Original Source: Zimmermann, Karl-Heinz: *Diskrete Mathematik*. Books on Demand GmbH, Norderstedt, 2006. TORE-DOI: 10.15480/882.1023 TORE-URI: https://hdl.handle.net/11420/1025

Graphs and Optimization

Graphs

Graph theory is an important tool for tackling complex problems in various scientific fields. The interdisciplinary nature of graph theory stems from the structure of graphs. Graph theory is applicable whenever a problem involves pairs of objects that are in relation to each other. Examples include road networks, electrical networks, and flow diagrams. In this chapter, fundamental concepts of graph theory are discussed.

21.1 Basic Concepts

A graph is a pair $G = (V, E)$ consisting of a non-empty set V and a set E of 2-element subsets of V. The elements of V are called vertices and the elements of E are called edges. An edge $e = \{u, v\}$ is also written as a word $e = uv$ (or $e = vu$). If an edge $e = uv$ exists, then u and v are incident with e, u and v are adjacent (or directly connected), and u and v are the endpoints of e.

In the following, only finite graphs are considered, i.e., graphs with a finite vertex set. The number of vertices of a graph G is called the order of G and the number of edges of G is called the size of G.

A graph can be represented by a diagram in which vertices are represented by points in the plane and edges by continuous line segments.
Example 21.1. The graph G with vertex set $V = \{v_1, \ldots, v_4\}$ and edge set $E = \{v_1v_3, v_2v_3, v_2v_4, v_3v_4\}$ is represented by the diagram in Fig. 21.1.

A graph $G = (V, E)$ has neither loops nor multiple edges. Loops are 1-element subsets of V, i.e., edges that are incident with only one vertex. Multiple edges are multisets of 2-element subsets of V, i.e., multiple connections between two vertices.

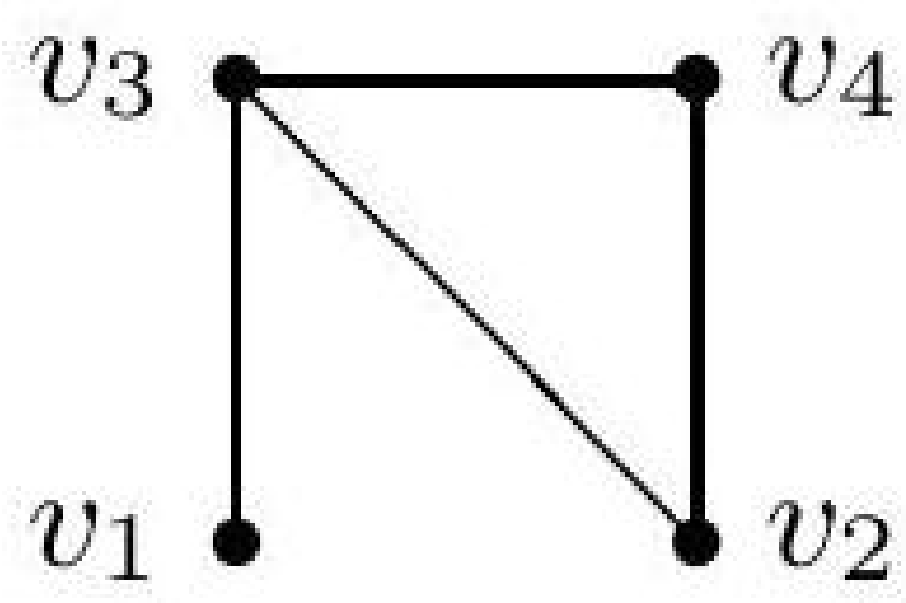


Fig. 21.1. A diagram of the graph in 21.1.

Let $G = (V, E)$ be a graph. The number of edges incident with a vertex $v \in V$ is called the degree of v and denoted by $d(v)$. If $d(v) = 0$, then v is isolated. If all vertices in G have the same degree k, then G is called k-regular.
Theorem 21.2. (Handshaking Lemma) For every graph $G = (V, E)$ it holds

$$\sum_{v \in V} d(v) = 2|E|. \tag{21.1}$$

Proof. Each edge is counted twice in the sum on the left-hand side, once for each incident vertex. □

Corollary 21.3. In every graph, the number of vertices with odd degree is always even.

Proof. According to the handshaking lemma, the sum of all degrees is even. By subtracting the sum of all even degrees, an even number is obtained, which represents the sum of all odd degrees. Therefore, the sum of all odd degrees must have an even number of summands. □

Example 21.4. Can 333 telephones be connected such that each telephone is directly connected to three other telephones? The answer is no, because the sum of the degrees of the telephone network would be odd ($333 \cdot 3$).

The degree sequence of a graph G is the decreasingly sorted sequence of the degrees of all vertices in G. For example, the graph in Fig. 21.1 has the degree sequence $(3, 2, 2, 1)$. Conversely, not every decreasing sequence of natural numbers corresponds to a graph. For example, there is no graph with degree sequence $(5, 3, 2, 2, 2, 1)$, because the sum of the degrees is odd.

Subgraphs

Let $G = (V, E)$ be a graph. A subgraph of G is a graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$. A subgraph G' of G can be viewed as the subgraph of G induced by its edge set E'. A subgraph G' of G is called the subgraph induced by its vertex set V' if every edge in G connecting two vertices in G' belongs to G'.

Example 21.5. In Fig. 21.2, a graph G along with two of its subgraphs G_1 and G_2 is shown. The subgraph G_2 is induced by the vertex set $\{b, c, d\}$, while the subgraph G_1 is not.

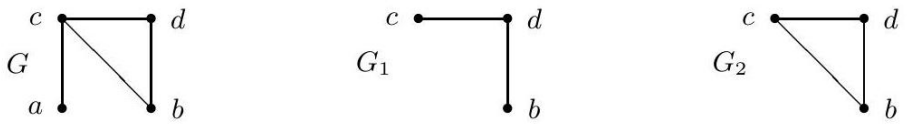


Fig. 21.2. A graph G with two subgraphs.

Isomorphisms

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. A mapping $\phi : V \to V'$ is called an isomorphism from G to G' if ϕ is bijective and for all $u, v \in V$ it holds $uv \in E$ if and only if $\phi(u)\phi(v) \in E'$. Two graphs G and G' are called isomorphic if there exists an isomorphism from G to G'.

Example 21.6. The two graphs in Fig. 21.3 are isomorphic, because an isomorphism is given by $\phi(a) = 1, \phi(b) = 2, \phi(c) = 3$ and $\phi(d) = 4$.

Isomorphic graphs have the same number of vertices (order), the same number of edges (size), and the same degree sequence. In other words, two graphs are not isomorphic if they have different order, size, or degree sequence. There exist non-isomorphic graphs with the same degree sequence (Fig. 21.4).

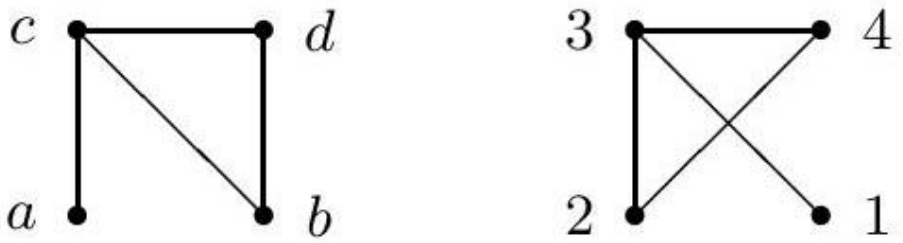


Fig. 21.3. Two isomorphic graphs.

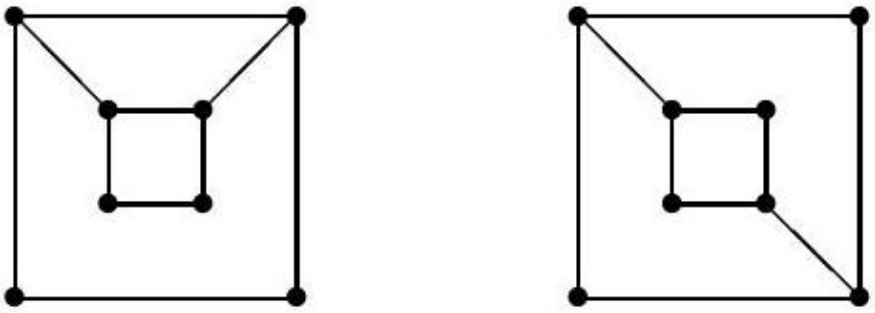


Fig. 21.4. Two non-isomorphic graphs with the same degree sequence.

Let G be a graph. An isomorphism from G to G is also called an automorphism of G.

Theorem 21.7. The set of all automorphisms of a graph forms a group under composition.

The group of all automorphisms of G is called the automorphism group of G and denoted by $\mathrm{Aut}(G)$.

Example 21.8. The automorphism group of a square (Fig. 21.5) is the dihedral group D_4 according to 15.30, consisting of four rotations id, (1234), $(1234)^2 = (13)(24)$, $(1234)^3 = (1432)$ and four reflections $(12)(34), (14)(23)$, (13), (24).

21.2 Paths, Cycles and Connectivity

Let $G = (V, E)$ be a graph. A sequence $W = (v_0, \ldots, v_k)$ of vertices $v_i \in V$ is called a path in G if for all $i, 1 \leq i \leq k$, it holds $v_{i-1}v_i \in E$. The vertex v_0 is the starting vertex and the vertex v_k is the ending vertex of W. The length of W is n, the number of its edges. A path W is called simple if W does not contain any vertex more than once.

Example 21.9. The graph in Fig. 21.6 contains, for example, the simple paths

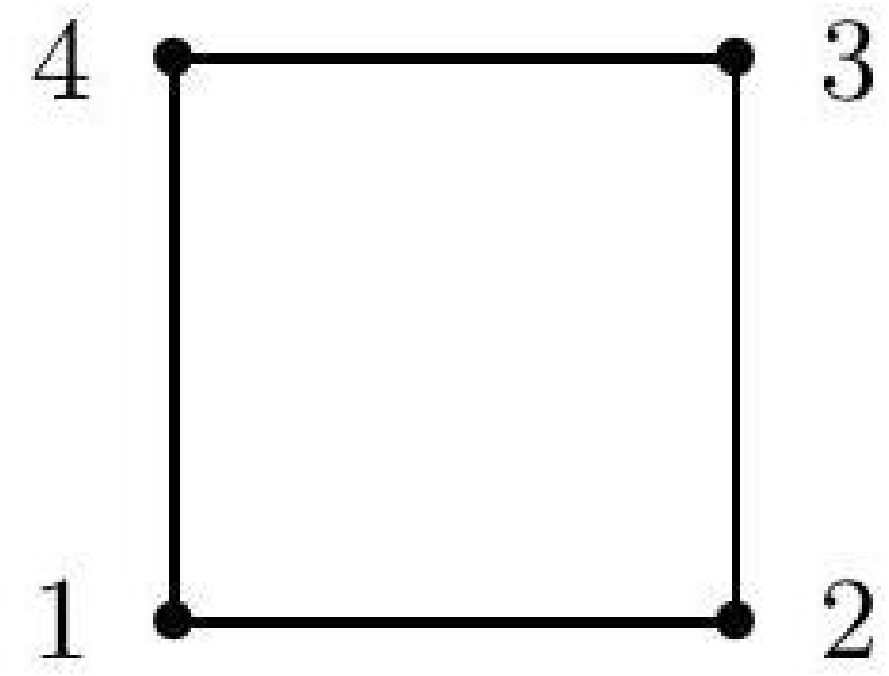


Fig. 21.5. A square.

(s, a, d, g, f, i, t) and (s, a, b, e, h, i, t) of length 6.

A cycle in G is a path in G in which the starting and ending vertices are identical. A cycle is called simple if it does not contain any vertex more than once (except for the starting and ending vertex). A back-and-forth traversal of an edge uv results in a simple cycle (u, v, u) of length 2.

Example 21.10. The graph in Fig. 21.6 contains, for example, the simple cycles (a, b, c, h, e, d, a) and (a, b, e, f, g, d, a) of length 6.

Connectivity

Let $G = (V, E)$ be a graph. Two vertices $u, v \in V$ are called connected in G, briefly $u \equiv_G v$, if $u = v$ or there exists a path from u to v. If every pair of vertices in G is connected, then G is called connected.

Lemma 21.11. Let $G = (V, E)$ be a graph. The connectivity $\equiv_G$ in G is an equivalence relation on V.

The equivalence classes of connectivity form, according to Theorem 5.5, a partition of V. The subgraphs spanned by the equivalence classes are called components of G. If G is connected, then there is only one component.

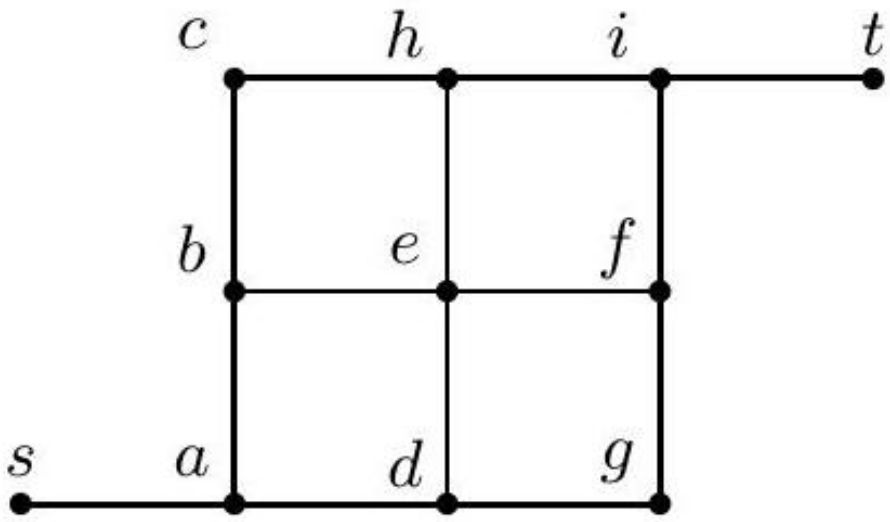


Fig. 21.6. A road network.

Example 21.12. The graph in Fig. 21.7 consists of two components, spanned by the equivalence classes $\{a, b\}$ and $\{c, d, e\}$.

Theorem 21.13. Let $G = (V, E)$ be a connected graph and $e \in E$ an edge lying on a simple cycle in G. The subgraph of G obtained by removing e is also connected.

Proof. Let G' denote the subgraph of G obtained by removing e. Let $u, v \in V$. Since G is connected, there exists a path W in G from u to v. If the path does not use the edge e, then W is also a path in G'. Otherwise, W can be modified so that instead of the edge e, the rest of the cycle on which e lies according to assumption is used. □

Distances

Let $G = (V, E)$ be a graph. For any two vertices $u, v \in V$, the distance $d_G(u, v)$ between u and v in G is defined by

$$d_G(u,v) = \begin{cases} 0 & \text{if } u = v \\ \infty & \text{if } u \text{ and } v \text{ are not connected,} \\ l & \text{if } l \text{ is the minimal length of a path in } G \text{ from } u \text{ to } v \end{cases}$$

Theorem 21.14. If G is a graph, then the distance d_G defines a metric on G.

Proof. We only show the triangle inequality. Let $u, v, w \in V$. From shortest paths between u and v as well as v and w, a path of length $d_G(u, v) + d_G(v, w)$ between u and w is obtained. For the shortest path between u and w, it holds by definition $d_G(u, w) \leq d_G(u, v) + d_G(v, w)$. □

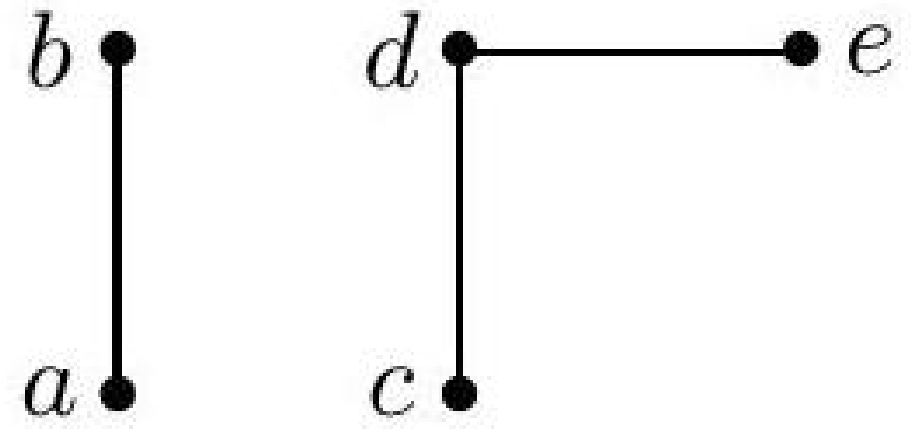


Fig. 21.7. A graph with two components.

Trees

A graph is called acyclic if it contains no simple cycle of length ≥ 3. Simple cycles of length 2 are described by back-and-forth traversals of edges and exist in every graph with at least one edge. An acyclic graph is called a forest. A connected forest is called a tree (Fig. 21.8).

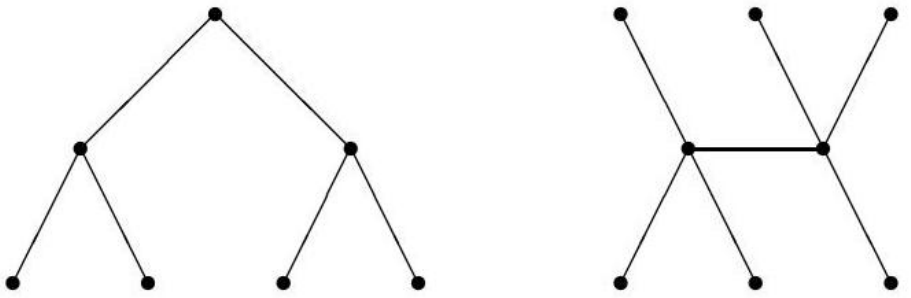


Fig. 21.8. A forest with two trees.

Theorem 21.15. A tree contains at least two vertices of degree 1.
Proof. Let G be a graph. Let vertices u and v in G be chosen such that the distance $d_G(u, v)$ is maximal. Let $W = (u, v_1, \ldots, v_{k-1}, v)$ be a shortest path in G from u to v. Assume that u has another adjacent vertex w besides v_1. Then by assumption $d_G(w, v) \leq d_G(u, v)$. Thus, there exists a shortest path from w to v that does not pass through u. Consequently, there is a contradiction, as there would be a simple cycle of length ≥ 3 in G. Thus, u has degree 1 and by

symmetry v also has degree 1. □

Theorem 21.16. For every tree $G = (V, E)$ it holds $|E| = |V| - 1$.

Proof. The case $|V| = 1$ is clear. Let G be a tree with $|V| > 1$ vertices. By Theorem 21.15, there exists a vertex v of degree 1 in G. By removing v, a subgraph $G' = (V', E')$ of G is obtained, which is also a tree. By the induction hypothesis, we get $1 = |V'| - |E'| = (|V| - 1) - (|E| - 1) = |V| - |E|$. □

Let $G = (V, E)$ be a graph. A spanning tree or spanning forest of G is a subgraph of G that is a tree and contains every vertex of G (Fig. 21.9).

Theorem 21.17. Every connected graph has a spanning tree.

Proof. Let $G = (V, E)$ be a connected graph. In the case $|E| = 1$, the statement is correct. Let $|E| > 1$. If G is a tree, then G is its own spanning tree. Otherwise, there exists a simple cycle in G. In this cycle, an arbitrary edge is removed. The resulting subgraph G' of G has $|E|-1$ edges and is connected by Theorem 21.13. Thus, G' has a spanning tree by the induction hypothesis. This spanning tree is also a spanning tree of G. □

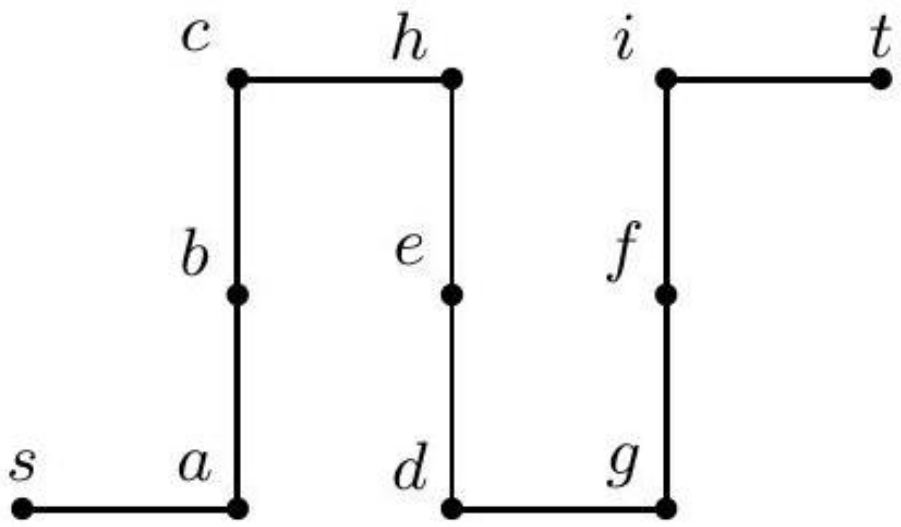


Fig. 21.9. A spanning tree for the graph in Fig. 21.6.

Theorem 21.18. A connected graph $G = (V, E)$ is a tree if and only if $|E| = |V| - 1$.

Proof. Let $|E| = |V| - 1$. Assume that G is not a tree. Then there exists a simple cycle in G. In this cycle, an arbitrary edge is removed. The resulting subgraph $G' = (V, E')$ of G is connected by Theorem 21.13. For the edge set of G', it holds $|E'| < |V| - 1$. On the other hand, G' contains a spanning tree with $|V| - 1$ edges by Theorems 21.16 and 21.17, whose edges contradictingly lie in E'. The converse has already been proven in Theorem 21.16. □

An edge e of a graph G is called a bridge or isthmus of G if G splits into two

components by removing e. For example, the graph in Fig. 21.1 has the edge v_1v_3 as its only bridge.

Bipartite Graphs

A graph $G = (V, E)$ is called bipartite if there exists a 2-partition of V into subsets V_1 and V_2 such that each edge in G has one endpoint in V_1 and one endpoint in V_2.

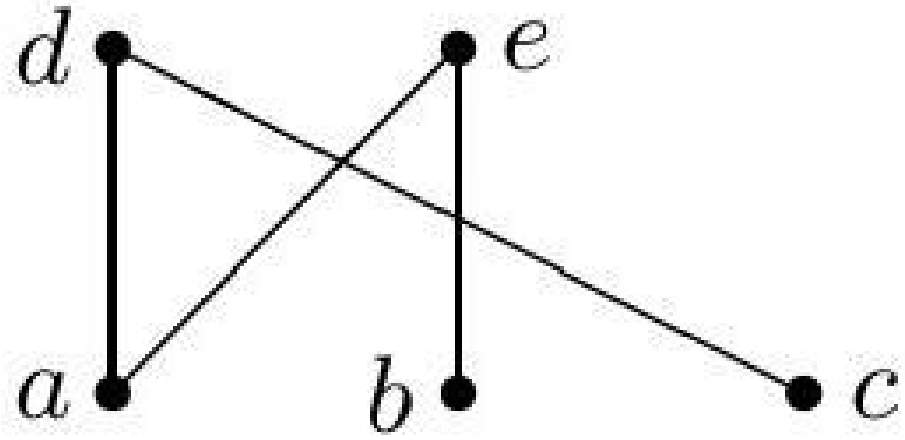


Fig. 21.10. A bipartite graph with the 2-partition $\{\{a, b, c\}, \{d, e\}\}$.

Theorem 21.19. A connected graph G is bipartite if and only if G contains no cycle of odd length.

Proof. Let $G = (V, E)$ be bipartite with the 2-partition $\{V_1, V_2\}$. Let $K = (v_0, v_1, \ldots, v_k)$ be a cycle in G. Without loss of generality, let $v_0 \in V_1$. Then $v_1 \in V_2$, $v_2 \in V_1$, $v_3 \in V_2$, etc. Thus, $v_k = v_0 \in V_1$. Hence, K has even length.

Conversely, suppose G contains no cycle of odd length. Let $v \in V$. We define

$$V_1 = \{u \in V \mid d_G(v, u) \equiv 1 \bmod 2\}$$

and

$$V_2 = \{u \in V \mid d_G(v, u) \equiv 0 \bmod 2\}.$$

It is clear that $\{V_1, V_2\}$ is a 2-partition of V. Assume there exists an edge uw in G with $u, w \in V_1$. Then there is a cycle consisting of the edge uw, a path of length $d_G(w, v)$ from w to v, and a path of length $d_G(v, u)$ from v to u. This cycle has length $1 + d_G(w, v) + d_G(v, u)$, which is contradictingly odd by assumption. Analogously, it is shown that there is no edge uv in G with $u, v \in V_2$. □

21.3 Planar Graphs

A graph is called planar if it has a diagram that can be drawn in the plane without crossings. Such a diagram is called plane. A plane diagram divides the drawing plane into faces or regions, where the drawn edges are considered as boundary lines.

Example 21.20. The hexahedron has a plane diagram that divides the drawing plane into six faces (Fig. 21.11). The region outside the plane diagram is also counted.

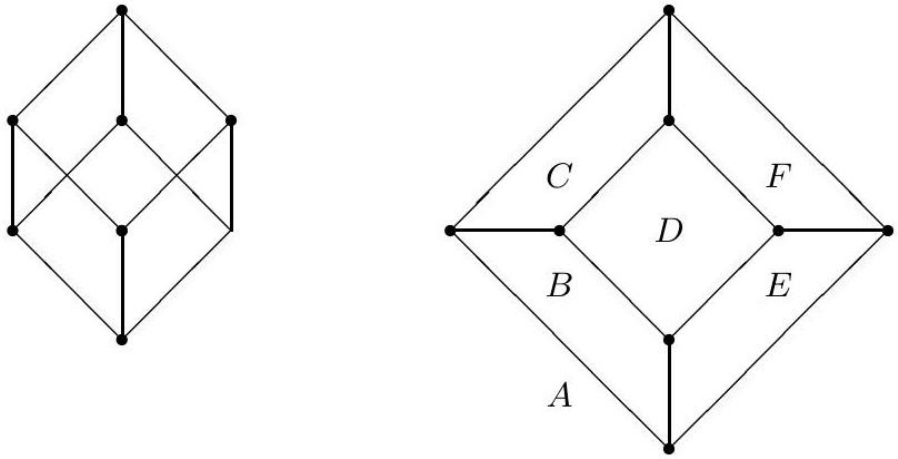


Fig. 21.11. A hexahedron and a plane diagram of the hexahedron (with faces $A, \ldots, F$).

Theorem 21.21. (Euler's polyhedron formula) For every plane diagram of a connected graph G with n vertices, m edges, and f faces it holds

$$n - m + f = 2. \tag{21.3}$$

Proof. In the case $f = 1$, G has a plane diagram with one face. Thus, G is acyclic, hence a tree. By Theorem 21.16, $m = n - 1$, from which the claim follows.

Let $f \geq 2$. Then G contains at least one simple cycle. If an edge lying on this cycle is removed, the resulting graph G' has $m - 1$ edges and $f - 1$ faces. By the induction hypothesis, $n - (m - 1) + (f - 1) = 2$, hence $n - m + f = 2$. □

Theorem 21.22. A planar connected graph G with $n \geq 3$ vertices has at most $3n - 6$ edges.

Proof. Let D be a plane diagram of $G = (V, E)$ and F the set of all faces in D. We consider a relation R between edges and faces. Let eRf if in D the edge e bounds the face f. Each edge bounds at most two faces and each face is bounded

by at least three edges. By the principle of double counting, $3|F| \leq 2|E|$. From this, Euler's polyhedron formula yields $|E| \leq 3n - 6$. □

A graph G is called complete if every two vertices in G are adjacent. A complete graph with n vertices has $\binom{n}{2}$ edges and is denoted by K_n (Fig. 21.12). The smallest non-planar complete graph is K_5. This is because this graph has 10 edges, while every planar graph with five vertices has at most $3 \cdot 5 - 6 = 9$ edges according to Theorem 21.22. The girth

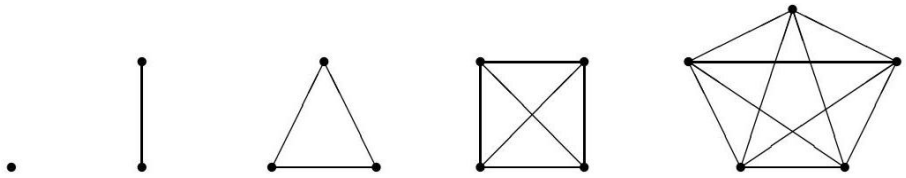


Fig. 21.12. The first five complete graphs $K_1, \ldots, K_5$.

of a graph G is the length of the shortest simple cycle in G. The girth of an acyclic graph is defined as ∞.

Theorem 21.23. A planar connected graph G with $n \geq 3$ vertices and girth $g \geq 3$ has at most $\max\{g(n-2)/(g-2), n-1\}$ edges.

Proof. If G is acyclic, then $g > n$ and the maximum is $|E| \leq n - 1$, which is correct according to Theorem 21.18. Otherwise, $g \leq n$. We distinguish two cases. First, suppose G has a bridge. Then G splits into components G_1 and G_2 by removing the bridge. Let G_i have n_i vertices and m_i edges. By the induction hypothesis, we get

$$\begin{aligned} m &= m_1 + m_2 + 1 \\ &\leq \max\left\{g\left(n_1 - 2\right)/(g-2), n_1 - 1\right\} + \max\left\{g\left(n_2 - 2\right)/(g-2), n_2 - 1\right\} \\ &\leq \max\{g(n-2)/(g-2), n-1\} \end{aligned}$$

Second, suppose G is bridgeless. Let f_i denote the number of faces in G bounded by i edges. Then

$$2m = \sum_i i f_i = \sum_{i \geq g} i f_i \geq g \sum_{i \geq g} f_i = gf.$$

With Euler's polyhedron formula, we get

$$m + 2 = n + f \leq n + \frac{2}{g} m,$$

from which $m \leq g(n-2)/(g-2)$ follows. □

A bipartite graph is called complete if for the 2-partition $\{V_1, V_2\}$ of its vertex set it holds that every vertex in V_1 is connected to every vertex in V_2. A complete bipartite graph with $|V_1| = m$ and $|V_2| = n$ is denoted by $K_{m,n}$.

The complete bipartite graph $K_{3,3}$ is not planar (Fig. 21.13). This is because this graph has 9 edges, while every planar graph with six vertices has at most $\max\{4(6-2)/(4-2), 6-2\} = 8$ edges according to Theorem 21.23.

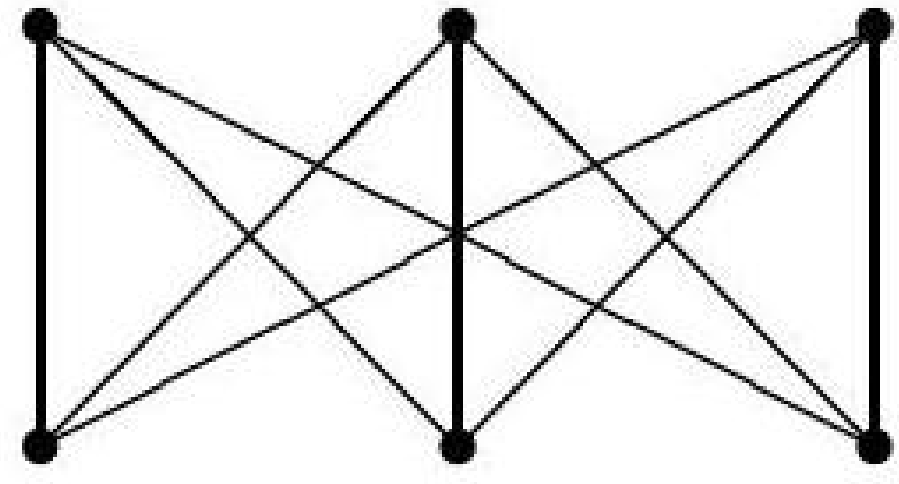


Fig. 21.13. The bipartite graph $K_{3,3}$.

We establish a necessary and sufficient criterion for the planarity of graphs. A subdivision of a graph G is a graph obtained from G by successively applying the following operation: Choose an edge uv in G and replace it by a path $(u, w_1, \ldots, w_k, v)$, where the vertices $w_i, 1 \leq i \leq k$, do not appear in G. Two graphs are called homeomorphic if they are subdivisions of the same graph (Fig. 21.14).

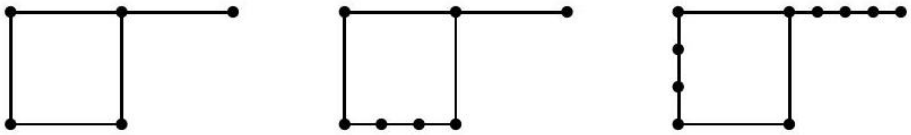


Fig. 21.14. A graph and two subdivisions of this graph.

Theorem 21.24. (Kurt Kuratowski, 1896-1980) A graph is planar if and only if it contains no subgraph homeomorphic to K_5 or $K_{3,3}$.

Example 21.25. The graph in Fig. 21.15 is not planar because it contains a subgraph homeomorphic to $K_{3,3}$.

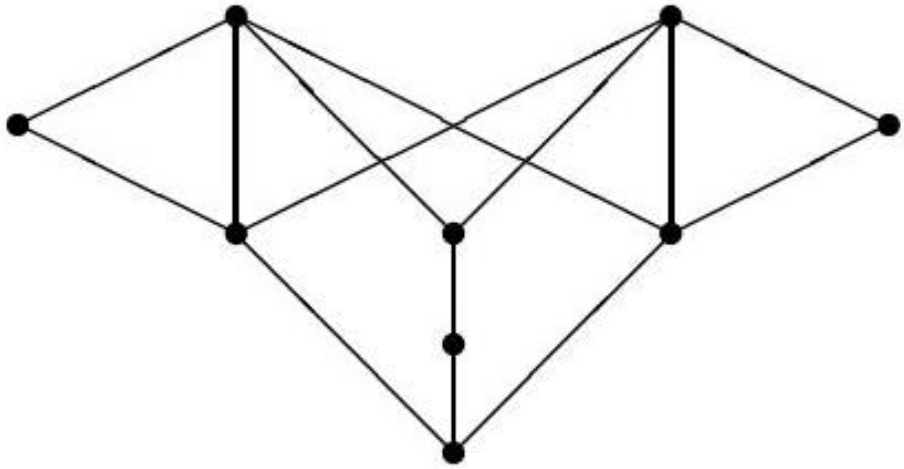


Fig. 21.15. A non-planar graph containing a subdivision of $K_{3,3}$.

21.4 Data Structures and Algorithms

Let $G = (V, E)$ be a graph with $V = \{v_1, \ldots, v_n\}$. The adjacency matrix of G is an $n \times n$ matrix $A(G) = (a_{ij})$ with

$$a_{ij} = \begin{cases} 1 \text{ if } v_i v_j \in E, \\ 0 \text{ otherwise.} \end{cases}$$

For example, the graph in Fig. 21.1 has the adjacency matrix

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

Theorem 21.26. Let A be the adjacency matrix of a graph G. The (i, j)-th entry of the k-th power of A gives the number of paths in G of length k from v_i to v_j.

Proof. We set $A^k = \left(a_{ij}^{(k)}\right)$. For $k = 1$, the statement is clear. For $k \geq 1$, by definition

$$a_{ij}^{(k+1)} = \sum_{l=1}^{n} a_{il}^{(k)} a_{lj},$$

where $a_{ij}^{(k)}$ is the number of paths in G of length k from v_i to v_j by the induction hypothesis. The term $\sum_{l=1}^{n} a_{il}^{(k)} a_{lj}$ describes the number of paths in G of length $k + 1$ from v_i to v_j that consist of a path of length k from v_i to v_l and an edge $v_l v_j$, summed over all appropriate intermediate vertices. No other paths of length k from v_i to v_j exist. Thus, the statement is proven. □

Let $G = (V, E)$ be a graph with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. The incidence matrix of G is an $n \times m$ matrix $B(G) = (b_{ij})$ with

$$b_{ij} = \begin{cases} 1 \text{ if } v_i \text{ is incident with } e_j, \\ 0 \text{ otherwise.} \end{cases}$$

For example, the graph in Fig. 21.1 has the incidence matrix ($e_1 = v_1v_3$, $e_2 = v_2v_3$, $e_3 = v_2v_4$, $e_4 = v_3v_4$)

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

An incidence matrix contains exactly two ones in each column and in each row as many ones as the degree of the corresponding vertex indicates.

A precursor to the matrix representation of graphs is the representation by lists. An adjacency list of a graph $G = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$ consists of lists $L_1, \ldots, L_n$, where list L_i contains the vertices adjacent to v_i. The graph in Fig. 21.1 is represented by the following adjacency list: $L_1 : v_3, L_2 : v_3, v_4, L_3 : v_1, v_2, v_4$ and $L_4 : v_2, v_4$.

The storage requirement (number of storage cells) for a graph $G = (V, E)$ depends on the data structure in which it is stored. The storage requirement for an adjacency matrix is $O\left(|V|^2\right)$, for an incidence matrix $O(|V||E|)$ and for an adjacency list $O(|V| + |E|)$. An algorithm for a graph G must read at least the vertices and edges of G. For this, the time requirement (number of computational operations) is $O(|V| + |E|)$. Therefore, a graph algorithm with linear time requirement $O(|V| + |E|)$ must represent the graph through an adjacency list (or a similarly space-efficient data structure).

We present two algorithms that construct a spanning tree of a graph. Let $G = (V, E)$ be a connected graph, $v^+$the set of vertices adjacent to $v \in V$ and $v_0 \in V$ fixed. Algorithm 21.1 uses a list L where elements are appended at the end and removed from the front. The vertices of the graph are traversed in breadth, i.e., after each vertex, all its adjacent vertices are processed first. This is managed by a list organized as a queue ("first-in, first-out"). This search principle is called breadth-first search.

Algorithm 21.2 uses a list L accessible only at one end. The vertices of the graph are traversed in depth, i.e., after each vertex, the adjacent vertices of its adjacent vertices are processed. This is managed by a list organized as a stack ("last-in, first-out"). This search principle is called depth-first search. If the graph is represented by an adjacency list, the running time of both algorithms is $O(|V| + |E|)$. This is because in both cases the adjacency list is processed exactly once (Fig. 21.16).

```
Algorithm 21.1 BreadthSearch $(G)$
Input: Graph $G=(V, E)$
Output: Sequence of all vertices in $G$.
    choose starting vertex $v_{0} \in V$
    $L:=\left\{v_{0}\right\}$
```

```
    $V^{\prime}:=\left\{v_{0}\right\}$
    $E^{\prime}:=\emptyset$
    while $L \neq \emptyset$ do
        remove the first vertex $v$ from $L$
        print $v$
        for all $w \in v^{+}$and $w$ has not yet been in $L$ do
            if $w \notin V^{\prime}$ then
                $V^{\prime}:=V^{\prime} \cup\{w\}$
                $E^{\prime}:=E^{\prime} \cup\{v w\}$
                append $w$ to the end of $L$
            end if
        end for
    end while
```

```
Algorithm 21.2 DepthSearch ( $G$ )
Input: Graph $G=(V, E)$
Output: Sequence of all vertices in $G$.
    choose starting vertex $v_{0} \in V$
    $L:=\left\{v_{0}\right\}$
    $V^{\prime}:=\left\{v_{0}\right\}$
    $E^{\prime}:=\emptyset$
    while $L \neq \emptyset$ do
        $v \in L$ is the top element of the list
        if there exists $w \in v^{+}$ and $w$ has not yet been in $L$ then
            insert $w$ into $L$
            $V^{\prime}:=V^{\prime} \cup\{w\}$
            $E^{\prime}:=E^{\prime} \cup\{v w\}$
        else
            remove $v$ from $L$
            print $v$
        end if
    end while
```

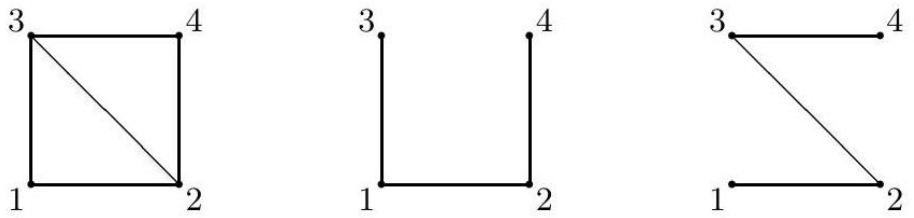
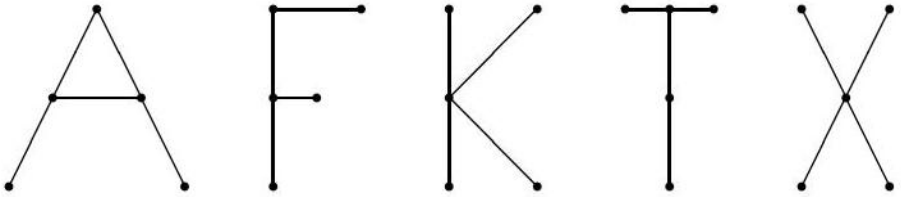


Fig. 21.16. A graph and two spanning trees of the graph, the first obtained by breadth-first search, the second by depth-first search, where the smallest vertex is chosen first in each case.

Self-test Problems

21.1. Prove the handshaking lemma using the principle of double counting.
21.2. Does there exist a graph with degree sequence $(3, 2, 2, 2)$?
21.3. What shape does a connected 2-regular graph have?
21.4. Draw a graph whose vertices are the 2-element subsets of $\underline{5}$ and two vertices are connected by an edge if the corresponding 2-element subsets are disjoint. This graph is called the Petersen graph. Is the Petersen graph planar?
21.5. A tree has three vertices of degree 3 and four vertices of degree 2. The remaining vertices are of degree 1. How many vertices of degree 1 are there?
21.6. Cut out the two diagonally opposite squares from a chessboard. Can this board (with 62 squares) be completely covered with dominoes, each covering two squares?
21.7. Prove Theorem 21.7.
21.8. Prove Lemma 21.11.
21.9. Give an infinite graph that is isomorphic to one of its proper subgraphs.
21.10. Which of the following graphs are isomorphic to each other?



21.11. Show that the graphs in Fig. 21.4 are not isomorphic.
21.12. Draw diagrams of all graphs with four vertices
21.13. Let $G = (V, E)$ be a graph. The complement of G is a graph $\bar{G} = (V, \bar{E})$ with edge set $\bar{E} = \binom{V}{2} \setminus E$. Consider all graphs with four vertices. Which of these graphs are self-complementary, i.e., isomorphic to their own complement?
21.14. Determine all spanning trees of K_4.
21.15. Let $n \geq 2$ be a natural number. Let $W_n = (\{0, 1\}^n, E)$ be the graph of the n-dimensional cube with $uv \in E$ if and only if there is exactly one $i, 1 \leq i \leq n$, such that $u_i \neq v_i$. Show that W_n contains a Hamiltonian cycle.
21.16. A simple cycle in a graph G is called Eulerian if the path contains every edge of G. A graph G is called Eulerian if G contains an Eulerian cycle. Show that a graph G is Eulerian if and only if every vertex in G has even degree.
21.17. Which of the graphs K_n and $K_{m,n}$ are Eulerian?
21.18. Specify an algorithm that tests whether a graph is a tree.

Networks

In this chapter, fundamental algorithms for networks are presented. These concern the construction of shortest paths, minimum spanning trees, maximum flows, minimum vertex and edge cutsets, minimum vertex covers, and maximum or complete matchings.

22.1 Shortest Paths

In this section, algorithms are presented that in a road network determine shortest paths between any two vertices or from one vertex to all others.

A (directed) road network is a pair (D, ω) consisting of a digraph $D = (V, E)$ and a cost function $\omega : E \to \mathbb{R}$ on the edges of D. The length of a directed path $W = (v_0, \ldots, v_k)$ in D is the sum of the costs of its edges

$$\omega(W) = \sum_{i=0}^{k-1} \omega\left(v_i v_{i+1}\right) \tag{22.1}$$

Let $u, v \in V$. A shortest path in D from u to v is a directed path in D from u to v with minimal length. For the distance between the vertices of a road network, we have

$$d_D(u,v) = \begin{cases} 0 & \text{if } u = v \\ \infty & \text{if no path from } u \text{ to } v \text{ exists in } D, \\ l & \text{if } l \text{ is the length of a shortest path in } D \text{ from } u \text{ to } v. \end{cases}$$

Theorem 22.1. If (D, ω) is a road network, then the distance d_D defines a metric.

Example 22.2. In the road network of Fig. 22.1, (v_1, v_2, v_3, v_4) is a shortest path from v_1 to v_4 of length 6.

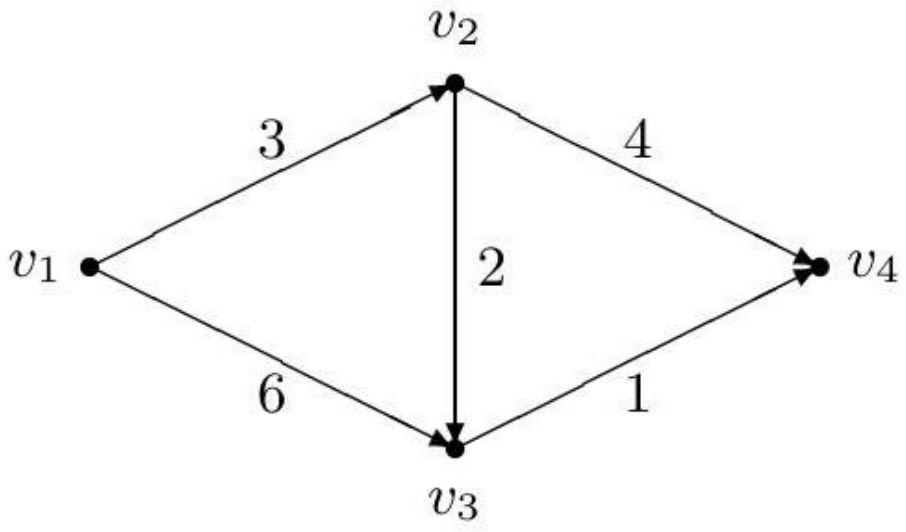


Fig. 22.1. A road network.

Shortest Paths Between Any Two Vertices

First, we treat an algorithm by R. Floyd and S. Warshall that determines the distances between any two vertices in a road network. Negative edge weights are allowed, but no cycles of negative length. A cycle of negative length can be traversed arbitrarily often and thus has arbitrarily small length. Let (D, ω) be a road network without cycles of negative length. The vertex set of the road network is denoted by $V = \{v_1, \ldots, v_n\}$.

```
Algorithm 22.1 Floyd-Warshall $(D, \omega)$
Input: Road network $(D, \omega)$ with $D=(V, E), V=\left\{v_{1}, \ldots, v_{n}\right\}$
Output: Shortest paths between all vertices of the road network
    for $i=1$ to $n$ do
        for $j=1$ to $n$ do
            if $v_{i} v_{j} \in E$ then
                $d^{(0)}\left(v_{i}, v_{j}\right):=\omega\left(v_{i} v_{j}\right)$
            else if $v_{i}=v_{j}$ then
                $d^{(0)}\left(v_{i}, v_{j}\right):=0$
            else
                $d^{(0)}\left(v_{i}, v_{j}\right):=\infty$
            end if
        end for
    end for
    for $k=1$ to $n$ do
        for $i=1$ to $n$ do
            for $j=1$ to $n$ do
                $d^{(k)}\left(v_{i}, v_{j}\right):=\min \left\{d^{(k-1)}\left(v_{i}, v_{j}\r
            end for
        end for
    end for
```

Theorem 22.3. (Floyd-Warshall) Algorithm 22.1 computes the distance $d_D(v_i, v_j) = d^{(n)}(v_i, v_j)$ for all vertices v_i, v_j in a road network (D, ω). The time requirement of the algorithm is $O\left(|V|^3\right)$.

Proof. In each iteration of the triply nested loop, a distance matrix $D_k = \left(d^{(k)}(v_i, v_j)\right)$ is computed. We show that $d^{(k)}(v_i, v_j)$ is the length of a shortest path from v_i to v_j that uses only $v_1, \ldots, v_k$ (except for v_i and v_j). For $k = 0$, the statement is clear. Let $k \geq 1$. We consider a shortest path in D from v_i to v_j that uses at most $v_1, \ldots, v_k$. If v_k is not used, then this path has length $d^{(k-1)}(v_i, v_j)$ by the induction hypothesis. Otherwise, this path consists of a path W_1 from v_i to v_k, a path W_2 from v_k to v_k, and a path W_3 from v_k to v_j, where W_1 and W_3 use only $v_1, \ldots, v_{k-1}$. Since D contains no cycles of negative length by assumption, the length of this path is given by the sum of the lengths of W_1 and W_3. Thus, by the induction hypothesis, $d^{(k)}(v_i, v_j) = d^{(k-1)}(v_i, v_k) + d^{(k-1)}(v_k, v_j)$. With $k = n$, the first claim follows. For the time requirement, the triply nested loop is decisive.

Example 22.4. For the road network in Fig. 22.1, Floyd-Warshall yields the following first and last distance matrices:

$$D_0 = \begin{pmatrix} 0 & 3 & 6 & \infty \\ \infty & 0 & 2 & 4 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}, \quad D_4 = \begin{pmatrix} 0 & 3 & 5 & 6 \\ \infty & 0 & 2 & 3 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Shortest Paths from One Vertex to All Others

Finally, an algorithm by E.W. Dijkstra is presented that computes the distances from a fixed vertex to all other vertices in a road network. Let (D, ω) be a road network, v_0 a vertex in D, and $\omega(e) \geq 0$ for every edge e.

Theorem 22.5. (Dijkstra) Algorithm 22.2 computes the distance $d_D(v_0, v) = d(v)$ from v_0 to all other vertices v in a road network (D, ω). The time requirement of the algorithm is $O\left(|V|^2\right)$.

Proof. The following loop invariant holds:

- For every vertex $v \in S$, $d(v) = d_D(v_0, v)$.
- For every vertex $v \in V \backslash S$, $d(v)$ is the length of a shortest path from v_0 to v that uses only vertices in S (except for the endpoint v).
  After termination, $S = V$ and thus the first statement is clear. The time requirement is determined by the while loop, which is executed $|V|$ times and requires at most $|V|$ steps in each iteration.

```
Algorithm 22.2 Dijkstra $\left(D, \omega, v_{0}\right)$
Input: Road network $(D, \omega)$ with $D=(V, E)$ and starting vertex $v_{0} \in V$
Output: Shortest paths from $v_{0}$ to all other vertices
    $d\left(v_{0}\right):=0$
    $S:=\emptyset$
    for all $v \in V \backslash\left\{v_{0}\right\}$ do
        $d(v):=\infty$
    end for
    while $S \neq V$ do
            choose $v \in V \backslash S$ with $d(v)=\min \{d(u) \mid u \in V \backslash S\}
            $S:=S \cup\{v\}$
            for all $u \in V \backslash S$ do
                $d(u):=\min \{d(u), d(v)+\omega(v u)\}$
            end for
    end while
```

Example 22.6. For the road network in Fig. 22.1 with starting vertex v_1, Dijkstra initially yields $S = \emptyset, d(v_1) = 0$ and $d(v_2) = d(v_3) = d(v_4) = \infty$. In the first step, we get $S = \{v_1\}, d(v_1) = 0, d(v_2) = 3, d(v_3) = 6$ and $d(v_4) = \infty$. In the second step, we obtain $S = \{v_1, v_2\}, d(v_1) = 0, d(v_2) = 3$, $d(v_3) = 5$ and $d(v_4) = 7$. In the third step, we compute $S = \{v_1, v_2, v_3\}$, $d(v_1) = 0, d(v_2) =$

$3, d(v_3) = 5, d(v_4) = 6$ and in the last step we have $S = \{v_1, v_2, v_3, v_4\}, d(v_1) = 0, d(v_2) = 3, d(v_3) = 5, d(v_4) = 6$.

22.2 Minimum Spanning Trees

In this section, an algorithm by J. Kruskal (1956) is presented that constructs a spanning tree with minimal cost for a weighted graph.

Let (G, ω) be a road network consisting of a graph $G = (V, E)$ and a cost function $\omega : E \to \mathbb{R}$ on the edges of G. The cost of a subgraph $G' = (V', E')$ of G is the sum of the costs of its edges

$$\omega(G') = \sum_{e \in E'} \omega(e) \tag{22.3}$$

A minimum spanning tree of a road network (G, ω) is a spanning tree of G that has minimal cost among all spanning trees of G (Fig. 22.2).

The number of spanning trees of a graph is so large that it is worthwhile to search for good algorithms to determine minimum spanning trees.

Theorem 22.7. (Arthur Cayley, 1821-1895) The number of spanning trees of the complete graph K_n is n^{n-2}.

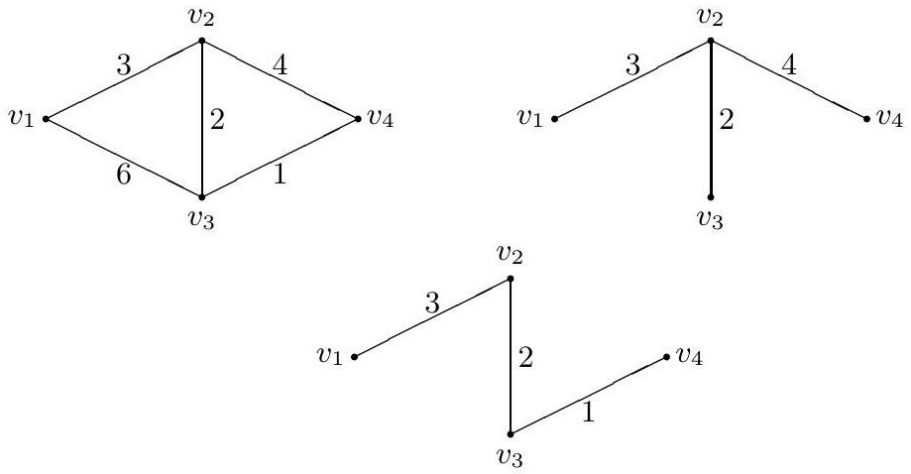


Fig. 22.2. A road network with two spanning trees, the first spanning tree has cost 9 and the second spanning tree has minimal cost 6.

Proof. Let $\{v_1, \ldots, v_n\}$ be the vertex set of K_n. Let A be a k-element subset of $\{v_1, \ldots, v_n\}$. Let $T_{n,k}$ be the number of forests with vertex set $\{v_1, \ldots, v_n\}$ consisting of k trees such that the vertices of A lie in different trees.

Let $A = \{v_1, \ldots, v_k\}$. Let G be a forest and let vertex v_1 be adjacent to i vertices. If vertex v_1 is removed, then $T_{n-1,k-1+i}$ forests with $A = \{v_2, \ldots, v_k\}$ are obtained. It follows

$$T_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} T_{n-1,k-1+i}. \tag{22.4}$$

Setting $T_{0,0} = 1$ and $T_{n,0} = 0$ for $n > 0$, we obtain by complete induction

$$T_{n,k} = kn^{n-k-1}. \tag{22.5}$$

In particular, $T_{n,1} = n^{n-2}$ is the number of spanning trees of K_n. □

Kruskal's Algorithm

Theorem 22.8. (Kruskal) Let (G, w) be a road network on a connected graph $G = (V, E)$. Algorithm 22.3 computes a minimum spanning tree for the road network (G, ω).

To prove the theorem, a criterion for the minimality of spanning trees is needed. MinSpannbaum has time complexity $O(|E| \log |E|) = O(|E| \log |V|)$ if the edges of the graph are sorted by their costs.

Example 22.9. For the road network in Fig. 22.2, MinSpannbaum successively yields the edges v_3v_4, v_2v_3 and v_1v_2. The spanning tree spanned by these edges is minimal with cost 6.

```
Algorithm 22.3 MinSpannbaum( $G, \omega$ )
Input: Road network $(G, \omega), G=(V, E)$ connected
Output: Edges of a minimum spanning tree
    $E^{\prime}:=\emptyset$
    while $E \neq \emptyset$ do
        choose $e \in E$ with $\omega(e)=\min \left\{\omega\left(e^{\prime}\right) \mid e^{\
        $E:=E \backslash\{e\}$
        if the graph spanned by $E^{\prime} \cup\{e\}$ is acyclic then
            $E^{\prime}:=E^{\prime} \cup\{e\}$
        end if
        return $E^{\prime}$
    end while
```

A Criterion for Minimum Spanning Trees

Let (G, ω) be a road network and $G = (V, E)$ connected, let $G' = (V, E')$ be a spanning tree of G. If we add an edge $e = uv \in E \backslash E'$ to the spanning tree, the resulting subgraph contains a cycle. This is because in G' there is a path between u and v, which together with the new edge e forms a cycle. The edge e is called a chord of G' and the cycle formed with e, $K_{G'}(e)$, is the fundamental cycle of the chord e (Fig. 22.3).

Theorem 22.10. Let (G, ω) be a road network and G connected. A spanning tree G' of G is minimal if and only if for every chord e of G' it holds $\omega(e) \geq \omega(e')$ for all edges e' in $K_{G'}(e)$.

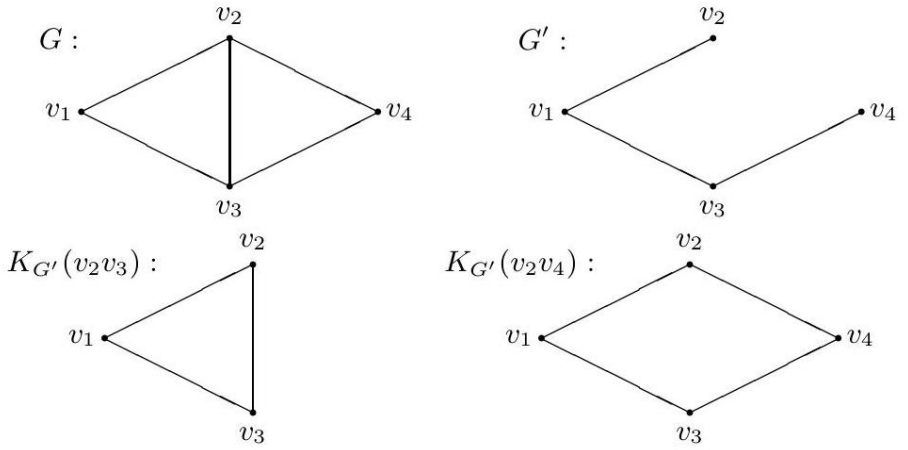


Fig. 22.3. A graph G with a spanning tree G' and fundamental cycles $K_{G'}(v_2v_3)$ and $K_{G'}(v_2v_4)$.

Proof. Let G' be minimal. Assume there exists a chord e of G' and an edge $e' \in K_{G'}(e)$ with $\omega(e) < \omega(e')$. We remove e' from G' and add e. This results in a spanning tree of G with lower cost, contradicting the assumption.

Conversely, suppose $\omega(e) \geq \omega(e')$ for every chord e of G' and all $e' \in K_{G'}(e)$. Let G_0 be a minimum spanning tree of G. We show that G' and G_0 have the same cost, thus G' is also minimal. If every edge of G_0 is also in G', then $G' = G_0$. Otherwise, there exists an edge e in G_0 not in G' and thus a chord of G'. By removing e, G_0 splits into two components and by adding an edge e' of the fundamental cycle $K_{G'}(e)$, a spanning tree G_1 is obtained. Since G_0 is minimal, $\omega(e) \leq \omega(e')$. On the other hand, e is a chord of G' and thus $\omega(e) \geq \omega(e')$. Therefore, $\omega(e) = \omega(e')$ and thus G_1 is also a minimum spanning tree. However, G_1 shares one more edge with G' than G_0. By complete induction, it follows that G' is also minimal.

Finally, Theorem 22.8 is shown.

Proof. The subgraph G' computed by MinSpannbaum contains all vertices of G, is acyclic and connected, thus a spanning tree of G. Furthermore, in the algorithm, edges are chosen such that the condition for chords in Theorem 22.10 is satisfied. Therefore, the computed spanning tree is minimal.

22.3 Maximum Flows

In this section, an algorithm by L.R. Ford and D.R. Fulkerson (1956) is presented that constructs a maximum flow from a source to a sink in a weighted graph.

A flow network is a quadruple $N = (D, \kappa, q, s)$ consisting of a digraph $D = (V, E)$, a cost function $\kappa : E \to \mathbb{R}_0^+$ called capacity, and vertices q and s in D such that there exists a directed path in D from q to s. The vertices in D

different from source and sink are called internal vertices (Fig. 22.4).

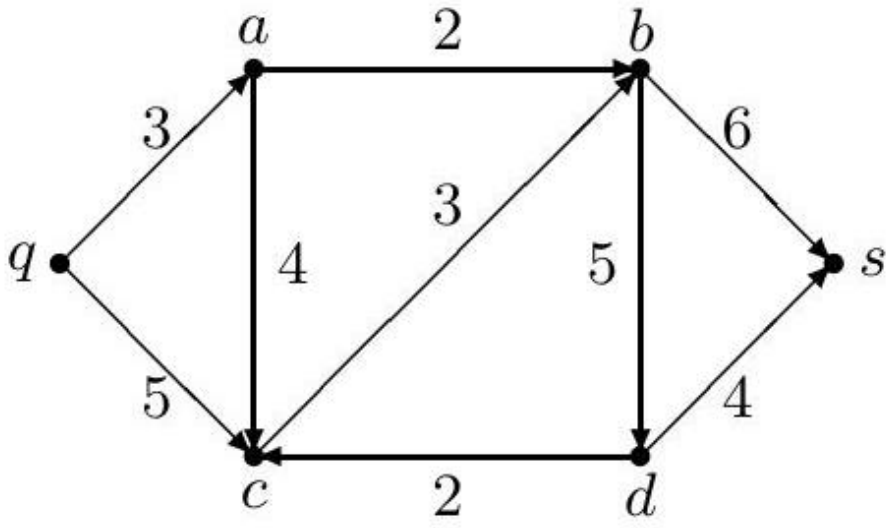


Fig. 22.4. A flow network.

Flows in Flow Networks

Let $N = (D, \kappa, q, s)$ be a flow network. A flow on N is a cost function $f : E \to \mathbb{R}_0^+$ with the following properties:

- $f \leq \kappa$, i.e., $f(e) \leq \kappa(e)$ for all edges $e \in E$.
- For every internal vertex $v \in V$ it holds

$$\sum_{w \in v^+} f(vw) = \sum_{u \in v^-} f(uv). \tag{22.6}$$

The first condition states that the flow does not exceed the capacity. The second condition means that the same amount flows out of every internal vertex as flows in. This condition is called Kirchhoff's condition in analogy to an electrical analog (Fig. 22.5).

Lemma 22.11. For every flow f on a flow network $N = (D, \kappa, q, s)$ it holds

$$\sum_{w \in q^+} f(qw) - \sum_{u \in q^-} f(uq) = \sum_{u \in s^-} f(us) - \sum_{w \in s^+} f(sw). \tag{22.7}$$

Proof. It holds

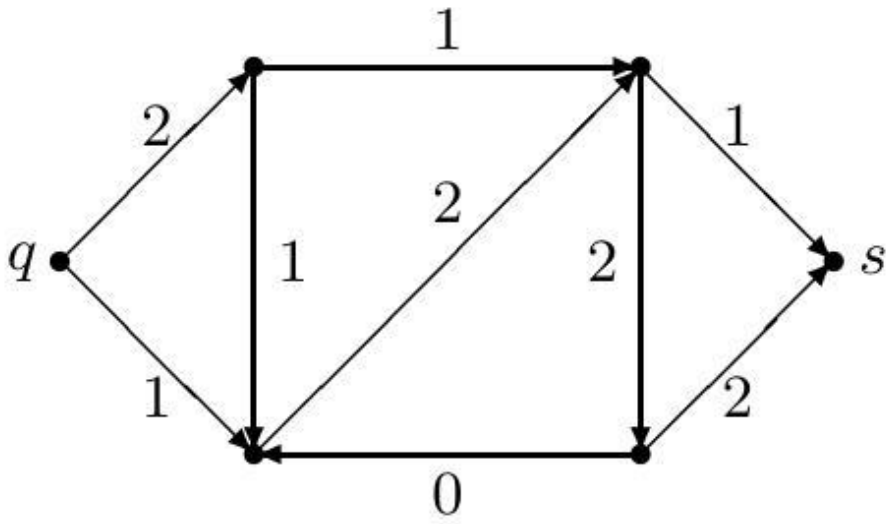


Fig. 22.5. A flow on the flow network from Fig. 22.4.

$$\sum_{v\in V}\sum_{w\in v^+} f(vw) = \sum_{v\in V}\sum_{u\in v^-} f(uv),$$

because both sides sum over all edges. From this, Kirchhoff's law implies

$$\begin{aligned} 0 &= \sum_{v\in V}\left(\sum_{w\in v^+} f(vw) - \sum_{u\in v^-} f(uv)\right) \\ &= \sum_{v\in\{q,s\}}\left(\sum_{w\in v^+} f(vw) - \sum_{u\in v^-} f(uv)\right). \end{aligned}$$

□

The number (22.7) is called the value of the flow f and denoted by $\Phi(f)$. It characterizes the total flow through the flow network. A flow f is called maximum on N if for every flow f' on N it holds $\Phi(f') \le \Phi(f)$.

Cuts in Flow Networks

Let $N = (D, \kappa, q, s)$ be a flow network. A subset S of V is called a cut in N if S contains the source q but not the sink s. The capacity of a cut S in N is the number

$$\kappa(S) = \sum_{\substack{e^-\in\bar{S}\\ e^+\in\bar{S}}} \kappa(e) \tag{22.8}$$

The capacity of a cut S is the sum of the capacities of edges whose starting vertex is in S and whose ending vertex is in $\bar{S} = V \backslash S$. A cut S in N is called minimal if for every cut S' in N it holds $\kappa(S) \leq \kappa(S')$. A minimal cut always exists since there are only finitely many cuts in a flow network.

Example 22.12. For the flow network in Fig. 22.4, some cuts are given in the following table:

S	$\{q\}$	$\{q,a\}$	$\{q,a,c\}$	$\{q,a,c,b\}$
$\kappa(S)$	8	11	5	11

Lemma 22.13. Let $N = (D, \kappa, q, s)$ be a flow network. For every flow f on N and every cut S in N it holds

$$\Phi(f) \leq \kappa(S) \tag{22.9}$$

Proof. It holds

$$\begin{aligned}
\Phi(f) &= \sum_{w \in q^+} f(qw) - \sum_{u \in q^-} f(uq) \\
&= \sum_{v \in S} \left(\sum_{w \in v^+} f(vw) - \sum_{u \in v^-} f(uv) \right) \\
&= \sum_{\substack{vw \in E \\ v,w \in S}} f(vw) + \sum_{\substack{vw \in E \\ v \in S, w \in \bar{S}}} f(vw) - \sum_{\substack{uv \in E \\ u,v \in S}} f(uv) - \sum_{\substack{uv \in E \\ v \in S, u \in \bar{S}}} f(uv) \\
&= \sum_{\substack{vw \in E \\ v \in S, w \in \bar{S}}} f(vw) - \sum_{\substack{uv \in E \\ v \in S, u \in \bar{S}}} f(uv) \\
&\leq \sum_{\substack{vw \in E \\ v \in S, w \in \bar{S}}} f(vw) \leq \kappa(S)
\end{aligned}$$

Lemma 22.14. In every flow network N there exists a maximum flow.
Proof. By Lemma 22.13, for every flow f on N it holds

$$\Phi(f) \leq \sum_{vw \in E} \kappa(vw)$$

Thus, the maximum flow value $\Phi = \sup\{\Phi(f) \mid f \text{ flow on } N\}$ is finite. If (f_i) is a sequence of flows with limit $\lim_i \Phi(f_i) = \Phi$, then for every (fixed) edge $vw \in E$ the subsequence $(f_i(vw))$ converges to a limit $f(vw)$. Thus, f is a flow with maximum value Φ.

The following theorem states that the maximum flow equals the minimum cut. In English, this is known as the "MaxFlow MinCut Theorem".

Theorem 22.15. (Ford-Fulkerson) In every network N the value of a maximum flow equals the capacity of a minimum cut

$$\max\{\Phi(f) \mid f \text{ flow on } N\} = \min\{\kappa(S) \mid S \text{ cut in } N\} \tag{22.10}$$

Proof. Let f be a maximum flow on N. We inductively define a cut S in N:

- $S := \{q\}$.
- $S := S \cup \{w\}$ for a $w \in \bar{S}$, if there exists a $v \in S$ such that $\kappa(vw) > f(vw)$, if $vw \in E$, or $f(wv) > 0$, if $wv \in E$.
  First, it is shown that S is a cut. Assume S is not a cut, i.e., $s \in S$. Then there exists a sequence $X = (x_0, \ldots, x_m)$ of vertices in N with $x_0 = q, x_m = s$ and for each $i, 1 \leq i \leq m$, one of the following conditions is satisfied:
- $\kappa(x_{i-1}x_i) > f(x_{i-1}x_i)$, if $x_{i-1}x_i \in E$,
- $f(x_ix_{i-1}) > 0$, if $x_ix_{i-1} \in E$.

The edge $x_{i-1}x_i$ is called a forward edge and the edge x_ix_{i-1} a backward edge of X. The sequence X is called an augmenting path from q to s (w.r.t. f).

Let ϵ be the minimum of all values $\kappa(e) - f(e)$ over all forward edges e and all values $f(e)$ over all backward edges e of X. By definition, $\epsilon > 0$ and a flow $f^* = f(X, \epsilon)$ on N is defined by

$$f^*(e) = \begin{cases} f(e) + \epsilon \text{ if } e \text{ is a forward edge of } X \\ f(e) - \epsilon \text{ if } e \text{ is a backward edge of } X \\ f(e) \quad \text{otherwise.} \end{cases}$$

For the value of the flow f^*, it holds contradictingly $\Phi(f^*) = \Phi(f) + \epsilon$.

It remains to compute the capacity of the cut S. For the value of the flow f, it holds by the proof of Lemma 22.13

$$\Phi(f) = \sum_{\substack{vw \in E \\ v \in S, w \in \bar{S}}} f(vw) - \sum_{\substack{uv \in E \\ v \in S, u \in \bar{S}}} f(uv) \tag{22.12}$$

For the cut S, it holds

- $\kappa(vw) = f(vw)$, if $vw \in E$ with $v \in S$ and $w \in \bar{S}$,
- $f(uv) = 0$, if $uv \in E$ with $v \in S$ and $u \in \bar{S}$.

Thus, from (22.12) it immediately follows $\Phi(f) = \kappa(S)$.

The proof provides an iterative algorithm for computing a maximum flow in a flow network. Starting with the zero flow, in each step an augmenting path from source to sink is constructed, with which the current flow is increased. The algorithm is implemented by two routines that call each other. In the first routine, an augmenting path is constructed, which is then used in the second routine to increase the flow. The vertices v of the flow network are marked with their predecessor vertex pred(v), the direction $R(v)$ of the edge incident with pred(v) and v, and the possible flow improvement $\epsilon(v)$. The set M contains the marked vertices whose neighbors have not all been marked yet. If the marking routine no longer reaches the sink, a maximum flow is present and S is a minimum cut.

Assume a flow network is given where all capacity values are integers. Then the algorithm terminates because the current flow is increased by a positive integer in each step and thus a flow is reached for which no augmenting path exists. This flow is then maximum. The algorithm is also applicable to flow networks where all capacity values are rational. This is because multiplying by the least common denominator of the capacity values yields an equivalent problem where all capacity values are integers.

For flow networks with irrational capacity values, it can happen that the improvements $\epsilon > 0$ are infinitesimally small and thus the algorithm does not terminate. The algorithm was modified by Edmonds and Karp (1972) so that the augmenting path is found via breadth-first search. This modified algorithm has time complexity $O\left(|V| \cdot |E|^2\right)$ even for irrational capacity values.

Example 22.16. We apply the algorithm to the flow network in Fig. 22.4. Starting with the zero flow, FlussMarkieren yields the marked flow network in Fig. 22.6. FlussVergrössern improves the zero flow along the augmenting path $X = (q, c, b, s)$ to a flow with value 3 and FlussMarkieren is called a second time (Fig. 22.7). By FlussVergrössern, the flow along the augmenting path

```
Algorithm 22.4 FlussMarkieren ( $D, \kappa, q, s f$ )
Input: Flow network ( $D, \kappa, q, s$ ) and flow $f$ on flow network
Output: Marked vertices, if sink is reached, otherwise minimum cut
    $S$ is output
    $S:=\{q\}$
    $M:=\{q\}$
    $\epsilon(q):=\infty$
    repeat
        choose vertex $v \in M$
        $M:=M \backslash\{v\}$
        for all $w \in V \backslash S$ with $v w \in E$ do
            if $f(v w)<\kappa(v w)$ then
                $\operatorname{pred}(w):=v$
                $R(w):={ }^{\prime} \rightarrow{ }^{\prime}$
                $\epsilon(w):=\min \{\kappa(v w)-f(v w), \epsilon(v)\}$
                $S:=S \cup\{w\}$
                $M:=M \cup\{w\}$
            end if
        end for
        for all $u \in V \backslash S$ with $u v \in E$ do
            if $f(u v)>0$ then
                $\operatorname{pred}(u):=v$
                $R(u):={ }^{\prime} \leftarrow^{\prime}$
                $\epsilon(u):=\min \{f(u v), \epsilon(v)\}$
                $S:=S \cup\{u\}$
                $M:=M \cup\{u\}$
            end if
        end for
```

```
until $M=\emptyset$ or $s \in S$
if $M=\emptyset$ then
    return $S$
end if
if $s \in S$ then
    FlussVergrössern $(D, \kappa, q, s f)$
end if
```

$X = (q, a, b, s)$ is increased by the value 2. Subsequent FlussMarkieren no longer reaches the sink (Fig. 22.8). Thus, the flow is maximum with value 5 and the corresponding minimum cut is $S = \{q, a, c\}$.

Theorem 22.17. (Integrality) In a network with integer capacity values, there exists an integer maximum flow.

Proof. In the algorithm, starting from the zero flow, the flow is increased by an integer $\epsilon > 0$ in each step. Thus, the flow at each step has only integer edge values. This particularly holds for the maximum flow.

```
Algorithm 22.5 FlussVergrössern( $D, \kappa, q, s, f$ )
Input: Flow network ( $D, \kappa, q, s$ ) and flow $f$ on flow network
Output: Flow $f$ increased by $\epsilon$
    $\epsilon:=\epsilon(s)$
    $v:=s$
    while $v \neq q$ do
        if $R(v)={ }^{\prime} \rightarrow^{\prime}$ then
            $f(\operatorname{pred}(v), v):=f(\operatorname{pred}(v), v)+\epsilon$
        end if
        if $R(v)=^{\prime} \leftarrow^{\prime}$ then
            $f(v, \operatorname{pred}(v)):=f(v, \operatorname{pred}(v))-\epsilon$
        end if
        $v:=\operatorname{pred}(v)$
    end while
    FlussMarkieren $(D, s, t, \kappa, f)$
```

22.4 Theorems of Hall, König-Egerváry, and Menger

In this section, a series of combinatorial applications of the Ford-Fulkerson theorem are treated.

The Theorem of Menger

The theorem of Karl Menger (1902-1985) determines the number of edge- and vertex-disjoint paths in a digraph. Let $D = (V, E)$ be a digraph and

let q and s be vertices in D. A set E' of edges in D is called an q and s separating edge set in D if every directed path from q to s contains at least one

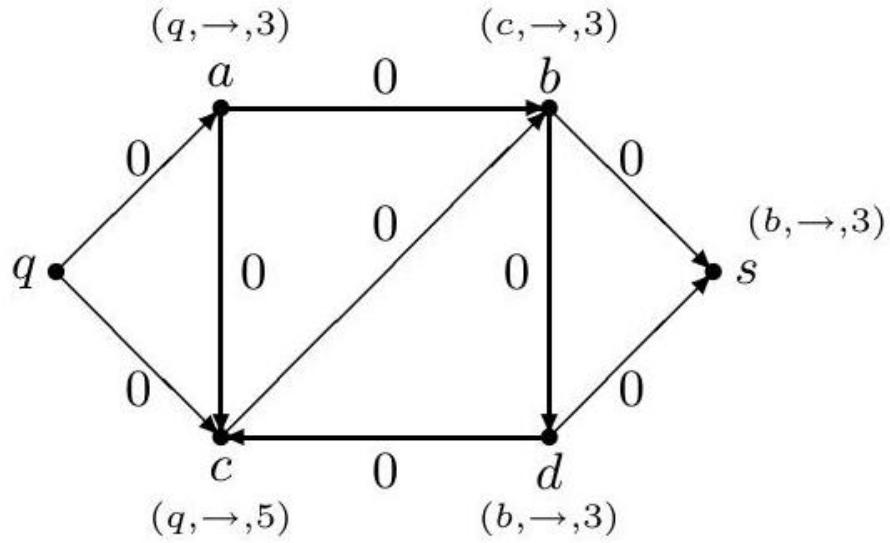


Fig. 22.6. First FlussMarkieren.

edge from E'. An q and s separating edge set E_0 in D is called minimal if for every q and s separating edge set E' in D it holds $|E_0| \leq |E'|$.

Theorem 22.18. (Menger, 1927) Let $D = (V, E)$ be a digraph and let q and s be distinct vertices in D. The maximum number of edge-disjoint directed paths from q to s equals the cardinality of a minimal q and s separating edge set in D.

Proof. We consider a flow network N on D where each edge has capacity 1. By the integrality theorem, there exists a maximum flow on N where each edge has value 0 or 1. Such a flow is called a 0-1 flow. Here, k edge-disjoint paths from q to s in D yield a 0-1 flow on N with value k, and vice versa.

Every cut S in N yields an q and s separating edge set $E' = \left\{e \in E \mid e^- \in S, e^+ \in \bar{S}\right\}$ with capacity $|E'| = \kappa(S)$. Conversely, let E' be an q and s separating edge set in D. Let $S(E')$ be the set of all vertices in D reachable from q via a directed path without using edges in E'. The set $S(E')$ forms a cut in N with capacity $|E'| = \kappa(S(E'))$. By the Ford-Fulkerson theorem, the claim follows.

Example 22.19. The digraph in Fig. 22.9 contains at most three edge-disjoint paths from q to s because q is adjacent to three vertices. On the other hand, every q and s separating edge set is at least 3-element, for example $\{ad, be, ce\}$ and $\{df, dg, be, ce\}$.

Let $D = (V, E)$ be a digraph and let q and s be vertices in D. A set V' of vertices in D is called an q and s separating vertex set in D if every directed path from q to s in D contains at least one vertex from V'. An q and s separating vertex set V_0 in D is called minimal if for every q and s separating vertex set

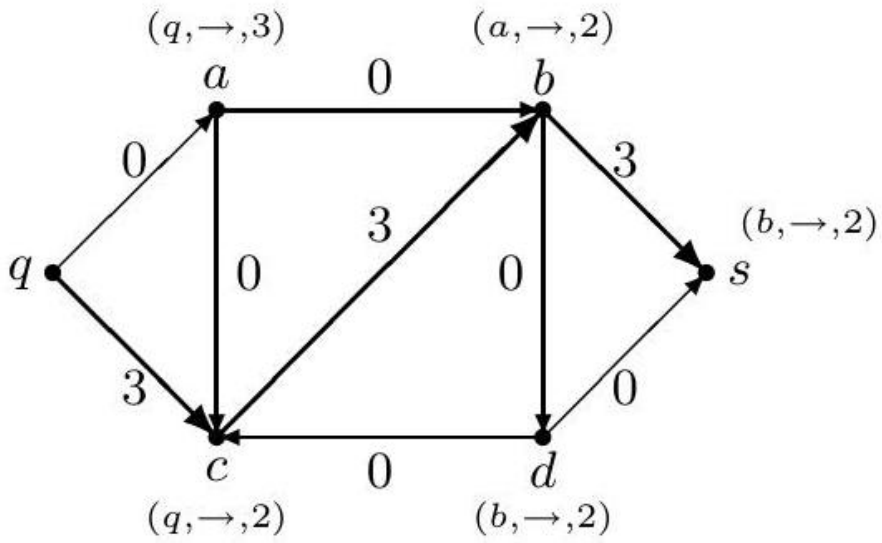


Fig. 22.7. First FlussVergrössern and second FlussMarkieren.

V' in D it holds $|V_0| \leq |V'|$.

Theorem 22.20. (Menger) Let $D = (V, E)$ be a digraph and let q and s be non-adjacent vertices in D. The maximum number of vertex-disjoint directed paths from q to s equals the cardinality of a minimal q and s separating vertex set in D.

Proof. We assign to the digraph D a digraph D' in which each vertex v (except q and s) is replaced by two vertices v' and v'' and an edge $v'v''$. The edges uv in D start at u'' and end at v' in the new digraph (Fig. 22.10).

By definition, the vertex-disjoint paths in D correspond to the edge-disjoint paths in D'. Thus, by Theorem 22.18, the maximum number of vertex-disjoint directed paths from q to s in D equals the cardinality of a minimal q and s separating edge set in D'. In a q and s separating edge set in D', only edges of the form $v'v''$ need to be considered, because an edge of the form $u''v'$ can always be replaced by $u'u''$. Such q and s separating edge sets in D' correspond to q and s separating vertex sets in D.

Example 22.21. The digraph in Fig. 22.9 contains at most two vertex-disjoint paths from q to s because each such path must pass through either d or e, for example (q, a, d, f, s) and (q, b, e, h, s). On the other hand, every q and s separating vertex set is at least 2-element, for example $\{a, b, c\}, \{d, e\}$ and $\{f, g, h, i\}$.

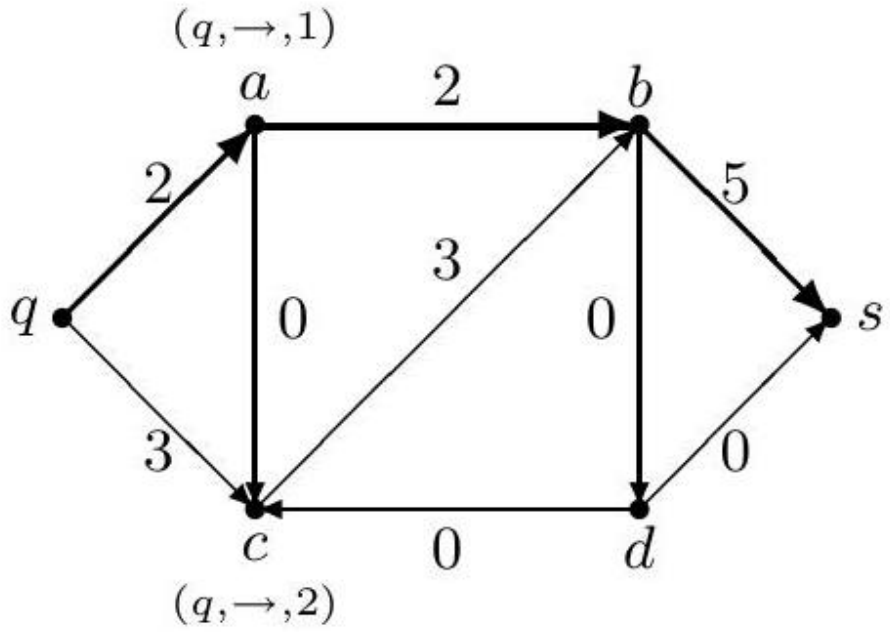


Fig. 22.8. Second FlussVergrössern and third FlussMarkieren. The sink is no longer reached and the algorithm terminates.

The Theorem of König and Egerváry

The theorem of König and Egerváry provides information about the cardinality of a maximum matching in a bipartite graph. Let G be a graph. A

set of edges P is called a matching in G if the edges in P have no common endpoint (Fig. 22.11). A matching P in G is called maximal if for every matching P' in G it holds $|P'| \leq |P|$.

A set U of vertices in G is called a vertex cover of G if for every edge uv in G it holds $u \in U$ or $v \in U$. A vertex cover U of G is called minimal if for every vertex cover U' of G it holds $|U| \leq |U'|$.

Theorem 22.22. (König-Egerváry, 1931) Let G be a bipartite graph. The cardinality of a maximum matching in G equals the cardinality of a minimum vertex cover of G.

Proof. Let G be a bipartite graph with 2-partition $\{V_1, V_2\}$. We assign to the graph G a digraph D in which two vertices q and s and directed edges $qv, v \in V_1$, and $vs, v \in V_2$ are added. Furthermore, each edge between V_1 and V_2 is directed from V_1 to V_2 (Fig. 22.12). By definition, a matching consisting of k edges in G corresponds to k vertex-disjoint paths from q to s in D. Furthermore, a vertex cover of G corresponds to a q and s separating vertex set in D. Thus, by Theorem 22.20, the claim follows. □

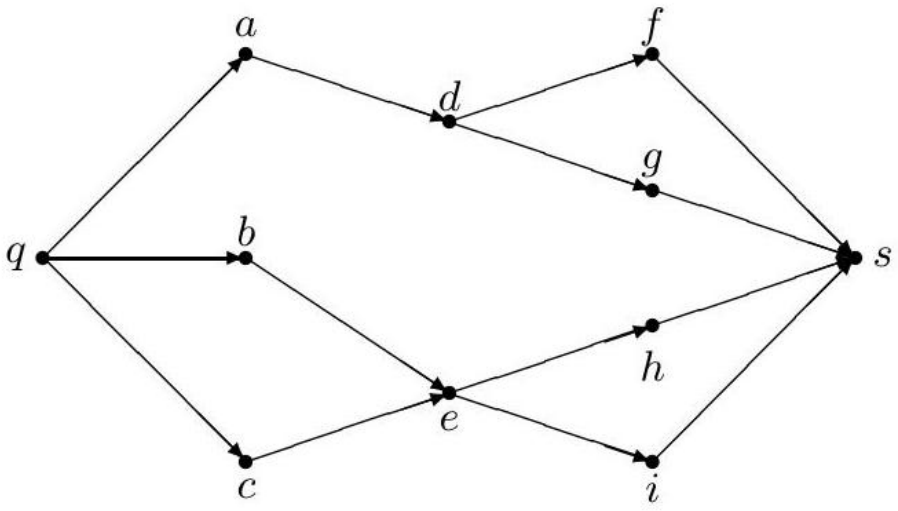


Fig. 22.9. A digraph with two vertex-disjoint and three edge-disjoint paths from q to s.

The Theorem of Hall

The theorem of Philip Hall (1904-1982) determines the cardinality of a complete matching in a bipartite graph. Let $G = (V, E)$ be a bipartite graph with 2-partition $\{V_1, V_2\}$. A matching P in G is called complete if every vertex in V_1 is incident with an edge in P (Fig. 22.11).

For every subset U of V_1, let U^+consist of all vertices in V_1 that are adjacent to some vertex in V_2, i.e.,

$$U^+ = \{v \in V_2 \mid \exists u \in U[uv \in E]\}\,, \tag{22.13}$$

Theorem 22.23. (Hall, 1935) Let G be a bipartite graph with 2-partition $\{V_1, V_2\}$ such that $|V_1| = |V_2|$. A complete matching in G exists if and only if for every subset U of V_1 it holds $|U^+| \geq |U|$.

Proof. Let P be a complete matching in G and U a subset of V_1. Then each vertex in U is incident with an edge in P and the other endpoint of such an edge lies in U^+. Since P is a matching, $|U| \leq |U^+|$.

Conversely, suppose $\left|U_1^+\right| \geq |U_1|$ for every subset U_1 of V_1. Let U be a vertex cover of G with $U_1 = U \cap V_1$ and $U_2 = U \cap V_2$. There is no edge uv with $u \in V_1 \backslash U_1$ and $v \in V_2 \backslash U_2$, because otherwise U would not be a vertex cover of G. Thus, $(V_1 \backslash U_1)^+ \subseteq U_2$, hence by assumption $|U_2| \geq \left|(V_1 \backslash U_1)^+\right| \geq |V_1 \backslash U_1|$. This implies $|U| = |U_1| + |U_2| \geq |U_1| + |V_1 \backslash U_2| = |V_1|$. Thus, every vertex cover of G has at least $|V_1|$ elements. Every minimum vertex cover of G therefore has cardinality $|V_1|$ by assumption. By the König-Egerváry theorem, $|V_1|$ is the cardinality of a

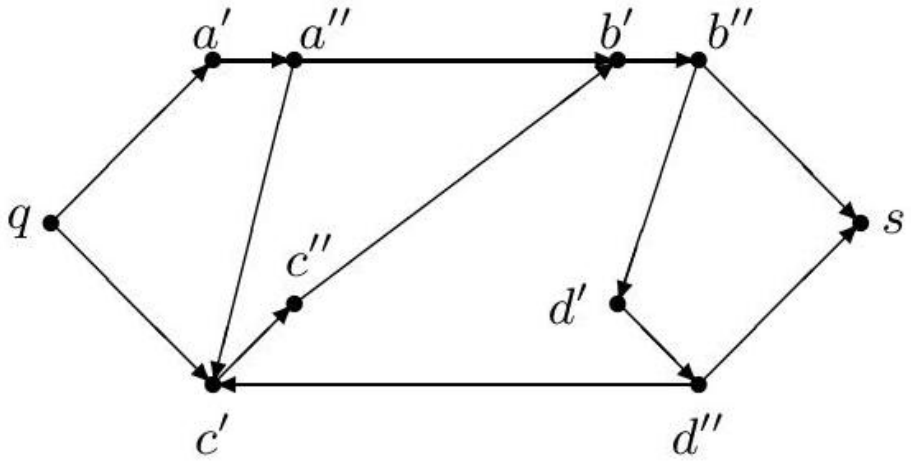


Fig. 22.10. The digraph D' associated with the digraph D in Fig. 22.4.

maximum matching in G. Such a matching in G is complete by definition. □

This theorem is called the marriage theorem because there is an obvious connection to a marriage problem.

Self-test Problems

22.1. Show Theorem 22.1.
22.2. Extend Floyd-Warshall so that not only distances but also shortest paths are computed.
22.3. Prove the statement about the loop invariant in Theorem 22.5.
22.4. Let (G, ω) be a road network where every two edges have different weights. Show that the network has exactly one minimum spanning tree.
22.5. In a data network, probabilities of failure are assigned to the lines (edges). The goal is to find paths between two stations (vertices) that have the highest probability of not failing. How can this problem be reduced to finding shortest paths?
22.6. Modify Floyd-Warshall so that it can compute the transitive closure of a homogeneous relation.
22.7. Prove Equation (22.5).
22.8. Show that MinSpannbaum has time complexity $O(|E| \log |E|)$.
22.9. Generalize the Ford-Fulkerson algorithm to networks with multiple sources and sinks.
22.10. The connectivity number of a graph $G = (V, E)$ is defined by

$$\kappa(G) = \min\{|F| \mid F \subseteq E, G \backslash F \text{ is not connected } \}$$

A graph G is called p-connected if $\kappa(G) \geq p$. Show that a graph G is p-

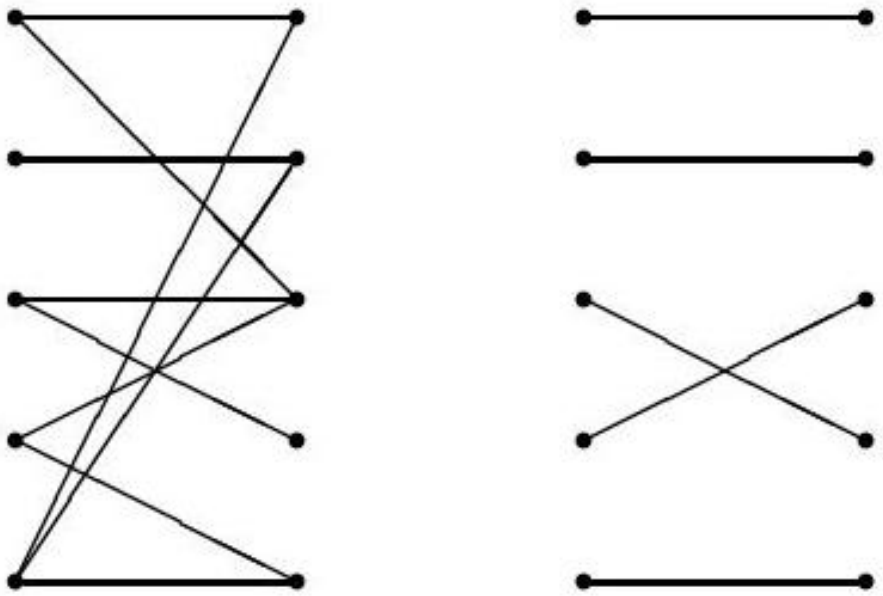


Fig. 22.11. A bipartite graph with a maximum and complete matching.

connected if and only if any two vertices in G are connected by at least p edge-disjoint paths.

22.11. Let S_1 and S_2 be minimal cuts in a flow network N. Show that $S_1 \cap S_2$ is also a minimal cut in N.

22.12. In a data network, data transmission occurs between two stations q and s. How can it be determined how many lines (edges) can fail at most while still maintaining a data connection between q and s?

22.13. Five boys, Alfred, Bert, Claus, and Detlef, and five girls, Rita, Susi, Thea, Ute, and Vera, attend a dance course. Alfred is friends with Rita, Susi, and Ute, Bert with Susi, Claus with Susi, Ute, and Thea, Detlef with Ute and Vera, and finally Egon with Rita, Susi, and Vera. Can each boy invite a girl he is friends with to the final ball without any girl having to choose between two boys?

22.14. Let A be a finite non-empty set and $M = (A_1, \ldots, A_n)$ a sequence of non-empty subsets of A. The goal is to find a representative a_i from each subset A_i such that different subsets are represented by different elements. Such a tuple is called a system of representatives of M.

Show that $M = (A_1, \ldots, A_n)$ has a system of representatives if and only if for every subset I of $\underline{n}$ it holds

$$\left| \bigcup_{i \in I} A_i \right| \geq |I|$$

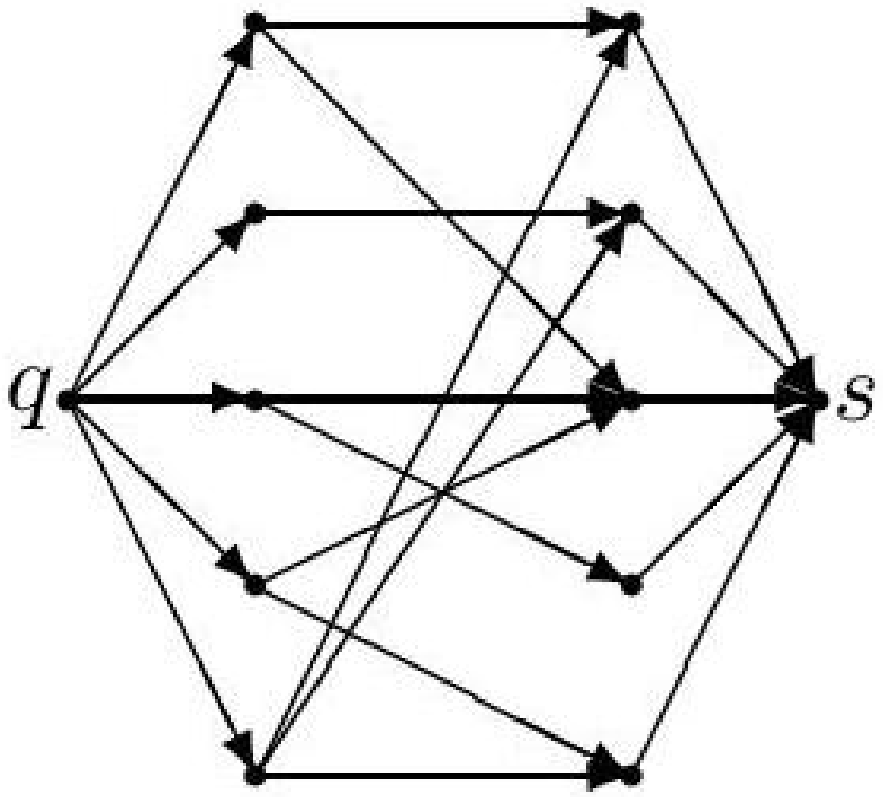


Fig. 22.12. The digraph associated with the graph in Fig. 22.11.

Combinatorial Optimization

Many optimization problems of great economic importance are so complex that an exact solution is not possible despite the enormous increase in computer performance. These optimization problems belong to the NP-hard problems and are characterized by requiring exponential effort to solve. In this chapter, methods are presented to solve such optimization problems. These include backtracking procedures that search the entire solution space, and heuristic procedures that generate approximate solutions. Furthermore, the optimality of special heuristics, so-called greedy algorithms, is characterized using matroids. Finally, the problem of vertex coloring of graphs is investigated.

23.1 Complexity Classes

The running time of algorithms is measured depending on the problem instance n. To solve practical problems, algorithms with polynomial running time $O\left(n^d\right), d \geq 0$, are required. In contrast, algorithms with exponential running time $O\left(c^n\right), c > 1$, are usually impractical.

Complexity of Decision Problems

A decision problem requires a "yesör "noänswer. Decision problems are divided into two classes. The class P consists of all decision problems that can be solved in polynomial time. The class NP includes all decision problems with the property that for every affirmative answer there exists a proof for the affirmative answer that can be verified in polynomial time. Obviously, P $\subseteq$ NP. It is likely that the class NP is larger than the class P. However, no proof for this exists.

Example 23.1. Let $G = (V, E)$ be a graph. A subset U of V is called independent in G if $uv \notin E$ for all $u, v \in U$. We consider the decision problem of finding an independent set U in G with $|U| \geq K$ for a given graph G and a natural number K. This problem is in NP. For a subset U of V, there are $\binom{|U|}{2}$ two-element subsets uv with $u, v \in U$ and with $n = |V|$ we have $\binom{|U|}{2} = O\left(n^2\right)$. The test whether uv is in E can be done in constant time. Thus, in $O\left(n^2\right)$ steps it can be tested whether U is independent in G.

Let D and D' be decision problems. A polynomial transformation from D to D' is an algorithm with polynomial running time that transforms every instance I of problem D into an instance I' of problem D' such that the answers of I and I' agree. If such a transformation exists, we write $D \propto D'$.

Example 23.2. Let $G = (V, E)$ be a graph. A subset $U \subseteq V$ is called a clique in G if $uv \in E$ for all $u, v \in U$. A clique is thus a complete subgraph of G. We consider the decision problem of finding a clique U in G with $|U| \geq K$ for a given graph G and a natural number K. This problem belongs to NP.

Let $G' = (V, E')$ be the complement of G, i.e., $uv \in E'$ if and only if $uv \notin E$. The independent sets in G are exactly the cliques in G', and vice versa. The graph G' can be constructed from G in $O\left(|V|^2\right)$ steps. Thus, the decision problem of finding an independent subset with at least K elements in a graph is polynomially transformable to the decision problem of finding a clique with at least K elements in a graph.

A decision problem D is called NP-complete if $D \in$ NP and for every problem $D' \in$ NP it holds $D' \propto D$.

Theorem 23.3. Let D be an NP-complete decision problem. If D is in P, then $P = NP$.

Proof. Let $D \in$ P be an NP-complete problem. Let A be a polynomial algorithm that solves D. Let $D' \in$ NP. By assumption, $D' \propto D$. Thus, there exists a polynomial algorithm B that transforms every instance of D' into an instance of D. Therefore, the algorithms A and B executed consecutively solve every instance of D' in polynomial time. Thus, $D' \in$ P. This implies the claim.

Example 23.4. Let n be a natural number. The satisfiability problem asks to find an assignment $b_1, \ldots, b_n \in \{0, 1\}$ of the variables of a n-ary switching function f such that $f(b_1, \ldots, b_n) = 1$. This problem is NP-complete. This was first proven by Stephen A. Cook (1971).

Currently, more than a thousand NP-complete problems are known, including many graph-theoretic problems.

Complexity of Optimization Problems

Combinatorial problems usually appear in the form of optimization problems. We classify such problems using Turing reduction.

A Turing reduction assigns a problem D a problem D' such that the solution algorithm A for D is used as a ſubalgorithmïn the solution algorithm B for D'. The algorithm A should have polynomial running time if and only if B has polynomial running time. Such a reduction is denoted by $D' \propto_T D$. A polynomial transformation is a special Turing reduction, i.e., $D' \propto D$ implies $D' \propto_T D$.

Example 23.5. The optimization problem corresponding to the decision problem of finding a clique with at least K elements in a graph is to find a clique of maximum cardinality in a graph.

Let A be an algorithm that solves the above decision problem. This algorithm is embedded in a loop that processes an instance $A(G, K)$ of the decision problem in each step for $K = n, \ldots, 1$. As soon as an instance $A(G, K)$ returns "yes", a maximum clique is found and thus the corresponding optimization problem is solved.

The analog of NP-complete decision problems are NP-hard optimization problems. A problem D is called NP-hard if there exists an NP-complete problem D' such that $D' \propto_T D$. By 23.5, the optimization problem of finding a maximum clique in a graph is NP-hard. In general, the optimization variants of NP-complete decision problems are NP-hard.

23.2 Backtracking Algorithms

Combinatorial Optimization

A combinatorial optimization problem is based on a finite set. This set is called the universe in practical terminology, and the elements of the universe are called solutions. The set of all solutions is divided into feasible and infeasible solutions. On a universe X, a discrete objective function $f : X \to \mathbb{Z}$ is defined. The goal is to find a feasible solution $x^* \in X$ with maximum value, i.e., $f(x^*) \geq f(x)$ for all feasible solutions $x \in X$. Such a solution x^* is called optimal. A combinatorial optimization problem thus has the following form

$$\begin{array}{ll} & \max f(x). \\ \text{s.t.} & x \in X \\ & x \text{ is feasible} \end{array}$$

Maximization and minimization problems are equivalent because maximizing f corresponds to minimizing $-f$, more precisely

$$\max\{f(x) \mid x \in X\} = -\min\{-f(x) \mid x \in X\} \tag{23.2}$$

Example 23.6. Let $G = \{1, \ldots, n\}$ be a set of items. Each item i has an integer value $f_i \geq 1$ and an integer size $g_i \geq 1$. Furthermore, let K_0 be a natural number, called capacity.

A knapsack is a subset G' of G. The value of a knapsack G' is the sum of the values of all packed items $\sum_{i \in G'} f_i$ and the size of G' is the sum of the sizes of all packed items $\sum_{i \in G'} g_i$. The knapsack problem asks to find a knapsack with maximum value such that its size does not exceed the capacity. The knapsack problem is NP-hard.

We assign to each knapsack G' its characteristic vector $x \in \{0,1\}^n$, i.e., $x_i = 1$ if $i \in G'$, and $x_i = 0$ if $i \notin G'$. Then a knapsack x has value $f(x) = \sum_i f_i x_i$ and size $g(x) = \sum_i g_i x_i$. The knapsack problem thus reads

$$\begin{array}{c} \max f(x). \\ \text{s.t. } x \in \{0,1\}^n \\ g(x) \leq K_0 \end{array} \tag{23.3}$$

Backtracking Procedures

A backtracking algorithm is a recursive procedure that generates all solutions of a combinatorial optimization problem step by step. Backtracking algorithms belong to the exhaustive search methods.

Rucksack-Backtrack1 is a backtracking algorithm for the knapsack problem. The algorithm starts with the so-called empty tuple $x = ()$ and generates all binary n-tuples in lexicographic order. Here, aktf is the value of the current n-tuple x, optx is the currently best feasible solution, and optf is the value of optx. The recursive calls of the routine Rucksack-Backtrack1 are illustrated by a depth-first call tree.

Example 23.7. We consider a knapsack problem with five items and capacity $K_0 = 35$. The items have sizes 14, 15, 11, 10, and 12 and values 26, 27, 18, 16, and 19. The call tree of Rucksack-Backtrack1 is shown in Fig. 23.1. The leaves of this tree correspond to the knapsacks. The weight of each knapsack is indicated at the leaves. An optimal solution is the knapsack $x = (1, 0, 1, 1, 0)$ with $f(x) = 60$ and $g(x) = 35$.

```
Algorithm 23.1 Rucksack-Backtrack1 $\left(x_{1}, \ldots, x_{k}, k\right)$
Input: $x_{1}, \ldots, x_{k} \in\{0,1\}, k \geq 0$
    global optf, optx
    if $k=n$ then
        if $g_{1} x_{1}+\ldots+g_{n} x_{n} \leq K_{0}$ then
            aktf $:=f_{1} x_{1}+\ldots+f_{n} x_{n}$
```

```
        if aktf $>$ optf then
            optf $:=$ aktf
            optx $:=\left(x_{1}, \ldots, x_{n}\right)$
        end if
    end if
else
    $x_{k+1}:=0$
    Rucksack-Backtrack1 $\left(x_{1}, \ldots, x_{k}, x_{k+1}, k+1\right)$
    $x_{k+1}:=1$
    Rucksack-Backtrack1 $\left(x_{1}, \ldots, x_{k}, x_{k+1}, k+1\right)$
end if
```

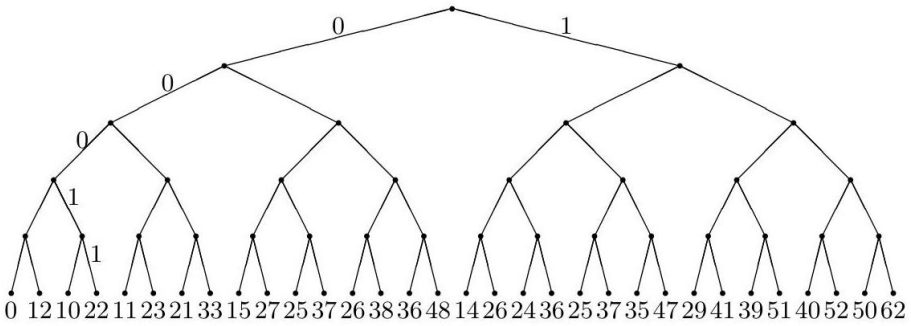


Fig. 23.1. A call tree of Rucksack-Backtrack1 for the knapsack problem in 23.7. The leaves are labeled with the sizes of the knapsacks.

Pruning of Subtrees

A call tree can be reduced by selectively pruning subtrees. The internal nodes of a call tree are called partial solutions. In the knapsack problem, partial solutions correspond to partially packed knapsacks $x = (x_1, \ldots, x_k), 0 \leq k \leq n$. Each partial solution x also has a value $f(x) = \sum_i f_i x_i$ and a size $g(x) = \sum_i g_i x_i$. A partial solution x is called feasible if $g(x) \leq K_0$. In the call tree, a subtree can be pruned if its root is not feasible. Then all partial solutions in this subtree are also not feasible. From Rucksack-Backtrack1 we obtain Rucksack-Backtrack2 in this way.

```
Algorithm 23.2 Rucksack-Backtrack2 $\left(x_{1}, \ldots, x_{k}, k\right)$
Input: $x_{1}, \ldots, x_{k} \in\{0,1\}, k \geq 0$
    global optf, optx
    if $k=n$ then
        if $g_{1} x_{1}+\ldots+g_{n} x_{n} \leq K_{0}$ then
            aktf $:=f_{1} x_{1}+\ldots+f_{n} x_{n}$
```

```
        if aktf $>$ optf then
            optf $:=$ aktf
            optx $:=\left(x_{1}, \ldots, x_{n}\right)$
        end if
    end if
else
    $x_{k+1}:=0$
    Rucksack-Backtrack2 $\left(x_{1}, \ldots, x_{k}, x_{k+1}, k+1\right)$
    $x_{k+1}:=1$
    if $g_{1} x_{1} \ldots+g_{k} x_{k}+g_{k+1} \leq K_{0}$ then
        Rucksack-Backtrack2 $\left(x_{1}, \ldots, x_{k}, x_{k+1}, k+1\right)$ \{Pruning\
    end if
end if
```

Bounding Functions

Let $x = (x_1, \ldots, x_k)$ be a partial solution of a call tree and $P(x)$ the maximum value of all feasible solutions lying in the subtree with root x. A bounding function is a real-valued mapping B on the nodes of a call tree such that for every feasible partial solution x it holds $B(x) \geq P(x)$. A call tree can be reduced using a bounding function. Let x be a feasible partial solution and optf the current optimal value. In the case $B(x) \leq$ optf, for every feasible solution y in the subtree with root x it holds

$$f(y) \leq B(x) \leq optf \tag{23.4}$$

The subtree with root x cannot improve the current optimal solution. Therefore, it can be pruned.

A bounding function for the knapsack problem is obtained from the rational knapsack problem. A knapsack $x = (x_1, \ldots, x_n)$ is called rational if it has rational components x_i with $0 \leq x_i \leq 1$. A rational knapsack x has value $f(x) = \sum_i f_i x_i$ and size $g(x) = \sum_i g_i x_i$. The rational knapsack problem reads

$$\begin{aligned} &\max f(x). \\ &\text{s.t. } x \text{ is rational} \\ &g(x) \leq K_0 \end{aligned} \tag{23.5}$$

Let the items be numbered such that $f_1/g_1 \geq f_2/g_2 \geq \ldots \geq f_n/g_n$. An optimal rational knapsack x^* can then be specified directly. In the case $g_1 + \ldots + g_n \leq K_0$, set $x^* = (1, \ldots, 1)$. Otherwise, there exists an index l with $g_1 + \ldots + g_l \leq K_0$ and $g_1 + \ldots + g_l + g_{l+1} > K_0$. Then set $x_1^* = \ldots = x_l^* = 1$, $x_{l+1}^* = \frac{K_0 - (g_1 + \ldots + g_l)}{g_{l+1}}$ and $x_{l+2}^* = \ldots = x_n^* = 0$. The optimal knapsack x^* is computed in $O(n)$ steps.

Example 23.8. The optimal rational knapsack of the knapsack problem in 23.7 is $x = (1, 1, 0.545455, 0, 0)$ with $g(x) = 35$ and $f(x) = 62.818182$.

The (integer) knapsack problem is a special case of the rational knapsack problem. Thus, the value of every feasible knapsack in a subtree with root $x = (x_1, \ldots, x_k)$ is upper bounded by

$$B(x) = f(x) + \text{RUCKSACK-RAT}\left(f_{k+1}, \ldots, f_n, g_{k+1}, \ldots, g_n, K_0 - g(x)\right),$$

where the second term gives the value of an optimal rational knapsack for the last $n - k$ items. Thus, $B(x)$ is a bounding function for the knapsack problem, from which the algorithm Rucksack-Backtrack3 results from Rucksack-Backtrack2.

```
Algorithm 23.3 Rucksack-Backtrack3 $\left(x_{1}, \ldots, x_{k}, k\right)$
Input: $x_{1}, \ldots, x_{k} \in\{0,1\}, k \geq 0$
    global optf, optx
    if $k=n$ then
        if $g_{1} x_{1}+\ldots+g_{n} x_{n} \leq K_{0}$ then
            aktf $:=f_{1} x_{1}+\ldots+f_{n} x_{n}$
            if aktf $>$ optf then
                optf $:=$ aktf
                optx $:=\left(x_{1}, \ldots, x_{n}\right)$
            end if
        end if
    else
        $B:=\left(f_{1} x_{1}+\ldots+f_{k} x_{k}\right)+\operatorname{RucKSACK-RAT}\left(f_{
        $\left.\sum_{i=1}^{k} g_{i} x_{i}\right)$ \{Bounding function\}
        if $B \leq optf$ then
            return
        end if
        $x_{k+1}:=0$
        Rucksack-Backtrack3 $\left(x_{1}, \ldots, x_{k}, x_{k+1}, k+1\right)$
        $x_{k+1}:=1$
        if $g_{1} x_{1} \ldots+g_{k} x_{k}+g_{k+1} \leq K_{0}$ then
            Rucksack-Backtrack3 $\left(x_{1}, \ldots, x_{k}, x_{k+1}, k+1\right)$ \{Pruning\
        end if
    end if
```

The Traveling Salesman Problem

Let $K_n = (V, E)$ be the complete graph with n vertices and $f : E \to \mathbb{N}$ a cost function on the edges of K_n. A tour or Hamiltonian cycle in K_n is a simple cycle in K_n that contains all vertices in K_n. The length of a tour $x = (x_1, \ldots, x_n, x_1)$ in K_n is

$$f(x) = \sum_{i=1}^{n} f\left(x_i x_{i+1}\right) + f\left(x_n x_1\right). \tag{23.6}$$

The traveling salesman problem for the weighted graph K_n is to find a tour in K_n with minimal length. The traveling salesman problem is NP-hard and can be interpreted as a traveling salesman problem or a component placement problem for printed circuit boards.

A tour $(x_1, \ldots, x_n, x_1)$ can be represented as a permutation $x = (x_1, \ldots, x_n)$ of the vertex set of K_n. Each cyclic shift of such a permutation represents the same tour. Rundreise-Backtrack is a backtracking algorithm for the traveling salesman problem. This algorithm

```
Algorithm 23.4 Rundreise-Backtrack $\left(x_{1}, \ldots, x_{k}, k\right)$
Input: Graph $G=(V, E), x_{1}, \ldots, x_{k} \in V, k \geq 0$
    global optf, optx
    if $k=n$ then
        akt $f:=f\left(x_{1} x_{2}\right)+\ldots+f\left(x_{n-1} x_{n}\right)+f\left(x_{n} x_
        if aktf $<$ optf then
            optf $:=$ aktf
            opt $x:=\left(x_{1}, \ldots, x_{n}\right)$
        end if
    else
        for all $x_{k+1} \notin\left\{x_{1}, \ldots, x_{k}\right\}$ do
            Rundreise-Backtrack $\left(x_{1}, \ldots, x_{k}, x_{k+1}, k+1\right)$
        end for
    end if
```

is started with the empty tuple $x = ()$ and generates all permutations of the vertex set of K_n.

The traveling salesman problem is a minimization problem. Therefore, a bounding function $B(x)$ for the traveling salesman problem must provide a lower bound for all feasible solutions containing a feasible partial solution x. Let $x = (x_1, \ldots, x_k)$ be a feasible partial solution. We consider a feasible solution $y = (x_1, \ldots, x_k, y_{k+1}, \ldots, y_n, x_1)$ of the traveling salesman problem that extends x. Then $(x_k, y_{k+1}, \ldots, y_n, x_1)$ is a path in the subgraph $G = G(x)$ of K_n spanned by $V \setminus \{x_2, \ldots, x_{k-1}\}$. A path that contains all vertices of a graph is a special spanning tree.

Thus, we obtain the following bounding function for the traveling salesman problem

$$B(x) = \sum_{i=1}^{k-1} f\left(x_i x_{i+1}\right) + \text{MinSpannbaum}(G(x), f)$$

23.3 Heuristic Algorithms

Backtracking algorithms provide all optimal solutions of a combinatorial optimization problem. If the optimization problem has a very large search space, backtracking algorithms are too expensive. In their place come heuristic algorithms that solve a combinatorial optimization problem approximately. The

quality of a heuristic algorithm, i.e., how far the approximate solution is from the optimal solution, cannot generally be estimated.

A heuristic algorithm for a combinatorial optimization problem is defined by a heuristic, which in turn is based on a neighborhood. A neighborhood for a universe X is a mapping $N : X \to P(X)$ that assigns to each solution $x \in X$ a set of neighboring solutions $N(x)$. Neighboring solutions should be similar in a way dependent on the specific problem.

Examples 23.9. In the knapsack problem, two knapsacks are considered neighboring if they differ by one item. That is, the neighborhood of a knapsack x is defined by the Hamming distance on $X = \{0,1\}^n$ as $N(x) = \{y \in X \mid d(x,y) = 1\}$.

In the traveling salesman problem, two tours are defined as neighboring if one tour can be transformed into the other by a Lin-2-Opt step (Fig. 23.2).

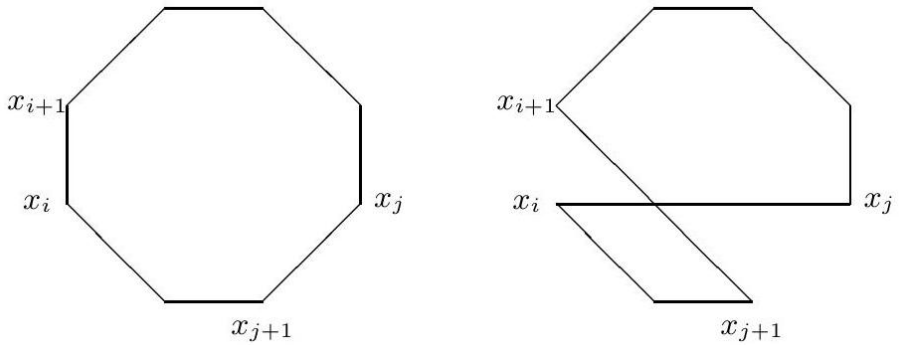


Fig. 23.2. A Lin-2-Opt step.

A heuristic for a combinatorial optimization problem is an algorithm H_N that for each feasible solution x determines a neighboring feasible solution y. If no neighboring feasible solution exists, the algorithm terminates with the output fail.

HeuristischeSuche is a generic heuristic algorithm typically implemented as an iterative procedure. The algorithm generates

```
Algorithm 23.5 HeuristischeSuche $\left(X, f, H_{N}, I\right)$
Input: Universe $X$, objective function $f$, heuristic $H_{N}$, natural number $I$
Output: locally optimal solution $x^{*}$
    $i:=1$
    choose feasible solution $x \in X$ \{starting point\}
    $x^{*}:=x$
    while $i \leq I$ do
        $y:=H_{N}(x)$
        if $y \neq$ fail then
            $x:=y$
```

```
            if $f(x)>f\left(x^{*}\right)$ then
                $x^{*}:=x$
            end if
        else
            return $x^{*}$ \{premature termination\}
        end if
        $i:=i+1$
    end while
    return $x^{*}$
```

a sequence of feasible solutions in which consecutive solutions are neighboring. The resulting approximate solution x^* usually depends on the starting point. A feasible solution $x^* \in X$ is called a local maximum of the optimization problem (23.1) if $f(x^*) \geq f(x)$ for all feasible $x \in N(x^*)$. A feasible solution $x^* \in X$ is called a global maximum if $f(x^*) \geq f(x)$ for all feasible $x \in X$. Combinatorial optimization problems usually have many local optima.

The Hill-Climbing Method

The hill-climbing method is a heuristic procedure comparable to a mountaineer striving for the summit without descending in between. In the heuristic of the hill-climbing method, for a feasible solution x a feasible solution $y \in N(x)$ with higher value $f(y) > f(x)$ is constructed. The hill-climbing method is called the method of steepest ascent if in each neighborhood always a feasible solution with maximum value is chosen.

```
Algorithm 23.6 HillClimbing $\left(X, f, H_{N}\right)$
Input: Universe $X$, objective function $f$, heuristic $H_{N}$
Output: locally optimal solution $x^{*}$
    choose feasible solution $x \in X$ \{starting point\}
    $x^{*}:=x$
    searching := true
    while searching do
        $y:=H_{N}(x)$
        if $y \neq$ fail then
            $x:=y$
            if $f(x)>f\left(x^{*}\right)$ then
                $x^{*}:=x$
            end if
        else
            searching := false
        end if
    end while
    return $x^{*}$
```

HillClimbing is a generic hill-climbing algorithm. A hill-climbing algorithm typically yields a local maximum depending on the starting value. Therefore, the

hill-climbing method belongs to the local optimization methods. Such methods are particularly suitable for problems that have only one optimum. Convex optimization problems have this property.

Example 23.10. We develop a hill-climbing procedure for the knapsack problem. The neighborhood is defined as in 23.9. The heuristic H_N is defined as follows: Let x be a feasible knapsack. Let $y \in N(x)$ be a random knapsack generated by a random number $j \in \{1, \ldots, n\}$

$$y_i = \begin{cases} x_i & \text{if } i \neq j \\ 1 - x_i & \text{if } i = j \end{cases}$$

The two knapsacks x and y differ only in the i-th item and are therefore neighboring. For the size of y it holds

$$g(y) = \begin{cases} g(x) + g_j \text{ if } x_j = 0 \\ g(x) - g_j \text{ if } x_j = 1 \end{cases}$$

The knapsack y is feasible if either $x_j = 1$ or $x_j = 0$ and $g(x) + g_j \leq K_0$. The heuristic outputs fail if $x_j = 0$ and $g(x) + g_j > K_0$. To avoid premature fail output, the heuristic $H_N(x)$ is extended so that it tries several random knapsacks for a given knapsack x until a feasible knapsack is found. The algorithm is started with the empty knapsack $(0, \ldots, 0)$.

Simulated Annealing

Annealing refers to an industrial process where a solid is heated and cooled. With rapid cooling, the atoms are very irregularly arranged, so that the structure still contains relatively high potential energy. In contrast, with slow cooling, a uniform lattice structure is formed, whose potential energy is very low. Simulated annealing is a heuristic procedure based on the annealing process and introduced by Kirkpatrick et al. (1983). Simulated annealing is based on the following heuristic: Let x be a feasible solution. Choose a feasible solution $y \in N(x)$. If $f(y) > f(x)$, then x is replaced by y. Otherwise, a random number $r \in [0, 1]$ is compared with the value $e^{(f(y)-f(x))/T}$. If $r < e^{(f(y)-f(x))/T}$, then x is replaced by y. Otherwise, either fail is output or a new $y \in N(x)$ is sought.

Simulated annealing allows not only an upward movement like in the hill-climbing method but also a downward movement, which depends on the current temperature T. At high temperature, the factor $e^{(f(y)-f(x))/T}$ is close to 1 and thus a downward movement is very likely, while at low temperature $e^{(f(y)-f(x))/T}$ is close to 0 and thus a downward movement is almost impossible. By allowing a downward movement, a local optimum can be left again, which is not possible with the hill-climbing method. Therefore, simulated annealing belongs to the global optimization methods.

The temperature is lowered according to a cooling schedule. The initial temperature T_0 is chosen high enough so that downward movements are possible

with high probability. The temperature is reduced by a percentage in each iteration until a final temperature T_f is reached. Our considerations are summarized in a generic algorithm for simulated annealing, SimulatedAnnealing.

Example 23.11. Let $G = (V, E)$ be a graph with a cost function $f : E \to \mathbb{N}_0$. A uniform partition of G is a 2-partition $\{V_1, V_2\}$ of V with $|V_1| = |V_2|$. The cost of a uniform partition $\{V_1, V_2\}$ of G is

$$f(V_1, V_2) = \sum_{\substack{uv \in E \\ u \in V_1, v \in V_2}} f(u, v)$$

The optimization problem for uniform graph partitions is: Find a uniform partition with minimal cost for a graph $G = (V, E)$ with an even number of vertices and cost function $f : E \to \mathbb{N}_0$. This problem is NP-hard.

To describe this optimization problem by simulated annealing, we use as universe X the set of all 2-partitions $\{V_1, V_2\}$ of V with $|V_1| = |V_2|$. The neighborhood of $\{V_1, V_2\}$ should consist of all partitions that arise from $\{V_1, V_2\}$ by swapping two vertices $v_1 \in V_1$ and $v_2 \in V_2$

```
Algorithm 23.7 SimulatedAnnealing $\left(X, f, H_{N}, T_{0}, T_{f}, \alpha\right)$
$\overline{\text { Input: Universe } X}$, objective function $f$, heuristic $H_{N}$, initial
    final temperature $T_{f}$, decrement $\alpha$
Output: locally optimal solution $x^{*}$
    $T:=T_{0}$
    choose feasible solution $x \in X$
    $x^{*}:=x$
    while $T \geq T_{f}$ do
        $y:=H_{N}(x)$
        if $y \neq$ fail then
            if $f(y)>f(x)$ then
                $x:=y$ \{upward movement\}
                if $f(x)>f\left(x^{*}\right)$ then
                    $x^{*}:=x$;
                end if
            else
                $r:=\operatorname{random}(0,1)$ \{downward movement\}
                if $r<e^{(f(y)-f(x)) / T}$ then
                    $x:=y$
                end if
            end if
        else
            return $x^{*}$ \{premature termination\}
        end if
        $T:=\alpha T\{\alpha=0.99\}$
    end while
    return $x^{*}$
```

The gain resulting from swapping v_1 with v_2 is defined by

$$F(v_1, v_2, V_1, V_2) = f(V_1, V_2) - f([V_1 \setminus \{v_1\}] \cup \{v_2\}, [V_2 \setminus \{v_2\}] \cup \{v_1\}).$$

In the heuristic, according to steepest ascent, a neighboring partition with maximum gain should be delivered for a partition.

The graph in Fig. 23.3 has the partition $\{\{a, b, c\}, \{d, e, f\}\}$ with cost $1 + 1 + 3 + 2 + 0 = 7$. Its neighboring partitions are

$$\begin{array}{l}
\{\{a, b, d\}, \{c, e, f\}\}, \{\{a, b, e\}, \{c, d, f\}\}, \\
\{\{a, b, f\}, \{c, d, e\}\}, \{\{a, d, c\}, \{b, e, f\}\}, \\
\{\{a, c, e\}, \{b, d, f\}\}, \{\{a, c, f\}, \{b, d, e\}\}, \\
\{\{b, c, d\}, \{a, e, f\}\}, \{\{b, c, e\}, \{a, d, f\}\}, \\
\{\{b, c, f\}, \{a, d, e\}\}.
\end{array}$$

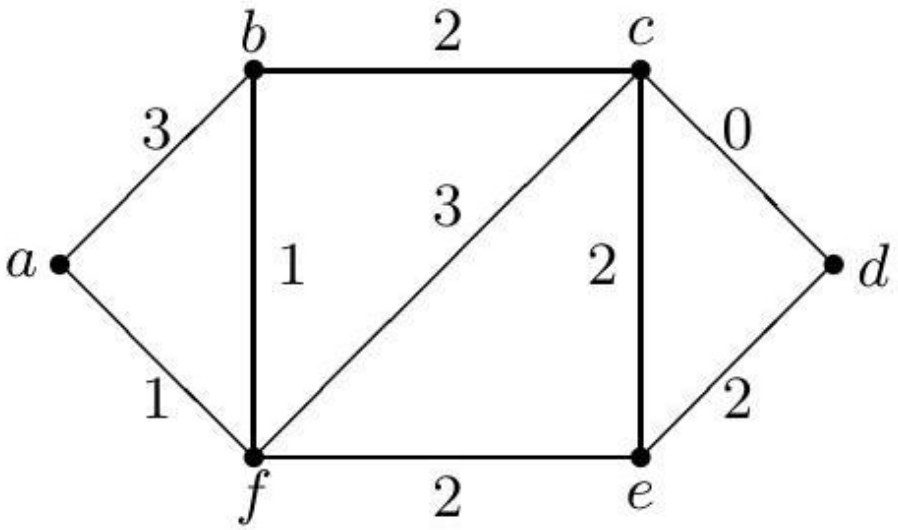


Fig. 23.3. A vertex-weighted graph.

Genetic Algorithms

Genetic algorithms work according to the model of biological evolution and belong to the global optimization methods. The basic patterns and terminology are taken from biology. A genetic algorithm manages a set of solutions of fixed size, called a population. A population can consist of feasible and infeasible solutions (higher and lower individuals). A genetic algorithm works as follows:

- Initialization: Choose a population P of N individuals.

- Iteration:
- Selection: Choose N individuals from the population P according to their fitness.
- Mutation: Replace each individual of the population P by a neighboring individual.
- Crossover: Pair the individuals of the population P and cross each pair so that new individuals are created.

Example 23.12. We develop a genetic algorithm for the traveling salesman problem. The neighborhood is defined as in 23.9.

- Selection: Each tour x in K_n is assigned a so-called Boltzmann weight $\exp(-\beta f(x)), \beta > 0$, and thus a Boltzmann probability

$$p(x) = \frac{\exp(-\beta f(x))}{\sum_{y \in P} \exp(-\beta f(y))}$$

The shorter a tour is, the higher its Boltzmann probability. From the population P, N individuals are selected according to their Boltzmann probabilities.

- Mutation or heuristic: A tour x is replaced by a random tour $x^{(ij)} \in N(x)$ generated by two random numbers $i, j \in \{1, \ldots, n\}$ via a Lin-2-Opt step.
- Crossover: Two tours $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are randomly crossed by generating a random number $j \in \{1, \ldots, n\}$ and thus forming two offspring

$$u = (x_1, \ldots, x_j, y_{j+1}, \ldots, y_n) \quad \text{and} \quad v = (y_1, \ldots, y_j, x_{j+1}, \ldots, x_n)\,.$$

The number j must be chosen such that u and v are again tours.

23.4 Greedy Algorithms and Matroids

Hill-climbing algorithms and Kruskal's algorithm are so-called greedy algorithms. Such algorithms choose an optimal improvement in each step and often end up in a local optimum due to their myopia. We want to investigate under which circumstances greedy algorithms solve an optimization problem.

Greedy Algorithms

Let A be a finite non-empty set, let M be a system of subsets of A with $\emptyset \in M$, and let $\omega : A \to \mathbb{R}_0^+$ be a cost function on A. The goal is to find an element T of the system M with maximum cost

$$\omega(T) = \max \{\omega(T') \mid T' \in M\}. \tag{23.7}$$

The cost of a subset T of A is equal to the sum of the costs of its elements

$$\omega(T) = \sum_{t \in T} \omega(t) \tag{23.8}$$

Greedy is a greedy algorithm for this problem. Algorithm 21.1 Greedy$(A, M, \cdot)$ is called optimal if for every cost function $\omega : A \to \mathbb{R}_0^+$ the computed solution $T = T(\omega)$ is optimal, i.e.,

$$\omega(T) = \max \{\omega(T') \mid T' \in M\} \tag{23.9}$$

We show that optimal greedy algorithms are characterized by matroids.

```
Algorithm 23.8 Greedy $(A, M, \omega)$
Input: Set $A$, system of sets $M$ on $A, \omega: A \rightarrow \mathbb{R}_{0}^{+}$.
Output: $T \in M$ with maximum cost
    $T:=\emptyset$
    while $A \neq \emptyset$ do
        choose $a \in A$ with $\omega(a)=\max \left\{\omega\left(a^{\prime}\right) \mid a^{\
        $A:=A \backslash\{a\};$
        if $T \cup\{a\} \in M$ then
            $T:=T \cup\{a\};$
        end if
    end while
    return $T$
```

Matroids

Let A be a non-empty set and M a system of subsets of A. The system M is called a filter on A if M is downward closed with respect to inclusion, i.e., from $S \in M$ and $T \subseteq S$ it always follows $T \in M$. A filter M on A is called a matroid if M has the exchange property, i.e., for all $S, T \in M$ it holds

$$|S| = |T| + 1 \quad \Longrightarrow \quad \exists s \in S \backslash T [T \cup \{s\} \in M]. \tag{23.10}$$

The elements of a matroid M are called independent sets. The independent sets of maximum cardinality are called bases of M. Matroids were introduced by Hassler Whitney (1907-1989).

Examples 23.13. - Let A be a finite-dimensional vector space. The set M of all linearly independent subsets of A forms a matroid, the linear matroid on

A. The exchange property corresponds to the well-known exchange theorem of Steinitz from linear algebra.

- Let A be a finite set and k a natural number. The set M of all subsets T of A with $|T| \leq k$ forms a matroid, the uniform matroid on A with respect to k.
- Let $G = (V, E)$ be a graph. The set M of all subsets of E that span a forest forms a matroid, the graphic matroid on G. The filter property of M is satisfied because a subgraph of a forest is again a forest. We prove the exchange property. Let $S, T \in M$ with $|S| = |T| + 1$. Then there exists an edge $uv \in S$ not in T. This edge can be chosen such that one of its endpoints lies in the forest spanned by S but not in the forest spanned by T. The subgraph of G spanned by $T \cup \{uv\}$ is then also a forest, so $T \cup \{uv\} \in M$.

All bases of a finite-dimensional vector space have the same cardinality. This holds more generally for the bases of a finite matroid.

Lemma 23.14. If M is a matroid on a finite set A, then all bases of M have the same cardinality.

Proof. Let S and T be bases of M. Assume $|S| \neq |T|$. Without loss of generality, let $|S| > |T|$. We reduce the basis S by $|S| - |T| - 1$ elements. The resulting set S' is also in M because M is a filter. It holds $|S'| = |T| + 1$. By the exchange property, there exists an element $s \in S' \backslash T$ with $T \cup \{s\} \in M$. Thus, T is contradictingly not a basis of M.

Theorem 23.15. Let M be a filter on a finite set A. The algorithm Greedy$(A, M, \cdot)$ is optimal if and only if M is a matroid on A.

Proof. Let M be a filter on A. Assume M does not satisfy the exchange property. Then there exist elements $S, T \in M$ with $|S| = |T| + 1$ such that for every element $s \in S \backslash T$ it holds $T \cup \{s\} \notin M$. We set $k = |T|$ and define a cost function $\omega : A \to \mathbb{R}_0^+$ by

$$\omega(a) = \begin{cases} k+2, & \text{if } a \in T \\ k+1, & \text{if } a \in S \backslash T \\ 0, & \text{otherwise.} \end{cases}$$

The algorithm Greedy(A, M, ω) chooses, by definition, all elements of the set T first because they all have maximum cost. The cost of T is

$$\omega(T) = |T| \cdot (k+2) = k(k+2)$$

According to the above assumption, the algorithm cannot select any further element to increase the cost. However,

$$\omega(S) \geq |S| \cdot (k+1) = (k+1)^2 > \omega(T)$$

Thus, Greedy(A, M, ω) is not optimal.

Conversely, let M be a matroid on A and $\omega : A \to \mathbb{R}_0^+$ a mapping. Let T

be the element of M delivered by Greedy(A, M, ω). Assume T is not a basis of M. Then, as in the proof of Lemma 23.14, there exists an element s such that $T \cup \{s\}$ belongs to the matroid M. But this contradicts the fact that the algorithm Greedy(A, M, ω) adds s to the set T.

We show that for every other basis S of M the inequality $\omega(S) \leq \omega(T)$ holds. Let $T = \{t_1, \ldots, t_k\}$ with $\omega(t_1) \geq \ldots \geq \omega(t_k)$ and $S = \{s_1, \ldots, s_k\}$ with $\omega(s_1) \geq \ldots \geq \omega(s_k)$. By definition of T, $\omega(t_1) \geq \omega(s_1)$. Let $T_{l-1} = \{t_1, \ldots, t_{l-1}\}$ and $S_{l-1} = \{s_1, \ldots, s_{l-1}\}$ with $\omega(T_{l-1}) \geq \omega(S_{l-1})$. In the case $\omega(t_l) \geq \omega(s_l)$, we immediately have $\omega(T_l) \geq \omega(S_l)$ for $T_l = \{t_1, \ldots, t_l\}$ and $S_l = \{s_1, \ldots, s_l\}$. Otherwise, the algorithm selects an element $s_i \in S_l \backslash T_l, 1 \leq i \leq l$, or even an element of another basis of M with cost $\geq \omega(s_i)$, such that $T_{l-1} \cup \{s_i\} \in M$. Because $\omega(s_i) \geq \omega(s_l) > \omega(t_l)$, the algorithm would have selected s_i before t_l, so this case cannot occur. Thus, Greedy$(A, M, \cdot)$ is optimal.

The fact that this theorem does not hold for filters is shown by the following Example 23.16. Let $A = \{a, b, c\}$. The set $M = \{\emptyset, \{a\}, \{b\}, \{c\}, \{b, c\}\}$ is a filter but not a matroid on A. With costs $\omega(a) = 3$ and $\omega(b) = \omega(c) = 2$, Greedy$(A, M, \omega)$ yields the set $T = \{a\}$ with $\omega(T) = 3$, although $S = \{b, c\}$ with $\omega(S) = 4$ is optimal.

Applying the above theorem to the graphic matroid yields an important

Corollary 23.17. Let G be a finite graph. MinSpannbaum$(G, \cdot)$ is an optimal greedy algorithm that for every cost function ω on the edges of G yields a minimum spanning tree of G.

23.5 Vertex Colorings

Finally, the problem of vertex coloring of graphs is investigated. Let $G = (V, E)$ be a graph and k a natural number. A coloring of G with k colors is a mapping $f : V \to \underline{k}$ that assigns different colors to every pair of adjacent vertices of G. If such a mapping exists, the graph G is called k-colorable. The chromatic number of G is the minimum number of colors needed to color G, denoted by $\chi(G)$.

Theorem 23.18. A graph G is 2-colorable if and only if G is bipartite.

Proof. A bipartite graph G with 2-partition $\{V_1, V_2\}$ of its vertex set is 2-colored by coloring all vertices in V_1 red and all vertices in V_2 blue. Conversely, a 2-colorable graph contains no cycles of odd length and is thus bipartite by Theorem 21.19.

The decision problem of finding a coloring of G with at most K colors for a given graph G and a natural number K is NP-complete. The corresponding optimization problem of determining the chromatic number of a graph is NP-hard.

A well-known greedy procedure for vertex coloring of a graph G is due to Welch and Powell. Here, the vertices of G are ordered in an arbitrary sequence. The first vertex is colored with color 1. If the i-th vertex is already colored, the $(i + 1)$-th vertex is colored with the smallest color not used by its neighbors. The resulting coloring depends on the order of the vertices.

```
Algorithm 23.9 Welch-Powell( $G$ )
```

```
Input: Graph $G$ with vertex sequence $\left(v_{1}, \ldots, v_{n}\right)$
Output: Vertex coloring $f$ of $G$
    $f\left(v_{1}\right):=1$;
    for $i:=2$ to $n$ do
            $F:=\emptyset$
            for $j:=1$ to $i-1$ do
                if $v_{j}$ is adjacent to $v_{i}$ then
                    $F:=F \cup\left\{f\left(v_{j}\right)\right\}$
                end if
            end for
            $l:=1$
            while $l \in F$ do
                $l:=l+1$
            end while
            $f\left(v_{i}\right):=l$
    end for
```

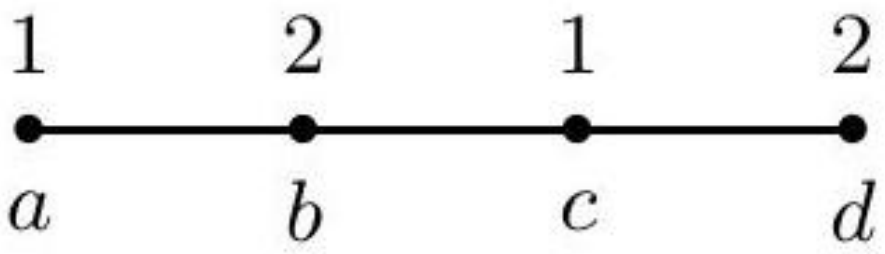


Fig. 23.4. Two vertex colorings of a graph.

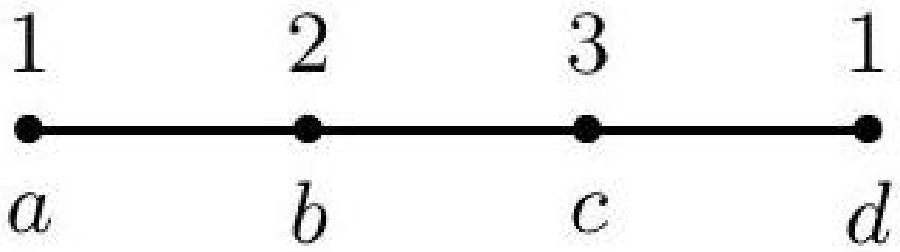


Fig. 23.4. Two vertex colorings of a graph.

Example 23.19. We consider the graph G in Fig. 23.4. The algorithm Welch-Powell (G) yields the following colorings for the vertex sequences (a, b, c, d) and (a, d, b, c)

v	a	b	c	d
$f(v)$	1	2	1	1
F		$\{a\}$	$\{b\}$	$\{c\}$

v	a	d	b	c
$f(v)$	1	1	2	3
F		$\emptyset$	$\{a\}$	$\{b, d\}$

The first coloring shows that the graph is 2-colorable.

Theorem 23.20. Let G be a graph. Let $\Delta(G)$ denote the maximum degree of G and $\omega(G)$ the number of vertices in a maximum clique in G. Then

$$\omega(G) \leq \chi(G) \leq \Delta(G) + 1 \tag{23.11}$$

If the graph G is not regular and connected, then

$$\chi(G) \leq \Delta(G) \tag{23.12}$$

Proof. To color a clique with k vertices, at least k colors are needed. Thus, $\omega(G) \leq \chi(G)$. Welch-Powell (G) yields a vertex coloring that uses at most $\Delta(G) + 1$ colors because each vertex has at most $\Delta(G)$ neighbors. Thus, the first statement is proven.

To prove the second statement, the vertices of G are ordered in a list. Let $k = \Delta(G)$. The last vertex v_n in the list is chosen such that its degree is less than k. Such a vertex exists because G is not regular. The vertices adjacent to v_n are inserted into the list from the back. Let v_{i+1} be the last vertex in the list whose adjacent vertices have already been entered. Then in the next step, all vertices adjacent to v_i are inserted into the list from the back. Since G is connected, all vertices are thus included in the list. In each step, the number of inserted vertices is at most $k - 1$. For this list, Welch-Powell (G) yields a k-coloring of G.

Let $p_G(k)$ be the number of k-colorings of a graph G. The chromatic number $\chi(G)$ is by definition the smallest natural number k with $p_G(k) \geq 1$.

Each k-coloring of the complete graph K_n is a n-permutation of k, thus

$$p_{K_n}(k) = k(k-1)\cdots(k-n+1) \tag{23.13}$$

Let G be a graph with two non-adjacent vertices u and v. Let G_{uv} be the graph obtained from G by adding the edge uv, and let $G_{u=v}$ be the graph obtained from G by identifying the two vertices (Fig. 23.5). The k-colorings of G split into two disjoint subsets,

depending on whether the vertices u and v are colored the same or differently. The k-colorings of G in which the vertices u and v are colored differently correspond to the k-colorings of G_{uv}, while the k-colorings of G in which the vertices u and v are colored the same correspond to the k-colorings of $G_{u=v}$. Thus, we have proven

Theorem 23.21. Let G be a graph. If u and v are non-adjacent vertices in G, then

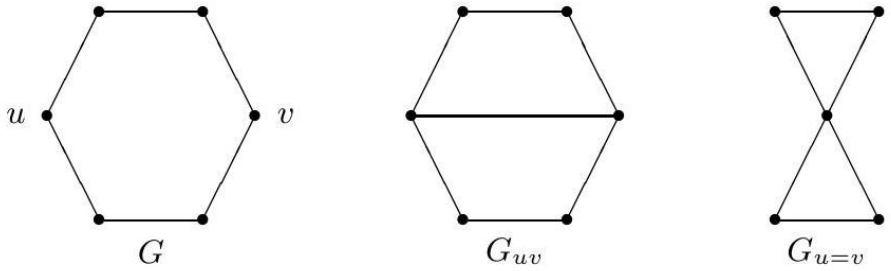


Fig. 23.5. A graph G and the derived graphs G_{uv} and $G_{u=v}$.

$$p_G(k) = p_{G_{uv}}(k) + p_{G_{u=v}}(k) \tag{23.14}$$

and

$$\chi(G) = \min\left\{\chi\left(G_{uv}\right), \chi\left(G_{u=v}\right)\right\} \tag{23.15}$$

The derived graphs G_{uv} and $G_{u=v}$ have at least one fewer pair of non-adjacent vertices than G. If we successively apply this procedure to G_{uv} and $G_{u=v}$, after finitely many steps complete graphs are obtained, for which the number of colorings is known according to (23.13) (Fig. 23.6).

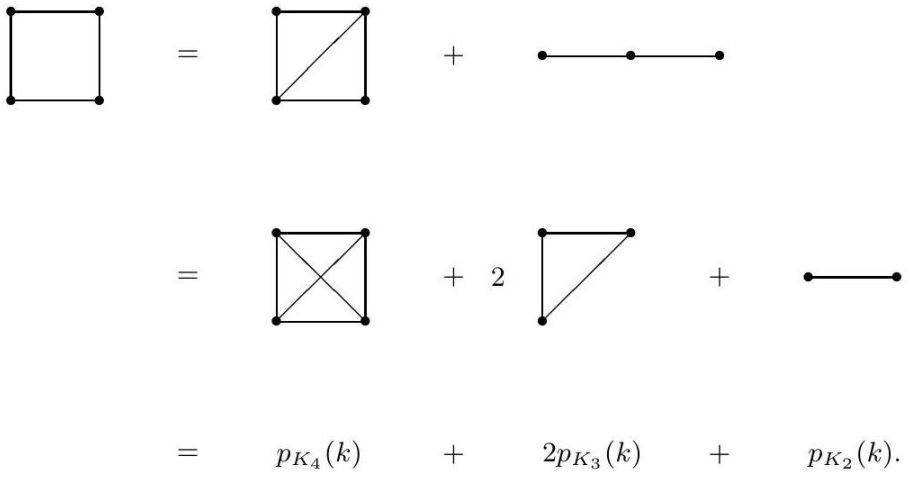


Fig. 23.6. Iterative computation of the chromatic polynomial of a square.

The expression $p_G(k)$ is an integer linear combination of expressions $p_{K_n}(k)$ for complete graphs K_n. The expression $p_{K_n}(k)$ is a polynomial function in k with integer coefficients. Thus, we have proven

Theorem 23.22. Let G be a graph. The expression $p_k(G)$ is a polynomial function in k with integer coefficients.

The polynomial function $p_G(k)$ is called the chromatic polynomial of G.

Self-test Problems

23.1. Show that the clique problem is in NP.
23.2. Develop a backtracking algorithm for the vertex cover problem and give a suitable bounding function.
23.3. Let $(M_1, \ldots, M_n)$ be a family of sets. The goal is to find a subfamily $(M_{i_1}, \ldots, M_{i_m})$ such that

$$\bigcup_{j=1}^{n} M_j = \bigcup_{j=1}^{m} M_{i_j}$$

Solve this set cover problem using a backtracking algorithm.
23.4. Compute all k-colorings of a finite graph using a backtracking algorithm.
23.5. Develop a hill-climbing algorithm for the above set cover problem.
23.6. What is the relationship between the design parameters of heuristic algorithms?
23.7. We consider a knapsack problem with 15 items and capacity $K_0 = 1019$. The items have the following value and size:

Values	535	440	418	120	512	386	48	574	376	146	143	106	55	124	266
Size	244	202	192	56	240	182	23	277	183	73	72	54	29	67	144

- Compute an optimal knapsack using a backtracking algorithm.
- Compute approximate solutions using the hill-climbing method, simulated annealing, and a genetic algorithm.
  23.8. Compute the chromatic polynomial of a pentagon.
  23.9. Let M be a matroid on a finite set A. The rank of a subset T of A is defined by

$$\mathrm{rg}(T) = \max\{|X| \mid X \subseteq T, X \in M\}$$

Show that for all $S, T \subseteq A$ and $a, b \in A$ it holds

- $0 \leq \mathrm{rg}(T) \leq |T|$.
- $T \in M \iff \mathrm{rg}(T) = |T|$.
- Monotonicity: $T \subseteq S \Longrightarrow \mathrm{rg}(T) \leq \mathrm{rg}(S)$.
- $\mathrm{rg}(T) \leq \mathrm{rg}(T \cup \{a\}) \leq \mathrm{rg}(T) + 1$.
- Submodularity: $\mathrm{rg}(T) + \mathrm{rg}(S) \geq \mathrm{rg}(T \cup S) + \mathrm{rg}(T \cap S)$.
- $\mathrm{rg}(T \cup \{a\}) = \mathrm{rg}(T \cup \{b\}) = \mathrm{rg}(T) \Longrightarrow \mathrm{rg}(T \cup \{a, b\}) = \mathrm{rg}(T)$.

Linear Optimization

Many optimization problems can be represented as linear programs. This particularly applies to production, transportation, assignment, and network problems. We develop the foundations of linear optimization and show that the theorems of Ford-Fulkerson and König-Egerváry are all special cases of the duality theorem of linear optimization.

24.1 Examples

We specify three previously treated optimization problems as linear optimization problems.

Knapsack Problem

The rational knapsack problem (23.5) is described by the following task

$$\begin{array}{ll} & \max \sum_i f_i x_i. \\ \text{s.t.} & \sum_i g_i x_i \leq K_0 \\ & x_i \geq 0 \quad \forall i \in \{1, \ldots, n\} \\ & x_i \leq 1 \quad \forall i \in \{1, \ldots, n\} \end{array}$$

Matching Problem

Let $G = (V, E)$ be a bipartite graph. We specify each subset P of E by a vector $x = (x_e)$ with

$$x_e = \begin{cases} 1 \text{ if } e \in P \\ 0 \text{ otherwise} \end{cases}$$

The problem of finding a maximum matching in G is given by the following task

$$\begin{array}{ll} & \max \sum_{e \in E} x_e. \\ \text{s.t.} & \sum_{v \in e} x_e \leq 1 \quad \forall v \in V \\ & x_e \in \{0, 1\} \quad \forall e \in E \end{array}$$

Network Problem

Let $N = (D, \kappa, q, s)$ be a flow network. Let f_{vw} be the value of a flow f on edge vw. The problem of finding a maximum flow on N is specified by the following task

$$\begin{array}{ll} & \max \sum_{v \in q^+} f_{qv} - \sum_{v \in q^-} f_{vq} \\ \text{s.t.} & f_{vw} \leq \kappa(vw) \quad \forall vw \in E \\ & \sum_{u \in v^-} f_{uv} = \sum_{w \in v^+} f_{vw} \quad \forall v \in V \backslash \{q, s\} \\ & f_{vw} \geq 0 \quad \forall vw \in E \end{array}$$

24.2 Linear Programs and Duality

In this section, the foundations of linear programming are discussed. All vectors are column vectors.

Linear Programs

Let $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. A linear program (LP) is the task of finding a vector $x \in \mathbb{R}^n$ such that

$$\begin{aligned} & \max c^T x. \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

A vector $x \in \mathbb{R}^n$ is called a feasible solution of (24.5) if it satisfies the constraints, i.e., $Ax \leq b$ and $x \geq 0$. The set of all feasible solutions of (24.5) is called the feasible region. A feasible solution x^* is called optimal if for every feasible solution x it holds $c^T x^* \geq c^T x$. An LP is called solvable if it has an optimal solution. An LP can be transformed into canonical or standard form, as specified in the first self-test problem.

An LP in the form (24.5) is called a primal LP. The corresponding dual LP reads

$$\begin{aligned} & \min b^T y. \\ & \text{s.t. } A^T y \geq c \\ & y \geq 0 \end{aligned} \tag{24.6}$$

Lemma 24.1. If x is a feasible solution of the primal LP (24.5) and y is a feasible solution of the dual LP (24.6), then

$$c^T x \leq b^T y \tag{24.7}$$

Proof. Since x is feasible and $y \geq 0$, it holds

$$b^T y = \sum_{i=1}^{m} b_i y_i \geq \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i. \tag{24.8}$$

Since y is feasible and $x \geq 0$, it follows

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j \geq \sum_{j=1}^{n} c_j x_j = c^T x \tag{24.9}$$

Thus, the statement is shown.

Example 24.2. We consider the LP

$$\begin{aligned} &\max x_1 + 3x_2 \\ &\text{s.t. } -x_1 + 3x_2 \leq 6 \\ &3x_1 + x_2 \leq 12 \\ &x_i \geq 0 \quad \forall i \end{aligned} \tag{24.10}$$

The feasible region of this LP is illustrated in Fig. 24.1. The objective function $f(x) = x_1 + 3x_2$ describes a family of parallel lines perpendicular to the vector $c = (1, 3)^T$. The optimal solution of the LP is $x^* = (3, 3)^T$ with objective function value $f(x^*) = 12$.

The corresponding dual LP reads

$$\begin{aligned} &\min 6y_1 + 12y_2. \\ &\text{s.t. } -y_1 + 3y_2 \leq 1 \\ &3y_1 + y_2 \leq 3 \\ &y_i \geq 0 \quad \forall i \end{aligned} \tag{24.11}$$

The feasible solutions of the primal LP and dual LP behave according to Lemma 24.1 as illustrated in Fig. 24.2.

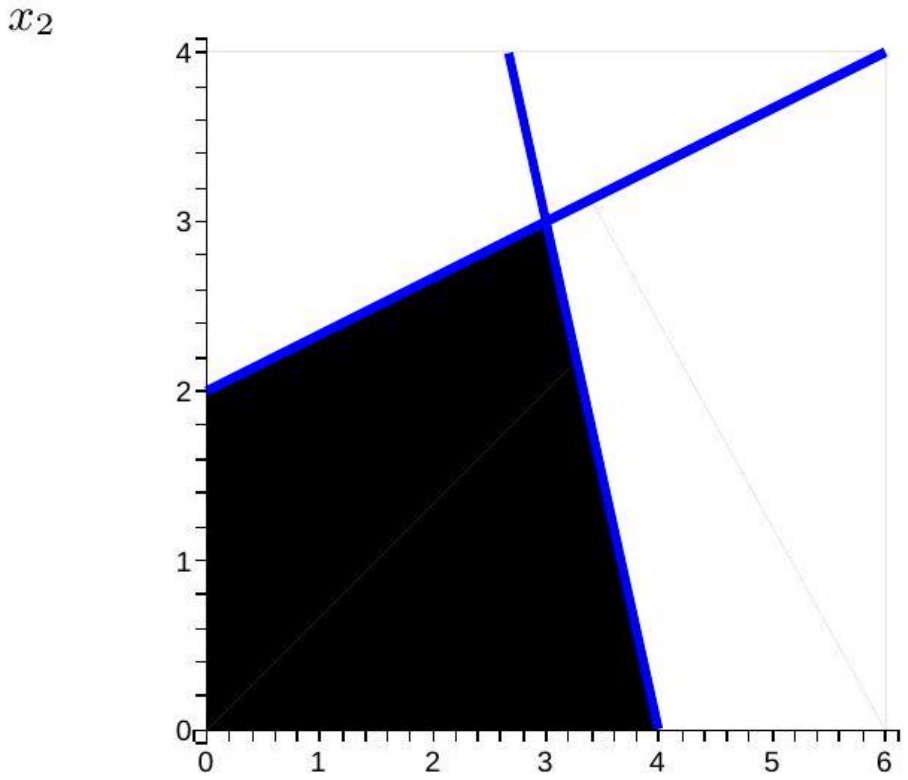
Corollary 24.3. Let x be a feasible solution of the primal LP (24.5) and y a feasible solution of the dual LP (24.6). If $c^T x = b^T y$, then x is an optimal solution of (24.5) and y is an optimal solution of (24.6).

Fig. 24.1. Feasible region (dark) of LP (24.10).

Theorem 24.4. (Farkas, 1902) The set $P = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$ is nonempty if and only if for every $y \in \mathbb{R}^m$ with $A^T y \geq 0$ it also holds $b^T y \geq 0$.
Proof. Let $\mathbf{x} \in P$. Then for every $y \in \mathbb{R}^m$ with $A^T y \geq 0$ it also holds $b^T y = (Ax)^T y = x^T \left(A^T y\right) \geq 0$.

Conversely, let P be empty. Let $a^{(i)}$ be the i-th column of A. Assume $Ax = b$ is not solvable. Then b cannot be represented as a linear combination of the columns of A. Thus, the matrix $(A \mid b)$ has rank $r(A \mid b) = r(A) + 1$ and the matrix $A' = \left(\frac{A|b}{0|1}\right)$ has rank $r(A') = r(A) + 1 = r(A \mid b)$. Thus, the last row $(0 \mid 1)$ is linearly dependent on the first m rows of A'. Therefore, there exist numbers $y_1, \ldots, y_m$ with $\sum_i {a^{(i)}}^T y_i = 0$ and $\sum_i b_i y_i = -1$, i.e., $y = (y_1, \ldots, y_m)^T$ satisfies $A^T y = 0$ and $b^T y = -1$.

Assume $Ax = b$ is solvable, but among the solutions there are no nonnegative ones. In the case $n = 1$, let $x_1 a^{(1)} = b$ and $x_1 < 0$. With $y = -b$ it follows ${a^{(1)}}^T y = -\frac{1}{x_1} b^T b \geq 0$ and $b^T y = -b^T b < 0$. Let the statement be true for all r-dimensional LPs with $r \leq n - 1$. By the induction hypothesis, there exists a $v \in \mathbb{R}^m$ with ${a^{(i)}}^T v \geq 0, 1 \leq i \leq n - 1$, and $b^T v < 0$. In the case ${a^{(n)}}^T v \geq 0$ we are done. Otherwise, set



$$\hat{a}^{(i)} = \left(a^{(i)^T} v\right) a^{(n)} - \left(a^{(n)^T} v\right) a^{(i)}, \quad 1 \leq i \leq n-1,$$
$$\hat{b} = \left(b^T v\right) a^{(n)} - \left(a^{(n)^T} v\right) b.$$

Assume there exist $x_i \geq 0, 1 \leq i \leq n-1$, such that $\sum_{i=1}^{n-1} x_i \hat{a}^{(i)} = \hat{b}$. Then it follows

$$-\frac{1}{a^{(n)^T} v} \left(\sum_{i=1}^{n-1} x_i \left(a^{(i)^T} v\right) - b^T v \right) a^{(n)} + \sum_{i=1}^{n-1} x_i a^{(i)} = b \tag{24.12}$$

The coefficients $a^{(i)}, 1 \leq i \leq n$, are non-negative by assumption. This contradicts the assumption that P is empty. Thus, $\sum_{i=1}^{n-1} x_i \hat{a}^{(i)} = \hat{b}, x_i \geq 0, 1 \leq i \leq n-1$, is not solvable. By the induction hypothesis, there exists a $w \in \mathbb{R}^m$ with $\hat{a}^{(i)T} w \geq 0, 1 \leq i \leq n-1$, and $\hat{b}^T w < 0$. For the vector $y = \left(a^{(n)^T} w\right) v - \left(a^{(n)^T} v\right) w$ it then holds $a^{(i)^T} y = \hat{a}^{(i)^T} w \geq 0, 1 \leq i \leq n-1, a^{(n)^T} y = 0$ and $b^T y = \hat{b}^T w < 0$.

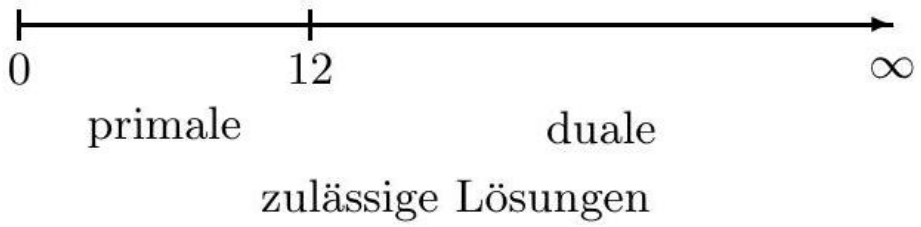


Fig. 24.2. The objective function values of the primal LP (24.10) and the dual LP (24.11).

Theorem 24.5. The set $P = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$ is non-empty if and only if for every $y \in \mathbb{R}^m$ with $A^T y \geq 0$ and $y \geq 0$ it also holds $b^T y \geq 0$.

Proof. If P is non-empty, the proof proceeds as in the previous theorem. Let P be empty. Then $P' = \{x \in \mathbb{R}^n \mid Ax + z = b, x \geq 0, z \geq 0\}$ is also empty. Let $B = (A \mid I_m)$, where I_m is the m-dimensional identity matrix. Then P' is empty if and only if $P'' = \{x \in \mathbb{R}^{n+m} \mid Bv = b, v \geq 0\}$ is empty. By the previous theorem, there exists a $y \in \mathbb{R}^m$ with $A^T y \geq 0, y = I_m y \geq 0$ and $b^T y < 0$.

Theorem 24.6. (Duality Theorem) Let a primal and the corresponding dual program be given

$$\begin{array}{ll} \max c^T x & \min b^T y. \\ \text{s.t. } Ax \leq b & \text{s.t. } A^T y \geq c \\ x \geq 0 & y \geq 0 \end{array}$$

- If both programs have at least one feasible solution, then both programs are solvable. For the optimal solutions x^* and y^* it holds $c^T x^* = b^T y^*$.
- If one of the two programs has no feasible solution, then neither of the two programs has an optimal solution.

Proof. We assume that both LPs have feasible solutions. Let $P = \big\{(x, y) \mid Ax \leq b, x \geq 0, A^T y \geq c, y \geq 0, c^T x$ We set

$$A' = \begin{pmatrix} A & 0 \\ 0 & -A^T \\ -c^T & b^T \end{pmatrix} \quad \text{and} \quad d = \begin{pmatrix} b \\ -c \\ 0 \end{pmatrix}.$$

Then $P = \{z \mid A'z \leq d, z \geq 0\}$. Assume P is empty. Then by Theorem 24.5 there exists a $y \in \mathbb{R}^{m+n+1}$ with $A^T y \geq 0, y \geq 0$ and $d^T y < 0$. That is, there exist $u \in \mathbb{R}^m, v \in \mathbb{R}^n$ and $\kappa \in \mathbb{R}$ with $u \geq 0, v \geq 0$ and $\kappa \geq 0$ such that

$$A^T u \geq \kappa c, \quad Av \leq \kappa b, \quad b^T u < c^T v \tag{24.14}$$

Assume $\kappa = 0$. Let x be a feasible solution of the primal LP and y a feasible solution of the dual LP. Then it follows contradictingly

$$0 \leq x^T \left(A^T u\right) = (Ax)^T u \leq b^T u < c^T v \leq \left(A^T y\right)^T v = y^T (Av) \leq 0$$

Thus, $\kappa > 0$ must hold. Set $x = \frac{1}{\kappa} v$ and $y = \frac{1}{\kappa} u$. Then $Ax \leq b, A^T y \geq c, x \geq 0$ and $y \geq 0$. By Lemma 24.1, $c^T v = \kappa \left(c^T x\right) \leq \kappa \left(b^T y\right) = b^T u$, which contradicts (24.14). Thus, P is not empty and by Corollary 24.3 the first statement follows.

Let the primal LP be unsolvable. Then by Theorem 24.5 there exists a $y \in \mathbb{R}^m$ with $A^T y \geq 0, y \geq 0$ and $b^T y < 0$. Let y' be a feasible solution of the dual LP. Then $y' + \kappa y$ is also a feasible solution of the dual LP for every $\kappa \geq 0$. It holds $b^T (y' + \kappa y) = b^T y' + \kappa \left(b^T y\right)$. Since $b^T y < 0$, this expression can become arbitrarily small. Thus, the dual LP is unsolvable. The converse is proven analogously.

Example 24.7. Let $G = (V, E)$ be a graph with incidence matrix $A = (a_{v,e})$. The problem of finding a maximum matching in G reads according to (24.3)

$$\begin{aligned} & \max 1^T x \\ \text{s.t.} \quad & Ax \leq 1 \\ & x_e \in \{0, 1\} \quad \forall e \in E \end{aligned}$$

We relax this problem by allowing real-valued solutions and obtain

$$\begin{aligned} & \max 1^T x. \\ & \text{s.t. } Ax \leq 1 \\ & x \geq 0 \end{aligned} \tag{24.16}$$

The dual problem reads

$$\begin{aligned} & \min 1^T y. \\ & \text{s.t. } A^T y \geq 1 \\ & y \geq 0 \end{aligned} \tag{24.17}$$

We restrict the variables and obtain the integer LP

$$\begin{aligned} & \min 1^T y \\ \text{s.t.} \quad & A^T y \geq 1 \\ & y_v \in \{0, 1\} \quad \forall v \in V \end{aligned}$$

Each edge $e = uv$ in G yields a constraint $y_u + y_v \geq 1$ in $A^T y \geq 1$. An optimal solution $y = (y_v)$ of (24.18) describes a minimum vertex cover of G.

24.3 The Feasible Region and the Role of Vertices

A subset A of $\mathbb{R}^n$ is called convex if with $x, y \in A$ also $\lambda x + (1 - \lambda) y \in A$ for all $\lambda \in [0, 1]$. A convex set thus contains the line segment between any two of its points.

Lemma 24.8. Let $A \subseteq \mathbb{R}^n$ be a convex set and r a natural number. If $x^{(1)}, \ldots, x^{(r)} \in \bar{A}$, then $\sum_i \lambda_i x^{(i)} \in A$ for all $\lambda_1, \ldots, \lambda_r \geq 0$ with $\sum_{i=1}^r \lambda_i = 1$.

The proof is carried out by complete induction on r.
The convex hull of a set $A \subseteq \mathbb{R}^n$ is defined by

$$\operatorname{conv}(A) = \left\{ \sum_{i=1}^{r} \lambda_i x^{(i)} \mid \lambda_i \geq 0, x^{(i)} \in A, \sum_{i=1}^{r} \lambda_i = 1, r \geq 1 \right\} \tag{24.19}$$

The convex hull of A is the smallest convex set containing A. If A is finite, $\operatorname{conv}(A)$ is called a convex polyhedron.

Let P be a convex polyhedron. A point $x \in P$ is called a vertex of P if for every representation

$$x = \sum_{i=1}^{r} \lambda_i x^{(i)}, \quad \text{with } x^{(i)} \in P, \lambda_i \geq 0, \text{ and } \sum_{i=1}^{r} \lambda_i = 1 \tag{24.20}$$

it holds $x = x^{(i)}$ for all i with $\lambda_i > 0$. A vertex x cannot thus be written as a convex linear combination (i.e., $\lambda_i \geq 0$ and $\sum_{i=1}^{r} \lambda_i = 1$) of other points in P. In particular, a vertex of P does not lie between two other points in P.

Example 24.9. The vertices of the LP (24.10) are $(0,0)^T, (4,0)^T, (3,3)^T$ and $(0,2)^T$.

Theorem 24.10. A convex polyhedron is the convex hull of its vertices.
Proof. Let $A = \left\{x^{(1)}, \ldots, x^{(r)}\right\}$ and $P = \operatorname{conv}(A)$. All points can be removed from A that can be represented as a convex linear combination of the remaining points. For example, if

$$x^{(1)} = \sum_{i=2}^{r} \lambda_i x^{(i)} \quad \text{with } \lambda_i \geq 0, \text{ and } \sum_{i=2}^{r} \lambda_i = 1$$

For every $y \in P$ it holds

$$y = \sum_{i=1}^{r} \mu_i x^{(i)} \quad \text{with } \mu_i \geq 0, \text{ and } \sum_{i=1}^{r} \mu_i = 1$$

It follows $y = \sum_{i=2}^{r} (\mu_1 \lambda_i + \mu_i) x^{(i)}$ and $\sum_{i=2}^{r} (\mu_1 \lambda_i + \mu_i) = 1$. Thus, $P = \operatorname{conv}\left(\left\{x^{(2)}, \ldots, x^{(r)}\right\}\right)$. It can be assumed that no point in A is a convex linear combination of the other points in A. For if

$$x^{(1)} = \sum_{j=1}^{t} \mu_j y^{(j)} \quad \text{with } y^{(j)} \in P, \mu_j \geq 0, \text{ and } \sum_{j=1}^{t} \mu_j = 1$$

and

$$y^{(j)} = \sum_{i=1}^{r} \lambda_{ji} x^{(i)} \quad \text{with } \lambda_{ji} \geq 0, \text{ and } \sum_{i=1}^{r} \lambda_{ji} = 1$$

Then $x^{(1)} = \sum_{i=1}^{r} \sum_{j=1}^{t} \mu_j \lambda_{ji} x^{(i)}$, so

$$\left(1-\sum_{j=1}^{t}\mu_j\lambda_{j1}\right)x^{(1)}=\sum_{i=2}^{r}\sum_{j=1}^{t}\mu_j\lambda_{ji}x^{(i)}$$

Since $x^{(1)}$ is not a convex linear combination of $x^{(2)},\ldots,x^{(r)}$, it follows $\sum_{j=1}^{t}\mu_j\lambda_{j1}=1$ and thus $\lambda_{j1}=1$ for every j with $\mu_j>0$. Therefore, $x^{(1)}=y^{(j)}$ for every j with $\mu_j>0$. Thus, $x^{(1)}$ is a vertex of P.

We now examine the LP

$$\begin{aligned}&\max c^Tx.\\ \text{s.t.}\quad &Ax=b\\ &x\geq 0\end{aligned}$$

We can assume $m\leq n$ and that A has rank $r(A)=m$. In the case $m>n$, $m-r(A)$ equations are redundant. Let $P=\{x\mid Ax=b,x\geq 0\}$ be the feasible region of (24.21) and denote $a^{(i)}$ the i-th column of A. Let $x\in P$ and $I=\{i\mid x_i>0\}$ the set of all indices i with $x_i>0$. The set $\left\{a^{(i)}\mid i\in I\right\}$ is called the basis of A corresponding to x. For the basis of A corresponding to x it holds

$$\sum_{i\in I}x_ia^{(i)}=b \tag{24.22}$$

Theorem 24.11. A point $x\in P$ is a vertex of P if and only if the basis of A corresponding to x is linearly independent.

Proof. Let x be a vertex of P and $I=\{i\mid x_i>0\}$. Assume $\left\{a^{(i)}\mid i\in I\right\}$ is linearly dependent. Then there exist numbers $d_i,i\in I$, not all zero, such that $\sum_{i\in I}d_ia^{(i)}=0$. Set $d_j=0$ for all $j\in\{1,\ldots,n\}\backslash I$, then $Ad=0$ for $d=(d_1,\ldots,d_n)^T$. We show that $y=x-\theta d$ and $z=x+\theta d$ are in P, where

$$\theta=\min\left\{\left.\frac{x_i}{|d_i|}\right|\ i\in I,d_i\neq 0\right\}>0$$

Since $Ad=0$, we have $Ay=b=Az$. Let $1\leq i\leq n$. If $i\notin I$, then $d_i=0$ and thus $y_i=0=z_i$. If $i\in I$ and $d_i=0$, then $y_i=x_i=z_i$. If $i\in I$ and $d_i\neq 0$, then $\theta\,|d_i|\leq\frac{x_i}{|d_i|}\cdot|d_i|=x_i$ and thus $y_i=x_i-\theta d_i\geq 0$ and $z_i=x_i+\theta d_i\geq 0$. Therefore, y and z are in P. The points y and z are different from x due to $\theta>0$ and it holds $x=\frac{1}{2}y+\frac{1}{2}z$. Thus, x is contradictingly not a vertex of P.

Conversely, suppose $\left\{a^{(i)}\mid i\in I\right\}$ is linearly independent. Assume $x=\lambda y+(1-\lambda)z$, where $0<\lambda<1$ and $y,z\in P$. By assumption, x is the unique solution of the linear system $Ax=b$ with $x_i=0$ for all $i\notin I$. Since y satisfies the equation $Ay=b$, there exists an index $i\notin I$ with $y_i>0$. Since $z\geq 0$, the i-th component of $\lambda y+(1-\lambda)z$ is contradictingly positive.

Thus, every vertex of P has at most m positive coordinates. The number of vertices of P is therefore finite. A vertex of P is called non-degenerate if it has m positive coordinates, otherwise it is called degenerate.

Theorem 24.12. If the LP (24.21) is solvable and its feasible region is a convex polyhedron, then among the optimal solutions there is a vertex.

Proof. Let $A = \{x^{(1)}, \ldots, x^{(r)}\}$ be the set of all vertices of P and $P = \text{conv}(A)$. By Theorem 24.10, every $x \in P$ has the form

$$x = \sum_{i=1}^{r} \lambda_i x^{(i)} \quad \text{with } \lambda_i \geq 0, \sum_{i=1}^{r} \lambda_i = 1$$

Let $x^{(k)}$ be a vertex with $c^T x^{(k)} = \max\{c^T x^{(i)} \mid 1 \leq i \leq r\}$. Then it follows

$$c^T x = \sum_{i=1}^{r} \lambda_i c^T x_i^{(i)} \leq \sum_{i=1}^{r} \lambda_i c^T x^{(k)} = c^T x^{(k)}$$

Thus, $x^{(k)}$ is an optimal solution of the LP.

In the above theorem, it was assumed that the feasible region of (24.21) is a convex polyhedron. In general, the feasible region of an LP is given by a sum

$$B = P + K = \{x + y \mid x \in A, y \in K\} \tag{24.23}$$

where P is a convex polyhedron and $K = \{x \mid Ax = 0, x \geq 0\}$ is a finitely generated convex cone. It can be shown that Theorem 24.12 remains valid in this generalization.

Example 24.13. We write the LP (24.10) in standard form

$$\begin{array}{ll} & \max x_1 + 3x_2 \\ \text{s.t.} & -x_1 + 3x_2 + x_3 = 6 \\ & 3x_1 + x_2 + x_4 = 12 \\ & x_i \geq 0 \quad \forall i \end{array}$$

by transforming each inequality into an equation using a slack variable ($x_3 \geq 0$ and $x_4 \geq 0$). The vertices of this LP can be determined by Theorem 24.11 through $\binom{4}{2} = 6$ systems of equations

$-x_1 + 3x_2 = 6$ $3x_1 + x_2 = 12$	$-x_1 + x_3 = 6$ $3x_1 = 12$ $\mathbf{x}^{(1)} = (3, 3, 0, 0)^T$
$\mathbf{x}^{(2)} = 6$	
$3x_1 + x_4 = 12$ $\mathbf{x}^{(3)} = (-6, 0, 0, 30)^T$	$3x_2 + x_3 = 6$ $x_2 = 12$
$3x_2 = 6$ $x_2 + x_4 = 12$ $\mathbf{x}^{(5)} = (0, 2, 0, 10)^T$ $\mathbf{x}^{(6)} = (0, 0, 6, 12)^T$	$\mathbf{x}^{(4)} = (0, 12, -30, 0)^T$ $x_3 = 6$ $x_4 = 12$

The optimal solution is $\mathbf{x}^{(1)} = (3, 3, 0, 0)^T$ with objective function value 12. Thus, $(3, 3)^T$ is the optimal solution of (24.10).

24.4 Integer Programming

An integer linear program (ILP) has the form

$$\begin{aligned} & \max c^T x. \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x_i \in \mathbb{Z} \quad \forall i \end{aligned}$$

We relax the problem by allowing real-valued solutions and obtain

$$\begin{aligned} & \max c^T x. \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned} \tag{24.25}$$

The feasible region $P = \{x \mid Ax = b, x \geq 0\}$ of this LP is called integer if every vertex of P is integer. We develop a sufficient condition for the integrality of P.

A square matrix A is called unimodular if A has determinant ± 1. An integer matrix A is called totally unimodular if every non-singular square submatrix of A is unimodular. A totally unimodular matrix thus consists only of entries 0, 1, and -1.

Lemma 24.14. If A is a totally unimodular $m \times n$ matrix, then the following matrices are also totally unimodular: $-A, A^T, (A \mid I_m)$ and $(A \mid -A)$. Similarly, every matrix is totally unimodular that results from multiplying a row or column of A by -1.

Theorem 24.15. If A is a totally unimodular $m \times n$ matrix and $b \in \mathbb{Z}^m$, then $P = \{x \mid Ax = b, x \geq 0\}$ is integer.

Proof. Let x be a vertex of P and $\left\{a^{(i)} \mid i \in I\right\}$ the basis of A corresponding to x. By (24.22), $\sum_{i \in I} x_i a^{(i)} = b$. For the components of this system of equations, by Cramer's rule

$$x_i = \frac{\det\left(B^{(i)}\right)}{\det(B)}, \quad i \in I,$$

where B is the matrix formed by the columns $a^{(i)}, i \in I$, and $B^{(i)}$ is obtained from B by replacing the i-th column with b. Since A is totally unimodular, it follows immediately from Theorem 24.11 $\det(B) = \pm 1$. By assumption, all entries of $B^{(i)}$ are integers, so $\det\left(B^{(i)}\right)$ is an integer. Thus, x is integer.

Lemma 24.16. The incidence matrix of a graph G is totally unimodular if and only if G is bipartite.

Proof. Let G not be bipartite and A the incidence matrix of G. By Theorem 21.19, G contains a simple cycle of odd length. The corresponding square submatrix of A has determinant ± 2.

Let $G = (V, E)$ be bipartite with vertex partition $V = V_1 \cup V_2$ and A the incidence matrix of G. For every 1-row submatrix of A, the statement is true because A contains only entries 0 and 1. Assume the statement is true for t-row

submatrices of A. Let B be a $(t+1)$-row submatrix of A. We distinguish three cases:

- The matrix B contains a zero column. Then $\det(B) = 0$.
- The matrix B contains a column j with exactly one 1, for example $a_{ij} = 1$. Then $\det(B) = 1 \cdot \det(B')$, where B' is the matrix obtained from B by removing the i-th row and j-th column. By the induction hypothesis on B', the claim follows.
- Otherwise, B contains two 1s per column, because A is an incidence matrix. Since G is bipartite, the rows of B can be divided into two sets $\left\{b^{(v)} \mid v \in V_1'\right\}, V_1' \subseteq V_1$, and $\left\{b^{(v)} \mid v \in V_2'\right\}, V_2' \subseteq V_2$, such that

$$\sum_{v \in V_1'} b^{(v)} = \sum_{v \in V_2'} b^{(v)}$$

Thus, $\det(B) = 0$.

Example 24.17. Let $G = (V, E)$ be a bipartite graph with incidence matrix $A = (a_{v,e})$. By Lemmas 24.14 and 24.16, A and A^T are totally unimodular. Thus, the relaxation (24.16) of the problem of finding a maximum matching in G has an optimal integer solution x^* by Theorem 24.15. Similarly, the relaxation (24.17) of the problem of finding a minimum vertex cover of G has an optimal integer solution y^*. Due to the constraints, $x^* \in \{0,1\}^{|E|}$ and $y^* \in \{0,1\}^{|V|}$. Thus, x^* describes a maximum matching in G and y^* a minimum vertex cover of G. By the duality theorem, the two relaxed LPs (24.16) and (24.17) have the same objective function value, i.e., $1^T x^* = 1^T y^*$. Thus, we have proven the König-Egerváry theorem using linear optimization.

Let $D = (V, E)$ be a digraph. The incidence matrix of $D = (V, E)$ is a matrix $A = (a_{v,e})$ defined by

$$a_{v,e} = \begin{cases} 1 \text{ if } v = e^- \\ -1 \text{ if } v = e^+ \\ 0 \text{ otherwise.} \end{cases}$$

Each column of A contains exactly one entry 1 and one entry -1, the rest are 0.

Lemma 24.18. The incidence matrix of a digraph is totally unimodular.

Proof. Let $D = (V, E)$ be a digraph with incidence matrix $A = (a_{v,e})$. For every 1-row submatrix of A, the statement is true since A contains only entries 0, 1, and -1. Assume the statement is true for t-row submatrices of A. Let B be a $(t+1)$-row submatrix of A. We distinguish three cases:

- The matrix B contains a zero column. Then $\det(B) = 0$.
- The matrix B contains a column j with exactly one 1, for example $a_{ij} = 1$. Then $\det(B) = 1 \cdot \det(B')$, where B' is the matrix obtained from B by removing the i-th row and j-th column. By the induction hypothesis on B', the claim follows.

- Otherwise, B contains one 1 and one -1 per column. Then the sum of the rows of B is 0 and thus $\det(B) = 0$.

Example 24.19. Let $N = (D, \kappa, q, s)$ be a flow network and $A = (a_{v,e})$ the incidence matrix of D. Let A' be the matrix obtained from A by removing the rows q and s. Then the Kirchhoff's law constraints read

$$A'x = 0.$$

Let w^T be the row q of A and $c \geq 0$ describe the capacity of N. Then the problem of finding a maximum flow on N reads

$$\begin{aligned} &\max w^T x. \\ &\text{s.t. } A'x = 0 \\ &x \leq c \\ &x \geq 0 \end{aligned} \tag{24.26}$$

The coefficient matrix A' of this LP is totally unimodular because A is totally unimodular by Lemma 24.18. By Theorem 24.15, this LP has an optimal integer solution x^*. The dual problem reads

$$\begin{aligned} &\min c^T y. \\ &\text{s.t. } A'^T z + y \geq w \\ &y \geq 0 \end{aligned} \tag{24.27}$$

For the constraints, it holds

$$\begin{pmatrix} A^T & I \\ 0 & I \end{pmatrix} \begin{pmatrix} z \\ y \end{pmatrix} \geq \begin{pmatrix} w \\ 0 \end{pmatrix}$$

Since A is totally unimodular by Lemma 24.18, this coefficient matrix is also totally unimodular. Thus, the dual problem has an optimal integer solution (y^*, z^*) by Theorem 24.15. We extend the vector z^* by components q and s as

$$\hat{z}_v^* = \begin{cases} -1 \text{ if } v = q \\ 0 \text{ if } v = s \\ z_v^* \text{ otherwise.} \end{cases}$$

The set $S = \{v \in V \mid \hat{z}_v^* \leq -1\}$ is by definition a cut in N. It holds $A^T \hat{z}^* + y^* = \big(A^T z^* - w\big) + y^* \geq 0$. Thus, for every edge vw the inequality $y_{vw}^* + \hat{z}_v^* - \hat{z}_w^* \geq 0$ holds. From this, for every edge vw with $v \in S$ and $w \notin S$ it immediately follows $y_{vw}^* \geq 1$, which by optimality implies $y_{vw}^* = 1$. Furthermore, for every edge vw with $w \in S$ and $v \notin S$, it holds $y_{vw}^* + \hat{z}_v^* - \hat{z}_w^* \geq y_{vw}^* \geq 0$, thus by optimality $y_{vw}^* = 0$. Finally, by optimality $y_{vw}^* = 0$ for every edge with $v, w \in S$ or $v, w \notin S$. Thus, S defines a minimum cut in N with capacity $c^T y^*$. By the duality theorem, the two relaxed LPs (24.26) and (24.27) have the same objective function value, i.e., $w^T x^* = c^T y^*$. Thus, we have shown the Ford-Fulkerson theorem using linear optimization.

24.5 The Simplex Method

We consider the LP

s.t.

where $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ with $m \leq n$ and $r(A) = m$.
Let $x^{(0)}$ be a vertex of $P = \{x \mid Ax = b, x \geq 0\}$ with basis $\left\{a^{(i)} \mid i \in I\right\}$. Then there are uniquely determined numbers $\alpha_{ij} \in \mathbb{R}, i \in I$ and $j \in \underline{n}$, with

$$a^{(j)} = \sum_{i\in I} \alpha_{ij} a^{(i)}, \quad 1 \leq j \leq n \tag{24.29}$$

Since $\left\{a^{(i)} \mid i \in I\right\}$ is a basis, it follows $\alpha_{ii} = 1$ for all $i \in I$ and $\alpha_{ij} = 0$ for all $i, j \in I$ with $i \neq j$.
Lemma 24.20. For every $x \in P$ it holds

$$c^T x = c^T x^{(0)} - \sum_{j\notin I} (d_j - c_j)\, x_j \tag{24.30}$$

where

$$d_j = \sum_{i\in I} \alpha_{ij} c_i, \quad j \notin I \tag{24.31}$$

Proof. For the vertex $x^{(0)}$ of P, it holds $\sum_{i\in I} x_i^{(0)} a^{(i)} = b$. On the other hand, by (24.30)

$$b = \sum_{j=1}^{n} x_j a^{(j)} = \sum_{j=1}^{n} x_j \left(\sum_{i\in I} \alpha_{ij} a^{(i)}\right) = \sum_{i\in I} \left(\sum_{j=1}^{n} \alpha_{ij} x_j\right) a^{(i)}.$$

By coefficient comparison, $x_i^{(0)} = \sum_{j=1}^{n} \alpha_{ij} x_j = x_i + \sum_{j\notin I} \alpha_{ij} x_j$ for all $i \in I$. Thus, it follows

$$c^T x = \sum_{j=1}^{n} c_j x_j = \sum_{i\in I} c_i x_i + \sum_{j\notin I} c_j x_j = \sum_{i\in I} c_i x_i^{(0)} - \sum_{j\notin I} \sum_{i\in I} (\alpha_{ij} c_i - c_j)\, x_j.$$

Thus, the statement is proven.
Theorem 24.21. Under the above assumptions:

1. If $d_j \geq c_j$ for all $j \notin I$, then $x^{(0)}$ is an optimal solution of the LP (24.28).
2. If there exists a $j \notin I$ with $d_j < c_j$ such that $\alpha_{ij} \leq 0$ for all $i \in I$, then the LP (24.28) is not solvable.
3. For every $j \notin I$ with $d_j < c_j$ there exists an $i \in I$ with $\alpha_{ij} > 0$. Then there exists a vertex $x^{(1)}$ of P with $c^T x^{(1)} \geq c^T x^{(0)}$. The case $c^T x^{(1)} = c^T x^{(0)}$ can only occur if $x^{(0)}$ is degenerate.

Proof. Let $d_j \geq c_j$ for all $j \notin I$. Let $x \in P$. By Lemma 24.20 and the assumption, $c^T x \leq c^T x^{(0)}$. Thus, $x^{(0)}$ is optimal.

Let $l \notin I$ with $d_l < c_l$ and $\alpha_{il} \leq 0$ for all $i \in I$. Let $\delta > 0$. We define $x \in \mathbb{R}^n$ by

$$x_i = \begin{cases} x_i^{(0)} - \delta\alpha_{il} & \text{if } i \in I \\ \delta & \text{if } i = l \\ 0 & \text{otherwise.} \end{cases}$$

Due to the assumption, $x \geq 0$ and with (24.29) it holds

$$\begin{aligned} Ax &= \sum_{i \in I} x_i a^{(i)} + \delta a^{(l)} \\ &= \sum_{i \in I} x_i^{(0)} a^{(i)} - \delta \sum_{i \in I} \alpha_{il} a^{(i)} + \delta a^{(l)} \\ &= \sum_{i \in I} x_i^{(0)} a^{(i)} - \delta a^{(l)} + \delta a^{(l)} = b. \end{aligned}$$

Thus, $x \in P$. By (24.30), $c^T x = c^T x^{(0)} - (d_l - c_l)\,\delta$. Since $d_l < c_l$, $c^T x \geq c^T x^{(0)}$. As $\delta > 0$ can be chosen arbitrarily, $c^T x$ becomes arbitrarily large. Thus, the LP (24.28) is not solvable.

Let $i \in I$ and $l \notin I$ with $d_l < c_l$ and $\alpha_{il} > 0$. We set

$$\delta = \min \left\{ \left. \frac{x_i^{(0)}}{\alpha_{il}} \right| i \in I, \alpha_{il} > 0 \right\}$$

Let x be defined as in (24.32). Let $k \in I$ with $\delta = x_k^{(0)}/\alpha_{kl}$. For every $i \in I$ with $\alpha_{il} > 0$ it holds

$$x_i = x_i^{(0)} - \frac{x_k^{(0)}}{\alpha_{kl}} \alpha_{il} \geq x_i^{(0)} - \frac{x_i^{(0)}}{\alpha_{il}} \alpha_{il} = 0,$$

so $x \geq 0$. As in the second case, $x \in P$ and $c^T x = c^T x^{(0)} - (d_l - c_l)\,\delta$.

Since $x_k = x_k^{(0)} - \frac{x_k^{(0)}}{\alpha_{kl}} \alpha_{kl} = 0$, the basis corresponding to x is in $\left\{ a^{(i)} \mid i \in (I \backslash \{k\}) \cup \{l\} \right\}$. Assume this set is linearly dependent. Then

$$\sum_{i \in I, i \neq k} \kappa_i a^{(i)} + \kappa_l a^{(l)} = 0$$

where not all coefficients vanish. Since $\left\{ a^{(i)} \mid i \in I \right\}$ is a basis, it follows $\kappa_l \neq 0$ and thus

$$a^{(l)} = \sum_{i \in I, i \neq k} \left(-\frac{\kappa_i}{\kappa_l} \right) a^{(i)}$$

By coefficient comparison with (24.29), it follows contradictingly $\alpha_{kl} = 0$. Thus, the above set is linearly independent. By Theorem 24.11, the corresponding vector x is a vertex of P, which has the desired properties.

We examine the third case of the above theorem in more detail. In it, a new vertex $x^{(1)}$ is determined from a given vertex $x^{(0)}$ by a basis exchange. The basis $\left\{a^{(i)} \mid i \in I\right\}$ of $x^{(0)}$ is replaced by the basis $\left\{a^{(i)} \mid i \in I'\right\}$ of $x^{(1)}$ with $I' = (I \backslash \{k\}) \cup \{l\}$. The necessary data for this are arranged in a so-called tableau

		x_i	x_j	x_k	x_l	$\dots$
x_i	$x_i^{(0)}$	1	α_{ij}	0	α_{il}	
x_k	$x_k^{(0)}$	0	α_{kj}	1	α_{kl}	
$\vdots$	$\vdots$					
	$c^T x^{(0)}$	0	f_j	0	f_l	$\dots$

where $f_i = d_i - c_i$ for all $1 \leq i \leq n$. For the basis exchange, the so-called pivot element $\alpha_{kl} > 0$ plays an important role. The tableau for the new basis is obtained by a so-called exchange step involving the k-th row and l-th column

		x_i	x_j	x_k	x_l

$\dots$

By (24.29), it holds

$$a^{(k)} = \sum_{i \in I, i \neq k} \left(-\frac{\alpha_{il}}{\alpha_{kl}}\right) a^{(i)} + \frac{1}{\alpha_{kl}} a^{(l)} \tag{24.35}$$

and thus

$$\begin{aligned} a^{(j)} &= \sum_{i \in I, i \neq k} \alpha_{ij} a^{(i)} + \alpha_{kj} a^{(k)} \\ &= \sum_{i \in I, i \neq k} \left(\alpha_{ij} - \frac{\alpha_{il}}{\alpha_{kl}} \alpha_{kj}\right) a^{(i)} + \frac{\alpha_{kj}}{\alpha_{kl}} a^{(l)}, \quad j \neq l. \end{aligned} \tag{24.36}$$

Let A_i be the i-th row of $A = (\alpha_{ij})$ or (24.33) and A'_i the i-th row of $A' = \left(\alpha'_{ij}\right)$ or (24.34). With (24.35) and (24.36), it follows

$$A'_i = \begin{cases} A_i - \frac{\alpha_{il}}{\alpha_{kl}} A_k & \text{if } i \neq l \\ \frac{1}{\alpha_{kl}} A_k & \text{if } i = l \end{cases}$$

Furthermore, $x_l^{(1)} = \frac{1}{\alpha_{kl}} x_k^{(0)}$ and $x_i^{(1)} = x_i^{(0)} - \frac{\alpha_{il}}{\alpha_{kl}} x_k^{(0)}$ for all $i \in I$ with $i \neq l$. Thus, (24.37) also holds for the vertex column.

Let $f = d - c$, where $d_i = 0$ for all $i \in I$, and $f' = d' - c$, where $d'_i = 0$ for all $i \in I'$. By (24.31), $d = \sum_{i \in I} c_i A_i$ and $d' = \sum_{i \in I'} c_i A'_i$. With (24.31) and (24.37), it follows

$$
\begin{aligned}
f' &= \sum_{i\in I'} c_i A'_i - c \\
&= \sum_{i\in I, i\neq k} c_i \left(A_i - \frac{\alpha_{il}}{\alpha_{kl}} A_k \right) + \frac{1}{\alpha_{kl}} c_l A_k - c \\
&= \sum_{i\in I, i\neq k} c_i A_i - \sum_{i\in I, i\neq k} \frac{\alpha_{il} c_i}{\alpha_{kl}} A_k + \frac{c_l}{\alpha_{kl}} A_k - c \\
&= \left(\sum_{i\in I} c_i A_i - c_k A_k \right) - \left(\frac{d_l}{\alpha_{kl}} A_k - c_k A_k \right) + \frac{c_l}{\alpha_{kl}} A_k - c \\
&= d - \frac{f_l}{\alpha_{kl}} A_k - c \\
&= f - \frac{f_l}{\alpha_{kl}} A_k
\end{aligned} \tag{24.38}
$$

Finally, by (24.30), $c^T x^{(1)} = c^T x^{(0)} - \delta f_l = c^T x^{(0)} - \frac{f_l}{\alpha_{kl}} x_k^{(0)}$. Thus, the tableau (24.34) is obtained from the tableau (24.33) by the Gaussian elimination described in (24.37).

To construct the first tableau, two cases are distinguished:

1. Let an LP in canonical form be given

$$
\begin{aligned}
& \max c^T y. \\
\text{s.t.} \quad & Ay \leq b \\
& y \geq 0
\end{aligned}
$$

Let $b \geq 0$. This LP is transformed into standard form

$$
\begin{aligned}
& \max c^T x \\
& \text{s.t. } (A \mid I_m)\, x = b \\
& x \geq 0
\end{aligned} \tag{24.40}
$$

where I_m denotes the $m \times m$ identity matrix. This LP has $x = (0, b)^T$ as a feasible solution. By Theorem 24.11, x is a vertex with the unit vectors of I_m as basis. From this, we obtain the initial tableau

		x_1	$\ldots$	x_n	x_{n+1}	$\ldots$	x_{n+m}
x_{n+1}	b_1	α_{11}	$\ldots$	α_{1n}	1		0
$\ldots$						$\ddots$	
x_{n+m}	b_m	α_{m1}	$\ldots$	α_{mn}	0		1
	0	$-c_1$	$\ldots$	$-c_n$	0	$\ldots$	0

2. Let an LP in standard form be given

s.t.

Without loss of generality, let $b \geq 0$. We first consider the LP

$$\begin{aligned} & \min 1^T y \\ \text{s.t.} \quad & Ax + y = b \\ & x \geq 0, y \geq 0 \end{aligned}$$

This LP has a feasible solution $(x, y) = (0, b)$ due to $b \geq 0$, which by Theorem 24.11 is a vertex.

Lemma 24.22. The LP (24.42) has a feasible solution if and only if the LP (24.43) has optimal value 0.

Proof. Let x be feasible in (24.42). Then $y^* = b - Ax = 0$ and thus (x, y^*) is an optimal solution of (24.43). Conversely, let (x^*, y^*) be an optimal solution of (24.43) with $1^T y^* = 0$. Then $y^* = 0$ and thus $Ax^* = Ax^* + y^* = b$. Therefore, x^* is feasible in (24.42).

The LP (24.43) can be solved using the simplex method described below. If 0 is the optimal value, then the optimal solution (x^*, y^*) yields, by Theorem 24.11, a (possibly degenerate) vertex x^* of (24.42).

The simplex algorithm summarizes our considerations:

1. Construct an initial tableau.
2. Test f_j:
   a) If $f_j \geq 0$ for all $j \notin I$, then the solution is optimal.
   b) There exists a $j \notin I$ with $f_j < 0$ such that $\alpha_{ij} \leq 0$ for all $i \notin I$. Then the LP has no optimal solution.
   c) Otherwise, choose a $l \notin I$ with $f_l < 0$ and determine a $k \in I$ with $\frac{x_k^{(0)}}{\alpha_{kl}} \leq \frac{x_i^{(0)}}{\alpha_{il}}$ for all $i \in I$ with $\alpha_{il} > 0$.
3. Construct a new tableau by swapping x_k and x_l and go to 2.

Example 24.23. Given is the canonical LP

$$\begin{aligned} & \max x_1 + x_2. \\ \text{s.t.} \quad & x_1 + 2x_2 \leq 4 \\ 2x_1 - x_2 \quad & \leq 3 \\ x_2 \quad & \leq 1 \\ x_i \geq 0 \quad & \forall i \end{aligned}$$

This LP is transformed into a standard LP by introducing slack variables

$$\begin{aligned} & \max x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 + x_3 = 4 \\ & 2x_1 - x_2 + x_4 = 3 \\ & x_2 + x_5 = 1 \\ & x_i \geq 0 \quad \forall i \end{aligned}$$

This LP has $x^{(0)} = (0, 0, 4, 3, 1)^T$ as the starting vertex. The corresponding initial tableau is

		x_1	x_2	x_3	x_4	x_5
x_3	4	1	2	1	0	0
x_4	3	**2**	-1	0	1	0
x_5	1	0	1	0	0	1
	0	-1	-1	0	0	0

We choose $l = 1$ and determine k by $\min\left\{\frac{4}{1}, \frac{3}{2}\right\} = \frac{3}{2}$, so $k = 4$. The next tableau is

		x_1	x_2	x_3	x_4	x_5
x_3	$\frac{5}{2}$	0	$\frac{5}{2}$	1	$-\frac{1}{2}$	0
x_1	$\frac{3}{2}$	1	$-\frac{1}{2}$	0	$\frac{1}{2}$	0
x_5	1	0	**1**	0	0	1
	$\frac{3}{2}$	0	$-\frac{3}{2}$	0	$\frac{1}{2}$	0

The next vertex is $x^{(1)} = \left(\frac{3}{2}, 0, \frac{5}{2}, 0, 1\right)^T$. We choose $l = 2$ and determine k by $\min\left\{\frac{5}{2}/\frac{5}{2}, \frac{1}{1}\right\} = 1$, so $k = 5$. The next tableau is

		x_1	x_2	x_3	x_4
x_3	0	0	0	1	$-\frac{1}{2}$
x_1	2	1	0	0	$\frac{1}{2}$
x_2	1	0	1	0	$\frac{1}{2}$
	3	0	0	0	$\frac{1}{2}$

We obtain the vertex $x^{(2)} = (2, 1, 0, 0, 0)^T$. This solution is optimal. Thus, the original canonical LP has the optimal solution $(2, 1)^T$.

There are linear programs for which the simplex method can only solve in exponential time. However, it can be shown using a probabilistic model that the simplex algorithm has polynomial average running time. On the other hand, there are polynomial algorithms for solving linear programs. These include the algorithms by Khachiyan and Karmarkar. Linear programming is thus in the class P. In contrast, integer linear programming is NP-complete. However, someone who believes that the polynomial algorithms are better in terms of running time than the simplex method is mistaken. The simplex algorithm is superior to the polynomial algorithms on average.

Self-test Problems

24.1. The function

$$c^T x = c_1 x_1 + \ldots + c_n x_n$$

is to be maximized under the constraints

$$
\begin{aligned}
a_{11}x_1 + \ldots + a_{1n}x_n &= b_1 \\
&\vdots \\
a_{k1}x_1 + \ldots + a_{kn}x_n &= b_k \\
a'_{11}x_1 + \ldots + a'_{1n}x_n &\leq b'_1 \\
&\vdots \\
a'_{l1}x_1 + \ldots + a'_{ln}x_n &\leq b'_l
\end{aligned}
$$

and the sign constraints

$$x_1 \geq 0, \quad \ldots \quad , x_r \geq 0$$

Show that this problem can be brought into canonical form

$$
\begin{aligned}
&\max c^T x. \\
\text{s.t.} \quad &Ax \leq b \\
&x \geq 0
\end{aligned}
$$

Show that the above problem can be transformed into standard form

s.t.

$$
\begin{aligned}
&\max c^T x. \\
&x \geq 0
\end{aligned}
\tag{24.45}
$$

24.2. An alcoholic beverage is to be produced from three liquid ingredients, with restrictions on the proportions and mixtures of the ingredients:

	Ingredients			Content	
Components	A	B	C	min	max
Alcohol	10%	15%	5%	11%	12%
Sugar	8 g/l	10 g/l	22 g/l	10 g/l	16 g/l
Flavoring	2 E	6 E	10 E	2 E	7 E
A	1	0	0	20%	
B	0	1	0		40%
C	0	0	1		50%
Price / l	7	5	3		

24.3. A carpenter makes tables and chairs. He sells tables for 40 euros and chairs for 10 euros. The carpenter works 40 hours per week. He needs 8 hours for a table and 4 hours for a chair. He produces at least three times as many chairs as tables.

Formulate a corresponding LP for profit maximization and solve it. What is the optimal solution of the corresponding ILP?

24.4. Prove Theorem 24.8.

24.5. Show that the feasible region $P = \{x \mid \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ is convex.

24.6. Show that the set of all optimal solutions of an LP is convex.

24.7. Let $A \subseteq \mathbb{R}^n$. Show

- $A \subseteq \text{conv}(A)$.
- $\text{conv}(A)$ is convex.
- If $A' \subseteq \mathbb{R}^n$ is convex with $A \subseteq A'$, then $\text{conv}(A) \subseteq A'$.

24.8. Let $A \subseteq \mathbb{R}^n$ be a convex set and $f : \mathbb{R}^n \to \mathbb{R}^n$ a linear mapping. Show that $f(A)$ is also convex.

24.9. Solve the LP

$$\begin{aligned} & \max 49x_1 + 24x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 \leq 8 \\ & 25x_1 + 12x_2 \leq 67 \\ & x_i \geq 0 \quad \forall i \end{aligned}$$

Show that the optimal solution of the corresponding ILP cannot be obtained by rounding the optimal solution of the LP.

24.10. Solve the following LP by inspection of the vertices

$$\begin{aligned} & \min x_1 + 2x_2 - x_3 + x4 + 2x_5 \\ \text{s.t.} \quad & x_1 - x_2 + 5x_3 - x_4 + x_5 = 8 \\ & 5x_1 - 4x_2 + 13x_3 - 2x_4 + x_5 = 20 \\ & x_i \geq 0 \quad \forall i \end{aligned}$$

24.11. Solve the following LP using the simplex algorithm

$$\begin{aligned} & \max 2x_1 + 3x_2 \\ & x_1 - x_2 \leq 2 \\ & -x_1 + x_2 \leq 2 \\ \text{s.t.} \quad & 2x_1 + x_2 \leq 8 \\ & x_1 + 3x_2 \leq 12 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

24.12. Let $G = (V, E)$ be a graph. A edge cover of G is a set of edges $E' \subseteq E$ such that every vertex in G is incident with at least one edge in E'. A minimum edge cover of G is an edge cover of G with minimum cardinality.

Specify the problem of finding a minimum edge cover of G as an ILP. Relax this ILP and give the corresponding dual LP. Restrict the variables of the dual LP to integer values and interpret this problem.

Disclaimer