 'tape', of infinite length (in two ways). Moreover, there is a 'read-write head', that can read symbols on the tape (one at a time). Depending on the state of the processor and the symbol read, the processor passes to another (or the same) state, the symbol on the tape is changed (or not) and the tape is moved one position 'to the right' or 'to the left'.

The whole system can be described by just giving the dependence mentioned in the previous sentence. So, mathematically, a *Turing machine* is just a function

(17) $$T : M \times \Sigma \to M \times \Sigma \times \{+1, -1\}.$$

Here M and Σ are finite sets: M is interpreted as the set of states of the processor, while Σ is the set of symbols that can be written on the tape. The function T describes an 'iteration': $T(m, \sigma) = (m', \sigma', +1)$ should mean that if the processor is in state m and the symbol read on the tape is σ, then the next state will be m', the symbol σ is changed to the symbol σ' and the tape is moved one position to the right. $T(m, \sigma) = (m', \sigma', -1)$ has a similar meaning — now the tape is moved one position to the left.

Thus if the processor is in state m and has the word $w'\alpha'\sigma\alpha''w''$ on the tape, where the symbol indicated by σ is read, and if $T(m, \sigma) = (m', \sigma', +1)$, then next the processor will be in state m' and has the word $w'\alpha'\sigma'\alpha''w''$ on the tape, where the symbol indicated by α'' is read. Similarly if $T(m, \sigma) = (m', \sigma', -1)$.

We assume that M contains a certain 'start state' 0 and a certain 'halting state' ∞. Moreover, Σ is assumed to contain a symbol $\_$ meaning 'blank'. (This is necessary to identify the beginning and the end of a word on the tape.)

We say that the Turing machine T *accepts* a word $w \in (\Sigma \setminus \{\_\})^*$ if, when starting in state 0 and with word w on the tape (all other symbols being blank), so that the read-write head is reading the first symbol of w, then after a finite number of iterations, the processor is in the halting state ∞. (If w is the empty word, the symbol read initially is the blank symbol $\_$.)

Let Π be the set of words accepted by T. So Π is a problem. We say that T *solves* Π. Moreover, we say that T *solves* Π *in polynomial time* if there exists a polynomial $p(x)$ such that if T accepts a word w, it accepts w in at most $p(\text{size}(w))$ iterations.

It is not difficult to see that the concept of algorithm defined in Section 6.4 above is at least as powerful as that of a Turing machine. We can encode any state of the computer model (processor+tape+read-write head) by a word (w', m, w''). Here m is the state of the processor and $w'w''$ is the word on the tape, while the first symbol of w'' is read. We define an algorithm A by:

(18) replace subword $, m, \sigma$ by σ', m', whenever $T(m, \sigma) = (m', \sigma', +1)$ and $m \neq \infty$;
replace subword α, m, σ by $m', \alpha\sigma'$, whenever $T(m, \sigma) = (m', \sigma', -1)$ and $m \neq \infty$.

To be precise, we should assume here that the symbols indicating the states in M do not belong to Σ. Moreover, we assume that the symbols (and) are not in Σ. Furthermore, to give the algorithm a start, it contains the tasks of replacing subword $\_\alpha$ by the word $(,0,\alpha$, and subword $\alpha\_$ by $\alpha)$ (for any α in $\Sigma \setminus \{\_\}$). Then, when starting with a word w, the first two iterations transform it to the word $(,0,w)$. After that, the rules (18) simulate the Turing machine iterations. The iterations stop as soon as we arrive at state ∞.

So T accepts a word w if and only if A accepts w — in (about) the same number of iterations. That is, T solves a problem Π (in polynomial time), if and only if A solves Π (in polynomial time).

This shows that the concept of 'algorithm' defined in Section 6.4 is at least as powerful as that of a Turing machine. Conversely, it is not hard (although technically somewhat complicated) to simulate an algorithm by a Turing machine. But how powerful is a Turing machine?

One could think of several objections against a Turing machine. It uses only one tape, that should serve both as an input tape, and as a memory, and as an output tape. We have only limited access to the information on the tape (we can shift only one position at a time). Moreover, the computer program seems to be implemented in the 'hardware' of the computer model; the Turing machine solves only one problem.

To counter these objections, several other computer models have been proposed that model a computer more realistically: multi-tape Turing machines, random access machines (RAM's), the universal Turing machine. However, from a polynomial-time algorithmic point of view, these models all turn out to be equivalent. Any problem that can be solved in polynomial time by any of these computer models, can also be solved in polynomial time by some one-tape Turing machine, and hence by an algorithm in the sense of Section 6.4. We refer to Aho, Hopcroft, and Ullman [1974] and Papadimitriou [1994] for an extensive discussion.

7. Cliques, cocliques, and colourings

7.1. Introduction

We have seen in Chapter 5 that in any graph $G = (V, E)$, a matching of maximum cardinality can be found in polynomial time. Similarly, an edge-cover of minimum cardinality can be found in polynomial time.

On the other hand, it is NP-complete to find a maximum-cardinality coclique in a graph. That is, determining $\alpha(G)$ is NP-complete. To be more precise, the problem COCLIQUE is:

(1) given: a graph G and a natural number k,
decide: if $\alpha(G) \geq k$.

Then:

Theorem 7.1. *The problem* COCLIQUE *is* NP-*complete.*

Proof. We reduce SAT to COCLIQUE. Let $C_1 \wedge \cdots \wedge C_k$ be a boolean expression in the variables $x_1, \ldots, x_n$. Let $x_1, \neg x_1, \ldots, x_n, \neg x_n$ be the *literals*. Consider the graph $G = (V, E)$ with $V := \{(\sigma, i) \mid \sigma \text{ is a literal in } C_i\}$ and $E := \{\{(\sigma, i), (\tau, j)\} \mid i = j \text{ or } \sigma = \neg\tau\}$. Then the expression is satisfiable if and only if G has a coclique of size k. ∎

Since by Gallai's theorem Theorem 3.1, $\alpha(G) = |V| - \tau(G)$, also determining the vertex-cover number $\tau(G)$ is NP-complete.

A *clique* in a graph $G = (V, E)$ is a subset C of V such that u and w are adjacent for any two distinct u, w in C. The *clique number* of G, denoted by $\omega(G)$, is the maximum cardinality of any clique in G.

Observe that a subset C of V is a clique in G if and only if C is a coclique in the complementary graph $\overline{G}$. So finding a maximum-cardinality clique in G is equivalent to finding a maximum-cardinality coclique in $\overline{G}$, and $\omega(G) = \alpha(\overline{G})$. As determining $\alpha(G)$ is NP-complete, also determining $\omega(G)$ is NP-complete.

A *(vertex-)colouring* of a graph $G = (V, E)$ is a partition of V into cocliques $C_1, \ldots, C_k$. The sets $C_1, \ldots, C_k$ are called the *colours* of the colouring. The *(vertex-)colouring number*, or *(vertex-)chromatic number*, of G, denoted by $\gamma(G)$, is the minimum number of colours in any vertex-colouring of G. A graph G is called *k-colourable* if $\gamma(G) \leq k$.

Well-known is the *four-colour conjecture* (*4CC*), stating that $\gamma(G) \leq 4$ for each planar graph G. This conjecture was proved by Appel and Haken [1977] and Appel, Haken, and Koch [1977], and is now called the *four-colour theorem* (*4CT*).

Again, it is NP-complete to decide if a graph is k-colourable. In fact, it is NP-complete to decide if a planar graph is 3-colourable. [Note that one can decide in polynomial time if a graph G is 2-colourable, as bipartiteness can be checked in polynomial time.]

These NP-completeness results imply that if NP≠co-NP, then one may not expect a min-max relation characterizing the coclique number $\alpha(G)$, the vertex-cover number $\tau(G)$, the clique number $\omega(G)$, or the colouring number $\gamma(G)$ of a graph G.

There is a trivial upper bound on the colouring number:

(2) $$\gamma(G) \leq \Delta(G) + 1,$$

where $\Delta(G)$ denotes the maximum valency of G. Brooks [1941] sharpened this inequality as follows:

Theorem 7.2 (Brooks' theorem). *For any connected graph G one has $\gamma(G) \leq \Delta(G)$, except if $G = K_n$ or $G = C_{2n+1}$ for some $n \geq 1$.*[18]

Another inequality relates the clique number and the colouring number:

$$\omega(G) \leq \gamma(G). \tag{3}$$

This is easy, since in any clique all vertices should have different colours.

But there are several graphs which have strict inequality in (3). We mention the odd circuits C_{2k+1}, with $2k+1 \geq 5$: then $\omega(C_{2k+1}) = 2$ and $\gamma(C_{2k+1}) = 3$. Moreover, for the complement $\overline{C_{2k+1}}$ of any such graph we have: $\omega(\overline{C_{2k+1}}) = k$ and $\gamma(\overline{C_{2k+1}}) = k+1$.

It was a conjecture of Berge [1963] that these graphs are crucial, which was proved in 2002 by Chudnovsky, Robertson, Seymour, and Thomas: [19]

Strong perfect graph conjecture: Let G be a graph. If $\omega(G) < \gamma(G)$ then G contains C_n or $\overline{C_n}$, for some odd $n \geq 5$, as an *induced* subgraph.

Another conjecture is due to Hadwiger [1943]. Since there exist graphs with $\omega(G) < \gamma(G)$, it is not true that if $\gamma(G) \geq n$ then G contains the complete graph K_n on n vertices as a subgraph. However, Hadwiger conjectured the following, where a graph H is called a *minor* of a graph G if H arises from some subgraph of G by contracting some (possible none) edges.

Hadwiger's conjecture: If $\gamma(G) \geq n$ then G contains K_n as a minor.

In other words, for each n, the graph K_n is the only graph G with the property that G is not $(n-1)$-colourable and each proper minor of G is $(n-1)$-colourable.

Hadwiger's conjecture is trivial for $n = 1, 2, 3$, and was shown by Hadwiger for $n = 4$ (see Exercise 7.8). As planar graphs do not contain K_5 as a minor, Hadwiger's conjecture for $n = 5$ implies the four-colour theorem. In fact, Wagner [1937] showed that Hadwiger's conjecture for $n = 5$ is equivalent to the four-colour conjecture. Recently, Robertson, Seymour, and Thomas [1993] showed that Hadwiger's conjecture is true also for $n = 6$, by showing that in that case it is equivalent to the four-colour theorem. For $n \geq 7$ Hadwiger's conjecture is unsettled.

Application 7.1: Map colouring. A well-known application of colouring the vertices of a graph is that of colouring the countries in a map in such a way that adjacent countries obtain different colours. So the four-colour theorem implies that if each country is connected, then the map can be coloured using not more than four colours. (One should not consider countries as 'adjacent' if they have a common boundary of measure 0 only.)

There are several other cases where colouring a map amounts to finding a minimum vertex-colouring in a graph. For instance, consider a map of the Paris Métro network (Figure 7.1).

Suppose now that you want to print a coloured map of the network, indicating each of the 13 lines by a colour, in such a way that lines that cross each other or meet each other in a station, are indicated by different colours and in such a way that a minimum number of colours is used. This easily reduces to a graph colouring problem.

Application 7.2: Storage of goods, etc. Suppose you are the director of a circus and wish to transport your animals in a number of carriages, in such a way that no two of the animals put into one carriage eat each other, and in such a way that you use a minimum number of carriages.

This trivially reduces to a graph colouring problem. A similar problem is obtained if you have to store a number of chemicals in a minimum number of rooms of a storehouse, in such a way that no two of the chemicals stored in one room react upon each other in an unwanted way.

[18]Here C_k denotes the circuit with k vertices.

[19]Let $G = (V, E)$ be a graph and let $V' \subseteq V$. Then the subgraph of G *induced by* V', denoted by $G|V'$ is the graph (V', E'), where E' equals the set of all edges in E contained in V'. The graph $G|V'$ is called an *induced* subgraph of G.

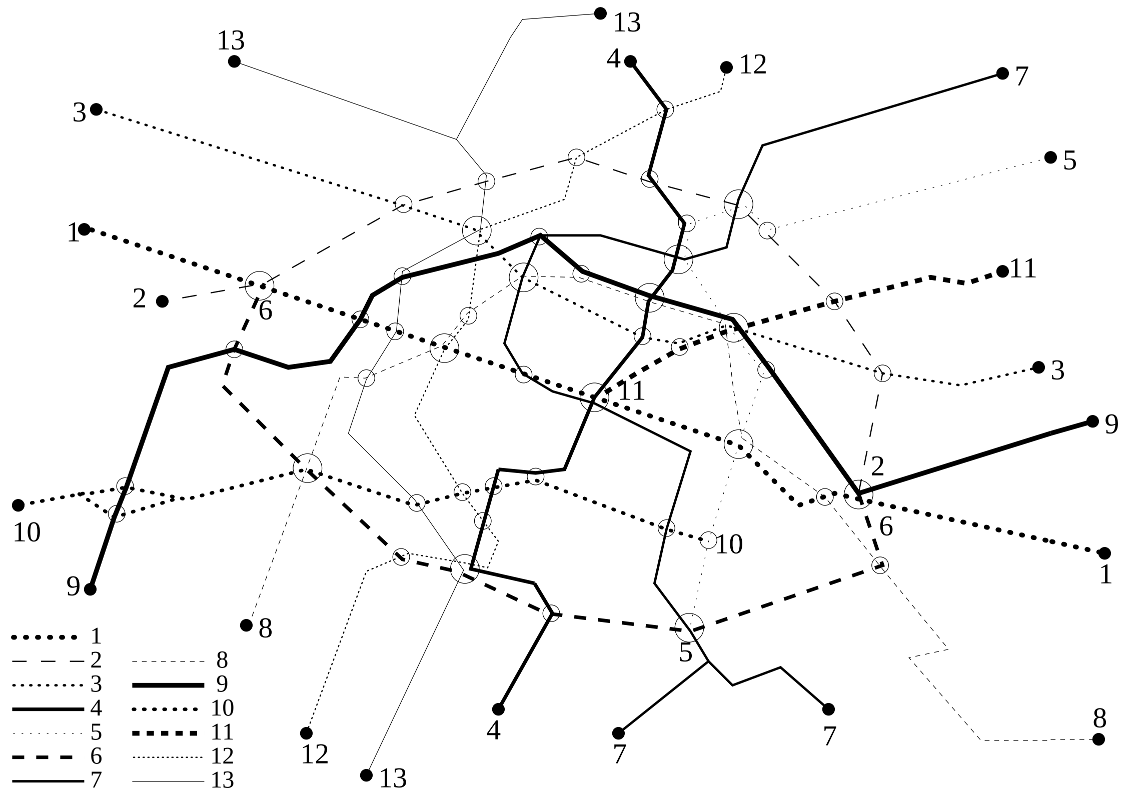


Figure 7.1

This problem may also occur when assigning multiple-bed rooms to school boys on a school trip.

Application 7.3: Assigning frequencies to radio stations, car phones, etc. Suppose one has to assign frequencies to radio stations in a certain area. Certain pairs of radio stations that are too close to each other cannot be assigned the same frequency as it would cause mutual interference. Such pairs of radio stations form the edge set of a graph G, with vertex set the set of radio stations. The chromatic number of G is equal to the minimum number of different frequencies that one needs in order to assign a frequency to each of the stations.

The problem occurs also when assigning frequencies to car phones, where often in a very short time new frequencies should be determined.

Exercises

7.1. Determine $\omega(G)$ and $\gamma(G)$ for the graph G obtained from the Paris Métro map given in Application 7.1.

7.2. Colour the map of Figure 7.2 (from the April 1975 issue of *Scientific American*).

7.3. Show that if G is a bipartite graph, then $\omega(G) = \gamma(G)$.

7.4. Derive from Kőnig's edge cover theorem (Corollary 3.2a) that if G is the complement of a bipartite graph, then $\omega(G) = \gamma(G)$.

7.5. Derive Kőnig's edge cover theorem (Corollary 3.2a) from the strong perfect graph theorem.

7.6. Let H be a bipartite graph and let G be the complement of the line-graph of H. Derive from Kőnig's matching theorem (Theorem 3.2) that $\omega(G) = \gamma(G)$.

7.7. Derive Kőnig's matching theorem (Theorem 3.2) from the strong perfect graph theorem.

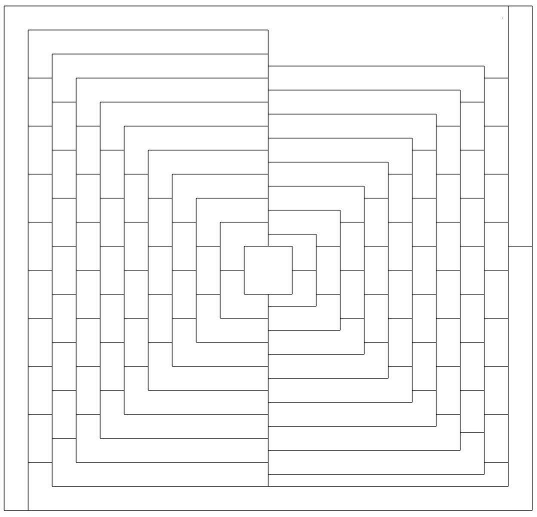


Figure 7.2

7.8. Let $G = (V, E)$ be a simple graph such that no minor of G is isomorphic to K_4. Show that $\gamma(G) \leq 3$. [*Hint:* One may assume that G is not a forest or a circuit. Then G has a circuit not covering all vertices of G. As G has no K_4-minor, G is not 3-connected, that is, G has a vertex cut set of size less than 3; then $\gamma(G) \leq 3$ follows by induction.]

7.2. Edge-colourings of bipartite graphs

For any graph $G = (V, E)$, an *edge-colouring* is a partition $\Pi = \{M_1, \ldots, M_p\}$ of the edge set E, where each M_i is a matching. Each of these matchings is called a *colour*. Define the *edge-colouring number* or *edge-chromatic number* $\chi(G)$ by

(4) $$\chi(G) := \min\{|\Pi| \mid \Pi \text{ is an edge-colouring of } G\}.$$

Let $\Delta(G)$ denote the maximum degree of (the vertices of) G. Clearly,

(5) $$\chi(G) \geq \Delta(G),$$

since at each vertex v, the edges incident with v should have different colours. Again the triangle K_3 has strict inequality. Kőnig [1916] showed that for bipartite graphs the two numbers are equal.

Theorem 7.3 (Kőnig's edge-colouring theorem)**.** *For any bipartite graph $G = (V, E)$ one has*

(6) $$\chi(G) = \Delta(G).$$

That is, the edge-colouring number of a bipartite graph is equal to its maximum degree.

Proof. First notice that the theorem is easy if $\Delta(G) \leq 2$. In that case, G consists of a number of vertex-disjoint paths and even circuits.

In the general case, colour as many edges of G as possible with $\Delta(G)$ colours, without giving the same colour to two intersecting edges. If all edges are coloured we are done, so suppose some edge $e = \{u, w\}$ is not coloured. At least one colour, say *red*, does not occur among the colours given to the edges incident with u. Similarly, there is a colour, say *blue*, not occurring at w. (Clearly, *red*$\neq$*blue*, since otherwise we could give edge e the colour *red*.)

Let H be the subgraph of G having as edges all *red* and *blue* edges of G, together with the edge e. Now $\Delta(H) = 2$, and hence $\chi(H) = \Delta(H) = 2$. So all edges occurring in H can be (re)coloured with *red* and *blue*. In this way we colour more edges of G than before. This contradicts the maximality assumption. ▮

This proof also gives a polynomial-time algorithm to find an edge-colouring with $\Delta(G)$ colours.

We remark here that Vizing [1964] proved that for general simple graphs G one has

(7) $$\Delta(G) \leq \chi(G) \leq \Delta(G) + 1.$$

Here 'simple' cannot be deleted, as is shown by the graph G with three vertices, where any two vertices are connected by two parallel edges: then $\Delta(G) = 4$ while $\chi(G) = 6$.

A theorem 'dual' to Kőnig's edge-colouring theorem was also shown by Kőnig. Note that the edge-colouring number $\chi(G)$ of a graph G is the minimum number of matchings needed to cover the edges of a bipartite graph. Dually, one can define:

(8) $\xi(G) :=$ the maximum number of pairwise disjoint edge covers in G.

So, in terms of colours, $\xi(G)$ is the maximum number of colours that can be used in colouring the edges of G in such a way that at each vertex all colours occur. Hence, if $\delta(G)$ denotes the minimum degree of G, then

(9) $$\xi(G) \leq \delta(G).$$

The triangle K_3 again is an example having strict inequality. For bipartite graphs however:

Corollary 7.3a. *For any bipartite graph $G = (V, E)$ one has*

(10) $$\xi(G) = \delta(G).$$

That is, the maximum number of pairwise disjoint edge covers is equal to the minimum degree.

Proof. One may derive from G a bipartite graph H, each vertex of which has degree $\delta(G)$ or 1, by repeated application of the following procedure:

(11) for any vertex v of degree larger than $\delta(G)$, add a new vertex u, and replace one of the edges incident with v, $\{v, w\}$ say, by $\{u, w\}$.

So there is a one-to-one correspondence between the edges of the final graph H and the edges of G. Since H has maximum degree $\delta(G)$, by Theorem 7.3 the edges of H can be coloured with $\delta(G)$ colours such that no two edges of the same colour intersect. So at any vertex of H of degree $\delta(G)$ all colours occur. This gives a colouring of the edges of G with $\delta(G)$ colours such that at any vertex of G all colours occur. ▮

Application 7.4: Scheduling classes. Suppose we have n classes and m teachers. In the following scheme it is indicated by an X which classes should be taught by which teachers (one lesson of one hour a day):

class:	1	2	3	4	5	6
teacher: *a*	X	X		X		
b	X		X	X	X	
c		X		X	X	
d		X				X
e	X		X		X	X
f	X		X	X		X
g		X	X		X	X

The question is: What is the minimum timespan in which all lessons can be scheduled?

Theorem 7.3 tells us that all lessons can be scheduled within a timespan of 4 hours. Indeed, make a bipartite graph G with colour classes $T :=$ set of teachers and $C :=$ set of classes, where $t \in T$ and $c \in C$ are connected if and only if teacher t should teach class c; that is, if there is an X in position (t, c) in the scheme.

In the above example G will have maximum degree $\Delta(G)$ equal to 4. Hence according to Theorem 7.3, the edge-colouring number $\chi(G)$ of G is also equal to 4. So we can colour the edges of G by 4 colours so that no two edges of the same colour have a vertex in common. That is, we can colour the X's in the scheme by 4 colours so that there are no two crosses of the same colour in any row or column. If every colour represent one hour, we obtain a schedule spanning 4 hours.

This application can be extended to the case where teachers can give more than one lesson a day to a class. In that case we obtain a bipartite graph with multiple edges.

For any k-edge-colouring of a graph $G = (V, E)$, we can assume that any two colours differ by at most 1 in size (if they differ more, one can exchange the two colours on one of the path components of the union of the two colours, to bring their cardinalities closer together). That is, each colour has size $\lfloor |E|/k \rfloor$ or $\lceil |E|/k \rceil$. It implies that there is a schedule in which no more than $\lceil |E|/k \rceil$ lessons are scheduled simultaneously. So the number of classrooms needed is $\lceil |E|/k \rceil$, which is clearly best possible if we want to schedule $|E|$ lessons within k hours.

Exercises

7.9. Determine a schedule for the following scheduling problems:

(i)

X	X	X	X	
X	X		X	X
X	X	X		X
	X	X	X	X
X		X	X	X

(ii)

X		X	X	X		
X	X	X			X	
			X	X	X	X
X	X				X	X
X		X	X	X		
	X		X		X	X
	X	X		X		X

(iii)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A	■	□	□	■	□	■	□	■	□	■	■	□	□	■	□	□	■	■
B	□	■	□	■	■	□	□	■	■	□	□	□	■	■	■	■	■	■
C	■	□	■	□	■	■	■	□	■	■	■	■	□	□	■	□	■	□
D	□	■	□	■	■	□	□	■	□	■	□	□	■	■	□	■	■	■
E	■	□	■	■	□	■	■	□	■	■	□	■	■	□	■	■	□	■
F	■	□	■	□	■	□	■	□	■	□	■	□	■	□	■	■	□	□
G	□	□	■	■	■	■	□	■	□	■	■	■	■	■	□	■	■	■
H	■	■	■	□	■	■	■	■	■	□	■	■	■	■	■	□	□	■
I	■	■	■	■	□	■	■	■	■	■	■	■	□	■	■	■	■	■
J	■	■	■	□	■	■	■	□	■	■	■	■	■	□	■	□	■	■
K	■	■	□	■	■	□	■	■	■	■	□	■	■	■	□	■	■	□
L	□	■	■	■	■	■	■	■	■	□	■	□	□	■	■	■	■	□
M	■	■	■	■	□	□	■	□	□	■	■	■	■	■	■	■	□	■
N	□	■	□	□	■	■	□	■	■	■	■	□	■	□	□	■	□	□
O	■	■	■	■	□	■	■	■	■	■	□	■	□	■	■	□	■	■
P	■	□	□	■	■	□	■	□	■	□	□	■	■	□	□	■	■	□
Q	□	■	■	□	□	■	□	■	■	■	■	■	□	■	■	■	■	■
R	■	■	■	■	□	■	■	■	□	□	■	■	■	■	■	□	■	■
S	■	■	□	■	■	□	■	■	■	■	□	■	■	□	■	■	■	■
T	■	□	■	■	■	■	■	■	■	■	■	■	□	■	□	■	□	■
U	□	■	■	■	■	■	■	□	□	■	■	□	■	■	■	■	■	□
V	■	□	■	□	□	■	□	■	■	□	■	■	■	■	■	□	■	■
W	□	■	■	■	■	□	■	□	■	■	□	■	□	■	□	■	□	■
X	■	□	■	□	■	■	■	■	□	□	■	□	□	■	□	□	■	□
Y	■	■	□	■	■	□	■	□	■	■	□	■	■	□	■	■	□	■
Z	□	■	■	■	□	■	□	■	■	■	■	■	■	■	■	□	■	□

(Here the slots to be scheduled are indicated by open cells.)

7.10. Let G be the line-graph of some bipartite graph H. Derive from Kőnig's edge-colouring theorem (Theorem 7.3) that $\omega(G) = \gamma(G)$.

7.11. Derive Kőnig's edge-colouring theorem (Theorem 7.3) from the strong perfect graph theorem.

7.12. Let $\mathcal{A} = (A_1, \ldots, A_n)$ and $\mathcal{B} = (B_1, \ldots, B_n)$ be partitions of a finite set X such that $|A_1| = \cdots = |A_n| = |B_1| = \cdots = |B_n| = k$. Show that $\mathcal{A}$ and $\mathcal{B}$ have k pairwise disjoint common transversals.

7.13. Let $\mathcal{A} = (A_1, \ldots, A_n)$ and $\mathcal{B} = (B_1, \ldots, B_n)$ be families of subsets of a finite set X.

(i) Let $k \in \mathbb{N}$. Suppose that X can be partitioned into k partial SDR's of $\mathcal{A}$, and that X also can be partitioned into k partial SDR's of $\mathcal{B}$. Derive that X can be partitioned into k *common* partial

SDR's for $\mathcal{A}$ and $\mathcal{B}$.

(ii) Show that the minimum number of common partial SDR's of $\mathcal{A}$ and $\mathcal{B}$ needed to cover X is equal to

$$\lceil \max_{Y\subseteq X} \max\{\frac{|Y|}{|\{i|A_i\cap Y\neq\emptyset\}|}, \frac{|Y|}{|\{i|B_i\cap Y\neq\emptyset\}|}\}\rceil. \tag{12}$$

(*Hint:* Use Exercise 3.8.)

7.14. Let $\mathcal{A}=(A_1,\ldots,A_n)$ and $\mathcal{B}=(B_1,\ldots,B_n)$ be families of subsets of a finite set X and let $k\in\mathbb{N}$. Suppose that X has a partition $(Y_1,\ldots,Y_k)$ such that each Y_i is an SDR of $\mathcal{A}$. Suppose moreover that X has a partition $(Z_1,\ldots,Z_k)$ such that each Z_i is an SDR of $\mathcal{B}$. Derive that X has a partition $(X_1,\ldots,X_k)$ such that each X_i is an SDR both of $\mathcal{A}$ and of $\mathcal{B}$.

7.15. Let $\mathcal{A}=(A_1,\ldots,A_n)$ and $\mathcal{B}=(B_1,\ldots,B_n)$ be families of subsets of a finite set X and let $k\in\mathbb{N}$. Suppose that X has a partition $(Y_1,\ldots,Y_n)$ such that $|Y_i|=k$ and $Y_i\subseteq A_i$ for $i=1,\ldots,n$. Suppose moreover that X has a partition $(Z_1,\ldots,Z_n)$ such that $|Z_i|=k$ and $Z_i\subseteq B_i$ for $i=1,\ldots,n$. Derive that X has a partition $(X_1,\ldots,X_k)$ such that each X_i is an SDR both of $\mathcal{A}$ and of $\mathcal{B}$.

7.16. Let $\mathcal{A}=(A_1,\ldots,A_n)$ and $\mathcal{B}=(B_1,\ldots,B_m)$ be families of subsets of a finite set and let k be a natural number. Prove that $\mathcal{A}$ and $\mathcal{B}$ have k pairwise disjoint common SDR's, if and only if for all $I,J\subseteq\{1,\ldots,n\}$:

$$|\bigcup_{i\in I}A_i\cap\bigcup_{j\in J}B_j|\geq k(|I|+|J|-n). \tag{13}$$

(*Hint:* Use Exercise 7.15.)

7.17. Let $\mathcal{A}=(A_1,\ldots,A_n)$ and $\mathcal{B}=(B_1,\ldots,B_n)$ be families of subsets of a finite set X.

(i) Let $k\in\mathbb{N}$. Suppose that $\mathcal{A}$ has k pairwise disjoint SDR's and that also $\mathcal{B}$ has k pairwise disjoint SDR's. Derive that X can be partitioned into k subsets $X_1,\ldots,X_k$ such that each X_i contains an SDR of $\mathcal{A}$ and contains an SDR of $\mathcal{B}$.

(ii) Show that the maximum number k for which there exists a partition as in (i) is equal to

$$\lfloor \min_{\emptyset\neq I\subseteq\{1,\ldots,n\}} \min\{\frac{|\bigcup_{i\in I}A_i|}{|I|}, \frac{|\bigcup_{i\in I}B_i|}{|I|}\}\rfloor. \tag{14}$$

(*Hint:* Use Exercise 3.7.)

7.3. Partially ordered sets

A *partially ordered set* is a pair $(X,\leq)$ where X is a set and where $\leq$ is a relation on X satisfying:

(15)
(i) $x\leq x$ for each $x\in X$;
(ii) if $x\leq y$ and $y\leq x$ then $x=y$;
(iii) if $x\leq y$ and $y\leq z$ then $x\leq z$.

A subset C of X is called a *chain* if for all $x,y\in C$ one has $x\leq y$ or $y\leq x$. A subset A of X is called an *antichain* if for all $x,y\in A$ with $x\neq y$ one has $x\not\leq y$ and $y\not\leq x$. Note that if C is a chain and A is an antichain then

$$|C\cap A|\leq 1. \tag{16}$$

First we observe the following easy min-max relation:

Theorem 7.4. *Let $(X,\leq)$ be a partially ordered set, with X finite. Then the minimum number of antichains needed to cover X is equal to the maximum cardinality of any chain.*

Proof. The fact that the maximum cannot be larger then the minimum follows easily from (16). To see that the two numbers are equal, define for any element $x \in X$ the *height* of x as the maximum cardinality of any chain in X with maximum x. For any $i \in \mathbb{N}$, let A_i denote the set of all elements of height i.

Let k be the maximum height of the elements of X. Then $A_1, \ldots, A_k$ are antichains covering X, and moreover there exists a chain of size k. ∎

Dilworth [1950] proved that the same theorem also holds when we interchange the words 'chain' and 'antichain':

Theorem 7.5 (Dilworth's decomposition theorem)**.** *Let $(X, \leq)$ be a partially ordered set, with X finite. Then the minimum number of chains needed to cover X is equal to the maximum cardinality of any antichain.*

Proof. We apply induction on $|X|$. The fact that the maximum cannot be larger then the minimum follows easily from (16). To see that the two numbers are equal, let α be the maximum cardinality of any antichain and let A be an antichain of cardinality α. Define

(17) $$\begin{aligned} A^{\downarrow} &:= \{x \in X \mid \exists y \in A : x \leq y\}, \\ A^{\uparrow} &:= \{x \in X \mid \exists y \in A : x \geq y\}. \end{aligned}$$

Then $A^{\downarrow} \cup A^{\uparrow} = X$ (since A is a maximum antichain) and $A^{\downarrow} \cap A^{\uparrow} = A$.

First assume $A^{\downarrow} \neq X$ and $A^{\uparrow} \neq X$. Then by induction $A^{\downarrow}$ can be covered with α chains. Since $A \subseteq A^{\downarrow}$, each of these chains contains exactly one element in A. For each $x \in A$, let C_x denote the chain containing x. Similarly, there exist α chains C'_x (for $x \in A$) covering $A^{\uparrow}$, where C'_x contains x. Then for each $x \in A$, $C_x \cup C'_x$ forms a chain in X, and moreover these chains cover X.

So we may assume that for each antichain A of cardinality α one has $A^{\downarrow} = X$ or $A^{\uparrow} = X$. It means that each antichain A of cardinality α is either the set of minimal elements of X or the set of maximal elements of X. Now choose a minimal element x and a maximal element y of X such that $x \leq y$. Then the maximum cardinality of an antichain in $X \setminus \{x, y\}$ is equal to $\alpha - 1$ (since each antichain in X of cardinality α contains x or y). By induction, $X \setminus \{x, y\}$ can be covered with $\alpha - 1$ chains. Adding the chain $\{x, y\}$ yields a covering of X with α chains. ∎

Application 7.5: Project scheduling. Suppose you have to perform a project consisting of several jobs. Each job takes one time-unit, say one hour. Certain jobs have to be done before other jobs; this relation is given by a partial order on the jobs. Assuming that you have sufficient workers, the time required to finish the project is equal to the size γ of the longest chain. Indeed, by Theorem 7.4, the jobs can be split into γ antichains $A_1, \ldots, A_\gamma$; in fact, these antichains can be chosen such that if $x \in A_i$ and $y \in A_j$ and $x < y$ then $i < j$. As in each of these antichains, the jobs can be done simultaneously, we obtain a feasible schedule.

This is an application quite similar to PERT-CPM (Application 1.4).

Application 7.6: Bungalow assignment. Suppose you are the manager of a bungalow park, with bungalows that can be rented out during the holiday season. There have been made a number of reservations, each for a connected period of some weeks, like in Figure 7.3. If the number of reservations during any of the weeks in the holiday season is not larger than the total number of bungalows available, then there exists an allocation of customers to bungalows, in such a way that no renter has to switch bungalows during his/her stay. This rule well-known to bungalow park managers, is a special case of Dilworth's decomposition theorem.

Indeed, one can make a partial order as follows. Let X be the set of reservations made, and for any $x, y \in X$ let $x < y$ if the last day for reservation x is earlier than or equal to the first day of reservation y.

Then the maximum size of any antichain of $(X, \leq)$ is equal to the maximum number n of reservations made for any week in the season. By Dilworth's decomposition theorem, X can be split into n chains. Each chain now gives a series of reservations that can be assigned to one and the same bungalow.

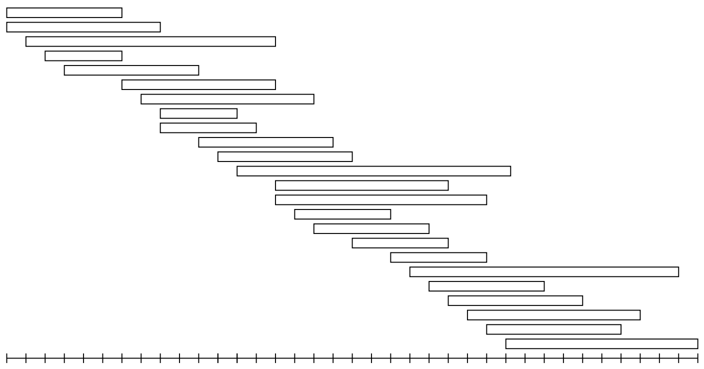


Figure 7.3

A similar problem occurs when assigning hotel rooms to hotel guests.

Application 7.7: Terminal and platform assignment. A similar problem as in Application 7.6 occurs when one has to assign airplanes to terminals at an airport, or trains or buses to platforms in a train or bus station. The model has to be adapted however, if one requires a *periodic* assignment; this occurs for instance if the trains or buses run a periodic timetable, say with period one hour.

Exercises

7.18. Let $(X, \leq)$ be a partially ordered set. Call a chain *maximal* if it is not contained in any other chain. Prove that the maximum number of pairwise disjoint maximal chains is equal to the minimum cardinality of a set intersecting all maximal chains.

7.19. Derive Kőnig's edge cover theorem from Dilworth's decomposition theorem.

7.20. Let $G = (V, E)$ be a bipartite graph, with colour classes V_1 and V_2, with $|V_1| = |V_2| = n$. Let k be a natural number. Derive from Dilworth's decomposition theorem that the edges of G can be covered by k perfect matchings, if and only if for each vertex cover $W \subseteq V$ the number of edges contained in W is at most $k(|W| - n)$.

7.21. Let $\mathcal{I} = (I_1, \ldots, I_n)$ be a family of intervals on $\mathbb{R}$, in such a way that each $x \in \mathbb{R}$ is contained in at most k of these intervals. Show that $\mathcal{I}$ can be partitioned into k classes $\mathcal{I}_1, \ldots, \mathcal{I}_k$ so that each $\mathcal{I}_j$ consists of pairwise disjoint intervals.

7.22. Let $D = (V, A)$ be an acyclic directed graph and let r and s be vertices of D such that each arc of D occurs in at least one $r - s$ path. Derive from Dilworth's decomposition theorem that the minimum number of $r - s$ paths needed to cover all arcs is equal to the maximum cardinality of $\delta^{\text{out}}(U)$, where U ranges over all subsets of V satisfying $r \in U, s \notin U$ and $\delta^{\text{in}}(U) = \emptyset$.

7.23. A graph $G = (V, E)$ is called a *comparability graph* if there exists a partial order $\leq$ on V such that for all u, w in V with $u \neq w$ one has:

$$(18) \qquad \{u, w\} \in E \Leftrightarrow u \leq w \text{ or } w \leq u.$$

(i) Show that if G is a comparability graph, then $\omega(G) = \gamma(G)$.

(ii) Show that if G is the complement of a comparability graph, then $\omega(G) = \gamma(G)$.
(*Hint:* Use Dilworth's decomposition theorem (Theorem 7.5).)

7.24. Let $(X, \leq)$ be a partially ordered set, with X finite. Let $\mathcal{C}$ and $\mathcal{A}$ denote the collections of chains and antichains in $(X, \leq)$, respectively. Let $w : X \to \mathbb{Z}_+$ be a 'weight' function.

(i) Show that the maximum weight $w(C)$ of any chain is equal to the minimum value of $\sum_{A\in\mathcal{A}}\lambda(A)$, where the $\lambda(A)$ range over all nonnegative integers satisfying

(19) $$\sum_{A\in\mathcal{A},x\in A}\lambda(A)=w(x)$$

for each $x \in X$.

(ii) Show that the maximum weight $w(A)$ of any antichain is equal to the minimum value of $\sum_{C\in\mathcal{C}}\lambda(C)$, where the $\lambda(C)$ range over all nonnegative integers satisfying

(20) $$\sum_{C\in\mathcal{C},x\in C}\lambda(C)=w(x)$$

for each $x \in X$.

(iii) Derive that the convex hull of the incidence vectors of antichains (as vectors in $\mathbb{R}^X$) is equal to the set of all vectors $f \in \mathbb{R}^X_+$ satisfying $f(C) \leq 1$ for each chain C.

[For any finite set X and any subset Y of X, define the *incidence vector* $\chi^Y \in \mathbb{R}^X$ of Y as:

(21) $$\begin{aligned}\chi^Y_x &:= 1 \quad \text{if } x \in Y;\\ &:= 0 \quad \text{if } x \notin Y.]\end{aligned}$$

(iv) Derive also that the convex hull of the incidence vectors of chains (as vectors in $\mathbb{R}^X$) is equal to the set of all vectors $f \in \mathbb{R}^X_+$ satisfying $f(A) \leq 1$ for each antichain A.

7.25. Derive Dilworth's decomposition theorem (Theorem 7.5) from the strong perfect graph theorem.

7.4. Perfect graphs

We now consider a general class of graphs, the 'perfect' graphs, that turn out to unify several results in combinatorial optimization, in particular, min-max relations and polyhedral characterizations.

As we saw before, the clique number $\omega(G)$ and the colouring number $\gamma(G)$ of a graph $G = (V, E)$ are related by the inequality:

(22) $$\omega(G) \leq \gamma(G).$$

There are graphs that have strict inequality; for instance, the circuit C_5 on five vertices.

Having equality in (22) does not say that much about the internal structure of a graph: any graph $G = (V, E)$ can be extended to a graph $G' = (V', E')$ satisfying $\omega(G') = \gamma(G')$, simply by adding to G a clique of size $\gamma(G)$, disjoint from V.

However, if we require that equality in (22) holds for each induced subgraph of G, we obtain a much more powerful condition. The idea for this was formulated by Berge [1963]. He defined a graph $G = (V, E)$ te be *perfect* if $\omega(G') = \gamma(G')$ holds for each induced subgraph G' of G.

Several classes of graphs could be shown to be perfect, and Berge [1961,1963] observed the important phenomenon that for several classes of graphs that were shown to be perfect, also the class of complementary graphs is perfect. (The *complement* or the *complementary graph* $\overline{G}$ of a graph $G = (V, E)$ is the graph with vertex set V, where any two distinct vertices in V are adjacent in $\overline{G}$ if and only if they are nonadjacent in G.)

Berge therefore conjectured that the complement of any perfect graph is perfect again. This conjecture was proved by Lovász [1972b], and his *perfect graph theorem* forms the kernel of perfect graph theory. It has several other theorems in graph theory as consequence. Lovász [1972a] gave the following stronger form of the conjecture, which we show with the elegant linear-algebraic proof found by Gasparian [1996].

Theorem 7.6. *A graph G is perfect if and only if $\omega(G')\alpha(G') \geq |V(G')|$ for each induced subgraph G' of G.*

Proof. Necessity is easy, since if G is perfect, then $\omega(G') = \gamma(G')$ for each induced subgraph G' of G, and since $\gamma(G')\alpha(G') \geq |V(G')|$ for any graph G'.

To see sufficiency, suppose to the contrary that there exists an imperfect graph G satisfying the condition, and choose such a graph with $|V(G)|$ minimal. So $\gamma(G) > \omega(G)$, while $\gamma(G') = \omega(G')$ for each induced subgraph $G' \neq G$ of G.

Let $\omega := \omega(G)$ and $\alpha := \alpha(G)$. We can assume that $V(G) = \{1, \ldots, n\}$.

We first construct

(23) cocliques $C_0, \ldots, C_{\alpha\omega}$ such that each vertex is covered by exactly α of the C_i.

Let C_0 be a coclique in G of size α. By the minimality of G, we know that for each $v \in C_0$, the subgraph of G induced by $V(G) \setminus \{v\}$ is perfect, and that hence its colouring number is at most ω (as its clique number is at most ω); therefore $V(G) \setminus \{v\}$ can be partitioned into ω cocliques. Doing this for each $v \in C_0$, we obtain cocliques as in (23).

Now for each $i = 0, \ldots, \alpha\omega$, there exists a clique K_i of size ω with $K_i \cap C_i = \emptyset$. Otherwise, the subgraph G' of G induced by $V(G) \setminus C_i$ would have $\omega(G') < \omega$, and hence it has colouring number at most $\omega - 1$. Adding C_i as a colour would give an ω-vertex colouring of G, contradicting the assumption that $\gamma(G) > \omega(G)$.

Then, if $i \neq j$ with $0 \leq i, j \leq \alpha\omega$, we have $|K_j \cap C_i| = 1$. This follows from the fact that K_j has size ω and intersects each C_i in at most one vertex, and hence, by (23), it intersects $\alpha\omega$ of the C_i. As $K_j \cap C_j = \emptyset$, we have that $|K_j \cap C_i| = 1$ if $i \neq j$.

Now consider the $(\alpha\omega + 1) \times n$ incidence matrices $M = (m_{i,j})$ and $N = (n_{i,j})$ of $C_0, \ldots, C_{\alpha\omega}$ and $K_0, \ldots, K_{\alpha\omega}$ respectively. So M and N are $0,1$ matrices, with $m_{i,j} = 1 \Leftrightarrow j \in C_i$, and $n_{i,j} = 1 \Leftrightarrow j \in K_i$, for $i = 0, \ldots, \alpha\omega$ and $j = 1, \ldots, n$. By the above, $MN^T = J - I$, where J is the $\alpha\omega \times \alpha\omega$ all-1 matrix, and I the $\alpha\omega \times \alpha\omega$ identity matrix. As $J - I$ has rank $\alpha\omega + 1$, we have $n \geq \alpha\omega + 1$. This contradicts the condition given in the theorem. ∎

This implies:

Corollary 7.6a ((Lovász's) perfect graph theorem)**.** *The complement of a perfect graph is perfect again.*

Proof. Directly from Theorem 7.6, as the condition given in it is maintained under taking the complementary graph. ∎

In fact, Berge [1963] also made an even stronger conjecture, which was proved in 2002 by Chudnovsky, Robertson, Seymour, and Thomas (we mentioned this in Section 7.1 in a different but equivalent form):

Strong perfect graph theorem. A graph G is perfect if and only if G does not contain any odd circuit C_{2k+1} with $k \geq 2$ or its complement as an induced subgraph.

We now show how several theorems we have seen before follow as consequences from the perfect graph theorem. First observe that trivially, any bipartite graph G is perfect. This implies Kőnig's edge cover theorem (Theorem 3.2a):

Corollary 7.6b (Kőnig's edge cover theorem)**.** *The complement of a bipartite graph is perfect. Equivalently, the edge cover number of any bipartite graph (without isolated vertices) is equal to its coclique number.*

Proof. Directly from the perfect graph theorem. Note that if G is a bipartite graph, then its cliques have size at most 2; hence $\gamma(\overline{G})$ is equal to the edge cover number of G if G has no isolated vertices.

Note moreover that the class of complements of bipartite graphs is closed under taking induced subgraphs. Hence the second statement in the Corollary indeed is equivalent to the first. ∎

We saw in Section 3.2 that by Gallai's theorem (Theorem 3.1), Kőnig's edge cover theorem directly implies Kőnig's matching theorem (Theorem 3.2), saying that the matching number of a bipartite graph G is equal to its vertex cover number. That is, the coclique number of the line graph $L(G)$ of G is equal to the minimum number of cliques of $L(G)$ that cover all vertices of $L(G)$. As this is true for any induced subgraph of $L(G)$ we know that the complement $\overline{L(G)}$ of the line graph $L(G)$ of any bipartite graph G is perfect.

Hence with the perfect graph theorem we obtain Kőnig's edge-colouring theorem (Theorem 7.3):

Corollary 7.6c (Kőnig's edge-colouring theorem)**.** *The line graph of a bipartite graph is perfect. Equivalently, the edge-colouring number of any bipartite graph is equal to its maximum degree.*

Proof. Again directly from Kőnig's matching theorem and the perfect graph theorem. ∎

We can also derive Dilworth's decomposition theorem (Theorem 7.5) easily from the perfect graph theorem. Let $(V, \leq)$ be a partially ordered set. Let $G = (V, E)$ be the graph with:

(24) $uv \in E$ if and only if $u < v$ or $v < u$.

Any graph G obtained in this way is called a *comparability graph.*

As Theorem 7.4 we saw the following easy 'dual' form of Dilworth's decomposition theorem:

Theorem 7.7. *In any partially ordered set $(V, \leq)$, the maximum size of any chain is equal to the minimum number of antichains needed to cover V.*

Proof. For any $v \in V$ define the *height* of v as the maximum size of any chain in V with maximum element v. Let k be the maximum height of any element $v \in V$. For $i = 1, \ldots, k$ let A_i be the set of elements of height i. Then $A_1, \ldots, A_k$ are antichains covering V, and moreover, there is a chain of size k, since there is an element of height k. ∎

Equivalently, we have $\omega(G) = \gamma(G)$ for any comparability graph. As the class of comparability graphs is closed under taking induced subgraphs we have:

Corollary 7.7a. *Any comparability graph is perfect.*

Proof. Directly from Theorem 7.7. ∎

So by the perfect graph theorem:

Corollary 7.7b. *The complement of any comparability graph is perfect.*

Proof. Directly from Corollary 7.7a and the perfect graph theorem (Corollary 7.6a). ∎

That is:

Corollary 7.7c (Dilworth's decomposition theorem)**.** *In any partially ordered set $(V, \leq)$, the maximum size of any antichain is equal to the minimum number of chains needed to cover V.*

Proof. Directly from Corollary 7.7b. ∎

A further application of the perfect graph theorem is to 'chordal graphs', which we describe in the next section.

We note here that it was shown with the help of the 'ellipsoid method' that there exists a polynomial-time algorithm for finding a maximum clique and a minimum vertex-colouring in any perfect graph (Grötschel, Lovász, and Schrijver [1981]). However no *combinatorial* polynomial-time algorithm is known for these problems.

Exercises

7.26. Show that the graph obtained from the Paris Métro network (see Application 7.1) is perfect.

7.27. Show that Theorem 7.6 is implied by the strong perfect graph theorem.

7.5. Chordal graphs

We finally consider a further class of perfect graphs, the 'chordal graphs' (or 'rigid circuit graphs' or 'triangulated graphs'). A graph G is called *chordal* if each circuit in G of length at least 4 has a chord. (A *chord* is an edge connecting two vertices of the circuit that do not form two neighbours in the circuit.)

For any set A of vertices let $N(A)$ denote the set of vertices not in A that are adjacent to at least one vertex in A. Call a vertex v *simplicial* if $N(\{v\})$ is a clique in G.

Dirac [1961] showed the following basic property of chordal graphs:

Theorem 7.8. *Each chordal graph G contains a simplicial vertex.*

Proof. We may assume that G has at least two nonadjacent vertices a, b. Let A be a maximal nonempty subset of V such that $G|A$ is connected and such that $A \cup N(A) \neq V$. Such a subset A exists as $G|\{a\}$ is connected and $\{a\} \cup N(\{a\}) \neq V$.

Let $B := V \setminus (A \cup N(A))$. Then each vertex v in $N(A)$ is adjacent to each vertex in B, since otherwise we could increase A by v. Moreover, $N(A)$ is a clique, for suppose that $u, w \in N(A)$ are nonadjacent. Choose $v \in B$. Let P be a shortest path in $A \cup N(A)$ connecting u and w. Then $P \cup \{u, v, w\}$ would form a circuit of length at least 4 without chords, a contradiction.

Now inductively we know that $G|B$ contains a vertex v that is simplicial in $G|B$. Since $N(A)$ is a clique and since each vertex in B is connected to each vertex in $N(A)$, v is also simplicial in G. ▮

This implies a result of Hajnal and Surányi [1958]:

Theorem 7.9. *The complement of any chordal graph is perfect.*

Proof. Let $G = (V, E)$ be a chordal graph. Since the class of chordal graphs is closed under taking induced subgraphs, it suffices to show $\omega(\overline{G}) \geq \gamma(\overline{G})$.

By Theorem 7.1, G has a simplicial vertex v. So $K := \{v\} \cup N(\{v\})$ is a clique. Let G' be the subgraph of G induced by $V \setminus K$. By induction we have $\omega(\overline{G'}) = \gamma(\overline{G'})$.

Now $\omega(\overline{G}) \geq \omega(\overline{G'}) + 1$, since we can add v to any clique of $\overline{G'}$. Similarly, $\gamma(\overline{G}) \leq \gamma(\overline{G'}) + 1$, since we can add K to any colouring of $\overline{G'}$. Hence $\omega(\overline{G}) \geq \gamma(\overline{G})$. ▮

With Lovász's perfect graph theorem, this implies the result of Berge [1960]:

Corollary 7.9a. *Any chordal graph is perfect.*

Proof. Directly from Theorem 7.9 and the perfect graph theorem (Corollary 7.6a). ▮

We can characterize chordal graphs in terms of subtrees of a tree T. Let $\mathcal{S}$ be a collection of nonempty subtrees of a tree T. The *intersection graph* of $\mathcal{S}$ is the graph with vertex set $\mathcal{S}$, where two vertices S, S' are adjacent if and only if they intersect (in at least one vertex).

The class of graphs obtained in this way coincides with the class of chordal graphs. To see this, we first show the following elementary lemma:

Lemma 7.1. *Let $\mathcal{S}$ be a collection of pairwise intersecting subtrees of a tree T. Then there is a vertex of T contained in all subtrees in $\mathcal{S}$.*

Proof. By induction on $|VT|$. If $|VT| = 1$ the lemma is trivial, so assume $|VT| \geq 2$. Let t be an end vertex of T. If there exists a subtree in $\mathcal{S}$ consisting only of t, the lemma is trivial. Hence we may assume that each subtree in $\mathcal{S}$ containing t also contains the neighbour of t. So deleting t from T and from all subtrees in $\mathcal{S}$ gives the lemma by induction. ▮

Then:

Theorem 7.10. *A graph is chordal if and only if it is isomorphic to the intersection graph of a collection of subtrees of some tree.*

Proof. *Necessity.* Let $G = (V, E)$ be chordal. By Theorem 7.8, G contains a simplicial vertex v. By induction, the subgraph $G - v$ of G is the intersection graph of a collection $\mathcal{S}$ of subtrees of some tree T. Let $\mathcal{S}'$ be the subcollection of $\mathcal{S}$ corresponding to the set N of neighbours of v in G. As N is a clique, $\mathcal{S}'$ consists of pairwise intersecting subtrees. Hence, by Lemma 7.1 these subtrees have a vertex t of T in common. Now we extend T and all subtrees in $\mathcal{S}'$ with a new vertex t' and a new edge tt'. Moreover, we introduce a new subtree $\{t'\}$ representing v. In this way we obtain a subtree representation for G.

Sufficiency. Let G be the intersection graph of some collection $\mathcal{S}$ of subtrees of some tree T. Suppose that G contains a chordless circuit C_k with $k \geq 4$. Let C_k be the intersection graph of $S_1, \ldots, S_k \in \mathcal{S}$, with S_1 and S_2 intersecting. Then $S_1, S_2, S_3 \cup \cdots \cup S_k$ are three subtrees of T that are pairwise intersecting. So by Lemma 7.1, T has a vertex v contained in each of these three subtrees. So $v \in S_1 \cap S_2 \cap S_i$ for some $i \in \{3, \ldots, k\}$. This yields a chord in C_k. ▮

This theorem enables us to interpret the perfectness of chordal graphs in terms of trees:

Corollary 7.10a. *Let $\mathcal{S}$ be a collection of nonempty subtrees of a tree T. Then the maximum number of pairwise vertex-disjoint trees in $\mathcal{S}$ is equal to the minimum number of vertices of T intersecting each tree in $\mathcal{S}$.*

Proof. Directly from Theorems 7.9 and 7.10, using Lemma 7.1. ▮

Similarly we have:

Corollary 7.10b. *Let $\mathcal{S}$ be a collection of subtrees of a tree T. Let k be the maximum number of times that any vertex of T is covered by trees in $\mathcal{S}$. Then $\mathcal{S}$ can be partitioned into subcollections $\mathcal{S}_1, \ldots, \mathcal{S}_k$ such that each $\mathcal{S}_i$ consists of pairwise vertex-disjoint trees.*

Proof. Directly from Corollary 7.9a and Theorem 7.10, again using Lemma 7.1. ▮

Exercises

7.28. Show that a graph $G = (V, E)$ is chordal if and only if each induced subgraph has a simplicial vertex.

7.29. Show that a graph is an interval graph if and only if it is chordal and its complement is a comparability graph.

7.30. Derive from the proof of Theorem 7.8 that each chordal graph is either a clique or contains two nonadjacent simplicial vertices.

7.31. Let G be a chordal graph. Derive from the proof of Theorem 7.8 that each vertex v that is nonadjacent to at least one vertex $w \neq v$, is nonadjacent to at least one simplicial vertex $w \neq v$.

7.32. Show that a graph $G = (V, E)$ is chordal if and only if the edges of G can be oriented so as to obtain a directed graph $D = (V, A)$ with the following properties:

(25)
(i) D is acyclic;
(ii) if (u, v) and (u, w) belong to A then (v, w) or (w, v) belongs to A.

8. Integer linear programming and totally unimodular matrices

8.1. Integer linear programming

Many combinatorial optimization problems can be described as maximizing a linear function c^Tx over the *integer* vectors in some polyhedron $P = \{x \mid Ax \leq b\}$. (A vector $x \in \mathbb{R}^n$ is called *integer* if each component is an integer, i.e., if x belongs to $\mathbb{Z}^n$.)

So this type of problems can be described as:

(1) $$\max\{c^Tx \mid Ax \leq b; x \in \mathbb{Z}^n\}.$$

Such problems are called *integer linear programming* problems. They consist of maximizing a linear function over the intersection $P \cap \mathbb{Z}^n$ of a polyhedron P with the set $\mathbb{Z}^n$ of integer vectors.

Example. Consider a graph $G = (V, E)$. Then the problem of finding a matching of maximum cardinality can be described as follows. Let A be the $V \times E$ incidence matrix of G. So the rows of A are indexed by the vertices of G, while the columns of A are indexed by the edges of G and for any $v \in V$ and $e \in E$:

(2) $$\begin{aligned} A_{v,e} &:= 1 \quad \text{if } v \in e;\\ &:= 0 \quad \text{if } v \notin e. \end{aligned}$$

Now finding a maximum-cardinality matching is equivalent to:

(3) $$\begin{array}{rll} \text{maximize} & \displaystyle\sum_{e \in E} x_e & \\ \text{subject to} & \displaystyle\sum_{e \ni v} x_e \leq 1 & \text{for each } v \in V,\\ & x_e \geq 0 & \text{for each } e \in E,\\ & x_e \in \mathbb{Z} & \text{for each } e \in E. \end{array}$$

This is the same as:

(4) $$\max\{\mathbf{1}^Tx \mid x \geq 0; Ax \leq \mathbf{1}; x \text{ integer}\},$$

where $\mathbf{1}$ denotes an all-one vector, of appropriate size. ▮

Clearly, always the following holds:

(5) $$\max\{c^Tx \mid Ax \leq b; x \text{ integer}\} \leq \max\{c^Tx \mid Ax \leq b\}.$$

The above example, applied to the graph K_3 shows that strict inequality can hold. This implies, that generally one will have strict inequality in the following duality relation:

(6) $$\max\{c^Tx \mid Ax \leq b; x \text{ integer}\} \leq \min\{y^Tb \mid y \geq 0; y^TA = c^T; y \text{ integer}\}.$$

A polytope P is called *integer* if each of its vertices is an integer vector. Clearly, if a polytope $P = \{x \mid Ax \leq b\}$ is integer, then the *LP*-problem

(7) $$\max\{c^Tx \mid Ax \leq b\}$$

has an integer optimum solution. So in that case,

(8) $$\max\{c^Tx \mid Ax \leq b; x \text{ integer}\} = \max\{c^Tx \mid Ax \leq b\}.$$

In Exercise 8.5 below we shall see that in a sense also the converse holds.

No polynomial-time algorithm is known to exist for solving an integer linear programming problem in general. In fact, the general integer linear programming problem is NP-complete, and it is conjectured that no polynomial-time algorithm exists.

However, for special classes of integer linear programming problems, polynomial-time algorithms have been found. These classes often come from combinatorial problems, like the matching problem above.

Exercises

8.1. Let P be a polytope. Prove that the set conv.hull$(P \cap \mathbb{Z}^n)$ is again a polytope.

8.2. Let $P = \{x \mid Ax \leq b\}$ be a polyhedron, where A is a rational matrix. Show that the set conv.hull$(P \cap \mathbb{Z}^n)$ is again a polyhedron.

8.3. Let $G = (V, E)$ be a graph. Describe the problem of finding a vertex cover of minimum cardinality as an integer linear programming problem.

8.4. Let $G = (V, E)$ be a graph. Describe the problem of finding a clique (= complete subgraph) of maximum cardinality as an integer linear programming problem.

8.5. Show that a polytope P is integer, if and only if for each vector c, the linear programming problem $\max\{c^T x \mid Ax \leq b\}$ has an integer optimum solution.

8.2. Totally unimodular matrices

Total unimodularity of matrices turns out to form an important tool in studying integer vectors in polyhedra.

A matrix A is called *totally unimodular* if each square submatrix of A has determinant equal to 0, +1, or −1. In particular, each entry of a totally unimodular matrix is 0, +1, or −1.

A link between total unimodularity and integer linear programming is given by the following fundamental result.

Theorem 8.1. *Let A be a totally unimodular $m \times n$ matrix and let $b \in \mathbb{Z}^m$. Then each vertex of the polyhedron*

$$P := \{x \mid Ax \leq b\} \tag{9}$$

is an integer vector.

Proof. Let A have order $m \times n$. Let z be a vertex of P. By Theorem 2.2, the submatrix A_z has rank n. So A_z has a nonsingular $n \times n$ submatrix A'. Let b' be the part of b corresponding to the rows of A that occur in A'.

Since, by definition, A_z is the set of rows a_i of A for which $a_i z = b_i$, we know $A'z = b'$. Hence $z = (A')^{-1}b'$. However, since $|\det A'| = 1$, all entries of the matrix $(A')^{-1}$ are integer. Therefore, z is an integer vector. ∎

As a direct corollary we have a similar result for polyhedra in general (not necessarily having vertices). Define a polyhedron P to be *integer* if for each vector c for which

$$\max\{c^T x \mid x \in P\} \tag{10}$$

is finite, the maximum is attained by some integer vector. So:

(11) if $P = \{x \mid Ax \leq b\}$ where A is an $m \times n$ matrix of rank n, then P is integer if and only if each vertex of P is integer.

Then we have:

Corollary 8.1a. *Let A be a totally unimodular $m \times n$ matrix and let $b \in \mathbb{Z}^m$. Then the polyhedron*

(12) $$P := \{x \mid Ax \leq b\}$$

is an integer polyhedron.

Proof. Let x^* be an optimum solution of (10). Choose integer vectors $d', d'' \in \mathbb{Z}^n$ such that $d' \leq x^* \leq d''$. Consider the polyhedron

(13) $$Q := \{x \in \mathbb{R}^n \mid Ax \leq b; d' \leq x \leq d''\}.$$

So Q is bounded.

Moreover, Q is the set of all vectors x satisfying

(14) $$\begin{pmatrix} A \\ -I \\ I \end{pmatrix} x \leq \begin{pmatrix} b \\ -d' \\ d'' \end{pmatrix}.$$

Now the matrix here is again totally unimodular (this follows easily from the total unimodularity of A). Hence by Theorem 8.1, Q is an integer polytope. This implies that the linear programming problem $\max\{c^T x \mid x \in \mathbb{Q}\}$ is attained by some integer vector $\tilde{x}$.

But then $\tilde{x}$ is also an optimum solution for the original LP-problem $\max\{c^T x \mid Ax \leq b\}$. Indeed, $\tilde{x}$ satisfies $A\tilde{x} \leq b$, as $\tilde{x}$ belongs to Q. Moreover,

(15) $$c^T \tilde{x} \geq c^T x^* = \max\{c^T x \mid Ax \leq b\},$$

implying that $\tilde{x}$ is an optimum solution. ▮

It follows that each linear programming problem with integer data and totally unimodular constraint matrix has integer optimum primal and dual solutions:

Corollary 8.1b. *Let A be a totally unimodular $m \times n$ matrix, let $b \in \mathbb{Z}^m$ and let $c \in \mathbb{Z}^n$. Then both problems in the LP-duality equation:*

(16) $$\max\{c^T x \mid Ax \leq b\} = \min\{y^T b \mid y \geq 0; y^T A = c^T\}$$

have integer optimum solutions (if the optima are finite).

Proof. Directly from Corollary 8.1a, using the fact that with A also the matrix

(17) $$\begin{pmatrix} -I \\ A^T \\ -A^T \end{pmatrix}$$

is totally unimodular. ▮

Hoffman and Kruskal [1956] showed, as we shall see below, that the above property more or less characterizes total unimodularity.

To derive this result, define an $m \times n$ matrix A to be *unimodular* if it has rank m and each $m \times m$ submatrix has determinant equal to 0, +1, or −1. It is easy to see that a matrix A is totally unimodular, if and only if the matrix $[I \;\; A]$ is unimodular.

We follow the proof of Hoffman and Kruskal's result given by Veinott and Dantzig [1968]. As a preparation one first shows:

Theorem 8.2. *Let A be an integer $m \times n$ matrix of rank m. Then A is unimodular, if and only if for each integer vector b the polyhedron*

(18) $$P = \{x \mid x \geq 0; Ax = b\}$$

is integer.

Proof. *Necessity.* First suppose that A is unimodular. Let b be an integer vector. Let D be the matrix

(19) $$D := \begin{pmatrix} -I \\ A \\ -A \end{pmatrix} \text{ and } f := \begin{pmatrix} 0 \\ b \\ -b \end{pmatrix}.$$

Note that the system $x \geq 0, Ax = b$ is the same as $Dx \leq f$.

Since D has rank n, we know that for each $c \in \mathbb{R}^n$, the linear programming problem

(20) $$\max\{c^T x \mid x \geq 0; Ax = b\} = \max\{c^T x \mid Dx \leq f\}$$

is attained by a *vertex* z of P (if the optima are finite).

Now consider the matrix D_z. By definition, this is the submatrix of D consisting of those rows D_i of D which have equality in $Dz \leq f$.

Clearly, D_z contains all rows of D that are in A and in $-A$. Since A has rank m, this implies that D_z contains a nonsingular $n \times n$ matrix B that fully contains A and moreover, part of $-I$. Since A is unimodular, $\det B$ equals $+1$ or -1. Let f' be the part of f corresponding to B. So $Bz = f'$, and hence $z = B^{-1}f'$. As $|\det B| = 1$, it follows that z is an integer vector.

Sufficiency. Suppose that $P = \{x \mid x \geq 0; Ax = b\}$ is integer, for each choice of an integer vector b. Let B be an $m \times m$ nonsingular submatrix of A. We must show that $\det B$ equals $+1$ or -1.

Without loss of generality, we may assume that B consists of the first m columns of A.

It suffices to show that $B^{-1}v$ is an integer vector for each choice of an integer vector v. (This follows from the fact that then B^{-1} itself is an integer matrix, and hence $(\det B)^{-1}$=det(B^{-1}) is an integer. This implies that $\det B$ equals $+1$ or -1.)

So let v be an integer vector. Then there exists an integer vector $u \in \mathbb{R}^m$ such that

(21) $$z := u + B^{-1}v > 0.$$

Define

(22) $$b := Bz.$$

So $b = Bz = Bu + BB^{-1}v = Bu + v$ is an integer vector.

Let z' arise from z by adding zero-components to z so as to obtain a vector in $\mathbb{R}^n$. So

(23) $$z' = \begin{pmatrix} z \\ \mathbf{0} \end{pmatrix},$$

where $\mathbf{0}$ is the all-zero vector in $\mathbb{R}^{n-m}$.

Then z' is a vertex of the polyhedron P (since $z' \in P$ and since there are n linearly independent rows in the matrix D for which $Dz \leq f$ holds with equality).

So z' is integer, and hence

(24) $$B^{-1}v = z - u$$

is an integer vector. ∎

This gives the result of Hoffman and Kruskal [1956]:

Corollary 8.2a (Hoffman-Kruskal theorem)**.** *Let A be an integer $m \times n$ matrix. Then A is totally unimodular, if and only if for each integer vector b the polyhedron*

(25) $$P = \{x \mid x \geq 0; Ax \leq b\}$$

is integer.

Proof. *Necessity.* Directly from Corollary 8.1a.

Sufficiency. Let P be an integer polyhedron, for each choice of an integer vector b. We show that, for each choice of $b \in \mathbb{Z}^m$, each vertex z of the polyhedron

(26) $$Q := \{z \mid z \geq 0; [I \quad A]z = b\}.$$

is integer. Indeed, z can be decomposed as

(27) $$z = \begin{pmatrix} z' \\ z'' \end{pmatrix},$$

where $z' \in \mathbb{R}^m$ and $z'' \in \mathbb{R}^n$. So $z' = b - Az''$.

Then z'' is a vertex of P. [This follows from the fact that if z'' would be equal to $\frac{1}{2}(v+w)$ for two other points v, w in P, then

(28) $$z' = b - Az'' = \frac{1}{2}(b - Av) + \frac{1}{2}(b - Aw).$$

Hence

(29) $$z = \begin{pmatrix} z' \\ z'' \end{pmatrix} = \frac{1}{2}\begin{pmatrix} b - Av \\ v \end{pmatrix} + \frac{1}{2}\begin{pmatrix} b - Aw \\ w \end{pmatrix}.$$

This contradicts the fact that z is a vertex of Q.]

So, by assumption, z'' is integer. Hence also $z' = b - Az''$ is integer, and hence z is integer.

So for each choice of b in $\mathbb{Z}^m$, the polyhedron Q is integer. Hence, by Theorem 8.2, the matrix $[I \quad A]$ is unimodular. This implies that A is totally unimodular. ∎

Exercises

8.6. Show that an integer matrix A is totally unimodular, if and only if for all integer vectors b and c, both sides of the linear programming duality equation

(30) $$\max\{c^T x \mid x \geq 0; Ax \leq b\} = \min\{y^T b \mid y \geq 0; y^T A \geq c^T\}$$

are attained by integer optimum solutions x and y (if the optima are finite).

8.7. Give an example of an integer matrix A and an integer vector b such that the polyhedron $P := \{x \mid Ax \leq b\}$ is integer, while A is not totally unimodular.

8.8. Let A be a totally unimodular matrix. Show that the columns of A can be split into two classes such that the sum of the columns in one class, minus the sum of the columns in the other class, gives a vector with entries 0, +1, and −1 only.

8.9. Let A be a totally unimodular matrix and let b be an integer vector. Let x be an integer vector satisfying $x \geq 0; Ax \leq 2b$. Show that there exist integer vectors $x' \geq 0$ and $x'' \geq 0$ such that $Ax' \leq b$, $Ax'' \leq b$ and $x = x' + x''$.

8.3. Totally unimodular matrices from bipartite graphs

Let A be the $V \times E$ incidence matrix of a graph $G = (V, E)$ (cf. (2)). The matrix A generally is not totally unimodular. E.g., if G is the complete graph K_3 on three vertices, then the determinant of A is equal to $+2$ or -2.

However, the following can be proved:

Theorem 8.3. *Graph G is bipartite, if and only if its incidence matrix A is totally unimodular.*

Proof. *Sufficiency.* Let A be totally unimodular. Suppose G is not bipartite. Then G contains an odd circuit, say with vertices $v_1, \ldots, v_k$ and edges $e_1, \ldots, e_k$. The submatrix of A on the rows indexed by $v_1, \ldots, v_k$ and the columns indexed by $e_1, \ldots, e_k$, is of type

$$(31) \qquad \begin{pmatrix} 1 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 1 & 1 \\ 1 & 0 & 0 & \cdots & \cdots & 0 & 1 \end{pmatrix},$$

up to permutation of rows and columns.

It is not difficult to see that matrix (31) has determinant 2. This contradicts the total unimodularity of A.

Necessity. Let G be bipartite. Let B be a square submatrix of A, of order $t \times t$, say. We show that $\det B$ equal 0 or ± 1 by induction on t. If $t = 1$, the statement is trivial.

So let $t > 1$. We distinguish three cases.

Case 1. *B has a column with only 0's.* Then $\det B$=0.

Case 2. *B has a column with exactly one* 1. In that case we can write (possibly after permuting rows or columns):

$$(32) \qquad B = \begin{pmatrix} 1 & b^T \\ \mathbf{0} & B' \end{pmatrix},$$

for some matrix B' and vector b, where $\mathbf{0}$ denotes the all-zero vector in $\mathbb{R}^{t-1}$. By the induction hypothesis, $\det B' \in \{0, \pm 1\}$. Hence, by (32), $\det B \in \{0, \pm 1\}$.

Case 3. *Each column of B contains exactly two 1's.* Then, since G is bipartite, we can write (possibly after permuting rows):

$$(33) \qquad B = \begin{pmatrix} B' \\ B'' \end{pmatrix},$$

in such a way that each column of B' contains exactly one 1 and each column of B'' contains exactly one 1. So adding up all rows in B' gives the all-one vector, and also adding up all rows in B'' gives the all-one vector. Therefore, the rows of B are linearly dependent, and hence $\det B$=0. ▮

As direct corollaries of this theorem, together with Corollary 8.1b, we obtain some theorems of Kőnig. First:

Corollary 8.3a (Kőnig's matching theorem)**.** *Let G be a bipartite graph. Then the maximum cardinality of a matching in G is equal to the minimum cardinality of a vertex cover in G.*

Proof. Clearly, the maximum cannot be larger than the minimum. To see that equality holds, let A be the $V \times E$ incidence matrix of G. Then by Corollary 8.1b, both optima in the LP-duality equation

$$(34) \qquad \max\{\mathbf{1}^T x \mid x \geq 0; Ax \leq \mathbf{1}\} = \min\{y^T \mathbf{1} \mid y \geq 0; y^T A \geq \mathbf{1}\}$$

are attained by integer optimum solutions x^* and y^*.

Since x^* is an integer vector satisfying $x \geq 0; Ax \leq \mathbf{1}$, x^* is a $\{0,1\}$ vector. Let M be the set of edges e of G for which $x^*_e = 1$. Then M is a matching, since $Ax^* \leq \mathbf{1}$ holds, implying that for each vertex v there is at most one edge e with $x^*_e = 1$. Moreover, the cardinality $|M|$ of M satisfies $|M| = \mathbf{1}^T x^*$. So $|M|$ is equal to the maximum in (34).

On the other hand, as vector y^* attains the minimum in (34), it should be a $\{0,1\}$ vector. (If some component would be 2 or larger, we could reduce it to 1, without violating $y^T A \geq \mathbf{1}$ but decreasing $y^T\mathbf{1}$. This contradicts the fact that y^* attains the minimum.)

Let W be the set of vertices of G for which $y^*_v = 1$. Then W is a vertex cover, since $y^{*T}A \geq \mathbf{1}$ holds, implying that for each edge e of G there is at least one vertex v with $y^*_v = 1$. Moreover, the cardinality $|W|$ of W satisfies $|W| = y^{*T}\mathbf{1}$. So $|W|$ is equal to the minimum in (34). ▮

One similarly derives:

Corollary 8.3b (Kőnig's edge cover theorem)**.** *Let G be a bipartite graph. Then the maximum cardinality of a coclique in G is equal to the minimum cardinality of an edge cover in G.*

Proof. Similar to the proof of Corollary 8.1a (now with A^T instead of A). ▮

One can also derive weighted versions of these two min-max relations. Let X be some finite set and let $w : X \to \mathbb{R}$ be a 'weight' function on X. The *weight* $w(Y)$ of some subset $Y \subseteq X$ is, by definition:

(35) $$w(Y) := \sum_{x \in Y} w(x).$$

Then:

Corollary 8.3c. *Let $G = (V,E)$ be a bipartite graph and let $w : V \to \mathbb{Z}_+$ be a weight function on E. Then:*

(i) *The maximum weight of a matching in G is equal to the minimum value of $\sum_{v \in V} f(v)$, where f ranges over all functions $f : V \to \mathbb{Z}_+$ such that $f(u) + f(v) \geq w(\{u,v\})$ for each edge $\{u,v\}$ of G;*

(ii) *The minimum weight of an edge cover in G is equal to the maximum value of $\sum_{v \in V} f(v)$, where f ranges over all functions $f : V \to \mathbb{Z}_+$ such that $f(u) + f(v) \leq w(\{u,v\})$ for each edge $\{u,v\}$ of G.*

Proof. The statements are equivalent to both sides in

(36) $$\max\{w^T x \mid x \geq 0; Ax \leq \mathbf{1}\} = \min\{y^T\mathbf{1} \mid y \geq 0; y^T A \geq w\}$$

and in

(37) $$\min\{w^T x \mid x \geq 0; Ax \geq \mathbf{1}\} = \max\{y^T\mathbf{1} \mid y \geq 0; y^T A \leq w\}$$

having integer optimum solutions. These facts follow from Theorem 8.3 and Corollary 8.1b. ▮

Similarly one has min-max relations for the maximum weight of a coclique and the minimum weight of a vertex cover in bipartite graphs (cf. Exercises 8.10 and 8.11).

Another corollary is as follows. For any finite set X and any subset Y of X, define the *incidence vector* $\chi^Y \in \mathbb{R}^X$ of Y as:

(38)
$$\begin{aligned} \chi^Y_x &:= 1 \quad \text{if } x \in Y; \\ &:= 0 \quad \text{if } x \notin Y. \end{aligned}$$

Now let $G = (V, E)$ be a graph. The *matching polytope* $P_{\text{matching}}(G)$ of G is, by definition, the convex hull (in $\mathbb{R}^E$) of the incidence vectors of all matchings in G. That is:

(39)
$$P_{\text{matching}}(G) = \text{conv.hull}\{\chi^M \mid M \text{ matching in } G\}.$$

Now with Theorem 8.3 we can give the linear inequalities describing $P_{\text{matching}}(G)$:

Corollary 8.3d. *If G is bipartite, the matching polytope $P_{\text{matching}}(G)$ of G is equal to the set of vectors x in $\mathbb{R}^E$ satisfying:*

(40)
$$\begin{aligned} &\text{(i)} & x_e &\geq 0 & &\textit{for each } e \in E; \\ &\text{(ii)} & \sum_{e \ni v} x_e &\leq 1 & &\textit{for each } v \in V. \end{aligned}$$

Proof. Let Q be the polytope defined by (40). Clearly, $P_{\text{matching}}(G) \subseteq Q$, since the incidence vector χ^M of any matching M satisfies (40).

To see that $Q \subseteq P_{\text{matching}}(G)$, observe that Q satisfies

(41)
$$Q = \{x \mid x \geq 0; Ax \leq \mathbf{1}\},$$

where A is the incidence matrix of A.

Since A is totally unimodular (Theorem 8.3), we know that Q is integer, i.e., that each vertex of Q is an integer vector (Corollary 8.1a). So Q is the convex hull of the integer vectors contained in Q. Now each integer vector in Q is equal to the incidence vector χ^M of some matching M in G. So Q must be contained in $P_{\text{matching}}(G)$. ▮

Again, one cannot delete the bipartiteness condition here, as for any odd circuit there exists a vector satisfying (40) but not belonging to the matching polytope $P_{\text{matching}}(G)$.

Similarly, let the *perfect matching polytope* $P_{\text{perfect matching}}(G)$ of G be defined as the convex hull of the incidence vectors of the *perfect* matchings in G. Then we have:

Corollary 8.3e. *If G is bipartite, the perfect matching polytope $P_{\text{perfect matching}}(G)$ of G is equal to the set of vectors x in $\mathbb{R}^E$ satisfying:*

(42)
$$\begin{aligned} &\text{(i)} & x_e &\geq 0 & &\textit{for each } e \in E; \\ &\text{(ii)} & \sum_{e \ni v} x_e &= 1 & &\textit{for each } v \in V. \end{aligned}$$

Proof. Similarly as above. ▮

Exercises

8.10. Give a min-max relation for the maximum weight of a coclique in a bipartite graph.

8.11. Give a min-max relation for the minimum weight of a vertex cover in a bipartite graph.

8.12. Let $G = (V, E)$ be a nonbipartite graph. Show that the inequalities (40) are not enough to define the matching polytope of G.

8.13. The *edge cover polytope* $P_{\text{edge cover}}(G)$ of a graph is the convex hull of the incidence vectors of the edge covers in G. Give a description of the linear inequalities defining the edge cover polytope of a bipartite graph.

8.14. The *coclique polytope* $P_{\text{coclique}}(G)$ of a graph is the convex hull of the incidence vectors of the cocliques in G. Give a description of the linear inequalities defining the coclique polytope of a bipartite graph.

8.15. The *vertex cover polytope* $P_{\text{vertex cover}}(G)$ of a graph is the convex hull of the incidence vectors of the vertex covers in G. Give a description of the linear inequalities defining the vertex cover polytope of a bipartite graph.

8.16. Derive from Corollary 8.3e that for each doubly stochastic matrix M there exist permutation matrices $P_1, \ldots, P_m$ and reals $\lambda_1, \ldots, \lambda_m \geq 0$ such that $\lambda_1 + \cdots + \lambda_m = 1$ and

(43) $$M = \lambda_1 P_1 + \cdots \lambda_m P_m.$$

(A matrix M is called *doubly stochastic* if each row sum and each column sum is equal to 1. A matrix P is called a *permutation matrix* if it is a $\{0,1\}$ matrix, with in each row and in each column exactly one 1.)

8.4. Totally unimodular matrices from directed graphs

A second class of totally unimodular matrices can be derived from directed graphs. Let $D = (V, A)$ be a directed graph. The $V \times A$ *incidence matrix* M of D is defined by:

(44) $$\begin{array}{rcrl} M_{v,a} & := & +1 & \text{if } a \text{ leaves } v, \\ & := & -1 & \text{if } a \text{ enters } v, \\ & := & 0 & \text{otherwise.} \end{array}$$

So each column of M has exactly one $+1$ and exactly one -1, while all other entries are 0.

Now we have:

Theorem 8.4. *The incidence matrix M of any directed graph D is totally unimodular.*

Proof. Let B be a square submatrix of M, of order t say. We prove that $\det B \in \{0, \pm 1\}$ by induction on t, the case $t = 1$ being trivial.

Let $t > 1$. We distinguish three cases.

Case 1. *B has a column with only zeros.* Then $\det B = 0$.

Case 2. *B has a column with exactly one nonzero.* Then we can write (up to permuting rows and columns):

(45) $$B = \begin{pmatrix} \pm 1 & b^T \\ \mathbf{0} & B' \end{pmatrix},$$

for some vector b and matrix B'.

Now by our induction hypothesis, $\det B' \in \{0, \pm 1\}$, and hence $\det B \in \{0, \pm 1\}$.

Case 3. *Each column of B contains two nonzeros.* Then each column of B contains one $+1$ and one -1, while all other entries are 0. So the rows of B add up to an all-zero vector, and hence $\det B = 0$. ▮

The incidence matrix M of a directed graph $D = (V, A)$ relates to flows and circulations in D. Indeed, any vector $x \in \mathbb{R}^A$ can be considered as a function defined on the arcs of D. Then the condition

(46) $$Mx = 0$$

is just the 'flow conservation law'. That is, it says:

(47) $$\sum_{a\in\delta^{\text{out}}(v)} x(a) = \sum_{a\in\delta^{\text{in}}(v)} x(a) \text{ for each } v \in V.$$

So we can derive from Theorem 8.4:

Corollary 8.4a. *Let $D = (V, A)$ be a directed graph and let $c : A \to \mathbb{Z}$ and $d : A \to \mathbb{Z}$. If there exists a circulation x on A with $c \le x \le d$, then there exists an integer circulation x on A with $c \le x \le d$.*

Proof. If there exists a circulation x with $c \le x \le d$, then the polytope

(48) $$P := \{x \mid c \le x \le d; Mx = 0\}$$

is nonempty. So it has at least one vertex x^*. Then, by Corollary 8.1a, x^* is an integer circulation satisfying $c \le x^* \le d$. ▮

In fact, one can derive Hoffman's circulation theorem— see Exercise 8.17. Another theorem that can be derived is the max-flow min-cut theorem.

Corollary 8.4b (max-flow min-cut theorem)**.** *Let $D = (V, A)$ be a directed graph, let r and s be two of the vertices of D, and let $c : A \to \mathbb{R}_+$ be a 'capacity' function on A. Then the maximum value of an $r - s$ flow subject to c is equal to the minimum capacity of an $r - s$ cut.*

Proof. Since the maximum clearly cannot exceed the minimum, it suffices to show that there exists an $r - s$ flow $x \le c$ and an $r - s$ cut, the capacity of which is not more than the value of x.

Let M be the incidence matrix of D and let M' arise from M by deleting the rows corresponding to r and s. So the condition $M'x = 0$ means that the flow conservation law should hold in any vertex $v \ne r, s$.

Let w be the row of M corresponding to vertex r. So $w_a = +1$ if arc a leaves r and $w_a = -1$ if arc a enters r, while $w_a = 0$ for all other arcs a.

Now the maximum value of an $r - s$ flow subject to c is equal to

(49) $$\max\{w^T x \mid 0 \le x \le c; M'x = 0\}.$$

By LP-duality, this is equal to

(50) $$\min\{y^T c \mid y \ge 0; y^T + z^T M' \ge w\}.$$

The inequality system in (50) is:

(51) $$(y^T \;\; z^T) \begin{pmatrix} I & I \\ 0 & M' \end{pmatrix} \ge (0 \;\; w).$$

The matrix here is totally unimodular, by Theorem 8.4.

Since w is an integer vector, this implies that the minimum (50) is attained by *integer* vectors y and z.

Now define

(52) $$W := \{v \in V \setminus \{r, s\} \mid z_v \le -1\} \cup \{r\}.$$

So W is a subset of V containing r and not containing s.

It suffices now to show that

(53) $$c(\delta^{\text{out}}(W)) \le y^T c,$$

since y^Tc is not more than the maximum flow value (49).

To prove (53) it suffices to show that

(54) if $a = (u, v) \in \delta^{\text{out}}(W)$ then $y_a \geq 1$.

Define $\tilde{z}_r := -1$, $\tilde{z}_s := 0$, and $\tilde{z}_u = z_u$ for all other u. Then $y^T + \tilde{z}^T M \geq 0$. Hence for all $a = (u, v) \in \delta^{\text{out}}(W)$ one has $y_a + \tilde{z}_u - \tilde{z}_v \geq 0$, implying $y_a \geq \tilde{z}_v - \tilde{z}_u \geq 1$. This proves (54). ▮

Similarly as in Corollary 8.4a it follows that if all capacities are integers, then there exists a maximum *integer* flow.

Next define a matrix to be an *interval matrix* if each entry is 0 or 1 and each row is of type

(55) $(0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0)$.

Corollary 8.4c. *Each interval matrix is totally unimodular.*

Proof. Let M be an interval matrix and let B be a $t \times t$ submatrix of M. Then B is again an interval matrix. Let N be the $t \times t$ matrix given by:

(56)
$$N := \begin{pmatrix} 1 & -1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 1 & -1 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}.$$

Then the matrix $N \cdot B^T$ is a $\{0, \pm 1\}$ matrix, with at most one $+1$ and at most one -1 in each column.

So it is a submatrix of the incidence matrix of some directed graph. Hence by Theorem 8.4, $\det(N \cdot B^T) \in \{0, \pm 1\}$. Moreover, $\det N = 1$. So $\det B = \det B^T \in \{0, \pm 1\}$. ▮

Exercises

8.17. Derive Hoffman's circulation theorem (Theorem 4.9) from Theorem 8.4.

8.18. Derive Dilworth's decomposition theorem (Theorem 7.5) from Theorem 8.4.

8.19. Let $D = (V, A)$ be a directed graph and let $T = (V, A')$ be a directed spanning tree on V.

Let C be the $A' \times A$ matrix defined as follows. Take $a' \in A'$ and $a = (u, v) \in A$, and define $C_{a',a} := +1$ if a' occurs in forward direction in the $u-v$ path in T and $C_{a',a} := -1$ if a' occurs in backward direction in the $u - v$ path in T. For all other $a' \in A'$ and $a \in A$ set $C_{a',a} := 0$.

(i) Prove that C is totally unimodular.
(*Hint:* Use a matrix similar to matrix N in Corollary 8.4c.)

(ii) Show that interval matrices and incidence matrices of directed graphs are special cases of such a matrix C.

9. Multicommodity flows and disjoint paths

9.1. Introduction

The problem of finding a maximum flow from one 'source' r to one 'sink' s is highly tractable. There is a very efficient algorithm, which outputs an integer maximum flow if all capacities are integer. Moreover, the maximum flow value is equal to the minimum capacity of a cut separating r and s. If all capacities are equal to 1, the problem reduces to finding arc-disjoint paths. Some direct transformations give similar results for vertex capacities and for vertex-disjoint paths.

Often in practice however, one is not interested in connecting only one pair of source and sink by a flow or by paths, but several pairs of sources and sinks simultaneously. One may think of a large communication or transportation network, where several messages or goods must be transmitted all at the same time over the same network, between different pairs of terminals. A recent application is the design of *very large-scale integrated* (VLSI) circuits, where several pairs of pins must be interconnected by wires on a chip, in such a way that the wires follow given 'channels' and that the wires connecting different pairs of pins do not intersect each other.

Mathematically, these problems can be formulated as follows. First, there is the *multicommodity flow problem* (or *k-commodity flow problem*):

(1) given: a directed graph $G = (V, E)$, pairs $(r_1, s_1), \ldots, (r_k, s_k)$ of vertices of G, a 'capacity' function $c : E \to \mathbb{Q}_+$, and 'demands' $d_1, \ldots, d_k$,

find: for each $i = 1, \ldots, k$, an $r_i - s_i$ flow $x_i \in \mathbb{Q}_+^E$ so that x_i has value d_i and so that for each arc e of G:

$$\sum_{i=1}^{k} x_i(e) \leq c(e).$$

The pairs (r_i, s_i) are called *commodities.* (We assume $r_i \neq s_i$ throughout.)

If we require each x_i to be an integer flow, the problem is called the *integer multicommodity flow problem* or *integer k-commodity flow problem.* (To distinguish from the integer version of this problem, one sometimes adds the adjective *fractional* to the name of the problem if no integrality is required.)

The problem has a natural analogue to the case where G is undirected. We replace each undirected edge $e = \{v, w\}$ by two opposite arcs (v, w) and (w, v) and ask for flows $x_1, \ldots, x_k$ of values $d_1, \ldots, d_k$, respectively, so that for each edge $e = \{v, w\}$ of G:

(2)
$$\sum_{i=1}^{k} (x_i(v, w) + x_i(w, v)) \leq c(e).$$

Thus we obtain the *undirected multicommodity flow problem* or *undirected k-commodity flow problem.* Again, we add *integer* if we require the x_i to be integer flows.

If all capacities and demands are 1, the integer multicommodity flow problem is equivalent to the *arc-* or *edge-disjoint paths problem*:

(3) given: a (directed or undirected) graph $G = (V, E)$, pairs $(r_1, s_1), \ldots, (r_k, s_k)$ of vertices of G,

find: pairwise edge-disjoint paths $P_1, \ldots, P_k$ where P_i is an $r_i - s_i$ path ($i = 1, \ldots, k$).

Related is the *vertex-disjoint paths problem*:

(4) given: a (directed or undirected) graph $G = (V, E)$, pairs $(r_1, s_1), \ldots, (r_k, s_k)$ of vertices of G,

find: pairwise vertex-disjoint paths $P_1, \ldots, P_k$ where P_i is an $r_i - s_i$ path $(i = 1, \ldots, k)$.

We leave it as an exercise (Exercise 9.1) to check that the vertex-disjoint paths problem can be transformed to the directed edge-disjoint paths problem.

The (fractional) multicommodity flow problem can be easily described as one of solving a system of linear inequalities in the variables $x_i(e)$ for $i = 1, \ldots, k$ and $e \in E$. The constraints are the flow conservation laws for each flow x_i separately, together with the inequalities given in (1). Therefore, the fractional multicommodity flow problem can be solved in polynomial time with any polynomial-time linear programming algorithm.

In fact, the only polynomial-time algorithm known for the fractional multicommodity flow problem is any general linear programming algorithm. Ford and Fulkerson [1958] designed an algorithm based on the simplex method, with column generation — see Section 9.6.

The following *cut condition* trivially is a necessary condition for the existence of a solution to the fractional multicommodity flow problem (1):

(5) for each $W \subseteq V$ the capacity of $\delta_E^{\mathrm{out}}(W)$ is not less than the demand of $\delta_R^{\mathrm{out}}(W)$,

where $R := \{(r_1, s_1), \ldots, (r_k, s_k)\}$. However, this condition is in general not sufficient, even not in the two simple cases given in Figure 9.1 (taking all capacities and demands equal to 1).

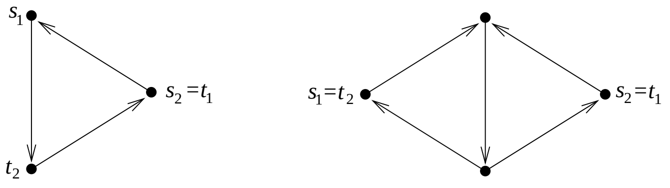


Figure 9.1

One may derive from the max-flow min-cut theorem that the cut condition *is* sufficient if $r_1 = r_2 = \cdots = r_k$ (similarly if $s_1 = s_2 = \cdots = s_k$) — see Exercise 9.3.

Similarly, in the undirected case a necessary condition is the following cut condition:

(6) for each $W \subseteq V$, the capacity of $\delta_E(W)$ is not less than the demand of $\delta_R(W)$

(taking $R := \{\{r_1, s_1\}, \ldots, \{r_k, s_k\}\}$). In the special case of the edge-disjoint paths problem (where all capacities and demands are equal to 1), the cut condition reads:

(7) for each $W \subseteq V, |\delta_E(W)| \geq |\delta_R(W)|$.

Figure 9.2 shows that this condition again is not sufficient.

However, Hu [1963] showed that the cut condition is sufficient for the existence of a fractional multicommodity flow, in the undirected case with $k = 2$ commodities. He gave an algorithm that yields a half-integer solution if all capacities and demands are integer. This result was extended by Rothschild and Whinston [1966]. We discuss these results in Section 9.2.

Similar results were obtained by Okamura and Seymour [1981] for arbitrary k, provided that the graph is planar and all terminals r_i, s_i are on the boundary of the unbounded face — see Section 9.5.

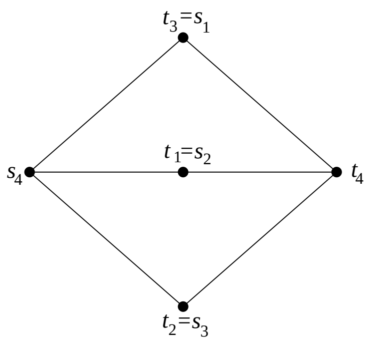


Figure 9.2

The integer multicommodity flow problem is NP-complete, even in the undirected case with $k = 2$ commodities and all capacities equal to 1, with arbitrary demands d_1, d_2 (Even, Itai, and Shamir [1976]). This implies that the undirected edge-disjoint paths problem is NP-complete, even if $|\{\{r_1, s_1\}, \ldots, \{r_k, s_k\}\}| = 2$.

In fact, the disjoint paths problem is NP-complete in all modes (directed/undirected, vertex/edge disjoint), even if we restrict the graph G to be planar (D.E. Knuth (see Karp [1975]), Lynch [1975], Kramer and van Leeuwen [1984]). For general directed graphs the arc-disjoint paths problem is NP-complete even for $k = 2$ 'opposite' commodities (r, s) and (s, r) (Fortune, Hopcroft, and Wyllie [1980]).

On the other hand, it is a deep result of Robertson and Seymour [1995] that the undirected vertex-disjoint paths problem is polynomially solvable for any fixed number k of commodities. Hence also the undirected edge-disjoint paths problem is polynomially solvable for any fixed number k of commodities.

Robertson and Seymour observed that if the graph G is planar and all terminals r_i, s_i are on the boundary of the unbounded face, there is an easy 'greedy-type' algorithm for the vertex-disjoint paths problem — see Section 9.4.

It is shown by Schrijver [1994] that for each fixed k, the k disjoint paths problem is solvable in polynomial time for directed planar graphs. For the directed planar arc-disjoint version, the complexity is unknown. That is, there is the following research problem:

Research problem. Is the directed arc-disjoint paths problem polynomially solvable for planar graphs with $k = 2$ commodities? Is it NP-complete?

Application 9.1: Multicommodity flows. Certain goods or messages must be transported through the same network, where the goods or messages may have different sources and sinks.

This is a direct special case of the problems described above.

Application 9.2: VLSI-routing. On a chip certain modules are placed, each containing a number of 'pins'. Certain pairs of pins should be connected by an electrical connection (a 'wire') on the chip, in such a way that each wire follows a certain (very fine) grid on the chip and that wires connecting different pairs of pins are disjoint.

Determining the routes of the wires clearly is a special case of the disjoint paths problem.

Application 9.3: Routing of railway stock. An extension of Application 4.5 is as follows. The stock of the railway company NS for the Amsterdam–Vlissingen line now consists of two types (1 and 2 say) of units, with a different number of seats s_1 and s_2 and different length l_1 and l_2. All units (also of different types) can be coupled with each other.

Again there is a schedule given, together with for each segment a minimum number of seats and a maximum length of the train. Moreover, the price p_i of buying any unit of type i is given.

Now the company wishes to determine the minimum costs of buying units of the two types so that the schedule can be performed and so that the total cost is minimized.

This can be considered as a 'min-cost integer multicommodity circulation problem'. That is we make the directed graph D as in Application 4.5. For each arc a corresponding to a segment we define $d(a)$ to be the minimum number of seats that should be offered on that segment, and $c(a)$ to be the maximum length possible at that segment. For all other arcs a we define $d(a) := 0$ and $c(a) := \infty$.

One should find two integer-valued circulations f_1 and f_2 in D such that

$$(8) \qquad s_1 f_1(a) + s_2 f_2(a) \geq d(a) \text{ and } l_1 f_1(a) + l_2 f_2(a) \leq c(a)$$

for each arc a and such that the sum $\sum(p_1 f_1(a) + p_2 f_2(a))$ is minimized, where a ranges over all 'overnight' arcs. Then $f_i(a)$ denotes the number of units of type i that should go on segment a.

Again several variations are possible, incorporating for instance the kilometer costs and maximum capacities of stock areas.

Exercises

9.1. Show that each of the following problems (a), (b), (c) can be reduced to problems (b), (c), (d), respectively:

(a) the undirected edge-disjoint paths problem,

(b) the undirected vertex-disjoint paths problem,

(c) the directed vertex-disjoint paths problem,

(d) the directed arc-disjoint paths problem.

9.2. Show that the undirected edge-disjoint paths problem for planar graphs can be reduced to the directed arc-disjoint paths problem for planar graphs.

9.3. Derive from the max-flow min-cut theorem that the cut condition (5) is sufficient for the existence of a fractional multicommodity flow if $r_1 = \cdots = r_k$.

9.4. Show that if the undirected graph $G = (V, E)$ is connected and the cut condition (7) is violated, then it is violated by some $W \subseteq V$ for which both W and $V \setminus W$ induce connected subgraphs of G.

9.5. (i) Show with Farkas' lemma: the fractional multicommodity flow problem (1) has a solution, if and only if for each 'length' function $l : E \to \mathbb{Q}_+$ one has:

$$(9) \qquad \sum_{i=1}^{k} d_i \cdot \text{dist}_l(r_i, s_i) \leq \sum_{e \in E} l(e)c(e).$$

(Here $\text{dist}_l(r, s)$ denotes the length of a shortest $r - s$ path with respect to l.)

(ii) Interprete the cut condition (5) as a special case of this condition.

9.2. Two commodities

Hu [1963] gave a direct combinatorial method for the undirected two-commodity flow problem and he showed that in this case the cut condition suffices. In fact, he showed that if the cut condition holds and all capacities and demands are integer, there exists a half-integer solution. We first give a proof of this result due to Sakarovitch [1973].

Consider a graph $G = (V, E)$, with commodities $\{r_1, s_1\}$ and $\{r_2, s_2\}$, a capacity function $c : E \to \mathbb{Z}_+$ and demands d_1 and d_2.

Theorem 9.1 (Hu's two-commodity flow theorem)**.** *The undirected two-commodity flow problem, with integer capacities and demands, has a half-integer solution, if and only if the cut condition* (6) *is satisfied.*

Proof. Suppose the cut condition holds. Orient the edges of G arbitrarily, yielding the directed graph $D = (V, A)$. For any $a \in A$ we denote by $c(a)$ the capacity of the underlying undirected edge.

Define for any $x \in R^A$ and any $v \in V$:

$$f(x, v) := \sum_{a\in\delta^{\text{out}}(v)} x(a) - \sum_{a\in\delta^{\text{in}}(v)} x(a). \tag{10}$$

So $f(x, v)$ is the 'net loss' of x in vertex v.

By the max-flow min-cut theorem there exists a function $x' : A \to \mathbb{Z}$ satisfying:

$$\begin{aligned} &f(x', r_1) = d_1, f(x', s_1) = -d_1, f(x', r_2) = d_2, f(x', s_2) = -d_2,\\ &f(x', v) = 0 \text{ for each other vertex } v,\\ &|x'(a)| \leq c(a) \text{ for each arc } a \text{ of } D. \end{aligned} \tag{11}$$

This can be seen by extending the undirected graph G by adding two new vertices r' and s' and four new edges $\{r', r_1\}, \{s_1, s'\}$ (both with capacity d_1) and $\{r', r_2\}, \{s_2, s'\}$ (both with capacity d_2) as in Figure 9.3.

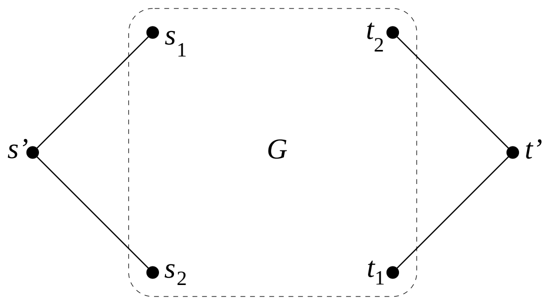


Figure 9.3

Then the cut condition for the two-commodity flow problem implies that the minimum capacity of any $r' - s'$ cut in the extended graph is equal to $d_1 + d_2$. Hence, by the max-flow min-cut theorem, there exists an integer-valued $r' - s'$ flow in the extended graph of value $d_1 + d_2$. This gives x' satisfying (11).

Similarly, the max-flow min-cut theorem implies the existence of a function $x'' : A \to \mathbb{Z}$ satisfying:

$$\begin{aligned} &f(x'', r_1) = d_1, f(x'', s_1) = -d_1, f(x'', r_2) = -d_2, f(x'', s_2) = d_2,\\ &f(x'', v) = 0 \text{ for each other vertex } v,\\ &|x''(a)| \leq c(a) \text{ for each arc } a \text{ of } D. \end{aligned} \tag{12}$$

To see this we extend G with vertices r'' and s'' and edges $\{r'', r_1\}, \{s_1, s''\}$ (both with capacity d_1) and $\{r'', s_2\}, \{r_2, s''\}$ (both with capacity d_2) (cf. Figure 9.4).
After this we proceed as above.

Now consider the vectors

$$x_1 := \tfrac{1}{2}(x' + x'') \text{ and } x_2 := \tfrac{1}{2}(x' - x''). \tag{13}$$

Since $f(x_1, v) = \frac{1}{2}(f(x', v) + f(x'', v))$ for each v, we see from (11) and (12) that x_1 satisfies:

$$f(x_1, r_1) = d_1, f(x_1, s_1) = -d_1, f(x_1, v) = 0 \text{ for all other } v. \tag{14}$$

So x_1 gives a half-integer $r_1 - s_1$ flow in G of value d_1. Similarly, x_2 satisfies:

$$f(x_2, r_2) = d_2, f(x_2, s_2) = -d_2, f(x_2, v) = 0 \text{ for all other } v. \tag{15}$$

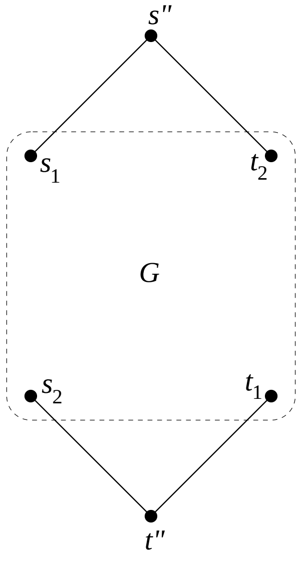


Figure 9.4

So x_2 gives a half-integer $r_2 - s_2$ flow in G of value d_2.

Moreover, x_1 and x_2 together satisfy the capacity constraint, since for each edge a of D:

(16) $$\begin{aligned}|x_1(a)| + |x_2(a)| &= \tfrac{1}{2}|x'(a) + x''(a)| + \tfrac{1}{2}|x'(a) - x''(a)| \\ &= \max\{|x'(a)|, |x''(a)|\} \leq c(a).\end{aligned}$$

(Note that $\frac{1}{2}|\alpha + \beta| + \frac{1}{2}|\alpha - \beta| = \max\{|\alpha|, |\beta|\}$ for all reals α, β.)

So we have a half-integer solution to the two-commodity flow problem. ▮

This proof also directly gives a polynomial-time algorithm for finding a half-integer flow.

The cut condition is not enough to derive an *integer* solution, as is shown by Figure 9.5 (taking all capacities and demands equal to 1).

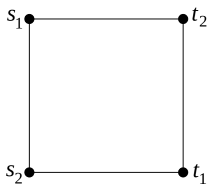


Figure 9.5

Moreover, as mentioned, the undirected integer two-commodity flow problem is NP-complete (Even, Itai, and Shamir [1976]).

However, Rothschild and Whinston [1966] showed that an integer solution exists if the cut condition holds, provided that the following *Euler condition* is satisfied:

(17) $$\begin{aligned}\textstyle\sum_{e\in\delta(v)} c(e) &\equiv 0 \pmod 2 \quad \text{if } v \neq r_1, s_1, r_2, s_2,\\ &\equiv d_1 \pmod 2 \quad \text{if } v = r_1, s_1,\\ &\equiv d_2 \pmod 2 \quad \text{if } v = r_2, s_2.\end{aligned}$$

(Equivalently, the graph obtained from G by replacing each edge e by $c(e)$ parallel edges and by adding d_i parallel edges connecting r_i and s_i ($i = 1, 2$), should be an Eulerian graph.)

Theorem 9.2. *If all capacities and demands are integer and the cut condition and the Euler condition are satisfied, then the undirected two-commodity flow problem has an integer solution.*

Proof. If the Euler condition holds, we can take x' in the proof of Theorem 9.1 so that the following further condition is satisfied:

(18) $$x'(a) \equiv c(a) \pmod 2 \qquad \text{for each } a \in A.$$

To see this, let x' satisfy (11) and let

(19) $$A' := \{a \in A \mid x'(a) \not\equiv c(a) \pmod 2\}.$$

Then each vertex v is incident with an even number δ of arcs in A', since

(20) $$\delta \equiv f(x', v) - f(c, v) \equiv 0 \pmod 2,$$

by (11) and (17). So if $A' \neq \emptyset$ then A' contains an (undirected) circuit. Increasing and decreasing x' by 1 on the arcs along this circuit (depending on whether the arc is forward or backward), we obtain a function again satisfying (11). Repeating this, we finally obtain a function x' satisfying (18).

Similarly, we can take x'' so that

(21) $$x''(a) \equiv c(a) \pmod 2 \qquad \text{for each } a \in A.$$

Conditions (18) and (21) imply that $x'(a) \equiv x''(a) \pmod 2$ for each $a \in A$. Hence $x_1 = \frac{1}{2}(x'+x'')$ and $x_2 = \frac{1}{2}(x' - x")$ are integer vectors. ∎

This proof directly yields a polynomial-time algorithm for finding the integer solution.

Exercises

9.6. Derive from Theorem 9.1 the following *max-biflow min-cut theorem* of Hu: Let $G = (V, E)$ be a graph, let r_1, s_1, r_2, s_2 be distinct vertices, and let $c : E \to \mathbb{Q}_+$ be a capacity function. Then the maximum value of $d_1 + d_2$ so that there exist $r_i - s_i$ flows x_i of value d_i ($i = 1, 2$), together satisfying the capacity constraint, is equal to the minimum capacity of a cut both separating r_1 and s_1 and separating r_2 and s_2.

9.7. Derive from Theorem 9.1 that the cut condition suffices to have a half-integer solution to the undirected k-commodity flow problem (with all capacities and demands integer), if there exist two vertices u and w so that each commodity $\{r_i, s_i\}$ intersects $\{u, w\}$. (Dinits (cf. Adel'son-Vel'skiĭ, Dinits, and Karzanov [1975]).)

9.8. Derive the following from Theorem 9.2. Let $G = (V, E)$ be a Eulerian graph and let r_1, s_1, r_2, s_2 be distinct vertices. Then the maximum number t of pairwise edge-disjoint paths $P_1, \ldots, P_t$, where each P_j connects either r_1 and s_1 or r_2 and s_2, is equal to the minimum cardinality of a cut both separating r_1 and s_1 and separating r_2 and s_2.

9.3. Disjoint paths in acyclic directed graphs

Fortune, Hopcroft, and Wyllie [1980] showed that the vertex-disjoint paths problem is NP-complete for directed graphs, even when fixing the number of paths to $k = 2$.

On the other hand they proved that if D is *acyclic*, then for each fixed k, the k vertex-disjoint paths problem can be solved in polynomial time. (A directed graph is called *acyclic* if it does not contain any directed circuit.)

The algorithm is contained in the proof of the following theorem:

Theorem 9.3. *For each fixed k there exists a polynomial-time algorithm for the k vertex-disjoint paths problem for acyclic directed graphs.*

Proof. Let $D = (V, A)$ be an acyclic digraph and let $r_1, s_1, \ldots, r_k, s_k$ be vertices of D, all distinct. In order to solve the vertex-disjoint paths problem we may assume that each r_i is a source and each s_i is a sink.

Make an auxiliary digraph $D' = (V', A')$ as follows. The vertex set V' consists of all k-tuples $(v_1, \ldots, v_k)$ of distinct vertices of D. In D' there is an arc from $(v_1, \ldots, v_k)$ to $(w_1, \ldots, w_k)$ if and only if there exists an $i \in \{1, \ldots, k\}$ such that:

(22) (i) $v_j = w_j$ for all $j \neq i$;
(ii) (v_i, w_i) is an arc of D;
(iii) for each $j \neq i$ there is no directed path in D from v_j to v_i.

Now the following holds:

(23) D contains k vertex-disjoint directed paths $P_1, \ldots, P_k$ such that P_i runs from r_i to s_i $(i = 1, \ldots, k)$
$\Longleftrightarrow$ D' contains a directed path P from $(r_1, \ldots, r_k)$ to $(s_1, \ldots, s_k)$.

To see $\Longrightarrow$, let P_i follow the vertices $v_{i,0}, v_{i,1}, \ldots, v_{i,t_i}$ for $i = 1, \ldots, k$. So $v_{i,0} = r_i$ and $v_{i,t_i} = s_i$ for each i. Choose $j_1, \ldots, j_k$ such that $0 \leq j_i \leq t_i$ for each i and such that:

(24) (i) D' contains a directed path from $(r_1, \ldots, r_k)$ to $(v_{1,j_1}, \ldots, v_{k,j_k})$,
(ii) $j_1 + \cdots + j_k$ is as large as possible.

Let $I := \{i \mid j_i < t_i\}$. If $I = \emptyset$ we are done, so assume $I \neq \emptyset$. Then by the definition of D' and the maximality of $j_1 + \cdots + j_k$ there exists for each $i \in I$ an $i' \neq i$ such that there is a directed path in D from $v_{i',j_{i'}}$ to v_{i,j_i}. Since $s_{i'}$ is a sink we know that $v_{i',j_{i'}} \neq s_{i'}$ and that hence i' belongs to I. So each vertex in $\{v_{i,j_i} \mid i \in I\}$ is end vertex of a directed path in D starting in another vertex in $\{v_{i,j_i} \mid i \in I\}$. This contradicts the fact that D is acyclic.

To see $\Longleftarrow$ in (23), let P be a directed path from $(r_1, \ldots, r_k)$ to $(s_1, \ldots, s_k)$ in D'. Let P follow the vertices $(v_{1,j}, \ldots, v_{k,j})$ for $j = 0, \ldots, t$. So $v_{i,0} = r_i$ and $v_{i,t} = s_i$ for $i = 1, \ldots, k$. For each $i = 1, \ldots, k$ let P_i be the path in D following $v_{i,j}$ for $j = 0, \ldots, t$, taking repeated vertices only once. So P_i is a directed path from r_i to s_i.

Moreover, $P_1, \ldots, P_k$ are pairwise disjoint. For suppose that P_1 and P_2 (say) have a vertex in common. That is $v_{1,j} = v_{2,j'}$ for some $j \neq j'$. Without loss of generality, $j < j'$ and $v_{1,j} \neq v_{1,j+1}$. By definition of D', there is no directed path in D from $v_{2,j}$ to $v_{1,j}$. However, this contradicts the facts that $v_{1,j} = v_{2,j'}$ and that there exists a directed path in D from $v_{2,j}$ to $v_{2,j'}$. ∎

One can derive from this that for fixed k also the k *arc*-disjoint paths problem is solvable in polynomial time for acyclic directed graphs (Exercise 9.9).

Application 9.4: Routing airplanes. This application extends Application 4.1. The data are similar, except that legal rules now prescribe the exact day of the coming week at which certain airplanes should be at the home basis for maintenance.

Again at Saturday 18.00h the company determines the exact routing for the next week. One can make the same directed graph as in Application 4.1. Now however it is prescribed that some of the paths P_i should start at a certain (c, t) (where c is the city where airplane a_i will be first after Saturday 18.00h) and that they should traverse the arc corresponding to maintenance on a prescribed day of the coming week (for instance Wednesday).

Now if for each airplane a_i which should be home for maintenance next week we can find this path P_i such that it traverses the for that plane required maintenance arc and in such a way that paths found for different airplanes are arc disjoint, then it is easy to see that these paths can be extended to paths $P_1, \ldots, P_n$ such that each arc is traversed exactly once.

As the directed graph D is acyclic, the problem can be solved with the algorithm described in the proof of Theorem 9.3, provided that the number of airplanes that should be home for maintenance the coming week is not too large.

Exercises

9.9. Derive from Theorem 9.3 that for each fixed k the k arc-disjoint paths problem is solvable in polynomial time for acyclic directed graphs.

9.10. Show that for fixed k, the following problem is solvable in polynomial time:

(25) given: an acyclic directed graph $D = (V, A)$, pairs $r_1, s_1, \ldots, r_k, s_k$ of vertices, and subsets $A_1, \ldots, A_k$ of A;
find: pairwise arc-disjoint directed paths $P_1, \ldots, P_k$, where P_i runs from r_i to s_i and traverses only arcs in A_i $(i = 1, \ldots, k)$.

9.4. Vertex-disjoint paths in planar graphs

Finding disjoint paths in *planar* graphs is of interest not only for planar communication or transportation networks, but especially also for the design of VLSI-circuits. The routing of wires should follow certain channels on layers of the chip. On each layer, these channels form a planar graph.

Unfortunately, even for planar graphs disjoint paths problems are in general hard. However, for some special cases, polynomial-time algorithms have been found. Such algorithms can be used, for example, as subroutines when solving any hard problem by decomposition. In Sections 9.4 and 9.5 we discuss some of these algorithms.

Let $G = (V, E)$ be a planar graph, embedded in the plane $\mathbb{R}^2$ and let $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ be pairwise disjoint pairs of vertices. Robertson and Seymour [1986] observed that there is an easy greedy-type algorithm for the vertex-disjoint paths problem if all vertices $r_1, s_1, \ldots, r_k, s_k$ belong to the boundary of one face I of G. That is, there exists a polynomial-time algorithm for the following problem:[20]

(26) given: a planar graph $G = (V, E)$ embedded in $\mathbb{R}^2$, a face I of G, pairs $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ of vertices on $\mathrm{bd}(I)$,
find: pairwise vertex-disjoint paths $P_1, \ldots, P_k$ in G, where P_i connects r_i and s_i $(i = 1, \ldots, k)$.

In fact, we may assume without loss of generality that I is the unbounded face.

Let us first describe the simple intuitive idea of the method, by explaining the recursive step in the 'ideal' case where G is connected and where $\mathrm{bd}(I)$ is a simple circuit.

We say that $\{r, s\}$ and $\{r', s'\}$ cross (around I) if r, r', s, s' are distinct and occur in this order cyclically around $\mathrm{bd}(I)$, clockwise or anti-clockwise (see Figure 9.6).

If any $\{r_i, s_i\}$ and $\{r_j, s_j\}$ cross around I (for some $i \neq j$), problem (26) clearly has no solution. So we may assume that no pair of commodities crosses. This implies that there exists an i so that at least one of the $r_i - s_i$ paths along $\mathrm{bd}(I)$ does not contain any r_j or s_j for $j \neq i$: just choose i so that the shortest $r_i - s_i$ path along $\mathrm{bd}(I)$ is shortest among all $i = 1, \ldots, k$.

Without loss of generality, $i = k$. Let Q be the shortest $r_k - s_k$ path along $\mathrm{bd}(I)$. Delete from G all vertices in Q, together with all edges incident with them. Denote the new graph by G'. Next solve

[20] $\mathrm{bd}(I)$ denotes the boundary of I.

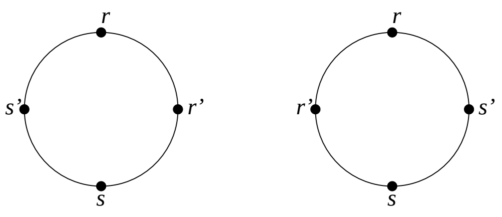


Figure 9.6

the vertex-disjoint paths problem for input G', $\{r_1, s_1\}, \ldots, \{r_{k-1}, s_{k-1}\}$. If this gives a solution $P_1, \ldots, P_{k-1}$, then $P_1, \ldots, P_{k-1}$, Q forms a solution to the original problem (trivially).

If the reduced problem turns out to have no solution, then the original problem also has no solution. This follows from the fact that if $P_1, \ldots, P_{k-1}$, P_k would be a solution to the original problem, we may assume without loss of generality that $P_k = Q$, since we can 'push' P_k 'against' the border $\mathrm{bd}(I)$. Hence $P_1, \ldots, P_{k-1}$ would form a solution to the reduced problem.

Although this might give a suggestive sketch of the algorithm, it is not completely accurate, since the ideal situation need not be preserved throughout the iteration process. Even if we start with a highly connected graph, after some iterations the reduced graph might have cut vertices or be disconnected. So one should be more precise.

Let us call a sequence $(v_1, \ldots, v_n)$ of vertices of a connected planar graph G a *border sequence* if it is the sequence of vertices traversed when following the boundary of G clockwise. Thus the graph in Figure 9.7 has border sequence $(a, b, c, d, e, c, f, c, g, b)$.

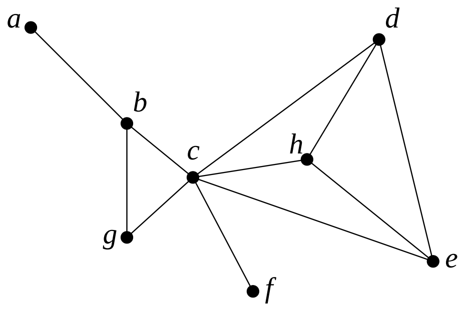


Figure 9.7

In fact, each cyclic permutation of a border sequence is again a border sequence.

Note that no border sequence will contain $\ldots r \ldots s \ldots r \ldots s \ldots$ for any two distinct vertices. Hence for any two vertices r and s on the boundary of G there is a unique sequence

(27) $$P(r, s) = (r, w_1, \ldots, w_t, s)$$

with the properties that $P(r, s)$ is part of a border sequence of G and that $w_1, \ldots, w_t$ all are distinct from r and s. Trivially, the vertices in $P(r, s)$ contain a simple $r - s$ path.

We say that two disjoint pairs $\{r, s\}$ and $\{r', s'\}$ cross (around G) if $\ldots r \ldots r' \ldots s \ldots s' \ldots$ or $\ldots r \ldots s' \ldots s \ldots r' \ldots$ occur in some border sequence of G. So the following *cross-freeness condition* is a necessary condition for (26) to have a solution:

(28) No two disjoint commodities $\{r_i, s_i\}, \{r_j, s_j\}$ cross (around the same component of G).

Now the algorithm can be described more precisely as follows. First check the cross-freeness condition. If it is violated, (26) has no solution. If it is satisfied, apply the following iterative step:

(29) Check for each $i = 1, \ldots, k$ if r_i and s_i belong to the same component of G. If not, the problem has no solution.
If so, choose $i \in \{1, \ldots, k\}$ for which the shortest among $P(r_i, s_i)$ and $P(s_i, r_i)$ is as short as possible. Without loss of generality, $i = k$ and $P(r_k, s_k)$ is shortest. Take for P_k any $r_k - s_k$ path using the vertices in $P(r_k, s_k)$ only.
If $k = 1$, stop. If $k > 1$, let G' be the graph obtained from G by deleting all vertices occurring in $P(r_k, s_k)$. Repeat this iterative step for G' and $\{r_1, s_1\}, \ldots, \{r_{k-1}, s_{k-1}\}$.
If it gives a solution $P_1, \ldots, P_{k-1}$, then $P_1, \ldots, P_{k-1}, P_k$ is a solution to the original problem. If it gives no solution, the original problem has no solution.

We leave it as a (technical) exercise to show the correctness of this algorithm. (The correctness can be derived also from the proof of Theorem 9.4 below.) It clearly is a polynomial-time method. Recently, Ripphausen-Lipa, Wagner, and Weihe [1997] found a linear-time algorithm.

Moreover, the method implies a characterization by means of a cut condition for the existence of a solution to (26). A *simple closed curve* C in $\mathbb{R}^2$ is by definition a one-to-one continuous function from the unit circle to $\mathbb{R}^2$. We will identify the function C with its image.

We say that C *separates* the pair $\{r, s\}$ if each curve connecting r and s intersects C. Now the following *cut condition* clearly is necessary for the existence of a solution to the vertex-disjoint paths problem in planar graphs:

(30) each simple closed curve in $\mathbb{R}^2$ intersects G at least as often as it separates pairs $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$.

Robertson and Seymour [1986] showed with this method:

Theorem 9.4. *Let $G = (V, E)$ be a planar graph embedded in $\mathbb{R}^2$ and let $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ be pairs of vertices on the boundary of G. Then there exist pairwise vertex-disjoint paths $P_1, \ldots, P_k$ where P_i connects r_i and s_i ($i = 1, \ldots, k$) if and only if the cross-freeness condition* (28) *and the cut condition* (30) *hold.*

Proof. Necessity of the conditions is trivial. We show sufficiency by induction on k, the case $k = 0$ being trivial. Let $k > 1$ and let (28) and (30) be satisfied. Suppose paths $P_1, \ldots, P_k$ as required do not exist. Trivially, $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ are pairwise disjoint (otherwise there would exist a simple closed curve C with $|C \cap G| = 1$ and intersecting two commodities, thus violating the cut condition).

The induction is based on the iterative step (29). To simplify the argument, we first show that we may assume that G is 2-connected.

First, we may assume that G is connected, as we can decompose G into its components. (If some r_i and s_i would belong to different components, there trivially exists a closed curve C violating the cut condition.)

Knowing that G is connected, the case $k = 1$ is trivial. So we may assume that $k \geq 2$. Suppose G contains a cut vertex v. We may assume that each component of $G - v$ intersects $\{r_1, s_1, \ldots, r_k, s_k\}$ (otherwise we could delete it from G without violating the cut condition). This implies that we can extend G planarly by an edge e connecting some vertices u' and u'' in different components of $G - v$, in such a way that $u' \in \{r_{i'}, s_{i'}\}$ and $u'' \in \{r_{i''}, s_{i''}\}$ for some $i' \neq i''$ and that $r_1, s_1, \ldots, r_k, s_k$ are still on the boundary of $G \cup e$. The cut condition holds for $G \cup e$ (a fortiori), but pairwise vertex-disjoint $r_i - s_i$ paths ($i = 1, \ldots, k$) do not exist in $G \cup e$ (since we cannot make use of edge e, as $i' \neq i''$). Repeating this we end up with a 2-connected graph.

If G is 2-connected, the boundary of G is a simple circuit. Now we apply the iterative step (29). That is, we find, without loss of generality, that the simple path $P(r_k, s_k)$ from r_k to s_k clockwise along the boundary of G does not contain any $r_1, s_1, \dots, r_{k-1}, s_{k-1}$. Let P_k be the corresponding $r_k - s_k$ path.

Again, let G' be the graph obtained from G by deleting all vertices in P_k, together with all edges incident with them. Let I and I' denote the unbounded faces of G and G', respectively (we take I and I' as *open* regions). So $I \subseteq I'$.

Now G' does not contain pairwise vertex-disjoint $r_i - s_i$ paths ($i = 1, \dots, k-1$), since by assumption G does not contain pairwise vertex-disjoint $r_i - s_i$ paths ($i = 1, \dots, k$). Hence, by the induction hypothesis, there exists a simple closed curve C with $|C \cap G'|$ smaller than the number of pairs $\{r_1, s_1\}, \dots, \{r_{k-1}, s_{k-1}\}$ separated by C. We may assume that C traverses each of the connected regions I', I and $I' \setminus I$ at most once. That is, each of $C \cap I'$, $C \cap I$ and $C \cap (I' \setminus I)$ is connected (possibly empty).

If $C \cap (I' \setminus I)$ is empty, then $C \cap G = C \cap G'$ and hence C violates the cut condition also for G. If $C \cap I$ is empty, then C does not separate any $\{r_i, s_i\}$ except for those intersected by C. Then C cannot violate the cut condition for G'.

If both $C \cap I$ and $C \cap (I' \setminus I)$ are nonempty, we may assume that $|C \cap G| = |C \cap G'| + 1$ and that C separates $\{r_k, s_k\}$ (since each face of G contained in I' is incident with at least one vertex on P_k). It follows that C violates the cut condition for G. ∎

Application 9.5: VLSI-routing. The VLSI-routing problem asks for the routes that wires should make on a chip so as to connect certain pairs of pins and so that wires connecting different pairs of pins are disjoint.

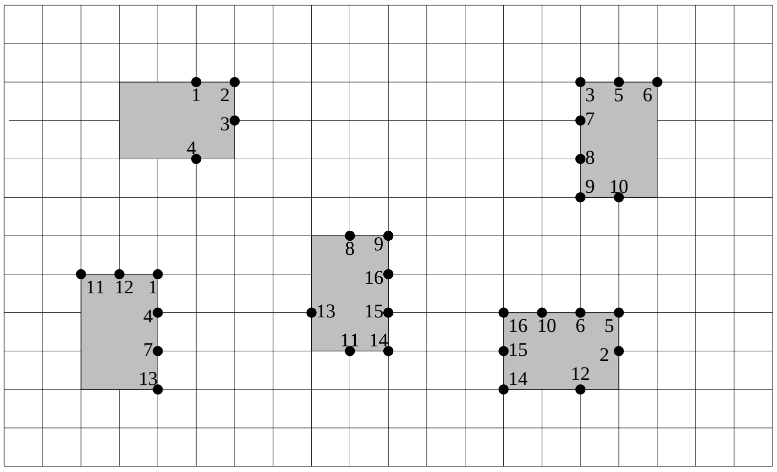


Figure 9.8

Since the routes that the wires potentially can make form a graph, the problem to be solved can be modeled as a disjoint paths problem. Consider an example of such a problem as in Figure 9.8 — relatively simple, since generally the number of pins to be connected is of the order of several thousands. The grey areas are 'modules' on which the pins are located. Points with the same label should be connected.

In the example, the graph is a 'grid graph', which is typical in VLSI-design since it facilitates the manufacturing of the chip and it ensures a certain minimum distance between disjoint wires. But even for such graphs the disjoint paths problem is NP-complete.

Now the following two-step approach was proposed by Pinter [1983]. First choose the 'homotopies' of the wires; for instance like in Figure 9.9. That is, for each i one chooses a curve C_i in the plane connecting the two vertices i, in such a way that they are pairwise disjoint, and such that the modules are not traversed (Figure 9.9).

Second, try to find disjoint paths $P_1, \dots, P_k$ in the graph such that P_i is homotopic to C_i, in the space obtained from the plane by taking out the rectangles forming the modules; that is, the paths P_i should be

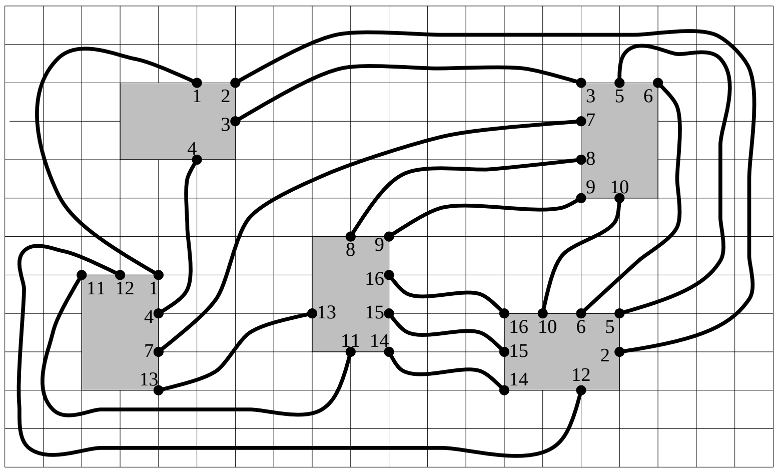


Figure 9.9

obtained from the curves C_i by shifting C_i over the surface, but not over any module, fixing the end points of the curve. In Figure 9.10 such a solution is given.

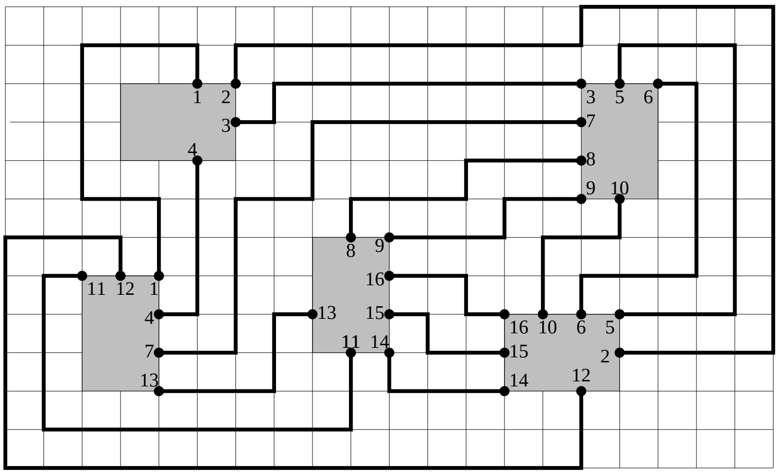


Figure 9.10

It was shown by Leiserson and Maley [1985] that this second step can be performed in polynomial time. So the hard part of the problem is the first step: finding the right topology of the layout.

Cole and Siegel [1984] proved a Menger-type cut theorem characterizing the existence of a solution in the second step. That is, if there is no solution for the disjoint paths problem given the homotopies, there is an 'oversaturated' cut: a curve D connecting two holes in the plane and intersecting the graph less than the number of times D necessarily crosses the curves C_i.

This can be used in a heuristic practical algorithm for the VLSI-routing problem: first guess the homotopies of the solution; second try to find disjoint paths of the guessed homotopies; if you find them you can stop; if you don't find them, the oversaturated cut will indicate a bottleneck in the chosen homotopies; amend the bottleneck and repeat.

Similar results hold if one wants to pack trees instead of paths (which is generally the case at VLSI-design), and the result can be extended to any planar graph (Schrijver [1991]). As a theoretical consequence one has that for each fixed number of modules, the planar VLSI-routing problem can be solved in polynomial time.

Exercises

9.11. Extend the algorithm and Theorem 9.4 to the directed case.

9.12. Extend the algorithm and Theorem 9.4 to the following *vertex-disjoint trees problem*:

(31) given: a planar graph $G = (V, E)$, sets $R_1, \ldots, R_k$ of vertices on the boundary of G,
find: pairwise vertex-disjoint subtrees $T_1, \ldots, T_k$ of G so that T_i covers R_i ($i = 1, \ldots, k$).

9.13. Extend the algorithm and Theorem 9.4 to the following problem:

(32) given: a planar graph $G = (V, E)$, pairs $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ of vertices on the boundary of G, subgraphs $G_1, \ldots, G_k$ of G,
find: pairwise vertex-disjoint paths $P_1, \ldots, P_k$ where P_i connects r_i and s_i and where P_i is in G_i ($i = 1, \ldots, k$).

9.14. (i) Reduce the edge-disjoint paths problem where all commodities are on the boundary of a planar graph so that the cross-freeness condition is satisfied, to the vertex-disjoint paths problem(26).

(ii) Show that the cut condition (7) is sufficient in this case of the edge-disjoint paths problem.

9.5. Edge-disjoint paths in planar graphs

The trivially necessary cross-freeness condition for the commodities if they are on the boundary of a planar graph, turned out to be of great help in handling the *vertex*-disjoint paths problem: it gives an ordering of the commodities, allowing us to handling them one by one.

As we saw in Exercise 9.14, the edge-disjoint analogue can be handled in the same way if the cross-freeness condition holds. In that case, the cut condition (7) is again sufficient. However, Figure 9.5 shows that without cross-freeness, the cut condition is not sufficient. That simple example shows that we may not hope for many other interesting cases where the cut condition is sufficient.

In fact, the complexity of the edge-disjoint paths problem for planar graphs with all commodities on the boundary, is open. Therefore, we put:

Research problem. Is the undirected edge-disjoint paths problem polynomially solvable for planar graphs with all commodities on the boundary? Is it NP-complete?

Okamura and Seymour [1981] showed that the problem is polynomially solvable if we pose the following *Euler condition*:

(33) the graph $(V, E \cup \{\{r_1, s_1\}, \ldots, \{r_k, s_k\}\})$ is Eulerian.

(We have parallel edges if some $\{r_i, s_i\}$ coincide or form an edge of G.) Moreover, they showed that with the Euler condition, the cut condition is a sufficient condition. (Thus we have an analogue to Rothschild and Whinston's theorem (Theorem 9.2).)

Theorem 9.5 (Okamura-Seymour theorem). *Let $G = (V, E)$ be a planar graph and let $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ be pairs of vertices on the boundary of G such that the Euler condition* (33) *holds. Then the edge-disjoint paths problem has a solution if and only if the cut condition holds.*

Proof. Necessity of the cut condition is trivial. Sufficiency is shown by induction on $|V| + |E|$, the case $|V| + |E| = 0$ being trivial. Let the cut condition be satisfied. We first show that we may assume that G is 2-connected.

If G is disconnected, we can deal with the components separately. If G is not 2-connected, consider a cut vertex v. We may assume that for no i the vertices r_i and s_i belong to different components of $G-v$, since otherwise we can replace the commodity $\{r_i, s_i\}$ by the two commodities $\{r_i, v\}$ and $\{v, s_i\}$ without violating the Euler or cut condition. For any component K of $G-v$ consider the graph induced by $K \cup v$. Again, the Euler and cut conditions are satisfied (with respect to those commodities fully contained in $K \cup v$). So by the induction hypothesis we know that paths as required exist in $K \cup v$. As this is the case for each component of $G-v$, we have paths as required in G.

So we may assume that G is 2-connected. For each $X \subseteq V$, let $\rho(X)$ be the set of all indices i for which X contains exactly one of r_i and s_i. Call a cut $\delta(X)$ *tight* if $|\delta(X)| = |\rho(X)|$. Choose an edge e on the boundary of G, say $e = \{u, w\}$ and let F be the bounded face incident with e.

Case 1. *Edge e is not contained in any tight cut.* Delete from G all edges incident with F, yielding graph G', say. Then G' with $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ again satisfies the Euler condition and the cut condition. A solution in G' gives directly a solution in G.

Case 2. *Edge e is contained in some tight cut.* Then there exists a tight cut $\delta(X)$ containing e so that both $< X >$ and $< V \setminus X >$ are connected (as follows from Exercise 9.4 by deleting e). Choose such a cut $\delta(X)$, with the additional properties that $w \in X$ and that X has smallest intersection with the boundary of G.

Since $|\rho(X)| = |\delta(X)| \geq 2$, we know $\rho(X) \neq \emptyset$. Without loss of generality, if $i \in \rho(X)$ then $r_i \notin X$, $s_i \in X$. Choose $i \in \rho(X)$ so that r_i is as close as possible to u when following the part of the boundary of G outside X. We may assume that $i = 1$.

Now delete e from G and replace the commodity $\{r_1, s_1\}$ by the two commodities $\{r_1, u\}$ and $\{w, s_1\}$. The new graph and commodities trivially satisfy the Euler condition. To see that they also satisfy the cut condition, suppose to the contrary that $Y \subseteq V$ is such that

(34) $$|\delta'(Y)| < |\rho'(Y)|$$

(where δ' and ρ' are the parameters for the converted structure) and so that both $< Y >$ and $< V \setminus Y >$ are connected (in $G-e$). Without loss of generality, $r_1 \notin Y$. By the Euler condition, (34) implies $|\delta'(Y)| \leq |\rho'(Y)| - 2$. Since

(35) $$|\rho'(Y)| \geq |\delta'(Y)| + 2 \geq |\delta(Y)| + 1 \geq |\rho'(Y)| - 1,$$

we know $|\rho'(Y)| > |\rho(Y)|$ and $|\delta(Y)| = |\rho(Y)|$ (using the Euler condition).

As $|\rho'(Y)| > |\rho(Y)|$, we know $s_1 \notin Y$, and at least one of u and w belongs to Y. By the choice of $i = 1$, there is no pair $\{r_j, s_j\}$ intersecting both $X \setminus Y$ and $Y \setminus X$. This implies:

(36) $$|\rho(X \cap Y)| + |\rho(X \cup Y)| = |\rho(X)| + |\rho(Y)|.$$

Moreover,

(37) $$|\delta(X \cap Y)| + |\delta(X \cup Y)| \leq |\delta(X)| + |\delta(Y)|.$$

Since $|\delta(X)| = |\rho(X)|$, $|\delta(Y)| = |\delta(X \cap Y)| \geq |\rho(X \cap Y)|$, and $|\delta(X \cup Y)| \geq |\rho(X \cup Y)|$, we know that $|\rho(X \cap Y)| = |\delta(X \cap Y)|$ should hold. This implies equality in (37). Hence there is no edge connecting $X \setminus Y$ and $Y \setminus X$. Therefore, $w \in Y$, and hence $\delta(X \cap Y)$ is a tight cut containing e. However, $s_1 \notin X \cap Y$, contradicting the minimality of the intersection of X with the boundary of G.

So the converted structure satisfies the cut condition. Hence, by induction, there exist pairwise edge-disjoint paths $P_1', P_1'', P_2, \ldots, P_k$ where P_1' connects r_1 and u, P_1'' connects w and s_1, and P_j connects r_j and s_j ($j = 2, \ldots, k$). Taking $P_1 := P_1' e P_1''$ gives a solution to the original edge-disjoint paths problem. ▮

Clearly, this method gives a polynomial-time algorithm for finding the paths, since we can determine a minimum-cardinality cut containing e' and e'', for any pair of edges e', e'' on the boundary of G (cf. Exercise 9.16).

Becker and Mehlhorn [1986] and Matsumoto, Nishizeki, and Saito [1985] gave implementations with running time of order $O(|E|^2)$. Recently, Wagner and Weihe [1995] found a linear-time algorithm.

Exercises

9.15. Let $G = (V, E)$ be a finite subgraph of the rectangular grid graph in $\mathbb{R}^2$, such that each bounded face of G is a square of area 1. Let $\{r_1, s_1\}, \ldots, \{r_k, s_k\}$ be pairs of vertices on the boundary of G such that each vertex of $(V, E \cap \{\{r_1, s_1\}, \ldots, \{r_k, s_k\}\})$ has degree even and at most 4. A cut is called a *1-bend cut* if it is the set of edges crossed by the union of some horizontal and some vertical half-line with one common end vertex.

Show that the cut condition holds whenever it holds for all 1-bend cuts.

9.16. Let G be a planar graph and let e' and e'' be edges on the boundary of G. Reduce the problem of finding a minimum-cardinality cut containing e' and e'' to a shortest path problem.

9.6. A column generation technique for multicommodity flows

The fractional multicommodity flow problem (1) asks for flows $x_1, \ldots, x_k$ of given values $d_1, \ldots, d_k$ such that the total amount of flow through any arc e does not exceed the capacity of e. So it amounts to finding a solution to the following system of linear inequalities in the $k|E|$ variables $x_i(e)$ $(i = 1, \ldots, k;\ e \in E)$:

$$(38)\quad \begin{array}{lll}
\text{(i)} & \displaystyle\sum_{e\in\delta^{\text{out}}(v)} x_i(e) - \sum_{e\in\delta^{\text{in}}(v)} x_i(e) = 0 & (i = 1, \ldots, k;\ v \in V, v \neq r_i, s_i), \\
\text{(ii)} & \displaystyle\sum_{e\in\delta^{\text{out}}(r_i)} x_i(e) - \sum_{e\in\delta^{\text{in}}(r_i)} x_i(e) = d_i & (i = 1, \ldots, k), \\
\text{(iii)} & \displaystyle\sum_{i=1}^{k} x_i(e) \leq c(e) & (e \in E), \\
\text{(iv)} & x_i(e) \geq 0 & (i = 1, \ldots, k;\ e \in E).
\end{array}$$

Thus any linear programming method can solve the multicommodity flow problem. In particular, the problem is solvable in polynomial time.

Since for each fixed $i = 1, \ldots, k$, a solution x_i to (38) is an $r_i - s_i$ flow, we can decompose x_i as a nonnegative combination of $r_i - s_i$ paths. That is, there exist $r_i - s_i$ paths $P_{i1}, \ldots, P_{in_i}$ and nonnegative reals $z_{i1}, \ldots, z_{in_i}$ satisfying:

$$(39)\quad \begin{array}{lll}
\text{(i)} & \displaystyle\sum_{j=1}^{n_i} z_{ij}\mathcal{X}^{P_{ij}}(e) = x_j(e) & (e \in E), \\
\text{(ii)} & \displaystyle\sum_{j=1}^{n_i} z_{ij} = d_i. &
\end{array}$$

Here $\mathcal{X}^P$ denotes the *incidence vector* of P in $\mathbb{Q}^E$, that is, $\mathcal{X}^P(e) = 1$ if P traverses e, and $= 0$ otherwise.

Hence the multicommodity flow problem amounts to finding paths P_{ij} and nonnegative reals z_{ij}, where P_{ij} is an $r_i - s_i$ path, such that:

$$(40)\quad \begin{array}{lll}
\text{(i)} & \displaystyle\sum_{j=1}^{n_i} z_{ij} = d_i & (i = 1, \ldots, k), \\
\text{(ii)} & \displaystyle\sum_{i=1}^{k} \sum_{j=1}^{n_i} z_{ij}\mathcal{X}^{P_{ij}}(e) \leq c(e) & (e \in E).
\end{array}$$

This formulation applies to both the directed and the undirected problems.

Solving (40) again amounts to solving a system of linear inequalities, albeit with an enormous number of variables: one variable for each $i = 1, \ldots, k$ and each $r_i - s_i$ path.

Ford and Fulkerson [1958] showed that this large number of variables can be avoided when solving the problem with the simplex method. The variables can be handled implicitly by using a *column generation technique* as follows.

First convert the problem to a maximization problem. To this end, add, for each $i = 1, \ldots, k$, a vertex r'_i and an arc $r'_i r_i$, with capacity equal to d_i. Then we can delete the constraint (40)(i), and maximize $\sum_{i,j} z_{ij}$ over the remaining constraints (replacing r_i by r'_i). If the maximum value is equal to $\sum_i d_i$ we have a solution to (40). If the maximum value is less, then (40) has no nonnegative solution z_{ij}.

Having this reduction, we see that the problem is equivalent to the following LP-problem. Let $\mathcal{P}$ be the collection of all $r_i - s_i$ paths for all $i = 1, \ldots, k$. Then:

(41)
$$\begin{array}{lll} \text{maximize:} & \displaystyle\sum_{P \in \mathcal{P}} z_P & \\ \text{subject to:} & \text{(i)} \quad \displaystyle\sum_{P \in \mathcal{P}} z_P \mathcal{X}^P(e) \leq c(e) & (e \in E), \\ & \text{(ii)} \quad z_P \geq 0 & (P \in \mathcal{P}). \end{array}$$

When solving (41) with the simplex method we first should add a slack variable z_e for each $e \in E$. Thus if A denotes the $E \times \mathcal{P}$ matrix with the incidence vectors of all paths in $\mathcal{P}$ as its columns (in some order) and w is the vector in $\mathbb{R}^{\mathcal{P}} \times \mathbb{R}^E$ with $w_P = 1$ $(P \in \mathcal{P})$ and $w_e = 0$ $(e \in E)$, we solve:

(42)
$$\begin{array}{ll} \text{maximize:} & w^T z \\ \text{subject to:} & [A\ I]z = c, \\ & z \geq 0. \end{array}$$

Now each simplex tableau is completely determined by the set of variables in the current basis. So knowing subsets $\mathcal{P}'$ of $\mathcal{P}$ and E' of E, giving the indices of variables in the basis, is enough to know implicitly the whole tableau. Note that $|\mathcal{P}'| + |E'| = E$. So although the tableau is exponentially large, it can be represented in a concise way.

Let B be the matrix consisting of those columns of $[A\ I]$ corresponding to $\mathcal{P}'$ and E'. So the rows of B are indexed by E and the columns by $\mathcal{P}' \cup E'$. The basic solution corresponding to B is easily computed: the vector $B^{-1}c$ gives the values for z_P if $P \in \mathcal{P}'$ and for z_e if $e \in E'$, while we set $z_P := 0$ if $P \notin \mathcal{P}'$ and $z_e := 0$ if $e \notin E'$. (Initially, $B = I$, that is $\mathcal{P}' = \emptyset$ and $E' = E$, so that $z_P = 0$ for all $P \in \mathcal{P}$ and $z_e = c(e)$ for all $e \in E$.)

Now we should describe pivoting (that is, finding variables leaving and entering the basis) and checking optimality. Interestingly, it turns out that this can be done by solving a set of shortest path problems.

First consider the dual variable corresponding to an edge e. It has value (in the current tableau):

(43)
$$w_B B^{-1} \varepsilon_e - w_e = w_B (B^{-1})_e$$

where as usual w_B denotes the part of vector w corresponding to B (that is, corresponding to $\mathcal{P}'$ and E') and where ε_e denotes the e-th unit basis vector in $\mathbb{R}^E$ (which is the column corresponding to e in $[A\ I]$). Note that the columns of B^{-1} are indexed by E; then $(B^{-1})_e$ is the column corresponding to e. Note also that $w_e = 0$ by definition.

Similarly, the dual variable corresponding to a path P in $\mathcal{P}$ has value:

(44)
$$w_B B^{-1} \mathcal{X}^P - w_P = [\sum_{e \in P} w_B (B^{-1})_e] - 1.$$

(Note that $\mathcal{X}^P$ is the column in $[A\ I]$ corresponding to P.)

In order to pivot, we should identify a negative dual variable. To this end, we first check if (43) is negative for some edge e. If so, we choose such an edge e and take z_e as the variable entering the basis. Selecting the variable leaving the basis now belongs to the standard simplex routine. We only have to consider that part of the tableau corresponding to $\mathcal{P}'$, E' and e. We select an element f in $\mathcal{P}' \cup E'$ for which the quotient $z_f/(B^{-1})_{fe}$ has positive denominator and is as small as possible. Then z_f is the variable leaving the basis.

Suppose next that (43) is nonnegative for each edge e. We consider $w_B(B^{-1})_e$ as the length $l(e)$ of e. Then for any path P,

$$\sum_{e\in P} w_B(B^{-1})_e \tag{45}$$

is equal to the length $\sum_{e\in P} l(e)$ of P. Hence, finding a dual variable (44) of negative value is the same as finding a path in $\mathcal{P}$ of length less than 1.

Such a path can be found by applying any shortest path algorithm: for each $i = 1,\ldots,k$, we find a shortest $r_i - s_i$ path (with respect to l). If each of these shortest paths has length at least 1, we know that all dual variables have nonnegative value, and hence the current basic solution is optimum.

If we find some $r_i - s_i$ path P of length less than 1, we choose z_P as variable entering the basis. Again selecting a variable leaving the basis is standard: we select an element f in $\mathcal{P}' \cup E'$ for which the quotient $z_f/(B^{-1}\mathcal{X}^P)_f$ has positive denominator and is as small as possible.

This describes pivoting. In order to avoid "cycling", we apply a lexicographic rule for selecting the variable leaving the basis. We order the edges of G arbitrarily. Now in case there is a tie in selecting the $f \in \mathcal{P}' \cup E'$ for which the corresponding quotient is as small as possible, we choose the $f \in \mathcal{P}' \cup E'$ for which the vector

$$\begin{aligned} (B^{-1})_f/(B^{-1})_{fe} &\quad \text{(if } e \text{ enters the basis)},\\ (B^{-1})_f/(B^{-1}\mathcal{X}^P)_f &\quad \text{(if } P \text{ enters the basis)}, \end{aligned} \tag{46}$$

is lexicographically as small as possible. In Exercise 9.17 we will see that this avoids cycling.

Exercises

9.17. (i) Apply the lexicographic rule above, and consider a simplex tableau, corresponding to $\mathcal{P}'$ and E' say. Show that for each $f \in \mathcal{P}' \cup E'$: if $z_f = 0$ then the first nonzero entry in the vector $(B^{-1})_f$ is positive. (Use induction on the number of pivot steps performed.)

(ii) Derive from (i) that, when applying the lexicographic rule, at each pivot iteration, if the objective value of the solution does not increase, then the vector $w_B B^{-1}$ increases lexicographically.

(iii) Derive that the lexicographic rule leads to termination of the method.

9.18. Modify the column generation technique to solve the following problem: given a directed graph $G = (V,E)$, a capacity function $c : E \to \mathbb{Q}_+$, commodities $(r_1,s_1),\ldots,(r_k,s_k)$ and 'profits' $p_1,\ldots,p_k \in \mathbb{Q}_+$, find vectors $x_1,\ldots,x_k$ in $\mathbb{Q}^E$ and rationals $d_1,\ldots,d_k$ so that:

$$\begin{aligned} &\text{(i) } x_i \text{ is an } r_i - s_i \text{ flow of value } d_i \ (i = 1,\ldots,k),\\ &\text{(ii) } \sum_{i=1}^{k} x_i(e) \le c(e) \ (e \in E),\\ &\text{(iii) } \sum_{i=1}^{k} p_i d_i \text{ is as large as possible.} \end{aligned} \tag{47}$$

9.19. Let P_{ij} and $z_{ij} > 0$ form a solution to the undirected form of (40) and let $W \subseteq V$ be so that the capacity of $\delta_E(W)$ is equal to the demand of $\delta_R(W)$. Show that each P_{ij} intersects $\delta_E(W)$ at most once.

9.20. Show that if the multicommodity flow problem has no solution, then Ford and Fulkerson's column generation technique yields a length function l violating (9).

10. Matroids

10.1. Matroids and the greedy algorithm

Let $G = (V, E)$ be a connected undirected graph and let $w : E \to \mathbb{Z}$ be a 'weight' function on the edges. In Section 1.4 we saw that a minimum-weight spanning tree can be found quite straightforwardly with Kruskal's so-called *greedy algorithm.*

The algorithm consists of selecting successively edges $e_1, e_2, \ldots, e_r$. If edges $e_1, \ldots, e_k$ have been selected, we select an edge $e \in E$ so that:

(1) (i) $e \not\in \{e_1, \ldots, e_k\}$ and $\{e_1, \ldots, e_k, e\}$ is a forest,

(ii) $w(e)$ is as small as possible among all edges e satisfying (i).

We take $e_{k+1} := e$. If no e satisfying (1)(i) exists, that is, if $\{e_1, \ldots, e_k\}$ forms a spanning tree, we stop, setting $r := k$. Then $\{e_1, \ldots, e_r\}$ is a spanning tree of minimum weight.

By replacing 'as small as possible' in (1)(ii) by 'as large as possible', one obtains a spanning tree of *maximum* weight.

It is obviously not true that such a greedy approach would lead to an optimal solution for any combinatorial optimization problem. We could think of such an approach to find a matching of maximum weight. Then in (1)(i) we replace 'forest' by 'matching' and 'small' by 'large'. Application to the weighted graph in Figure 10.1 would give $e_1 = cd, e_2 = ab$.

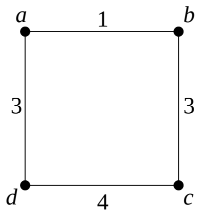


Figure 10.1

However, ab and cd do not form a matching of maximum weight.

It turns out that the structures for which the greedy algorithm *does* lead to an optimal solution, are the *matroids.* It is worth studying them, not only because it enables us to recognize when the greedy algorithm applies, but also because there exist fast algorithms for 'intersections' of two different matroids.

The concept of matroid is defined as follows. Let X be a finite set and let $\mathcal{I}$ be a collection of subsets of X. Then the pair $(X, \mathcal{I})$ is called a *matroid* if it satisfies the following conditions:

(2) (i) $\emptyset \in \mathcal{I}$,

(ii) if $Y \in \mathcal{I}$ and $Z \subseteq Y$ then $Z \in \mathcal{I}$,

(iii) if $Y, Z \in \mathcal{I}$ and $|Y| < |Z|$ then $Y \cup \{x\} \in \mathcal{I}$ for some $x \in Z \setminus Y$.

For any matroid $M = (X, \mathcal{I})$, a subset Y of X is called *independent* if Y belongs to $\mathcal{I}$, and *dependent* otherwise.

Let $Y \subseteq X$. A subset B of Y is called a *basis* of Y if B is an inclusionwise maximal independent subset of B. That is, for any set $Z \in \mathcal{I}$ with $B \subseteq Z \subseteq Y$ one has $Z = B$.

It is not difficult to see that condition (2)(iii) is equivalent to:

(3) for any subset Y of X, any two bases of Y have the same cardinality.

(Exercise 10.1.) The common cardinality of the bases of a subset Y of X is called the *rank* of Y, denoted by $r_M(Y)$.

We now show that if $G = (V, E)$ is a graph and $\mathcal{I}$ is the collection of forests in G, then $(E, \mathcal{I})$ indeed is a matroid. Conditions (2)(i) and (ii) are trivial. To see that condition (3) holds, let $E' \subseteq E$. Then, by definition, each basis Y of E' is an inclusionwise maximal forest contained in E'. Hence Y forms a spanning tree in each component of the graph (V, E'). So Y has $|V| - k$ elements, where k is the number of components of (V, E'). So each basis of E' has $|V| - k$ elements, proving (3).

A set is called simply a *basis* if it is a basis of X. The common cardinality of all bases is called the *rank* of the matroid. If $\mathcal{I}$ is the collection of forests in a connected graph $G = (V, E)$, then the bases of the matroid $(E, \mathcal{I})$ are exactly the spanning trees in G.

We next show that the matroids indeed are those structures for which the greedy algorithm leads to an optimal solution. Let X be some finite set and let $\mathcal{I}$ be a collection of subsets of X satisfying (2)(i) and (ii).

For any weight function $w : X \to \mathbb{R}$ we want to find a set Y in $\mathcal{I}$ maximizing

$$(4) \qquad w(Y) := \sum_{y \in Y} w(y).$$

The *greedy algorithm* consists of selecting $y_1, \ldots, y_r$ successively as follows. If $y_1, \ldots, y_k$ have been selected, choose $y \in X$ so that:

(5) (i) $y \notin \{y_1, \ldots, y_k\}$ and $\{y_1, \ldots, y_k, y\} \in \mathcal{I}$,
(ii) $w(y)$ is as large as possible among all y satisfying (i).

We stop if no y satisfying (5)(i) exist, that is, if $\{y_1, \ldots, y_k\}$ is a basis.

Now:

Theorem 10.1. *The pair $(X, \mathcal{I})$ satisfying (2)(i) and (ii) is a matroid, if and only if the greedy algorithm leads to a set Y in $\mathcal{I}$ of maximum weight $w(Y)$, for each weight function $w : X \to \mathbb{R}_+$.*

Proof. *Sufficiency.* Suppose the greedy algorithm leads to an independent set of maximum weight for each weight function w. We show that $(X, \mathcal{I})$ is a matroid.

Conditions (2)(i) and (ii) are satisfied by assumption. To see condition (2)(iii), let $Y, Z \in \mathcal{I}$ with $|Y| < |Z|$. Suppose that $Y \cup \{z\} \notin \mathcal{I}$ for each $z \in Z \setminus Y$.

Consider the following weight function w on X. Let $k := |Y|$. Define:

$$(6) \qquad \begin{aligned} w(x) &:= k + 2 && \text{if } x \in Y, \\ w(x) &:= k + 1 && \text{if } x \in Z \setminus Y, \\ w(x) &:= 0 && \text{if } x \in X \setminus (Y \cup Z). \end{aligned}$$

Now in the first k iterations of the greedy algorithm we find the k elements in Y. By assumption, at any further iteration, we cannot choose any element in $Z \setminus Y$. Hence any further element chosen, has weight 0. So the greedy algorithm will yield a basis of weight $k(k + 2)$.

However, any basis containing Z will have weight at least $|Z \cap Y|(k + 2) + |Z \setminus Y|(k + 1) \geq |Z|(k+1) \geq (k+1)(k+1) > k(k+2)$. Hence the greedy algorithm does not give a maximum-weight independent set.

Necessity. Now let $(X, \mathcal{I})$ be a matroid. Let $w : X \to \mathbb{R}_+$ be any weight function on X. Call an independent set Y *greedy* if it is contained in a maximum-weight basis. It suffices to show that if Y is greedy, and x is an element in $X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{I}$ and such that $w(x)$ is as large as possible, then $Y \cup \{x\}$ is greedy.

As Y is greedy, there exists a maximum-weight basis $B \supseteq Y$. If $x \in B$ then $Y \cup \{x\}$ is greedy again. If $x \notin B$, then there exists a basis B' containing $Y \cup \{x\}$ and contained in $B \cup \{x\}$. So

$B' = (B \setminus \{x'\}) \cup \{x\}$ for some $x' \in B \setminus Y$. As $w(x)$ is chosen maximum, $w(x) \geq w(x')$. Hence $w(B') \geq w(B)$, and therefore B' is a maximum-weight basis. So $Y \cup \{x\}$ is greedy. ▮

Note that by replacing "as large as possible" in (5) by "as small as possible", one obtains an algorithm for finding a *minimum*-weight basis in a matroid. Moreover, by ignoring elements of negative weight, the algorithm can be adapted to yield an independent set of maximum weight, for any weight function $w : X \to \mathbb{R}$.

Exercises

10.1. Show that condition (3) is equivalent to condition (2)(iii) (assuming (2)(i) and (ii)).

10.2. Let $M = (X, \mathcal{I})$ be a matroid. Two elements x, y of X are called *parallel* if $\{x, y\}$ is a circuit. Show that if x and y are parallel and Y is an independent set with $x \in Y$, then also $(Y \setminus \{x\}) \cup \{y\}$ is independent.

10.3. Let $M = (X, \mathcal{I})$ be a matroid, with $X = \{x_1, \ldots, x_m\}$. Define

(7) $$Y := \{x_i \mid r_M(\{x_1, \ldots, x_i\}) > r_M(\{x_1, \ldots, x_{i-1}\})\}.$$

Prove that Y belongs to $\mathcal{I}$.

10.2. Equivalent axioms for matroids

The definition of the notion of matroid given in the previous section is given by 'axioms' in terms of the independent sets. There are several other axioms that characterize matroids. In this section we give a number of them.

Let X be a finite set, and let $\mathcal{I}$ be a nonempty *down-monotone* collection of subsets of X; that is, if $F \in \mathcal{I}$ and $F' \subseteq F$, then $F' \in \mathcal{I}$. Let $\mathcal{B}$ be the collection of inclusionwise maximal sets in $\mathcal{I}$, and let $\mathcal{C}$ be the collection of inclusionwise minimimal sets that are *not* in $\mathcal{I}$. Finally, for any subset Y of X, define

(8) $$r(Y) := \max\{|Z| \mid Z \subseteq Y, Z \in \mathcal{I}\}.$$

Obviously, knowing one of the objects $\mathcal{I}$, $\mathcal{B}$, $\mathcal{C}$, r, we know all the other. Moreover, any nonempty antichain[21] $\mathcal{B}$ arises in this way from some nonempty down-monotone collection $\mathcal{I}$ of subsets. Similarly, any antichain $\mathcal{C}$ consisting of nonempty sets arises in this way. Finally, r arises in this way if and only if

(9)
(i) $r(\emptyset) = 0$,
(ii) if $Z \subseteq Y \subseteq X$ then $r(Z) \leq r(Y)$.

We can now characterize when such objects arise from a matroid $(X, \mathcal{I})$. That is, we obtain the following equivalent characterizations of matroids.

Theorem 10.2. *Let $\mathcal{I}$, $\mathcal{B}$, $\mathcal{C}$, and r be as above. Then the following are equivalent:*

(i) *if $F, F' \in \mathcal{I}$ and $|F'| > |F|$, then $F \cup \{x\} \in \mathcal{I}$ for some $x \in F' \setminus F$;*
(ii) *if $B, B' \in \mathcal{B}$ and $x \in B' \setminus B$, then $(B' \setminus \{x\}) \cup \{y\} \in \mathcal{B}$ for some $y \in B \setminus B'$;*
(iii) *if $B, B' \in \mathcal{B}$ and $x \in B' \setminus B$, then $(B \setminus \{y\}) \cup \{x\} \in \mathcal{B}$ for some $y \in B \setminus B'$;*
(iv) *if $C, C' \in \mathcal{C}$ with $C \neq C'$ and $x \in C \cap C'$, then $(C \cup C') \setminus \{x\}$ contains a set in $\mathcal{C}$;*
(v) *if $C, C' \in \mathcal{C}$, $x \in C \cap C'$, and $y \in C \setminus C'$, then $(C \cup C') \setminus \{x\}$ contains a set in $\mathcal{C}$ containing y;*

[21] An *antichain* is a collection of sets no two of which are contained in each other.

(vi) *for all* $Y, Z \subseteq X$ *one has*

$$r(Y \cap Z) + r(Y \cup Z) \leq r(Y) + r(Z). \tag{10}$$

Proof. (i)⇒(ii): (i) directly implies that all sets in $\mathcal{B}$ have the same size. Now let $B, B' \in \mathcal{B}$ and $x \in B' \setminus B$. Since $B' \setminus \{x\} \in \mathcal{I}$, by (i) there exists a $y \in B \setminus B'$ such that $B'' := (B' \setminus \{x\}) \cup \{y\} \in \mathcal{I}$. Since $|B''| = |B'|$, we know $B'' \in \mathcal{B}$.

(iii)⇒(i): Let F, F' form a counterexample to (i) with $|F \cap F'|$ as large as possible. Consider sets B, B' in $\mathcal{B}$ with $F \subseteq B$ and $F' \subseteq B'$.

As F, F' is a counterexample, we know $F \not\subseteq B'$. Choose $x \in F \setminus B'$. Then by (iii), $(B' \setminus \{y\}) \cup \{x\}$ for some $y \in B' \setminus B$. Hence replacing F' by $(F' \setminus \{y\}) \cup \{x\}$ we would keep a counterexample but increase $|F \cap F'|$, a contradiction.

(ii)⇒(iii): By the foregoing we know that (iii) implies (ii). Now axioms (ii) and (iii) interchange if we replace $\mathcal{B}$ by the collection of complements of sets in $\mathcal{B}$. Hence also the implication (ii)⇒(iii) holds.

(i)⇒(v): If (i) holds, then by the foregoing, also (ii) holds. Let $C, C' \in \mathcal{C}$ and $x \in C \cap C'$, $y \in C \setminus C'$. We can assume that $X = C \cup C'$. Let $B, B' \in \mathcal{B}$ with $B \supseteq C \setminus \{y\}$ and $B' \supseteq C' \setminus \{x\}$. Then $y \notin B$ and $x \notin B'$ (since $C \not\subseteq B$ and $C' \not\subseteq B'$).

We can assume that $y \notin B'$. Otherwise, $y \in B' \setminus B$, and hence by (ii), there exists a $z \in B \setminus B'$ with $B'' := (B' \setminus \{y\}) \cup \{z\} \in \mathcal{B}$. Then $z \neq x$, since otherwise $C' \subseteq B''$. Hence, replacing B' by B'' gives $y \notin B'$.

As $y \notin B'$, we know $B' \cup \{y\} \notin \mathcal{I}$, and hence there exists a $C'' \in \mathcal{C}$ contained in $B' \cup \{y\}$. As $C'' \not\subseteq B'$, we know $y \in C''$. Moreover, as $x \notin B'$ we know $x \notin C''$.

(v)⇒(iv): is trivial.

(iv)⇒(i): Let F, F' form a counterexample to (i) with $|F \cap F'|$ maximal. Then $F \not\subseteq F'$, and so we can choose $y \in F \setminus F'$. By the maximality of $|F \cap F'|$, we know $F' \cup \{x\} \notin \mathcal{I}$. So there is a $C \in \mathcal{C}$ contained in $F' \cup \{x\}$. As $C \not\subseteq F'$ we know $x \in C$. Then C is the unique set in $\mathcal{C}$ contained in $F' \cup \{x\}$. For suppose there is another, C' say. Again, $x \in C'$, and hence by (iv) there exists a $C'' \in \mathcal{C}$ contained in $(C \cup C') \setminus \{x\}$. But then $C'' \subseteq F'$, a contradiction.

As $C \not\subseteq F$, C intersects $F' \setminus F$. Choose $y \in C \cap (F' \setminus F)$. Then $F'' := (F' \cup \{x\}) \setminus \{y\}$ does not contain any set in $\mathcal{C}$ (as C is the only set in $\mathcal{C}$ contained in $F' \cup \{x\}$). Then replacing F' by F'', we would keep a counterexample while increasing $|F' \cap F|$, contradicting our assumption.

(i)⇒(vi): Choose $Y, Z \subseteq X$. Let F be an inclusionwise maximal set in $\mathcal{I}$ with $F \subseteq Y \cap Z$, and let F' be an inclusionwise maximal set in $\mathcal{I}$ with $F \subseteq F \subseteq Y \cup Z$. By (i) we know that $r(Y \cap Z) = |F|$ and $r(Y \cup Z) = |F'|$. Then

$$|F' \cap Y| + |F' \cap Z| = |F' \cap (Y \cap Z)| + |F' \cap (Y \cup Z)| \geq |F| + |F'|, \tag{11}$$

and hence we have (10).

(vi)⇒(i): Let $F, F' \in \mathcal{I}$ with $|F| < |F'|$. Let U be the largest subset of $F' \setminus F$ with $r(F \cup U) = |F|$. Then $U \neq F' \setminus F$, since $r(F \cup F') \geq |F'| > |F|$. So there exists an $x \in F' \setminus F \cup U$. If $F \cup \{x\} \in \mathcal{I}$ we are done, so we can assume that $F \cup \{x\} \notin \mathcal{I}$; equivalently, $r(F \cup \{x\}) = |F|$. Let $U' := U \cup \{x\}$. Then by (10),

$$r(F \cup U') \leq r(F \cup U) + r(F \cup \{x\}) - r(F) = |F|, \tag{12}$$

contradicting the maximality of U. ∎

Given a matroid $M = (X, \mathcal{I})$, any in $\mathcal{B}$ is called a *basis* and any set in $\mathcal{C}$ a *circuit* of M. The function r is called *rank function* of M (often denoted by r_M), and $r(Y)$ the *rank* of Y.

The symmetry of (ii) and (iii) in Theorem 10.2 immediately implies the following. Define

(13) $$\mathcal{B}^* := \{X \setminus B \mid B \in \mathcal{B}\}.$$

Then

Corollary 10.2a. *If $\mathcal{B}$ is the collection of bases of some matroid M, then $\mathcal{B}^*$ also is the collection of bases of some matroid on X, denoted by M^*.*

Proof. Directly from the equivalence of (ii) and (iii) in Theorem 10.2. ∎

The matroid M^* is called the *dual matroid* of M. Since $(\mathcal{B}^*)^* = \mathcal{B}$, we know $(M^*)^* = M$.

Theorem 10.3. *The rank function r_{M^*} of the dual matroid M^* satisfies:*

(14) $$r_{M^*}(Y) = |Y| + r_M(X \setminus Y) - r_M(X).$$

Proof.

(15) $$\begin{aligned} & r_{M^*}(Y) = \max\{|A \cap Y| \mid A \in \mathcal{B}^*\} = \\ & = |Y| - \min\{|B \cap Y| \mid B \in \mathcal{B}\} = |Y| - r_M(X) + \max\{|B \setminus Y| \mid B \in \mathcal{B}\} = \\ & |Y| - r_M(X) + r_M(X \setminus Y). \end{aligned}$$ ∎

Another way of constructing matroids from matroids is by 'deletion' and 'contraction'. Let $M = (X, \mathcal{I})$ be a matroid and let $Y \subseteq X$. Define

(16) $$\mathcal{I}' := \{Z \mid Z \subseteq Y, Z \in \mathcal{I}\}.$$

Then $M' = (Y, \mathcal{I}')$ is a matroid again, as one easily checks. M' is called the *restriction* of M to Y. If $Y = X \setminus Z$ with $Z \subseteq X$, we say that M' arises by *deleting* Z, and denote M' by $M \setminus Z$.

Contracting Z means replacing M by $(M^* \setminus Z)^*$. This matroid is denoted by M/Z. One may check that if G is a graph and e is an edge of G, then contracting edge $\{e\}$ in the cycle matroid $M(G)$ of G corresponds to contracting e in the graph. That is, $M(G)/\{e\} = M(G/\{e\})$, where $G/\{e\}$ denotes the graph obtained from G by contracting e.

If matroid M' arises from M by a series of deletions and contractions, M' is called a *minor* of M.

Exercises

10.4. (i) Let X be a finite set and let k be a natural number. Let $\mathcal{I} := \{Y \subseteq X \mid |Y| \leq k\}$. Show that $(X, \mathcal{I})$ is a matroid. Such matroids are called *k-uniform matroids*.

(ii) Show that k-uniform matroids are transversal matroids. Give an example of a k-uniform matroid that is neither graphic nor cographic.

10.5. Let $M = (X, \mathcal{I})$ be a matroid and let k be a natural number. Define $\mathcal{I}' := \{Y \in \mathcal{I} \mid |Y| \leq k\}$. Show that $(X, \mathcal{I}')$ is again a matroid (called the *k-truncation* of M).

10.6. Let $M = (X, \mathcal{I})$ be a matroid, let U be a set disjoint from X, and let $k \geq 0$. Define

(17) $$\mathcal{I}' := \{U' \cup Y \mid U' \subseteq U, Y \in \mathcal{I}, |U' \cup Y| \leq k\}.$$

Show that $(U \cup X, \mathcal{I}')$ is again a matroid.

10.7. Let $M = (X, \mathcal{I})$ be a matroid and let $x \in X$.

(i) Show that if x is not a loop, then a subset Y of $X \setminus \{x\}$ is independent in the contracted matroid $M/\{x\}$ if and only if $Y \cup \{x\}$ is independent in M.

(ii) Show that if x is a loop, then $M/\{x\} = M \setminus \{x\}$.

(iii) Show that for each $Y \subseteq X : r_{M/\{x\}}(Y) = r_M(Y \cup \{x\}) - r_M(\{x\})$.

10.8. Let $M = (X, \mathcal{I})$ be a matroid and let $Y \subseteq X$.

(ii) Let B be a basis of Y. Show that a subset U of $X \setminus Y$ is independent in the contracted matroid M/Y, if and only if $U \cup B$ is independent in M.

(ii) Show that for each $U \subseteq X \setminus Y$

(18) $$r_{M/Y}(U) = r_M(U \cup Y) - r_M(Y).$$

10.9. Let $M = (X, \mathcal{I})$ be a matroid and let $Y, Z \subseteq X$. Show that $(M \setminus Y)/Z = (M/Z) \setminus Y$. (That is, deletion and contraction commute.)

10.10. Let $M = (X, \mathcal{I})$ be a matroid, and suppose that we can test in polynomial time if any subset Y of X belongs to $\mathcal{I}$. Show that then the same holds for the dual matroid M^*.

10.3. Examples of matroids

In this section we describe some classes of examples of matroids.

I. Graphic matroids. As a first example we consider the matroids described in Section 10.1.

Let $G = (V, E)$ be a graph. Let $\mathcal{I}$ be the collection of all forests in G. Then $M = (E, \mathcal{I})$ is a matroid, as we saw in Section 10.1.

The matroid M is called the *cycle matroid* of G, denoted by $M(G)$. Any matroid obtained in this way, or isomorphic to such a matroid, is called a *graphic matroid.*

Note that the bases of $M(G)$ are exactly those forests F of G for which the graph (V, F) has the same number of components as G. So if G is connected, the bases are the spanning trees.

Note also that the circuits of $M(G)$, in the matroid sense, are exactly the circuits of G, in the graph sense.

II. Cographic matroids. There is an alternative way of obtaining a matroid from a graph $G = (V, E)$. It is in fact the matroid dual of the graphic matroid.

Let $\mathcal{B}$ be the set of subsets J of E such that $E \setminus J$ is an inclusionwise maximal forest. By Corollary 10.2a, $\mathcal{B}$ forms the collection of bases of a matroid. Its collection $\mathcal{I}$ of independent sets consists of those subsets J of E for which

(19) $$\kappa(V, E \setminus J) = \kappa(V, E).$$

where, for any graph H, let $\kappa(H)$ denote the number of components of H.

The matroid $(E, \mathcal{I})$ is called the *cocycle matroid* of G, denoted by $M^*(G)$. Any matroid obtained in this way, or isomorphic to such a matroid, is called a *cographic matroid.*

By definition, a subset C of E is a circuit of $M^*(G)$ if it is an inclusionwise minimal set with the property that $(V, E \setminus C)$ has more components than G. Hence C is a circuit of $M^*(G)$ if and only if C is an inclusionwise minimal nonempty cutset in G.

III. Linear matroids. Let A be an $m \times n$ matrix. Let $X = \{1, \ldots, n\}$ and let $\mathcal{I}$ be the collection of all those subsets Y of X so that the columns with index in Y are linearly independent. That is, so that the submatrix of A consisting of the columns with index in Y has rank $|Y|$.

Now:

Theorem 10.4. $(X, \mathcal{I})$ *is a matroid.*

Proof. Again, conditions (2)(i) and (ii) are easy to check. To see condition (2)(iii), let Y and Z be subsets of X so that the columns with index in Y are linearly independent, and similarly for Z, and so that $|Y| < |Z|$.

Suppose that $Y \cup \{x\} \notin \mathcal{I}$ for each $x \in Z \setminus Y$. This means that each column with index in $Z \setminus Y$ is spanned by the columns with index in Y. Trivially, each column with index in $Z \cap Y$ is spanned by the columns with index in Y. Hence each column with index in Z is spanned by the columns with index in Y. This contradicts the fact that the columns indexed by Y span an $|Y|$-dimensional space, while the columns indexed by Z span an $|Z|$-dimensional space, with $|Z| > |Y|$. ▮

Any matroid obtained in this way, or isomorphic to such a matroid, is called a *linear matroid.*

Note that the rank $r_M(Y)$ of any subset Y of X is equal to the rank of the matrix formed by the columns indexed by Y.

IV. Transversal matroids. Let $X_1, \ldots, X_m$ be subsets of the finite set X. A set $Y = \{y_1, \ldots, y_n\}$ is called a *partial transversal* (*of* $X_1, \ldots, X_m$), if there exist distinct indices $i_1, \ldots, i_n$ so that $y_j \in X_{i_j}$ for $j = 1, \ldots, n$. A partial transversal of cardinality m is called a *transversal* (or a *system of distinct representatives*, or an *SDR*).

Another way of representing partial transversals is as follows. Let $\mathcal{G}$ be the bipartite graph with vertex set $\mathcal{V} := \{1, \ldots, m\} \cup X$ and with edges all pairs $\{i, x\}$ with $i \in \{1, \ldots, m\}$ and $x \in X_i$. (We assume here that $\{1, \ldots, m\} \cap X = \emptyset$.)

For any matching M in $\mathcal{G}$, let $\rho(M)$ denote the set of those elements in X that belong to some edge in M. Then it is not difficult to see that:

(20) $Y \subseteq X$ is a partial transversal, if and only if $Y = \rho(M)$ for some matching M in $\mathcal{G}$.

Now let $\mathcal{I}$ be the collection of all partial transversals for $X_1, \ldots, X_m$. Then:

Theorem 10.5. $(X, \mathcal{I})$ *is a matroid.*

Proof. Again, conditions (2)(i) and (ii) are trivial. To see (2)(iii), let Y and Z be partial transversals with $|Y| < |Z|$. Consider the graph $\mathcal{G}$ constructed above. By (20) there exist matchings M and M' in $\mathcal{G}$ so that $Y = \rho(M)$ and $Z = \rho(M')$. So $|M| = |Y| < |Z| = |M'|$.

Consider the union $M \cup M'$ of M and M'. Each component of the graph $(\mathcal{V}, M \cup M')$ is either a path, or a circuit, or a singleton vertex. Since $|M'| > |M|$, at least one of these components is a path P with more edges in M' than in M. The path consists of edges alternatingly in M' and in M, with end edges in M'.

Let N and N' denote the edges in P occurring in M and M', respectively. So $|N'| = |N| + 1$. Since P has odd length, exactly one of its end vertices belongs to X; call this end vertex x. Then $x \in \rho(M') = Z$ and $x \notin \rho(M) = Y$. Define $M'' := (M \setminus N) \cup N'$. Clearly, M'' is a matching with $\rho(M'') = Y \cup \{x\}$. So $Y \cup \{x\}$ belongs to $\mathcal{I}$. ▮

Any matroid obtained in this way, or isomorphic to such a matroid, is called a *transversal matroid.* If the sets $X_1, \ldots, X_m$ form a partition of X, one speaks of a *partition matroid.*

These four classes of examples show that the greedy algorithm has a wider applicability than just for finding minimum-weight spanning trees. There are more classes of matroids (like 'algebraic matroids', 'gammoids'), for which we refer to Welsh [1976].

Exercises

10.11. Show that a partition matroid is graphic, cographic, and linear.

10.12. Let $M = (V, \mathcal{I})$ be the transversal matroid derived from subsets $X_1, \ldots, X_m$ of X as in Example IV.

(i) Show with Kőnig's matching theorem that:

$$r_M(X) = \min_{J \subseteq \{1,\ldots,m\}} (|\bigcup_{j \in J} X_j| + m - |J|). \tag{21}$$

(ii) Derive a formula for $r_M(Y)$ for any $Y \subseteq X$.

10.13. Let $G = (V, E)$ be a graph. Let $\mathcal{I}$ be the collection of those subsets Y of E so that F has at most one circuit. Show that $(E, \mathcal{I})$ is a matroid.

10.14. Let $G = (V, E)$ be a graph. Call a collection $\mathcal{C}$ of circuits a *circuit basis* of G if each circuit of G is a symmetric difference of circuits in $\mathcal{C}$. (We consider circuits as edge sets.)
Give a polynomial-time algorithm to find a circuit basis $\mathcal{C}$ of G that minimizes $\sum_{C \in \mathcal{C}} |C|$.
(The running time of the algorithm should be bounded by a polynomial in $|V| + |E|$.)

10.15. Let $G = (V, E)$ be a connected graph. For each subset E' of E, let $\kappa(V, E')$ denote the number of components of the graph (V, E'). Show that for each $E' \subseteq E$:

(i) $r_{M(G)}(E') = |V| - \kappa(V, E')$;

(ii) $r_{M^*(G)}(E') = |E'| - \kappa(V, E \setminus E') + 1$.

10.16. Let G be a planar graph and let G^* be a planar graph dual to G. Show that the cycle matroid $M(G^*)$ of G^* is isomorphic to the cocycle matroid $M^*(G)$ of G.

10.17. Show that the dual matroid of a linear matroid is again a linear matroid.

10.18. Let $G = (V, E)$ be a loopless undirected graph. Let A be the matrix obtained from the $V \times E$ incidence matrix of G by replacing in each column, exactly one of the two 1's by -1.

(i) Show that a subset Y of E is a forest if and only if the columns of A with index in Y are linearly independent.

(ii) Derive that any graphic matroid is a linear matroid.

(iii) Derive (with the help of Exercise 10.17) that any cographic matroid is a linear matroid.

10.4. Two technical lemmas

In this section we prove two technical lemmas as a preparation to the coming sections on matroid intersection.

Let $M = (X, \mathcal{I})$ be a matroid. For any $Y \in \mathcal{I}$ define a bipartite graph $H(M, Y)$ as follows. The graph $H(M, Y)$ has vertex set X, with colour classes Y and $X \setminus Y$. Elements $y \in Y$ and $x \in X \setminus Y$ are adjacent if and only if

$$(Y \setminus \{y\}) \cup \{x\} \in \mathcal{I}. \tag{22}$$

Then we have:

Lemma 10.1. *Let $M = (X, \mathcal{I})$ be a matroid and let $Y, Z \in \mathcal{I}$ with $|Y| = |Z|$. Then $H(M, Y)$ contains a perfect matching on $Y \triangle Z$.*[22]

Proof. Suppose not. By Kőnig's matching theorem there exist a subset S of $Y \setminus Z$ and a subset S' of $Z \setminus Y$ such that for each edge $\{y, z\}$ of $H(M, Y)$ satisfying $z \in S'$ one has $y \in S$ and such that $|S| < |S'|$.

[22] A *perfect matching on* a vertex set U is a matching M with $\bigcup M = U$.

As $|(Y \cap Z) \cup S| < |(Y \cap Z) \cup S'|$, there exists an element $z \in S'$ such that $T := (Y \cap Z) \cup S \cup \{z\}$ belongs to $\mathcal{I}$. This implies that there exists an $U \in \mathcal{I}$ such that $T \subseteq U \subseteq T \cup Y$ and $|U| = |Y|$. So $U = (Y \setminus \{x\}) \cup \{z\}$ for some $x \notin S$. As $\{x, z\}$ is an edge of $H(M, Y)$ this contradicts the choice of S and S'. ▮

The following forms a counterpart:

Lemma 10.2. *Let $M = (X, \mathcal{I})$ be a matroid and let $Y \in \mathcal{I}$. Let $Z \subseteq X$ be such that $|Y| = |Z|$ and such that $H(M, Y)$ contains a unique perfect matching N on $Y \triangle Z$. Then Z belongs to $\mathcal{I}$.*

Proof. By induction on $k := |Z \setminus Y|$, the case $k = 0$ being trivial. Let $k \geq 1$.

By the unicity of N there exists an edge $\{y, z\} \in N$, with $y \in Y \setminus Z$ and $z \in Z \setminus Y$, with the property that there is no $z' \in Z \setminus Y$ such that $z' \neq z$ and $\{y, z'\}$ is an edge of $H(M, Y)$.

Let $Z' := (Z \setminus \{z\}) \cup \{y\}$ and $N' := N \setminus \{\{y, z\}\}$. Then N' is the unique matching in $H(M, Y)$ with union $Y \triangle Z'$. Hence by induction, Z' belongs to $\mathcal{I}$.

There exists an $S \in \mathcal{I}$ such that $Z' \setminus \{y\} \subseteq S \subseteq (Y \setminus \{y\}) \cup Z$ and $|S| = |Y|$ (since $(Y \setminus \{y\}) \cup Z = (Y \setminus \{y\}) \cup \{z\} \cup Z'$ and since $(Y \setminus \{y\}) \cup \{z\}$ belongs to $\mathcal{I}$). Assuming $Z \notin \mathcal{I}$, we know $z \notin S$ and hence $r((Y \cup Z') \setminus \{y\}) = |Y|$. Hence there exists an $z' \in Z' \setminus Y$ such that $(Y \setminus \{y\}) \cup \{z'\}$ belongs to $\mathcal{I}$. This contradicts the choice of y. ▮

Exercises

10.19. Let $M = (X, \mathcal{I})$ be a matroid, let B be a basis of M, and let $w : X \to \mathbb{R}$ be a weight function. Show that B is a basis of maximum weight, if and only if $w(B') \leq w(B)$ for every basis B' with $|B' \setminus B| = 1$.

10.20. Let $M = (X, \mathcal{I})$ be a matroid and let Y and Z be independent sets with $|Y| = |Z|$. For any $y \in Y \setminus Z$ define $\delta(y)$ as the set of those $z \in Z \setminus Y$ which are adjacent to y in the graph $H(M, Y)$.

(i) Show that for each $y \in Y \setminus Z$ the set $(Z \setminus \delta(y)) \cup \{y\}$ belongs to $\mathcal{I}$.
(*Hint:* Apply inequality (10) to $X' := (Z \setminus \delta(y)) \cup \{y\}$ and $X'' := (Z \setminus \delta(y)) \cup (Y \setminus \{y\})$.)

(ii) Derive from (i) that for each $y \in Y \setminus Z$ there exists an $z \in Z \setminus Y$ so that $\{y, z\}$ is an edge both of $H(M, Y)$ and of $H(M, Z)$.

10.5. Matroid intersection

Edmonds [1970] discovered that the concept of matroid has even more algorithmic power, by showing that there exist fast algorithms also for *intersections* of matroids.

Let $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ be two matroids, on the same set X. Consider the collection $\mathcal{I}_1 \cap \mathcal{I}_2$ of *common independent sets*. The pair $(X, \mathcal{I}_1 \cap \mathcal{I}_2)$ is generally *not* a matroid again (cf. Exercise 10.21).

What Edmonds showed is that, for any weight function w on X, a maximum-weight common independent set can be found in polynomial time. In particular, a common independent set of maximum cardinality can be found in polynomial time.

We consider first some applications.

Example 10.5a. Let $G = (V, E)$ be a bipartite graph, with colour classes V_1 and V_2, say. Let $\mathcal{I}_1$ be the collection of all subsets F of E so that no two edges in F have a vertex in V_1 in common. Similarly, let $\mathcal{I}_2$ be the collection of all subsets F of E so that no two edges in F have a vertex in V_2 in common. So both $(X, \mathcal{I}_1)$ and $(X, \mathcal{I}_2)$ are partition matroids.

Now $\mathcal{I}_1 \cap \mathcal{I}_2$ is the collection of matchings in G. Finding a maximum-weight common independent set amounts to finding a maximum-weight matching in G.

Example 10.5b. Let $X_1, \dots, X_m$ and $Y_1, \dots, Y_m$ be subsets of X. Let $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ be the corresponding transversal matroids.

Then common independent sets correspond to common partial transversals. The collections $(X_1, \dots, X_m)$ and $(Y_1, \dots, Y_m)$ have a common transversal, if and only if the maximum cardinality of a common independent set is equal to m.

Example 10.5c. Let $D = (V, A)$ be a directed graph. Let $M_1 = (A, \mathcal{I}_1)$ be the cycle matroid of the underlying undirected graph. Let $\mathcal{I}_2$ be the collection of subsets Y of A so that each vertex of D is entered by at most one arc in Y. So $M_2 := (A, \mathcal{I}_2)$ is a partition matroid.

Now the common independent sets are those subsets Y of A with the property that each component of (V, Y) is a rooted tree. Moreover, D has a rooted spanning tree, if and only if the maximum cardinality of a set in $\mathcal{I}_1 \cap \mathcal{I}_2$ is equal to $|V| - 1$.

Example 10.5d. Let $G = (V, E)$ be a connected undirected graph. Then G has two edge-disjoint spanning trees, if and only if the maximum cardinality of a common independent set in the cycle matroid $M(G)$ of G and the cocycle matroid $M^*(G)$ of G is equal to $|V| - 1$.

In this section we describe an algorithm for finding a maximum-cardinality common independent sets in two given matroids. In the next section we consider the more general maximum-weight problem.

For any two matroids $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ and any $Y \in \mathcal{I}_1 \cap \mathcal{I}_2$, we define a directed graph $H(M_1, M_2, Y)$ as follows. Its vertex set is X, while for any $y \in Y, x \in X \setminus Y$,

(23) $$\begin{aligned} &(y, x) \text{ is an arc of } H(M_1, M_2, Y) \text{ if and only if } (Y \setminus \{y\}) \cup \{x\} \in \mathcal{I}_1, \\ &(x, y) \text{ is an arc of } H(M_1, M_2, Y) \text{ if and only if } (Y \setminus \{y\}) \cup \{x\} \in \mathcal{I}_2. \end{aligned}$$

These are all arcs of $H(M_1, M_2, Y)$. In fact, this graph can be considered as the union of directed versions of the graphs $H(M_1, Y)$ and $H(M_2, Y)$ defined in Section 10.4.

The following is the basis for finding a maximum-cardinality common independent set in two matroids.

Cardinality common independent set augmenting algorithm

input: matroids $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ and a set $Y \in \mathcal{I}_1 \cap \mathcal{I}_2$;
output: a set $Y' \in \mathcal{I}_1 \cap \mathcal{I}_2$ with $|Y'| > |Y|$, if it exists.
description of the algorithm: We assume that M_1 and M_2 are given in such a way that for any subset Z of X we can check in polynomial time if $Z \in \mathcal{I}_1$ and if $Z \in \mathcal{I}_2$.

Consider the sets

(24) $$\begin{aligned} X_1 &:= \{y \in X \setminus Y \mid Y \cup \{y\} \in \mathcal{I}_1\}, \\ X_2 &:= \{y \in X \setminus Y \mid Y \cup \{y\} \in \mathcal{I}_2\}. \end{aligned}$$

Moreover, consider the directed graph $H(M_1, M_2, Y)$ defined above. There are two cases.

Case 1. *There exists a directed path P in $H(M_1, M_2, Y)$ from some vertex in X_1 to some vertex in X_2.* (Possibly of length 0 if $X_1 \cap X_2 \neq \emptyset$.)

We take a shortest such path P (that is, with a minimum number of arcs). Let P traverse the vertices $y_0, z_1, y_1, \dots, z_m, y_m$ of $H(M_1, M_2, Y)$, in this order. By construction of the graph $H(M_1, M_2, Y)$ and the sets X_1 and X_2, this implies that $y_0, \dots, y_m$ belong to $X \setminus Y$ and $z_1, \dots, z_m$ belong to Y.

Now output

(25) $$Y' := (Y \setminus \{z_1, \dots, z_m\}) \cup \{y_0, \dots, y_m\}.$$

Case 2. *There is no directed path in $H(M_1, M_2, Y)$ from any vertex in X_1 to any vertex vertex in X_2.* Then Y is a maximum-cardinality common independent set. ▮

This finishes the description of the algorithm. The correctness of the algorithm is given in the following two theorems.

Theorem 10.6. *If Case 1 applies, then $Y' \in \mathcal{I}_1 \cap \mathcal{I}_2$.*

Proof. Assume that Case 1 applies. By symmetry it suffices to show that Y' belongs to $\mathcal{I}_1$.

To see that $Y' \setminus \{y_0\}$ belongs to $\mathcal{I}_1$, consider the graph $H(M_1, Y)$ defined in Section 10.4. Observe that the edges $\{z_j, y_j\}$ form the only matching in $H(M_1, Y)$ with union equal to $\{z_1, \ldots, z_m, y_1, \ldots, y_m\}$ (otherwise P would have a shortcut). So by Lemma 10.2, $Y' \setminus \{y_0\} = (Y \setminus \{z_1, \ldots, z_m\}) \cup \{y_1, \ldots, y_m\}$ belongs to $\mathcal{I}_1$.

To show that Y' belongs to $\mathcal{I}_1$, observe that $r_{M_1}(Y \cup Y') \geq r_{M_1}(Y \cup \{y_0\}) = |Y| + 1$, and that, as $(Y' \setminus \{y_0\}) \cap X_1 = \emptyset$, $r_{M_1}((Y \cup Y') \setminus \{y_0\}) = |Y|$. As $Y' \setminus \{y_0\} \in \mathcal{I}_1$, we know $Y' \in \mathcal{I}_1$. ▮

Theorem 10.7. *If Case 2 applies, then Y is a maximum-cardinality common independent set.*

Proof. As Case 2 applies, there is no directed $X_1 - X_2$ path in $H(M_1, M_2, Y)$. Hence there is a subset U of X containing X_2 such that $U \cap X_1 = \emptyset$ and such that no arc of $H(M_1, M_2, Y)$ enters U. (We can take for U the set of vertices that are not reachable by a directed path from X_1.)

We show

(26) $$r_{M_1}(U) + r_{M_2}(X \setminus U) = |Y|.$$

To this end, we first show

(27) $$r_{M_1}(U) = |Y \cap U|.$$

Clearly, as $Y \cap U \in \mathcal{I}_1$, we know $r_{M_1}(U) \geq |Y \cap U|$. Suppose $r_{M_1}(U) > |Y \cap U|$. Then there exists an x in $U \setminus Y$ so that $(Y \cap U) \cup \{x\} \in \mathcal{I}_1$. Since $Y \in \mathcal{I}_1$, this implies that there exists a set $Z \in \mathcal{I}_1$ with $|Z| \geq |Y|$ and $(Y \cap U) \cup \{x\} \subseteq Z \subseteq Y \cup \{x\}$. Then $Z = Y \cup \{x\}$ or $Z = (Y \setminus \{y\}) \cup \{x\}$ for some $y \in Y \setminus U$.

In the first alternative, $x \in X_1$, contradicting the fact that x belongs to U. In the second alternative, (y, x) is an arc of $H(M_1, M_2, Y)$ entering U. This contradicts the definition of U (as $y \notin U$ and $x \in U$).

This shows (27). Similarly we have that $r_{M_2}(X \setminus U) = |Y \setminus U|$. Hence we have (26).

Now (26) implies that for any set Z in $\mathcal{I}_1 \cap \mathcal{I}_2$ one has

(28) $$|Z| = |Z \cap U| + |Z \setminus U| \leq r_{M_1}(U) + r_{M_2}(X \setminus U) = |Y|.$$

So Y is a common independent set of maximum cardinality. ▮

The algorithm clearly has polynomially bounded running time, since we can construct the auxiliary directed graph $H(M_1, M_2, Y)$ and find the path P (if it exists), in polynomial time.

It implies the result of Edmonds [1970]:

Theorem 10.8. *A maximum-cardinality common independent set in two matroids can be found in polynomial time.*

Proof. Directly from the above, as we can find a maximum-cardinality common independent set after applying at most $|X|$ times the common independent set augmenting algorithm. ▮

The algorithm also yields a min-max relation for the maximum cardinality of a common independent set, as was shown again by Edmonds [1970].

Theorem 10.9 (Edmonds' matroid intersection theorem). *Let $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ be matroids. Then*

$$\max_{Y \in \mathcal{I}_1 \cap \mathcal{I}_2} |Y| = \min_{U \subseteq X} (r_{M_1}(U) + r_{M_2}(X \setminus U)). \tag{29}$$

Proof. The inequality $\leq$ follows similarly as in (28). The reverse inequality follows from the fact that if the algorithm stops with set Y, we obtain a set U for which (26) holds. Therefore, the maximum in (29) is at least as large as the minimum. ∎

Exercises

10.21. Give an example of two matroids $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ so that $(X, \mathcal{I}_1 \cap \mathcal{I}_2)$ is *not* a matroid.

10.22. Derive Kőnig's matching theorem from Edmonds' matroid intersection theorem.

10.23. Let $(X_1, \ldots, X_m)$ and $(Y_1, \ldots, Y_m)$ be subsets of the finite set X. Derive from Edmonds' matroid intersection theorem: $(X_1, \ldots, X_m)$ and $(Y_1, \ldots, Y_m)$ have a common transversal, if and only if

$$|(\bigcup_{i \in I} X_i) \cap (\bigcup_{j \in J} Y_j)| \geq |I| + |J| - m \tag{30}$$

for all subsets I and J of $\{1, \ldots, m\}$.

10.24. Reduce the problem of finding a Hamiltonian cycle in a directed graph to the problem of finding a maximum-cardinality common independent set in *three* matroids.

10.25. Let $G = (V, E)$ be a graph and let the edges of G be coloured with $|V| - 1$ colours. That is, we have partitioned E into classes $X_1, \ldots, X_{|V|-1}$, called *colours*. Show that there exists a spanning tree with all edges coloured differently, if and only if (V, E') has at most $|V| - t$ components, for any union E' of t colours, for any $t \geq 0$.

10.26. Let $M = (X, \mathcal{I})$ be a matroid and let $X_1, \ldots, X_m$ be subsets of X. Then $(X_1, \ldots, X_m)$ has an independent transversal, if and only if the rank of the union of any t sets among $X_1, \ldots, X_m$ is at least t, for any $t \geq 0$. (Rado [1942].)

10.27. Let $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ be matroids. Define

$$\mathcal{I}_1 \vee \mathcal{I}_2 := \{Y_1 \cup Y_2 \mid Y_1 \in \mathcal{I}_1, Y_2 \in \mathcal{I}_2\}. \tag{31}$$

(i) Show that the maximum cardinality of a set in $\mathcal{I}_1 \vee \mathcal{I}_2$ is equal to

$$\min_{U \subseteq X} (r_{M_1}(U) + r_{M_2}(U) + |X \setminus U|). \tag{32}$$

(*Hint:* Apply the matroid intersection theorem to M_1 and M_2^*.)

(ii) Derive that for each $Y \subseteq X$:

$$\max\{|Z| \mid Z \subseteq Y, Z \in \mathcal{I}_1 \vee \mathcal{I}_2\} = \min_{U \subseteq Y} (r_{M_1}(U) + r_{M_2}(U) + |Y \setminus U|). \tag{33}$$

(iii) Derive that $(X, \mathcal{I}_1 \vee \mathcal{I}_2)$ is again a matroid.
(*Hint:* Use axiom (vi) in Theorem 10.2.)
This matroid is called the *union* of M_1 and M_2, denoted by $M_1 \vee M_2$. (Edmonds and Fulkerson [1965], Nash-Williams [1967].)

(iv) Let $M_1 = (X, \mathcal{I}_1), \ldots, M_k = (X, \mathcal{I}_k)$ be matroids and let

$$\mathcal{I}_1 \vee \ldots \vee \mathcal{I}_k := \{Y_1 \cup \ldots \cup Y_k \mid Y_1 \in \mathcal{I}_1, \ldots, Y_k \in \mathcal{I}_k\}. \tag{34}$$

Derive from (iii) that $M_1 \vee \ldots \vee M_k := (X, \mathcal{I}_1 \vee \ldots \vee \mathcal{I}_k)$ is again a matroid and give a formula for its rank function.

10.28. (i) Let $M = (X, \mathcal{I})$ be a matroid and let $k \geq 0$. Show that X can be covered by k independent sets, if and only if $|U| \leq k \cdot r_M(U)$ for each subset U of X.

(*Hint:* Use Exercise 10.27.) (Edmonds [1965b].)

(ii) Show that the problem of finding a minimum number of independent sets covering X in a given matroid $M = (X, \mathcal{I})$, is solvable in polynomial time.

10.29. Let $G = (V, E)$ be a graph and let $k \geq 0$. Show that E can be partitioned into k forests, if and only if each nonempty subset W of V contains at most $k(|W| - 1)$ edges of G.

(*Hint:* Use Exercise 10.28.) (Nash-Williams [1964].)

10.30. Let $X_1, \ldots, X_m$ be subsets of X and let $k \geq 0$.

(i) Show that X can be partitioned into k partial transversals of $(X_1, \ldots, X_m)$, if and only if

$$k(m - |I|) \geq |X \setminus \bigcup_{i \in I} X_i| \tag{35}$$

for each subset I of $\{1, \ldots, m\}$.

(ii) Derive from (i) that $\{1, \ldots, m\}$ can be partitioned into classes $I_1, \ldots, I_k$ so that $(X_i \mid i \in I_j)$ has a transversal, for each $j = 1, \ldots, k$, if and only if Y contains at most $k|Y|$ of the X_i as a subset, for each $Y \subseteq X$.

(*Hint:* Interchange the roles of $\{1, \ldots, m\}$ and X.) (Edmonds and Fulkerson [1965].)

10.31. (i) Let $M = (X, \mathcal{I})$ be a matroid and let $k \geq 0$. Show that there exist k pairwise disjoint bases of M, if and only if $k(r_M(X) - r_M(U)) \geq |X \setminus U|$ for each subset U of X.

(*Hint:* Use Exercise 10.27.) (Edmonds [1965b].)

(ii) Show that the problem of finding a maximum number of pairwise disjoint bases in a given matroid, is solvable in polynomial time.

10.32. Let $G = (V, E)$ be a connected graph and let $k \geq 0$. Show that there exist k pairwise edge-disjoint spanning trees, if and only if for each t, for each partition $(V_1, \ldots, V_t)$ of V into t classes, there are at least $k(t - 1)$ edges connecting different classes of this partition.

(*Hint:* Use Exercise 10.31.) (Nash-Williams [1961], Tutte [1961].)

10.33. Let M_1 and M_2 be matroids so that, for $i = 1, 2$, we can test in polynomial time if a given set is independent in M_i. Show that the same holds for the union $M_1 \vee M_2$.

10.34. Let $M = (X, \mathcal{I})$ be a matroid and let B and B' be two disjoint bases. Let B be partitioned into sets Y_1 and Y_2. Show that there exists a partition of B' into sets Z_1 and Z_2 so that both $Y_1 \cup Z_1 \cup Z_2$ and $Z_1 \cup Y_2$ are bases of M.

(*Hint:* Assume without loss of generality that $X = B \cup B'$. Apply the matroid intersection theorem to the matroids $(M \setminus Y_1)/Y_2$ and $(M^* \setminus Y_1)/Y_2$.)

10.35. The following is a special case of a theorem of Nash-Williams [1985]:

Let $G = (V, E)$ be a simple, connected graph and let $b : V \to \mathbb{Z}_+$. Call a graph $\tilde{G} = (\tilde{V}, \tilde{E})$ a *b-detachment* of G if there is a function $\phi : \tilde{V} \to V$ such that $|\phi^{-1}(v)| = b(v)$ for each $v \in V$, and such that there is a one-to-one function $\psi : \tilde{E} \to E$ with $\psi(e) = \{\phi(v), \phi(w)\}$ for each edge $e = \{v, w\}$ of $\tilde{G}$.

Then there exists a connected b-detachment, if and only if for each $U \subseteq V$ the number of components of the graph induced by $V \setminus U$ is at most $|E_U| - b(U) + 1$.

Here E_U denotes the set of edges intersecting U.

10.6. Weighted matroid intersection

We next consider the problem of finding a maximum-weight common independent set, in two given matroids, with a given weight function. The algorithm, again due to Edmonds [1970], is an extension of the algorithm given in Section 10.5. In each iteration, instead of finding a path P with a minimum number of arcs in H, we will now require P to have minimum length with respect to some length function defined on H.

To describe the algorithm, if matroid $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ and a weight function $w : S \to \mathbb{R}$ are given, call a set $Y \in \mathcal{I}_1 \cap \mathcal{I}_2$ *extreme* if $w(Z) \leq w(Y)$ for each $Z \in \mathcal{I}_1 \cap \mathcal{I}_2$ satisfying $|Z| = |Y|$.

Weighted common independent set augmenting algorithm

input: matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$, a weight function $w : S \to \mathbb{Q}$, and an extreme common independent set Y;
output: an extreme common independent set Y' with $|Y'| = |Y| + 1$, if it exists
description of the algorithm: Consider again the sets X_1 and X_2 and the directed graph $G(M_1, M_2, Y)$ on S as in the cardinality case.

For any $x \in S$ define the 'length' $l(x)$ of x by:

$$(36) \qquad \begin{array}{ll} l(x) := w(x) & \text{if } x \in Y, \\ l(x) := -w(x) & \text{if } x \notin Y. \end{array}$$

The *length* of a path P, denoted by $l(P)$, is equal to the sum of the lengths of the vertices traversed by P, counting multiplicities.

We consider two cases.

Case 1. *$G(M_1, M_2, Y)$ has an $X_1 - X_2$ path P.* We choose P so that $l(P)$ is minimal and so that it has a minimum number of arcs among all minimum-length $X_1 - X_2$ paths. Set $Y' := Y \triangle VP$.

Case 2. *$G(M_1, M_2, Y)$ has no $X_1 - X_2$ path.* Then Y is a maximum-size common independent set. ∎

This finishes the description of the algorithm. The correctness of the algorithm if Case 2 applies follows directly from Theorem 10.7. In order to show the correctness if Case 1 applies, we first prove the following basic property of the length function l.

Theorem 10.10. *Let C be a simple directed circuit in $G(M_1, M_2, Y)$, and let $t \in VC$. Define $Z := Y \triangle VC$. If $Z \notin \mathcal{I}_1 \cap \mathcal{I}_2$ then there exists a directed cycle C' with $VC' \subset VC$ such that $l(C') < 0$, or $l(C') \leq l(C)$ and $t \in VC'$.*

Proof. By symmetry we can assume that $Z \notin \mathcal{I}_1$. Let N_1 and N_2 be the sets of arcs in C belonging to $G(M_1, Y)$ and $G(M_2, Y)$ respectively. If $Z \notin \mathcal{I}_1$, there is, by Theorem 10.2 a matching N_1' in $G(M_1, Y)$ on VC with $N_1' \neq N_1$. Consider the directed graph $D = (VC, A)$ formed by the arcs in N_1, N_1' (taking arcs in $N_1 \cap N_1'$ multiple), and by the arcs in N_2 taking each of them twice (parallel). Now each vertex in VC is entered and left by exactly two arcs of D. Moreover, since $N_1' \neq N_1$, D contains a simple directed circuit C_1 with $VC_1 \subset VC$. We can extend this to a decomposition of A into simple directed cycles $C_1, \ldots, C_k$. Then

$$(37) \qquad \chi^{VC_1} + \cdots + \chi^{VC_k} = 2 \cdot \chi^{VC}.$$

Since $VC_1 \neq VC$ we know that $VC_j = VC$ for at most one j. If, say $VC_k = VC$, then (37) implies that either $l(VC_j) < 0$ for some $j < k$ or $l(VC_j) \leq l(VC)$ for all $j < k$, implying the proposition.

If $VC_j \neq VC$ for all j, then $l(VC_j) < 0$ for some $j \leq k$ or $l(VC_j) \leq l(VC)$ for all $j \leq k$, again implying the proposition. ∎

This implies:

Theorem 10.11. *Let $Y \in \mathcal{I}_1 \cap \mathcal{I}_2$. Then Y is extreme if and only if $G(M_1, M_2, Y)$ has no directed cycle of negative length.*

Proof. To see necessity, suppose $G(M_1, M_2, Y)$ has a cycle C of negative length. Choose C with $|VC|$ minimal. Consider $Z := Y \triangle VC$. Since $w(Z) = w(Y) - l(C) > w(Y)$, while $|Z| = |Y|$, we know that $Z \notin \mathcal{I}_1 \cap \mathcal{I}_2$. Hence by Proposition 10.10, $G(M_1, M_2, Y)$ has a negative-length directed cycle covering fewer than $|VC|$ vertices, contradicting our assumption.

To see sufficiency, consider a $Z \in \mathcal{I}_1 \cap \mathcal{I}_2$ with $|Z| = |Y|$. By Corollary 10.1, both $G(M_1, Y)$ and $G(M_2, Y)$ have a perfect matching on $Y \triangle Z$. These two matchings together form a disjoint union of a number of directed cycles $C_1, \ldots, C_t$. Then

$$w(Y) - w(Z) = \sum_{j=1}^{t} l(C_j) \geq 0, \tag{38}$$

implying $w(Z) \leq w(Y)$. So Y is extreme. ∎

This theorem implies that we can find in the algorithm a shortest path P in polynomial time (with the Bellman-Ford method).

This also gives:

Theorem 10.12. *If Case 1 applies, Y' is an extreme common independent set.*

Proof. We first show that $Y' \in \mathcal{I}_1 \cap \mathcal{I}_2$. To this end, let t be a new element, and extend (for each $i = 1, 2$), M_i to a matroid $M'_i = (S + t, \mathcal{I}'_i)$, where for each $T \subseteq S + t$:

$$T \in \mathcal{I}'_i \text{ if and only if } T - t \in \mathcal{I}_i. \tag{39}$$

Note that $G(M'_1, M'_2, Y + t)$ arises from $G(M_1, M_2, Y)$ by extending it with a new vertex t and adding arcs from each vertex in X_1 to t, and from t to each vertex in X_2.

Let P be the path found in the algorithm. Define

$$w(t) := l(t) := -l(P). \tag{40}$$

As P is a shortest $X_1 - X_2$ path, this makes that $G(M'_1, M'_2, Y + t)$ has no negative-length directed cycle. Hence, by Theorem 10.11, $Y + t$ is an extreme common independent set in M'_1 and M'_2.

Let P run from $z_1 \in X_1$ to $z_2 \in X_2$. Extend P by the arcs (t, z_1) and (z_2, t) to a directed cycle C. So $Z = (Y + t) \triangle VC$. As P has a minimum number of arcs among all shortest $X_1 - X_2$ paths, and as $G(M'_1, M'_2, Y + t)$ has no negative-length directed circuits, by Proposition 10.10 we know that $Z \in \mathcal{I}_1 \cap \mathcal{I}_2$.

Moreover, Z is extreme, since $Y + t$ is extreme and $w(Z) = w(Y + t)$. ∎

So the weighted common independent set augmenting algorithm is correct. It obviously has polynomially bounded running time. Therefore:

Theorem 10.13. *A maximum-weight common independent set in two matroids can be found in polynomial time.*

Proof. Starting with the extreme common independent set $Y_0 := \emptyset$ we can find iteratively extreme common independent sets $Y_0, Y_1, \ldots, Y_k$, where $|Y_i| = i$ for $i = 0, \ldots, k$ and where Y_k is a maximum-size common independent set. Taking one among $Y_0, \ldots, Y_k$ of maximum weight, we have an extreme common independent set. ∎

Exercises

10.36. Give an example of two matroids $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ and a weight function $w : X \to \mathbb{Z}_+$ so that there is no maximum-weight common independent set which is also a maximum-cardinality common independent set.

10.37. Reduce the problem of finding a maximum-weight common basis in two matroids to the problem of finding a maximum-weight common independent set.

10.38. Let $D = (V, A)$ be a directed graph, let $r \in V$, and let $l : A \to \mathbb{Z}_+$ be a length function. Reduce the problem of finding a minimum-length rooted tree with root r, to the problem of finding a maximum-weight common independent set in two matroids.

10.39. Let B be a common basis of the matroids $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ and let $w : X \to \mathbb{Z}$ be a weight function. Define length function $l : X \to \mathbb{Z}$ by $l(x) := w(x)$ if $x \in B$ and $l(x) := -w(x)$ if $x \notin B$.

Show that B has maximum-weight among all common bases of M_1 and M_2, if and only if $H(M_1, M_2, B)$ has no directed circuit of negative length.

10.7. Matroids and polyhedra

The algorithmic results obtained in the previous sections have interesting consequences for polyhedra associated with matroids.

Let $M = (X, \mathcal{I})$ be a matroid. The *matroid polytope* $P(M)$ of M is, by definition, the convex hull of the incidence vectors of the independent sets of M. So $P(M)$ is a polytope in $\mathbb{R}^X$.

Each vector z in $P(M)$ satisfies the following linear inequalities:

$$(41)\qquad \begin{array}{rcll} z(x) & \geq & 0 & \text{for } x \in X, \\ z(Y) & \leq & r_M(Y) & \text{for } Y \subseteq X. \end{array}$$

This follows from the fact that the incidence vector χ^Y of any independent set Y of M satisfies (41).

Note that if z is an integer vector satisfying (41), then z is the incidence vector of some independent set of M.

Edmonds [1970] showed that system (41) in fact fully determines the matroid polytope $P(M)$. It means that for each weight function $w : X \to \mathbb{R}$, the linear programming problem

$$(42)\qquad \begin{array}{lrcll} \text{maximize} & w^T z, & & & \\ \text{subject to} & z(x) & \geq & 0 & (x \in X) \\ & z(Y) & \leq & r_M(Y) & (Y \subseteq X) \end{array}$$

has an integer optimum solution z. This optimum solution necessarily is the incidence vector of some independent set of M. In order to prove this, we also consider the LP-problem dual to (42):

$$(43)\qquad \begin{array}{lrcll} \text{minimize} & \displaystyle\sum_{Y \subseteq X} y_Y r_M(Y), & & & \\ \text{subject to} & y_Y & \geq & 0 & (Y \subseteq X) \\ & \displaystyle\sum_{Y \subseteq X, x \in Y} y_Y & \geq & w(x) & (x \in X). \end{array}$$

We show:

Theorem 10.14. *If w is integer, then* (42) *and* (43) *have integer optimum solutions.*

Proof. Order the elements of X as $y_1, \ldots, y_m$ in such a way that $w(y_1) \geq w(y_2) \geq \ldots w(y_m)$. Let n be the largest index for which $w(y_n) \geq 0$. Define $X_i := \{y_1, \ldots, y_i\}$ for $i = 0, \ldots, m$ and

$$(44)\qquad Y := \{y_i \mid i \leq n; r_M(X_i) > r_M(X_{i-1})\}.$$

Then Y belongs to $\mathcal{I}$ (cf. Exercise 10.3). So $z := \chi^Y$ is an integer feasible solution of (42).

Moreover, define a vector y in $\mathbb{R}^{\mathcal{P}(X)}$ by:

$$\begin{array}{lll} y_Y & := & w(y_i) - w(y_{i+1}) \quad \text{if } Y = X_i \text{ for some } i = 1, \ldots, n-1, \\ y_Y & := & w(y_n) \quad \text{if } Y = X_n, \\ y_Y & := & 0 \quad \text{for all other } Y \subseteq X \end{array} \tag{45}$$

Then y is an integer feasible solution of (43).

We show that z and y have the same objective value, thus proving the theorem:

$$\begin{aligned} w^T z = w(Y) &= \sum_{x \in Y} w(x) = \sum_{i=1}^{n} w(y_i) \cdot (r_M(X_i) - r_M(X_{i-1})) \\ &= w(y_n) \cdot r_M(X_n) + \sum_{i=1}^{n} (w(y_i) - w(y_{i+1})) \cdot r_M(X_i) = \sum_{Y \subseteq X} y_Y r_M(Y). \end{aligned} \tag{46}$$

∎

So system (41) is totally dual integral. This directly implies:

Corollary 10.14a. *The matroid polytope $P(M)$ is determined by* (41).

Proof. Immediately from Theorem 10.14. ∎

An even stronger phenomenon occurs at intersections of matroid polytopes. It turns out that the intersection of two matroid polytopes gives exactly the convex hull of the common independent sets, as was shown again by Edmonds [1970].

To see this, we first derive a basic property:

Theorem 10.15. *Let $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ be matroids, let $w : X \to \mathbb{Z}$ be a weight function and let B be a common basis of maximum weight $w(B)$. Then there exist functions $w_1, w_2 : X \to \mathbb{Z}$ so that $w = w_1 + w_2$, and so that B is both a maximum-weight basis of M_1 with respect to w_1 and a maximum-weight basis of M_2 with respect to w_2.*

Proof. Consider the directed graph $H(M_1, M_2, B)$ with length function l as defined in Exercise 10.39. Since B is a maximum-weight basis, $H(M_1, M_2, B)$ has no directed circuits of negative length. Hence there exists a function $\phi : X \to \mathbb{Z}$ so that $\phi(y) - \phi(x) \leq l(y)$ for each arc (x, y) of $H(M_1, M_2, B)$. Using the definition of $H(M_1, M_2, B)$ and l, this implies that for $y \in B, x \in X \setminus B$:

$$\begin{array}{lll} \phi(x) - \phi(y) & \leq & -w(x) \quad \text{if } (B \setminus \{y\}) \cup \{x\} \in \mathcal{I}_1, \\ \phi(y) - \phi(x) & \leq & w(x) \quad \text{if } (B \setminus \{y\}) \cup \{x\} \in \mathcal{I}_2. \end{array} \tag{47}$$

Now define

$$\begin{array}{llllll} w_1(y) & := & \phi(y), & w_2(y) & := & w(y) - \phi(y) \quad \text{for } y \in B \\ w_1(x) & := & w(x) + \phi(x), & w_2(x) & := & -\phi(x) \quad \text{for } x \in X \setminus B. \end{array} \tag{48}$$

Then $w_1(x) \leq w_1(y)$ whenever $(B \setminus \{y\}) \cup \{x\} \in \mathcal{I}_1$. So by Exercise 10.19, B is a maximum-weight basis of M_1 with respect to w_1. Similarly, B is a maximum-weight basis of M_2 with respect to w_2. ∎

Note that if B is a maximum-weight basis of M_1 with respect to some weight function w, then also after adding a constant function to w this remains the case.

This observation will be used in showing that a theorem similar to Theorem 10.15 holds for independent sets instead of bases.

Theorem 10.16. *Let $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ be matroids, let $w : X \to \mathbb{Z}$ be a weight function, and let Y be a maximum-weight common independent set. Then there exist weight functions $w_1, w_2 : X \to \mathbb{Z}$ so that $w = w_1 + w_2$ and so that Y is both a maximum-weight independent set of M_1 with respect to w_1 and a maximum-weight independent set of M_2 with respect to w_2.*

Proof. Let U be a set of cardinality $|X| + 2$ disjoint from X. Define

$$\begin{aligned}(49)\qquad &\mathcal{J}_1 := \{Y \cup W \mid Y \in \mathcal{I}_1, W \subseteq U, |Y \cup W| \leq |X| + 1\},\\ &\mathcal{J}_2 := \{Y \cup W \mid Y \in \mathcal{I}_2, W \subseteq U, |Y \cup W| \leq |X| + 1\}.\end{aligned}$$

Then $M_1' := (X \cup U, \mathcal{J}_1)$ and $M_2 := (X \cup U, \mathcal{J}_2)$ are matroids again. Define $\tilde{w} : X \to \mathbb{Z}$ by

$$(50)\qquad \begin{array}{lcll}\tilde{w}(x) & := & w(x) & \text{if } x \in X,\\ \tilde{w}(x) & := & 0 & \text{if } x \in U.\end{array}$$

Let W be a subset of U of cardinality $|X \setminus Y| + 1$. Then $Y \cup W$ is a common basis of M_1' and M_2'. In fact, $Y \cup W$ is a maximum-weight common basis with respect to the weight function $\tilde{w}$. (Any basis B of larger weight would intersect X in a common independent set of M_1 and M_2 of larger weight than Y.)

So by Theorem 10.15, there exist functions $\tilde{w}_1, \tilde{w}_2 : X \to \mathbb{Z}$ so that $\tilde{w}_1 + \tilde{w}_2 = \tilde{w}$ and so that $Y \cup W$ is both a maximum-weight basis of M_1' with respect to $\tilde{w}_1$ and a maximum-weight basis of M_2' with respect to $\tilde{w}_2$.

Now, $\tilde{w}_1(u'') \leq \tilde{w}_1(u')$ whenever $u' \in W, u'' \in U \setminus W$. Otherwise we can replace u' in $Y \cup W$ by u'' to obtain a basis of M_1' of larger $\tilde{w}_1$-weight. Similarly, $\tilde{w}_2(u'') \leq \tilde{w}_2(u')$ whenever $u' \in W, u'' \in U \setminus W$.

Since $\tilde{w}_1(u) + \tilde{w}_2(u) = \tilde{w}(u) = 0$ for all $u \in U$, this implies that $\tilde{w}_1(u'') = \tilde{w}_1(u')$ whenever $u' \in W, u'' \in U \setminus W$. As $\emptyset \neq W \neq U$, it follows that $\tilde{w}_1$ and $\tilde{w}_2$ are constant on U. Since we can add a constant function to $\tilde{w}_1$ and subtracting the same function from $\tilde{w}_2$ without spoiling the required properties, we may assume that $\tilde{w}_1$ and $\tilde{w}_2$ are 0 on U.

Now define $w_1(x) := \tilde{w}_1(x)$ and $w_2(x) := \tilde{w}_2(x)$ for each $x \in X$. Then Y is both a maximum-weight independent set of M_1 with respect to w_1 and a maximum-weight independent set of M_2 with respect to w_2. ∎

Having this theorem, it is quite easy to derive that the intersection of two matroid polytopes has integer vertices, being incidence vectors of common independent sets.

By Theorem 10.14 the intersection $P(M_1) \cap P(M_2)$ of the matroid polytopes associated with the matroids $M_1 = (X, \mathcal{I}_1)$ and $M_2 = (X, \mathcal{I}_2)$ is determined by:

$$(51)\qquad \begin{array}{rcll} z(x) & \geq & 0 & (x \in X),\\ z(Y) & \leq & r_{M_1}(Y) & (Y \subseteq X),\\ z(Y) & \leq & r_{M_2}(Y) & (Y \subseteq X),\end{array}$$

The corresponding linear programming problem is, for any $w : X \to \mathbb{R}$:

$$(52)\qquad \begin{array}{llcll} \text{maximize} & w^T z, & & & \\ \text{subject to} & z(x) & \geq & 0 & (x \in X),\\ & z(Y) & \leq & r_{M_1}(Y) & (Y \subseteq X),\\ & z(Y) & \leq & r_{M_2}(Y) & (Y \subseteq X).\end{array}$$

Again we consider the dual linear programming problem:

$$
\text{(53)} \qquad \begin{array}{lrcll}
\text{minimize} & \displaystyle\sum_{Y \subseteq X} (y'_Y r_{M_1}(Y) + y''_Y r_{M_2}(Y)) & & & \\
\text{subject to} & y'_Y & \geq & 0 & (Y \subseteq X), \\
 & y''_Y & \geq & 0 & (Y \subseteq X), \\
 & \displaystyle\sum_{Y \subseteq X, x \in Y} (y'_Y + y''_Y) & \geq & w(x) & (x \in X).
\end{array}
$$

Now

Theorem 10.17. *If w is integer, then* (52) *and* (53) *have integer optimum solutions.*

Proof. Let Y be a common independent set of maximum weight $w(Y)$. By Theorem 10.15, there exist functions $w_1, w_2 : X \to \mathbb{Z}$ so that $w_1 + w_2 = w$ and so that Y is a maximum-weight independent set in M_i with respect to w_i $(i = 1, 2)$.

Applying Theorem 10.14 to w_1 and w_2, respectively, we know that there exist integer optimum solutions y' and y'', respectively, for problem (43) with respect to M_1, w_1 and M_2, w_2, respectively. One easily checks that y', y'' forms a feasible solution of (53). Optimality follows from:

$$
\text{(54)} \qquad \begin{aligned}
w(Z) = w_1(Z) + w_2(Z) &= \sum_{Y \subseteq X} y'_Y r_{M_1}(Y) + \sum_{Y \subseteq X} y''_Y r_{M_2}(Y) \\
&= \sum_{Y \subseteq X} (y'_Y r_{M_1}(Y) + y''_Y r_{M_2}(Y)).
\end{aligned}
$$

∎

So system (51) is totally dual integral. Again, this directly implies:

Corollary 10.17a. *The convex hull of the common independent sets of two matroids M_1 and M_2 is determined by* (51).

Proof. Directly from Theorem 10.17. ∎

Exercises

10.40. Give an example of three matroids M_1, M_2, and M_3 on the same set X so that the intersection $P(M_1) \cap P(M_2) \cap P(M_3)$ is *not* the convex hull of the common independent sets.

10.41. Derive Edmonds' matroid intersection theorem (Theorem 10.9) from Theorem 10.17.

References

[1975] G.M. Adel'son-Vel'skiĭ, E.A. Dinits, A.V. Karzanov, *Potokovye algoritmy* [Russian; Flow Algorithms], Izdatel'stvo "Nauka", Moscow, 1975.

[1974] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Massachusetts, 1974.

[1993] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows — Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[1977] K. Appel, W. Haken, Every planar map is four colorable Part I: discharging, *Illinois Journal of Mathematics* 21 (1977) 429–490.

[1977] K. Appel, W. Haken, J. Koch, Every planar map is four colorable Part II: reducibility, *Illinois Journal of Mathematics* 21 (1977) 491–567.

[1969] M.L. Balinski, Labelling to obtain a maximum matching, in: *Combinatorial Mathematics and Its Applications* (Proceedings Conference Chapel Hill, North Carolina, 1967; R.C. Bose, T.A. Dowling, eds.), The University of North Carolina Press, Chapel Hill, North Carolina, 1969, pp. 585–602.

[1957] T.E. Bartlett, An algorithm for the minimum number of transport units to maintain a fixed schedule, *Naval Research Logistics Quarterly* 4 (1957) 139–149.

[1986] M. Becker, K. Mehlhorn, Algorithms for routing in planar graphs, *Acta Informatica* 23 (1986) 163–176.

[1958] R. Bellman, On a routing problem, *Quarterly of Applied Mathematics* 16 (1958) 87–90.

[1958] C. Berge, Sur le couplage maximum d'un graphe, *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences [Paris]* 247 (1958) 258–259.

[1960] C. Berge, Les problèmes de coloration en théorie des graphes, *Publications de l'Institut de Statistique de l'Université de Paris* 9 (1960) 123–160.

[1961] C. Berge, Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind, *Wissenschaftliche Zeitschrift der Martin-Luther-Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe* 10 (1961) 114–115.

[1963] C. Berge, Some classes of perfect graphs, in: *Six Papers on Graph Theory* [related to a series of lectures at the Research and Training School of the Indian Statistical Institute, Calcutta, March-April 1963], Research and Training School, Indian Statistical Institute, Calcutta, [1963,] pp. 1–21.

[1946] G. Birkhoff, Tres observaciones sobre el algebra lineal, *Revista Facultad de Ciencias Exactas, Puras y Aplicadas Universidad Nacional de Tucuman, Serie A (Matematicas y Fisica Teorica)* 5 (1946) 147–151.

[1955] A.W. Boldyreff, Determination of the maximal steady state flow of traffic through a railroad network, *Journal of the Operations Research Society of America* 3 (1955) 443–465.

[1926] O. Borůvka, O jistém problému minimálním [Czech, with German summary; On a minimal problem], *Práce Moravské Přírodovědecké Společnosti Brno [Acta Societatis Scientiarum Naturalium Moravi[c]ae]* 3 (1926) 37–58.

[1941] R.L. Brooks, On colouring the nodes of a network, *Proceedings of the Cambridge Philosophical Society* 37 (1941) 194–197.

[1911] C. Carathéodory, Über den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen, *Rendiconti del Circolo Matematico di Palermo* 32 (1911) 193–217 [reprinted in: *Constantin Carathéodory, Gesammelte Mathematische Schriften, Band III* (H. Tietze, ed.), C.H. Beck'sche Verlagsbuchhandlung, München, 1955, pp. 78–110].

[1975] N. Christofides, *Graph Theory — An Algorithmic Approach*, Academic Press, New York, 1975.

[1976] N. Christofides, *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem*, Management Sciences Research Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1976.

[1984] R. Cole, A. Siegel, River routing every which way, but loose, in: *25th Annual Symposium on Foundations of Computer Science* (25th FOCS, Singer Island, Florida, 1984), IEEE, New York, 1984, pp. 65–73.

[1971] S.A. Cook, The complexity of theorem-proving procedures, in: *Conference Record of Third Annual ACM Symposium on Theory of Computing* (3rd STOC, Shaker Heights, Ohio, 1971), The Association for Computing Machinery, New York, 1971, pp. 151–158.

[1978] W.H. Cunningham, A.B. Marsh, III, A primal algorithm for optimum matching, [in: *Polyhedral Combinatorics — Dedicated to the Memory of D.R. Fulkerson* (M.L. Balinski, A.J. Hoffman, eds.)] *Mathematical Programming Study* 8 (1978) 50–72.

[1951a] G.B. Dantzig, Application of the simplex method to a transportation problem, in: *Activity Analysis of Production and Allocation — Proceedings of a Conference* (Proceedings Conference on Linear Programming, Chicago, Illinois, 1949; Tj.C. Koopmans, ed.), Wiley, New York, 1951, pp. 359–373.

[1951b] G.B. Dantzig, Maximization of a linear function of variables subject to linear inequalities, in: *Activity Analysis of Production and Allocation — Proceedings of a Conference* (Proceedings Conference on Linear Programming, Chicago, Illinois, 1949; Tj.C. Koopmans, ed.), Wiley, New York, 1951, pp. 339–347.

[1959] E.W. Dijkstra, A note on two problems in connexion with graphs, *Numerische Mathematik* 1 (1959) 269–271.

[1950] R.P. Dilworth, A decomposition theorem for partially ordered sets, *Annals of Mathematics* (2) 51 (1950) 161–166 [reprinted in: *The Dilworth Theorems — Selected Papers of Robert P. Dilworth* (K.P. Bogart, R. Freese, J.P.S. Kung, eds.), Birkhäuser, Boston, Massachusetts, 1990, pp. 7–12].

[1970] E.A. Dinits, Algoritm resheniya zadachi o maksimal'nom potoke v seti so stepennoĭ otsenkoĭ [Russian], *Doklady Akademii Nauk SSSR* 194 (1970) 754–757 [English translation: Algorithm for solution of a problem of maximum flow in a network with power estimation *Soviet Mathematics Doklady* 11 (1970) 1277–1280].

[1961] G.A. Dirac, On rigid circuit graphs, *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 25 (1961) 71–76.

[1965a] J. Edmonds, Maximum matching and a polyhedron with 0,1-vertices, *Journal of Research National Bureau of Standards Section B* 69 (1965) 125–130.

[1965b] J. Edmonds, Minimum partition of a matroid into independent subsets, *Journal of Research National Bureau of Standards Section B* 69 (1965) 67–72.

[1965c] J. Edmonds, Paths, trees, and flowers, *Canadian Journal of Mathematics* 17 (1965) 449–467.

[1970] J. Edmonds, Submodular functions, matroids, and certain polyhedra, in: *Combinatorial Structures and Their Applications* (Proceedings Calgary International Conference on Combinatorial Structures and Their Applications, Calgary, Alberta, 1969; R. Guy, H. Hanani, N. Sauer, J. Schönheim, eds.), Gordon and Breach, New York, 1970, pp. 69–87.

[1965] J. Edmonds, D.R. Fulkerson, Transversals and matroid partition, *Journal of Research National Bureau of Standards Section B* 69 (1965) 147–153.

[1972] J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, *Journal of the Association for Computing Machinery* 19 (1972) 248–264.

[1931] J. Egerváry, Matrixok kombinatorius tulajdonságairól [Hungarian, with German summary], *Matematikai és Fizikai Lapok* 38 (1931) 16–28 [English translation [by H.W. Kuhn]: On combinatorial properties of matrices, Logistics Papers, George Washington University, issue 11 (1955), paper 4, pp. 1–11].

[1976] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multicommodity flow problems, *SIAM Journal on Computing* 5 (1976) 691–703.

[1975] S. Even, O. Kariv, An $O(n^{2.5})$ algorithm for maximum matching in general graphs, in: *16th Annual Symposium on Foundations of Computer Science* (16th FOCS, Berkeley, California, 1975), IEEE, New York, 1975, pp. 100–112.

[1975] S. Even, R.E. Tarjan, Network flow and testing graph connectivity, *SIAM Journal on Computing* 4 (1975) 507–518.

[1894] Gy. Farkas, A Fourier-féle mechanikai elv alkalmazásai [Hungarian], *Mathematikai és Természettudományi Értesitő* 12 (1894) 457–472 [German translation, with slight alterations: J. Farkas, Über die Anwendungen des mechanischen Princips von Fourier, *Mathematische und naturwissenschaftliche Berichte aus Ungarn* 12 (1895) 263–281].

[1896] Gy. Farkas, A Fourier-féle mechanikai elv alkalmazásának algebrai alapja [Hungarian; On the algebraic foundation of the applications of the mechanical principle of Fourier], *Mathematikai és Physikai Lapok* 5 (1896) 49–54 [German translation, with some additions: Section I of: J. Farkas, Die algebraischen Grundlagen der Anwendungen des Fourier'schen Princips in der Mechanik, *Mathematische und naturwissenschaftliche Berichte aus Ungarn* 15 (1897-99) 25–40].

[1898] Gy. Farkas, A Fourier-féle mechanikai elv algebrai alapja [Hungarian], *Mathématikai és Természettudományi Értesitő* 16 (1898) 361–364 [German translation: J. Farkas, Die algebraische Grundlage der Anwendungen des mechanischen Princips von Fourier, *Mathematische und naturwissenschaftliche Berichte aus Ungarn* 16 (1898) 154–157].

[1957] G.J. Feeney, The empty boxcar distribution problem, *Proceedings of the First International Conference on Operational Research (Oxford 1957)* (M. Davies, R.T. Eddison, T. Page, eds.), Operations Research Society of America, Baltimore, Maryland, 1957, pp. 250–265.

[1955] A.R. Ferguson, G.B. Dantzig, The problem of routing aircraft, *Aeronautical Engineering Review* 14 (1955) 51–55.

[1956] L.R. Ford, Jr, *Network Flow Theory,* Paper P-923, The RAND Corporation, Santa Monica, California, [August 14,] 1956.

[1956] L.R. Ford, Jr, D.R. Fulkerson, Maximal flow through a network, *Canadian Journal of Mathematics* 8 (1956) 399–404.

[1957] L.R. Ford, Jr, D.R. Fulkerson, A simple algorithm for finding maximal network flows and an application to the Hitchcock problem, *Canadian Journal of Mathematics* 9 (1957) 210–218.

[1958] L.R. Ford, Jr, D.R. Fulkerson, A suggested computation for maximal multi-commodity network flows, *Management Science* 5 (1958) 97–101.

[1962] L.R. Ford, Jr, D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, New Jersey, 1962.

[1980] S. Fortune, J. Hopcroft, J. Wyllie, The directed subgraph homeomorphism problem, *Theoretical Computer Science* 10 (1980) 111–121.

[1984] M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, in: *25th Annual Symposium on Foundations of Computer Science* (25th FOCS, Singer Island, Florida, 1984), IEEE, New York, 1984, pp. 338–346.

[1917] G. Frobenius, Über zerlegbare Determinanten, *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin* (1917) 274–277 [reprinted in: *Ferdinand Georg Frobenius, Gesammelte Abhandlungen*, Band III (J.-P. Serre, ed.), Springer, Berlin, 1968, pp. 701–704].

[1973] H.N. Gabow, *Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs*, Ph.D. Thesis, Department of Computer Science, Stanford University, Stanford, California, 1973.

[1990] H.N. Gabow, Data structures for weighted matching and nearest common ancestors with linking, in: *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, California, 1990), Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1990, pp. 434–443.

[1986] Z. Galil, S. Micali, H. Gabow, An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs, *SIAM Journal on Computing* 15 (1986) 120–130.

[1958] T. Gallai, Maximum-minimum Sätze über Graphen, *Acta Mathematica Academiae Scientiarum Hungaricae* 9 (1958) 395–434.

[1959] T. Gallai, Über extreme Punkt- und Kantenmengen, *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Mathematica* 2 (1959) 133–138.

[1979] M.R. Garey, D.S. Johnson, *Computers and Intractability — A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, California, 1979.

[1996] G.S. Gasparian, Minimal imperfect graphs: a simple approach, *Combinatorica* 16 (1996) 209–212.

[1990] A.V. Goldberg, É. Tardos, R.E. Tarjan, Network flow algorithms, in: *Paths, Flows, and VLSI-Layout* (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, eds.), Springer, Berlin, 1990, pp. 101–164.

[1988] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum-flow problem, *Journal of the Association for Computing Machinery* 35 (1988) 921–940.

[1990] A.V. Goldberg, R.E. Tarjan, Finding minimum-cost circulations by successive approximation, *Mathematics of Operations Research* 15 (1990) 430–466.

[1873] P. Gordan, Ueber die Auflösung linearer Gleichungen mit reellen Coefficienten, *Mathematische Annalen* 6 (1873) 23–28.

[1981] M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* 1 (1981) 169–197 [corrigendum: *Combinatorica* 4 (1984) 291–295].

[1960] M.-g. Guan, Graphic programming using odd or even points [in Chinese], *Acta Mathematica Sinica* 10 (1960) 263–266 [English translation: *Chinese Mathematics* 1 (1962) 273–277].

[1943] H. Hadwiger, Über eine Klassifikation der Streckenkomplexe, *Vierteljahrsschrift der naturforschenden Gesellschaft in Zürich* 88 (1943) 133–142.

[1958] A. Hajnal, J. Surányi, Über die Auflösung von Graphen in vollständige Teilgraphen, *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae — Sectio Mathematica* 1 (1958) 113–121.

[1935] P. Hall, On representatives of subsets, *The Journal of the London Mathematical Society* 10 (1935) 26–30 [reprinted in: *The Collected Works of Philip Hall* (K.W. Gruenberg, J.E. Roseblade, eds.), Clarendon Press, Oxford, 1988, pp. 165–169].

[1960] A.J. Hoffman, Some recent applications of the theory of linear inequalities to extremal combinatorial analysis, in: *Combinatorial Analysis* (New York, 1958; R. Bellman, M. Hall, Jr, eds.) [Proceedings of Symposia in Applied Mathematics, Volume X], American Mathematical Society, Providence, Rhode Island, 1960, pp. 113–127.

[1956] A.J. Hoffman, J.B. Kruskal, Integral boundary points of convex polyhedra, in: *Linear Inequalities and Related Systems* (H.W. Kuhn, A.W. Tucker, eds.) [Annals of Mathematics Studies 38], Princeton University Press, Princeton, New Jersey, 1956, pp. 223–246.

[1973] J. Hopcroft, R.M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, *SIAM Journal on Computing* 2 (1973) 225–231.

[1961] T.C. Hu, The maximum capacity route problem, *Operations Research* 9 (1961) 898–900.

[1963] T.C. Hu, Multi-commodity network flows, *Operations Research* 11 (1963) 344–360.

[1977] D.B. Johnson, Efficient algorithms for shortest paths in sparse networks, *Journal of the Association for Computing Machinery* 24 (1977) 1–13.

[1984] N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica* 4 (1984) 373–395.

[1972] R.M. Karp, Reducibility among combinatorial problems, in: *Complexity of Computer Computations* (Proceedings of a symposium on the Complexity of Computer Computations, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1972; R.E. Miller, J.W. Thatcher, eds.), Plenum Press, New York, 1972, pp. 85–103.

[1975] R.M. Karp, On the computational complexity of combinatorial problems, *Networks* 5 (1975) 45–68.

[1974] A.V. Karzanov, Nakhozhdenie maksimal'nogo potoka v seti metodom predpotokov [Russian; Determining the maximal flow in a network by the method of preflows], *Doklady Akademii Nauk SSSR* 215 (1974) 49–52 [English translation: *Soviet Mathematics Doklady* 15 (1974) 434–437].

[1979] L.G. Khachiyan, Polinomialnyĭ algoritm v lineĭnom programmirovanii [Russian], *Doklady Akademii Nauk SSSR* 244 (1979) 1093–1096 [English translation: A polynomial algorithm in linear programming, *Soviet Mathematics Doklady* 20 (1979) 191–194].

[1980] L.G. Khachiyan, Polinomial'nye algoritmy v lineĭnom programmirovanii [Russian], *Zhurnal Vychislitel'noĭ Matematiki i Matematicheskoĭ Fiziki* 20 (1980) 51–86 [English translation: Polynomial algorithms in linear programming, *U.S.S.R. Computational Mathematics and Mathematical Physics* 20 (1980) 53–72].

[1968] D.E. Knuth, *The Art of Computer Programming, Volume I Fundamental Algorithms*, Addison-Wesley, Reading, Massachusetts, 1968.

[1916] D. Kőnig, Graphok és alkalmazásuk a determinánsok és a halmazok elméletére [Hungarian], *Mathematikai és Természettudományi Értesitő* 34 (1916) 104–119 [German translation: Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre, *Mathematische Annalen* 77 (1916) 453–465].

[1931] D. Kőnig, Graphok és matrixok [Hungarian; Graphs and matrices], *Matematikai és Fizikai Lapok* 38 (1931) 116–119.

[1932] D. Kőnig, Über trennende Knotenpunkte in Graphen (nebst Anwendungen auf Determinanten und Matrizen), *Acta Litterarum ac Scientiarum Regiae Universitatis Hungaricae Francisco-Josephinae, Sectio Scientiarum Mathematicarum [Szeged]* 6 (1932-34) 155–179.

[1948] Tj.C. Koopmans, Optimum utilization of the transportation system, in: *The Econometric Society Meeting* (Washington, D.C., 1947; D.H. Leavens, ed.) [Proceedings of the International Statistical Conferences — Volume V], 1948, pp. 136–146 [reprinted in: *Econometrica* 17 (Supplement) (1949) 136–146] [reprinted in: *Scientific Papers of Tjalling C. Koopmans*, Springer, Berlin, 1970, pp. 184–193].

[1984] M.R. Kramer, J. van Leeuwen, The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits, in: *VLSI-Theory* (F.P. Preparata, ed.) [Advances in Computing Research, Volume 2], JAI Press, Greenwich, Connecticut, 1984, pp. 129–146.

[1956] J.B. Kruskal, Jr, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proceedings of the American Mathematical Society* 7 (1956) 48–50.

[1955] H.W. Kuhn, The Hungarian method for the assignment problem, *Naval Research Logistics Quarterly* 2 (1955) 83–97.

[1976] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[1985] C.E. Leiserson, F.M. Maley, Algorithms for routing and testing routability of planar VLSI layouts, in: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing* (17th STOC, Providence, Rhode Island, 1985), The Association for Computing Machinery, New York, 1985, pp. 69–78.

[1972a] L. Lovász, A characterization of perfect graphs, *Journal of Combinatorial Theory, Series B* 13 (1972) 95–98.

[1972b] L. Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Mathematics* 2 (1972) 253–267 [reprinted as: Normal hypergraphs and the weak perfect graph conjecture, in: *Topics on Perfect Graphs* (C. Berge, V. Chvátal, eds.) [Annals of Discrete Mathematics 21], North-Holland, Amsterdam, 1984, pp. 29–42].

[1979] L. Lovász, Graph theory and integer programming, in: *Discrete Optimization I* (Proceedings Advanced Research Institute on Discrete Optimization and Systems Applications and Discrete Optimization Symposium, Banff, Alta, and Vancouver, B.C., Canada, 1977; P.L. Hammer, E.L. Johnson, B.H. Korte, eds.) [Annals of Discrete Mathematics 4], North-Holland, Amsterdam, 1979, pp. 141–158.

[1986] L. Lovász, M.D. Plummer, *Matching Theory*, Akadémiai Kiadó, Budapest [also: North-Holland Mathematics Studies Volume 121, North-Holland, Amsterdam], 1986.

[1975] J.F. Lynch, The equivalence of theorem proving and the interconnection problem, *(ACM) SIGDA Newsletter* 5:3 (1975) 31–36.

[1978] V.M. Malhotra, M.P. Kumar, S.N. Maheshwari, An $O(|V|^3)$ algorithm for finding maximum flows in networks, *Information Processing Letters* 7 (1978) 277–278.

[1985] K. Matsumoto, T. Nishizeki, N. Saito, An efficient algorithm for finding multicommodity flows in planar networks, *SIAM Journal on Computing* 14 (1985) 289–302.

[1927] K. Menger, Zur allgemeinen Kurventheorie, *Fundamenta Mathematicae* 10 (1927) 96–115.

[1980] S. Micali, V.V. Vazirani, An $O(\sqrt{|v|}|E|)$ algorithm for finding maximum matching in general graphs, in: *21st Annual Symposium on Foundations of Computer Science* (21st FOCS, Syracuse, New York, 1980), IEEE, New York, 1980, pp. 17–27.

[1784] G. Monge, Mémoire sur la théorie des déblais et des remblais, *Histoire de l'Académie Royale des Sciences* [année 1781. Avec les Mémoires de Mathématique & de Physique, pour la même Année] (2e partie) (1784) [*Histoire:* 34–38, *Mémoire:*] 666–704.

[1936] T.S. Motzkin, *Beiträge zur Theorie der linearen Ungleichungen*, Inaugural Dissertation Basel, Azriel, Jerusalem, 1936 [English translation: *Contributions to the theory of linear inequalities*, RAND Corporation Translation 22, The RAND Corporation, Santa Monica, California, 1952 [reprinted in: *Theodore S. Motzkin: Selected Papers* (D. Cantor, B. Gordon, B. Rothschild, eds.), Birkhäuser, Boston, Massachusetts, 1983, pp. 1–80]].

[1961] C.St.J.A. Nash-Williams, Edge-disjoint spanning trees of finite graphs, *The Journal of the London Mathematical Society* 36 (1961) 445–450.

[1964] C.St.J.A. Nash-Williams, Decomposition of finite graphs into forests, *The Journal of the London Mathematical Society* 39 (1964) 12.

[1967] C.St.J.A. Nash-Williams, An application of matroids to graph theory, in: *Theory of Graphs — International Symposium — Théorie des graphes — Journées internationales d'étude* (Rome, 1966; P. Rosenstiehl, ed.), Gordon and Breach, New York, and Dunod, Paris, 1967, pp. 263–265.

[1985] C.St.J.A. Nash-Williams, Connected detachments of graphs and generalized Euler trails, *The Journal of the London Mathematical Society* (2) 31 (1985) 17–29.

[1947] J. von Neumann, *Discussion of a maximum problem*, typescript dated November 15–16, 1947, Institute for Advanced Study, Princeton, New Jersey, 1947 [reprinted in: *John von Neumann, Collected Works, Volume VI* (A.H. Taub, ed.), Pergamon Press, Oxford, 1963, pp. 89–95].

[1953] J. von Neumann, A certain zero-sum two-person game equivalent to the optimal assignment problem, in: *Contributions to the Theory of Games Volume II* (H.W. Kuhn, A.W. Tucker, eds.) [Annals of Mathematics Studies 28], Princeton University Press, Princeton, New Jersey, 1953, pp. 5–12 [reprinted in: *John von Neumann, Collected Works, Volume VI* (A.H. Taub, ed.), Pergamon Press, Oxford, 1963, pp. 44–49].

[1968] A.R.D. Norman, M.J. Dowling, *Railroad Car Inventory: Empty Woodrack Cars on the Louisville and Nashville Railroad*, Technical Report 320-2926, IBM New York Scientific Center, New York, 1968.

[1981] H. Okamura, P.D. Seymour, Multicommodity flows in planar graphs, *Journal of Combinatorial Theory, Series B* 31 (1981) 75–81.

[1988] J.B. Orlin, A faster strongly polynomial minimum cost flow algorithm, in: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing* (20th STOC, Chicago, Illinois, 1988), The Association for Computing Machinery, New York, 1988, pp. 377–387.

[1993] J.B. Orlin, A faster strongly polynomial minimum cost flow algorithm, *Operations Research* 41 (1993) 338–350.

[1994] C.H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.

[1983] R.Y. Pinter, River routing: methodology and analysis, in: *Third CalTech Conference on Very Large Scale Integration* (Pasadena, California, 1983; R. Bryant, ed.), Springer, Berlin, 1983, pp. 141–163.

[1957] R.C. Prim, Shortest connection networks and some generalizations, *The Bell System Technical Journal* 36 (1957) 1389–1401.

[1942] R. Rado, A theorem on independence relations, *The Quarterly Journal of Mathematics (Oxford)* (2) 13 (1942) 83–89.

[1965] J.W.H.M.T.S.J. van Rees, Een studie omtrent de circulatie van materieel, *Spoor- en Tramwegen* 38 (1965) 363–367.

[1997] H. Ripphausen-Lipa, D. Wagner, K. Weihe, The vertex-disjoint Menger problem in planar graphs, *SIAM Journal on Computing* 26 (1997) 331–349.

[1956] J.T. Robacker, *Min-Max Theorems on Shortest Chains and Disjoint Cuts of a Network*, Research Memorandum RM-1660, The RAND Corporation, Santa Monica, California, [12 January] 1956.

[1986] N. Robertson, P.D. Seymour, Graph minors. VI. Disjoint paths across a disc, *Journal of Combinatorial Theory, Series B* 41 (1986) 115–138.

[1995] N. Robertson, P.D. Seymour, Graph minors. XIII. The disjoint paths problem, *Journal of Combinatorial Theory, Series B* 63 (1995) 65–110.

[1993] N. Robertson, P. Seymour, R. Thomas, Hadwiger's conjecture for K_6-free graphs, *Combinatorica* 13 (1993) 279–361.

[1966] B. Rothschild, A. Whinston, Feasibility of two commodity network flows, *Operations Research* 14 (1966) 1121–1129.

[1973] M. Sakarovitch, Two commodity network flows and linear programming, *Mathematical Programming* 4 (1973) 1–20.

[1991] A. Schrijver, Disjoint homotopic paths and trees in a planar graph, *Discrete & Computational Geometry* 6 (1991) 527–574.

[1994] A. Schrijver, Finding k disjoint paths in a directed planar graph, *SIAM Journal on Computing* 23 (1994) 780–788.

[1915] E. Stiemke, Über positive Lösungen homogener linearer Gleichungen, *Mathematische Annalen* 76 (1915) 340–342.

[1974] R. Tarjan, Finding dominators in directed graphs, *SIAM Journal on Computing* 3 (1974) 62–89.

[1984] R.E. Tarjan, A simple version of Karzanov's blocking flow algorithm, *Operations Research Letters* 2 (1984) 265–268.

[1950] R.L. Thorndike, The problem of the classification of personnel, *Psychometrika* 15 (1950) 215–235.

[1937] A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society* (2) 42 (1937) 230–265 [correction: 43 (1937) 544–546] [reprinted in: *The Undecidable — Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions* (M. Davis, ed.), Raven Press, Hewlett, New York, 1965, pp. 116–154].

[1947] W.T. Tutte, The factorization of linear graphs, *The Journal of the London Mathematical Society* 22 (1947) 107–111 [reprinted in: *Selected Papers of W.T. Tutte Volume I* (D. McCarthy, R.G. Stanton, eds.), The Charles Babbage Research Centre, St. Pierre, Manitoba, 1979, pp.89–97].

[1961] W.T. Tutte, On the problem of decomposing a graph into n connected factors, *The Journal of the London Mathematical Society* 36 (1961) 221–230 [reprinted in: *Selected Papers of W.T. Tutte Volume I* (D. McCarthy, R.G. Stanton, eds.), The Charles Babbage Research Centre, St. Pierre, Manitoba, 1979, pp. 214–225].

[1968] A.F. Veinott, Jr, G.B. Dantzig, Integral extreme points, *SIAM Review* 10 (1968) 371–372.

[1964] V.G. Vizing, Ob otsenke khromaticheskogo klassa p-grafa [Russian; On an estimate of the chromatic class of a p-graph], *Diskretnyĭ Analiz* 3 (1964) 25–30.

[1937] K. Wagner, Über eine Eigenschaft der ebenen Komplexe, *Mathematische Annalen* 114 (1937) 570–590.

[1995] D. Wagner, K. Weihe, A linear-time algorithm for edge-disjoint paths in planar graphs, *Combinatorica* 15 (1995) 135–150.

[1976] D.J.A. Welsh, *Matroid Theory*, Academic Press, London, 1976.

[1969] W.W. White, A.M. Bomberault, A network algorithm for empty freight car allocation, *IBM Systems Journal* 8 (1969) 147–169.

Name index

Subject index