Appendix A
Some NP-Complete Problems

To ask the hard question is simple.
But what does it mean?
What are we going to do?
W.H. Auden

In this appendix we present a brief list of NP-complete problems; we restrict ourselves to problems which either were mentioned before or are closely related to subjects treated in the book. A much more extensive list can be found in Garey and Johnson [GarJo79].

Chinese postman (cf. Sect. 14.5)

Let $G=(V,A,E)$ be a mixed graph, where A is the set of directed edges and E the set of undirected edges of G. Moreover, let w be a nonnegative length function on $A\cup E$, and c be a positive number. Does there exist a cycle of length at most c in G which contains each edge at least once and which uses the edges in A according to their given orientation?

This problem was shown to be NP-complete by Papadimitriou [Pap76], even when G is a planar graph with maximal degree 3 and $w(e)=1$ for all edges e. However, it is polynomial for graphs and digraphs; that is, if either $A=\emptyset$ or $E=\emptyset$. See Theorem 14.5.4 and Exercise 14.5.6.

Chromatic index (cf. Sect. 9.3)

Let G be a graph. Is it possible to color the edges of G with k colors, that is, does $\chi'(G)\le k$ hold?

Holyer [Hol81] proved that this problem is NP-complete for each $k\ge 3$; this holds even for the special case where $k=3$ and G is 3-regular.

Chromatic number (cf. Sect. 9.1)

Let G be a graph. Is it possible to color the vertices of G with k colors, that is, does $\chi(G)\le k$ hold?

Karp [Kar72] proved that this problem is NP-complete for each $k\ge 3$. Even the special case where $k=3$ and G is a planar graph with maximal degree 4 remains NP-complete; see [GarJS76]. Assuming P $\neq$ NP, there is

DOI 10.1007/978-3-642-32278-5, © Springer-Verlag Berlin Heidelberg 2013

not even a polynomial approximative algorithm which always needs fewer than $2\chi(G)$ colors; see [GarJo76]. For perfect graphs, the chromatic number can be computed in polynomial time; see [GroLS93].

As noted in Lemma 9.2.1, the clique number $\omega(G)$ is a (trivial) lower bound for $\chi(G)$. However, it is (for general graphs) NP-complete to decide if one has equality, even if some colouring of G with $\chi(G)$ colours is given; this is equivalent to the result of [BusPa06] on the clique partition and independence numbers, using Lemma 9.2.1.

Clique (cf. Exercise 2.8.4)

Let $G = (V,E)$ be a graph and $c \leq |V|$ a positive integer. Does G contain a clique consisting of c vertices?

This problem is NP-complete by a result of Karp [Kar72]; thus determining the clique number $\omega(G)$ is an NP-hard problem. Also, the related question of whether G contains a clique with at least $r|V|$ vertices, where $0 < r < 1$, is NP-complete for fixed r. Assuming P $\neq$ NP, there is not even a polynomial ε-approximative algorithm for determining a maximal clique; see [AroSa02]. However, the problem can be solved in polynomial time for perfect (in particular, for bipartite) graphs; see [GroLS93].

Clique Partition (cf. Sect. 9.2)

Let $G = (V,E)$ be a graph and $c \leq |V|$ a positive integer. Does V admit a partition into at most c cliques?

This problem is NP-complete by a result of Karp [Kar72]. Thus determining the clique partition number $\theta(G)$ is an NP-hard problem, but the problem can be solved in polynomial time for perfect (in particular, for bipartite) graphs; see [GroLS93]. As noted in Lemma 9.2.1, the independence number $\alpha(G)$ is a (trivial) lower bound for $\theta(G)$. However, it is (for general graphs) NP-complete to decide if one has equality, even if some minimum clique partition is given; see [BusPa06].

Diameter (cf. Sect. 3.9)

Let $G = (V,E)$ be a connected graph, and let $c \leq |V|$ be a positive integer. Recall that the diameter of G can be determined efficiently; see Sect. 3.9. However, the following two related problems are NP-complete; see [ChvTh78] and [GarJo79].

(1) Does there exist a strongly connected orientation H of G with diameter at most c? Note that Robbins' theorem allows us to check efficiently whether a strongly connected orientation exists (and to find such an orientation, if possible); see Sect. 1.6.
(2) Let C be a given set of at most $|E|$ nonnegative integers. Does there exist a mapping $w : E \to C$ such that G has *weighted diameter* at most c, that is, such that any two vertices u, v have distance $d(u,v) \leq c$ in the network (G,w)? This problem remains NP-complete even for $C = \{0,1\}$.

Discrete metric realization (cf. Sect. 3.2)

Let $D = (d_{xy})$ be an $n \times n$ matrix with integer entries representing distances in a finite metric space. Is there a network (G, w) of total length $\leq k$ realizing D?

Winkler [Win88] proved that this problem—and also the analogous real problem—is NP-complete.

Disjoint connecting paths (cf. Sect. 7.1)

Let G be a graph, k an integer, and $(s_1, t_1), \dots, (s_k, t_k)$ pairs of vertices (usually called *terminals*). Are there disjoint paths $P_1, P_2, \dots, P_k$ such that P_i connects s_i with t_i?

Here *disjoint* may be interpreted as either edge disjoint and vertex disjoint. In both cases, this problem is NP-complete as stated (that is, if k is included in the input), even when G is assumed to be planar; these results are due to Karp [Kar75] and Lynch [Lyn75]. However, the problem becomes polynomial for every fixed k, a result first proved in the framework of the fundamental theory of graph minors due to Robertson and Seymour [RobSe95]. Recently, a quadratic time algorithm was given by Kawarabayashi, Kobayashi and Reed [KawKR12]; this paper also contains a good discussion of the problem, including many further references.

Disjoint paths (cf. Sect. 7.1)

Let $G = (V, E)$ be a graph, s and t be two vertices of G, and k and c be two positive integers. Does G contain k vertex disjoint paths of length at most c from s to t?

Itai, Pearl and Shiloach [ItaPS82] proved that this problem is NP-complete for each fixed $k \geq 5$ (whereas it is polynomial for fixed $k \leq 4$). Similar results hold for edge disjoint paths from s to t, and for the analogous problems where each path should contain precisely c edges. In contrast, the maximal number of (edge or vertex) disjoint paths from s to t can be determined efficiently using network flow methods if no restrictions are added; see Sect. 7.1.

Dominating set (cf. Sect. 2.8)

Let $G = (V, E)$ be a graph and k a positive integer. Does G admit a dominating set D with $|D| \leq k$? The variant where D is required to be connected is likewise NP-complete, see Exercise 2.8.7.

Graph partitioning (cf. Sect. 9.2)

Let $G = (V, E)$ be a graph and c a positive integer. The question of whether G can be partitioned into at most c subgraphs of a given type is NP-complete for many classes of subgraphs: for triangles and, more generally, for subgraphs with a given isomorphism type, for Hamiltonian subgraphs, for forests, for cliques, and for matchings. We refer the reader to [GarJo79, §A1.1] and the references given there.

In particular, determining the clique partition number $\theta(G)$ is an NP-hard problem in general. For perfect graphs, this problem can be solved in polynomial time; see [GroLS93].

Hamiltonian cycle (cf. Sects. 1.4 and 2.8)

Let $G=(V,E)$ be a graph. Does G contain a Hamiltonian cycle?

Karp [Kar72] proved that this problem is NP-complete; it remains NP-complete even if we know a Hamiltonian path of G [PapSt77]; see Theorem 15.7.3. The special cases for bipartite graphs and for planar, 3-connected, 3-regular graphs are still NP-complete; see [Kri75] and [GarJT76]. The analogous problem for directed Hamiltonian cycles in digraphs likewise is NP-complete [Kar72]; see Exercise 2.7.6.

Hamiltonian path (cf. Exercise 2.7.7)

Does the graph $G=(V,E)$ contain a Hamiltonian path? This problem and the analogous directed problem are NP-complete, even if the start and end vertices of the Hamiltonian path are fixed; see [GarJo79].

Independent set (cf. Exercise 2.8.4)

Let $G=(V,E)$ be a graph and $c \leq |V|$ a positive integer. Does G contain an independent set with c elements? Note that the independent sets of G are precisely the cliques of the complementary graph $\overline{G}$. This problem is therefore NP-complete in general, but polynomial for perfect graphs (see **Clique** and **Vertex cover**).

The independent set problem remains NP-complete when restricted to 3-regular planar graphs; see [GarJS76].

Induced subgraph

Let $G=(V,E)$ be a graph and c a positive integer. The problem of whether G contains an induced subgraph on c vertices that belongs to a prescribed class of graphs is often NP-complete: for cliques and independent sets (see **Clique** and **Independent set**), and also for planar subgraphs, for bipartite subgraphs, for subforests, etc. We refer to [GarJo79, § A1.2] and the references given there.

Integer linear programming (cf. Sect. 14.3)

Let A be an $m \times n$ integer matrix, $\mathbf{c} \in \mathbb{Z}^n$ and $\mathbf{b} \in \mathbb{Z}^m$ integer vectors, and d an integer. Does there exist an integer vector $\mathbf{x} \in \mathbb{Z}^n$ satisfying $\mathbf{x} \geq \mathbf{0}$, $A\mathbf{x}^T \leq \mathbf{b}^T$, and $\mathbf{c}\mathbf{x}^T \geq d$? This problem is NP-complete by a result of Karp [Kar72], whereas the corresponding linear program (where $\mathbf{x}$ may have rational entries) can be solved in polynomial time by the work of Khachyan [Kha79].

Longest cycle

Let $N = (G, w)$ be a network with a nonnegative length function w, where G is a graph, and let c be a positive integer. Does N contain a cycle of length at least c?

This problem is NP-complete even when all edges have length 1; see Exercise 2.7.8. Similar results hold for the analogous directed problem.

Longest path (cf. Sects. 2.7 and 3.1)

Let s and t be two vertices in a network $N = (G, w)$ on a graph G, where w is a nonnegative length function, and let c be a positive integer. Does there exist a path from s to t of length at least c?

This problem is NP-complete even when all edges have length 1; see Exercise 2.7.8. Similar results hold for the analogous directed problem.

Matroid intersection (cf. Sect. 5.4)

Let $(E, \mathbf{S}_i)$ $(i = 1, 2, 3)$ be three matroids on the same set E, and let c be a positive integer. Does E have a subset U of cardinality c which is an independent set for all three matroids?

This problem is NP-complete; see Theorem 5.4.13. Note that the corresponding problem for the intersection of two matroids is solvable in polynomial time (even in the weighted case); see [Law75, Law76, Edm79, Cun86], and [Whi87].

Max cut (cf. Chap. 6)

Let $G = (V, E)$ be a graph with a nonnegative capacity function c, and let b be a positive integer. Does there exist a cut (S, T) of G with capacity $c(S, T) \geq b$?

This problem is NP-complete by a result of Karp [Kar72]; this holds even in the special case where G has maximal degree 3 and $c(e) = 1$ for all edges e; see [Yan78]. Thus determining a cut of maximal capacity is an NP-hard problem, whereas the analogous problem for cuts of minimal capacity is easy.

However, the max cut problem is polynomial for planar graphs; see [Had75].

Min cut (cf. Chap. 6)

Let $G = (V, E)$ be a graph with a nonnegative capacity function c, let s and t be two vertices of G, and $b \leq |V|$ and k be two positive numbers. Does there exist a cut (S, T) of G with $s \in S$, $t \in T$, $|S| \leq b$, $|T| \leq b$, and capacity $c(S, T) \leq k$?

This problem is NP-complete, even when $c(e) = 1$ for all edges e; see [GarJS76]. Note that omitting the bounds on $|S|$ and $|T|$ (that is, putting $b = |V|$) yields one of the fundamental easy problems: again, we have a case of an easy problem becoming hard due to additional constraints.

Minimum k-connected subgraph (cf. Chap. 8)

Let $G = (V, E)$ be a graph, and let $k \leq |V|$ and $b \leq |E|$ be two positive integers. Does there exist a subset E^* of E with $|E^*| \leq b$ such that $G^* = (V, E^*)$ is k-connected?

This problem—and also the analogous problem for k-fold line connectivity—is NP-complete for each fixed $k \geq 2$; see [GarJo76]. Thus determining a minimal k-connected subgraph of G is NP-hard. Note that the case $k = 1$ can be solved with complexity $O(|E|)$ using BFS, for example: then we merely have to find a spanning tree of G.

Minimum spanning tree (cf. Chap. 4)

Let $N = (G, w)$ be a network with a nonnegative weight function w on a connected graph G. As we saw in Sect. 4.3, determining a minimal spanning tree T is one of the fundamental easy problems of algorithmic graph theory. As for the problem of determining spanning trees in general, we obtain NP-complete problems by adding side constraints, for example by restricting the diameter of T or asking for many leaves. See Sect. 4.7 and [GarJo79, §A.2.1].

Network flow (cf. Chaps. 6 and 10)

The flow problems we treated in this book are all solvable in polynomial time. Again, adding side constraints will often result in NP-complete problems. We refer the reader to [GarJo79, §A.2.4] and the references given there.

Network reliability (cf. Example 3.1.2)

Let $G = (V, E)$ be a graph, V^* a subset of V, p a mapping from E to the rational numbers in $[0, 1]$ (the *failure probability*), and $q \leq 1$ a positive rational number. Is the probability that any two vertices in V^* are connected by at least one *reliable* path (that is, a path which does not contain an edge which fails) at least q?

This problem is NP-complete by a result of Rosenthal [Ros77]; see also [Val79b] for related questions. Provan [Pro86] showed that it is NP-hard to determine the probability for the existence of a reliable path from s to t in a planar acyclic digraph G, and also in a planar graph G with maximal degree $\Delta(G) = 3$.

Permanent evaluation (cf. Sect. 7.4)

Let A be an $n \times n$ matrix with entries 0 and 1, and let $k \leq n!$ be a positive integer. Does $\mathrm{per} A = k$ hold?

This problem and the corresponding problems about $\mathrm{per} A \leq k$ and $\mathrm{per} A \geq k$ are NP-hard, which is due to Valiant [Val79a]; we note that it is not known whether these problems actually belongs to NP.

Recall that determining the number of perfect matchings in a bipartite graph is equivalent to determining the permanent of an appropriate matrix, so that this problem is likewise NP-hard.

Restricted matching (cf. Sect. 14.7)

Let $G=(V,E)$ be a graph, and consider a decomposition of E into subsets E_i $(i=1,\ldots,k)$. Also, let c and b_i $(i=1,\ldots,k)$ be positive integers. Does there exist a matching K with c edges such that $|K\cap E_i|\leq b_i$ holds for $i=1,\ldots,k$?

This problem is NP-complete, even when all b_i are 1; see [ItaRT78].

Satisfiability (cf. Sect. 2.7)

Let $C_1\ldots C_m$ be a formula involving n Boolean variables in conjunctive normal form. Does there exist an assignment of the values *true* and *false* to the n variables such that the given formula takes the value *true*?

This problem is NP-complete, even when each of the C_i involves precisely three of the n Boolean variables (3-SAT). This celebrated result due to [Coo71] was the starting point of the theory of NP-completeness.

Shortest cycle (cf. Sects. 3.3 and 10.6)

Let $N=(G,w)$ be a network on a graph G, where w is a length function that may take negative values, and c an integer. Does G contain a cycle of length at most c?

This problem is NP-complete; see [GarJo79]. It can be solved in polynomial time for nonnegative length functions; see, for example, [ItaRo78] and [Mon83]. Similar results hold for the analogous directed problem. Note that determining a cycle of minimum cycle mean is easy for arbitrary length functions w; see Sect. 10.6.

Shortest path (cf. Chap. 3 and Sect. 14.6)

Let s and t be two vertices in a network $N=(G,w)$ on a graph G, where w is a length function that may take negative values, and let c be an integer. Does there exist a path from s to t of length at most c?

This problem is NP-complete, and this also holds for the analogous directed problem; see [GarJo79]. As we saw in Sect. 14.6, the problem becomes polynomial if we assume that N does not contain any cycles of negative length. Particularly good algorithms exist for the special case where all edges have nonnegative length; see Chap. 3.

Spanning tree (cf. Chap. 4)

We know that a spanning tree in a connected graph G can be determined with linear complexity using either BFS or DFS; see Sects. 3.3 and 8.2.

However, the problem usually becomes NP-complete if we add extra constraints such as either a lower bound on the number of leaves, or an upper bound on the maximal degree of the tree; see Sect. 4.7 and [GarJo79, § 4.7]. The same conclusion holds if we ask whether the sum of the distances $d(u,v)$ in T (taken over all pairs (u,v) of vertices) can be bounded by c; see [JohLR78].

Steiner network (cf. Sect. 4.6)

Let $N = (G, w)$ be a network on a graph $G = (V, E)$, where $V = R \dot{\cup} S$ and where $w : E \to \mathbb{R}^+$ is a positive weight function, and let c be a positive integer. Does there exist a minimal spanning tree T for some induced subgraph whose vertex set has the form $R \cup S'$ with $S' \subset S$ so that $w(T) \leq c$?

This problem is NP-complete by a result of Karp [Kar72]. The problem becomes polynomial when either $|R|$ or $|S|$ is fixed.

Steiner tree (cf. Sect. 4.6)

For a given set of n points in the Euclidean plane, we want to find a minimal Steiner tree (that is, a tree of minimal length with respect to the Euclidean distance) which contains the given n points. This problem was shown to be NP-hard by Garey, Graham and Johnson [GarGJ77].

Travelling salesman problem (TSP) (cf. Chap. 15)

Let $w : E \to \mathbb{R}^+$ be a positive length function on the complete graph K_n. Given a positive integer b, is there a tour (that is, a Hamiltonian cycle) of length at most b?

Recall that the TSP served as our standard example for an NP-complete problem. It remains NP-complete in the metric case, in the asymmetric case, and for length functions restricted to the values 1 and 2. The related questions of whether a tour is suboptimal or whether an optimal tour contains a given edge are likewise NP-hard.

The associated approximation problem is NP-hard in the general case, but easy in the metric case: there is a polynomial ε-approximative algorithm with $\varepsilon = 1/2$ for the ΔTSP. The existence of such an algorithm for a value $\varepsilon < 1/219$ would already imply $\mathrm{P} = \mathrm{NP}$; see [PapVe06].

See Chap. 15 and the monographs [LawLRS85] and [GusPa02].

Unextendable matching (cf. Sect. 7.2 and Chap. 13)

Let $G = (V, E)$ be an arbitrary graph, and let c be a positive integer. Does G contain an unextendable matching of cardinality at most c?

This problem is NP-complete by a result of Yannakakis and Gavril [YanGa80]. The problem remains NP-complete in the special cases of planar graphs and of bipartite graphs (even when the maximal degree is restricted to 3).

Recall that a matching which cannot be extended does not have to have maximal cardinality in general. As we have seen, it is easy to determine a *maximal* matching (that is, a matching of maximal cardinality) in G. Hence the existence of an unextendable matching of cardinality *at least* c is easy to decide.

Vertex cover (cf. Sect. 2.8)

Let $G = (V, E)$ be a graph, and let c be a positive integer. Does G have a vertex cover of cardinality at most c?

This problem is NP-complete by a result of Karp [Kar72]; see Theorem 2.8.3. It can be solved in polynomial time for perfect graphs (hence, for bipartite graphs); see [GroLS93].

Note that **Vertex cover** is equivalent to **Independent set**: the complement of a vertex cover is an independent set.

Appendix B
Solutions

People of quality know everything without ever having been taught anything.
MOLIÈRE

This appendix contains solutions (or extended hints) to virtually all the exercises. For difficult exercises, we include more details; if an exercise is of a purely computational nature, we usually state only the result.

B.1 Solutions for Chap. 1

1.1.2 As $2n-1$ is odd, $2i$ $(i=1,\dots,2n-1)$ runs through all residue classes modulo $2n-1$. Therefore the sets F_i are pairwise disjoint. Clearly, each F_i is a factor of K_{2n}. As $F_1,\dots,F_{2n-1}$ contain altogether $n(2n-1)$ edges, they must form a factorization.

1.1.3 Note $T_3=K_3$. The graph T_4 is K_6 with one 1-factor removed. The complement of T_5 is shown in Fig. 1.12; cf. Exercise 1.5.11.

A vertex $\{x,y\}$ of T_n is adjacent precisely to the $2(n-2)$ vertices of the form $\{x,z\}$ and $\{y,z\}$, where $z\neq x,y$. Two distinct vertices $\{x,y\}$ and $\{x,z\}$ are adjacent precisely to the $n-3$ vertices $\{x,w\}$ for $w\neq x,y,z$ and to $\{y,z\}$. Finally, the common neighbors of two vertices $\{x,y\}$ and $\{z,w\}$, where x,y,z,w are distinct, are precisely $\{x,z\}$, $\{x,w\}$, $\{y,z\}$, and $\{y,w\}$.

1.1.4 For a given vertex x, there are exactly $a'=n-a-1$ vertices which are not adjacent to x in G. If x and y are vertices adjacent in G, there are precisely $a-c-1$ vertices which are adjacent to x but not to y, and precisely $(n-a-1)-(a-c-1)$ vertices which are adjacent neither to x nor to y. Thus $\overline{G}$ has parameters $a'=n-a-1$ and $d'=n-2a+c$. Similar arguments give $c'=n-2a+d-2$.

To prove the validity of the equation in question, choose some vertex x. Then there are $n-a-1$ vertices z which are not adjacent to x. For each such vertex z, there are precisely d vertices y which are adjacent to x as well as to z. On the other hand, there are a vertices y adjacent to x, and for each such vertex y, there are $a-c-1$ vertices z adjacent to y but not adjacent to x.

1.2.1 Let $W=(v_0,\dots,v_n)$ be a walk with start vertex $a=v_0$ and end vertex $b=v_n$. If W is not a path, it contains repeated vertices. Let $x=v_i$ be the first such vertex, and let v_j be the next occurrence of x on W. Then the subwalk of W from v_i to v_j is a closed walk, and omitting it from W yields a shorter walk W' from a to b. Using induction on the length of W gives the assertion.

Now let $W=(v_0,\dots,v_n)$ be a closed walk of odd length which is not a cycle. Suppose there exists some index $i\neq 0,n$ such that $v_0=v_i=v_n$. Then one of the closed walks $(v_0,\dots,v_i)$ or $(v_i,\dots,v_n)$ has odd length, and the assertion follows by induction. In the general case, there are indices $i,j\neq 0,n$ with $i\neq j$ and $v_i=v_j$; again, the assertion follows by induction.

We obtain a closed walk of even length not containing any cycle if we append to some path P (from u to v, say) the same path P traversed in the opposite direction (that is, from v to u).

1.2.2 Let x and y be any two vertices. Then the connected components of x and y contain at least $(n+1)/2$ vertices each; hence they cannot be disjoint, and therefore they coincide.

1.2.3 Trivially, the condition is necessary. To show that it is also sufficient, choose some vertex s and let V_1 be its connected component. Then $V_2=V\setminus V_1$ has to be empty: otherwise, the hypothesis would provide an edge vw with $v\in V_1$ and $w\in V_2$, and w would after all be in V_1, a contradiction.

1.2.4 If neither G nor $\overline{G}$ are connected, choose some vertex s and denote the connected components of G and $\overline{G}$ containing s by S and T, respectively. As each vertex $v\neq s$ is either adjacent to s in S or in T, we must have $V=S\cup T$. It can be seen by similar arguments that there cannot exist a pair (v,w) of vertices with $v\in S\setminus T$ and $w\in T\setminus S$, a contradiction.

1.2.5 The assertion follows from $\sum_v \deg v = 2n-2$; see the proof of Lemma 1.1.1.

1.2.9 If $G\setminus e$ is connected, the assertion follows using induction on $|E|$. Otherwise, G consists of two connected components V_1 and V_2. Using induction on n, the assertion holds for the induced graphs $G|V_1$ and $G|V_2$. Hence

$$|E| = \big|(E|V_1)\big| + \big|(E|V_2)\big| + 1 \geq \big(|V_1|-1\big) + \big(|V_2|-1\big) + 1 = n-1.$$

1.2.15 See Fig. B.1.

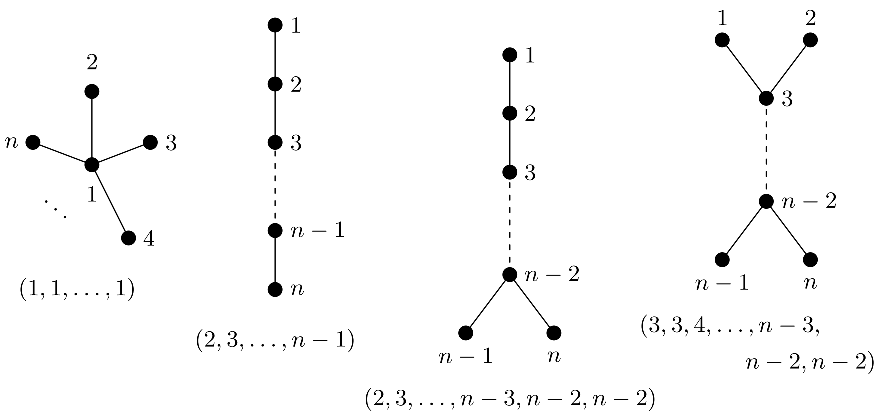


Fig. B.1 Solution to Exercise 1.2.15

1.2.16 The symbol u occurs precisely $\deg u - 1$ times in $\pi_V(G)$; this is similar to the proof of Lemma 1.2.12. In particular, stars are precisely those trees G for which all entries of $\pi_V(G)$ agree, whereas paths are the trees having a Prüfer code with distinct entries.

1.2.17 As a tree on n vertices has $n-1$ edges, condition (1.6) is certainly necessary. By the solution to Exercise 1.2.16, the degree of a vertex u in a tree T equals the number of entries u in the Prüfer code $\pi_V(T)$ plus 1. Now let $d_1, \dots, d_n$ be a sequence of positive integers satisfying (1.6); then

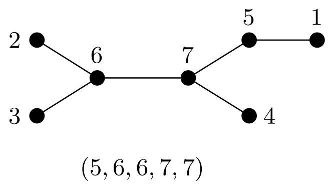
$$(d_1 - 1) + (d_2 - 1) + \cdots + (d_n - 1) = n - 2.$$

Hence there are words of length $n-2$ over the alphabet $\{1, \dots, n\}$ which contain exactly $d_i - 1$ entries i (for $i = 1, \dots, n$), and the corresponding trees under the Prüfer code have the prescribed degree sequence. For the sequence $(1,1,1,1,2,3,3)$, we may use the Prüfer code $(5,6,6,7,7)$ to obtain the tree shown in Fig. B.2.

1.3.3 Denote the vertices of odd degree by x_i and y_i (for $i = 1, \dots, k$). Adding the edges $x_i y_i$ to G yields an Eulerian multigraph H. The desired trails arise by omitting the edges $x_i y_i$ from a Euler tour for H.

1.3.4 Note that an edge uv of G has degree $\deg u + \deg v - 2$ when considered as a vertex of $L(G)$. In particular, $L(K_{m,n})$ is $(m+n-2)$-regular. If x, y, z, w are distinct vertices of $K_{m,n}$, edges of the form xy and zw are always adjacent to precisely two edges in $L(K_{m,n})$. Edges of the form xy and xz are adjacent to $m-2$ or $n-2$ edges, depending on which part of $K_{m,n}$ contains x. Hence $L(K_{m,n})$ is an SRG if and only if $m = n$.

Fig. B.2 Tree with prescribed degree sequence



1.3.5 Note first that $L(G)$ is connected, since G is assumed to be connected. By Exercise 1.3.4, an edge uv of G has degree $\deg u + \deg v - 2$ in $L(G)$; by Theorem 1.3.1, $L(G)$ is Eulerian if and only if this number is always even. As G is connected, this requires that the degrees of all the vertices of G have the same parity. In particular, this condition is met if G is Eulerian: then all vertices of G have even degree. Finally, $L(K_{2n})$ is Eulerian while K_{2n} is not Eulerian, as all vertices have odd degree.

1.4.4 The existence of non-adjacent vertices u and v with $\deg u + \deg v < n$ would imply $m < \frac{1}{2}(n-2)(n-3) + n = \frac{1}{2}(n-1)(n-2) + 2$, since the maximal number of edges arises if the remaining $n-2$ vertices induce a complete graph.

1.4.5 As K_6 is Hamiltonian, G also has to be Hamiltonian by Theorem 1.4.1. Therefore G contains a cycle of length 6. We have to add at least two edges to this cycle to obtain a graph where $\deg u + \deg v \geq 6$ holds for some pair of non-adjacent vertices u and v. On the other hand, it is easy to check that the closure of such a graph G is indeed K_6. Hence eight edges are needed.

1.4.6 Let $(e_1, \ldots, e_m)$ be an Euler tour of G; then the sequence $(e_1, \ldots, e_m, e_1)$ is a Hamiltonian cycle in $L(G)$. The converse is false; for example, K_4 is not Eulerian even though $L(K_4) = T_4$ is Hamiltonian.

1.4.8 We color the squares of a chess board alternately black and white, as usual. Note that a knight always moves from a black square to a white one, and from a white square to a black one; in the corresponding graph, all edges connect a black and a white vertex. (This means that G is bipartite; see Sect. 3.3.) Obviously, G can only contain a Hamiltonian cycle if the numbers of white and black vertices are equal, which is impossible if n and m are both odd. This accounts for case (a).

In case (b), the preceding necessary condition is satisfied. However, the cases $m = 1$ and $m = 2$ are trivially impossible. In order to show that a knight's cycle is also impossible for $m = 4$, we consider a second coloring of the chess board: the squares of the first and fourth rows are green, whereas the squares in rows two and three are red. Then a knight can move from a green square only to a red square; from a red square, green squares as well as red squares are accessible. Now assume the existence of a knight's cycle. Then the knight has to reach and to leave each of the green squares precisely once. As the green squares are only accessible from the red ones, the $2n$

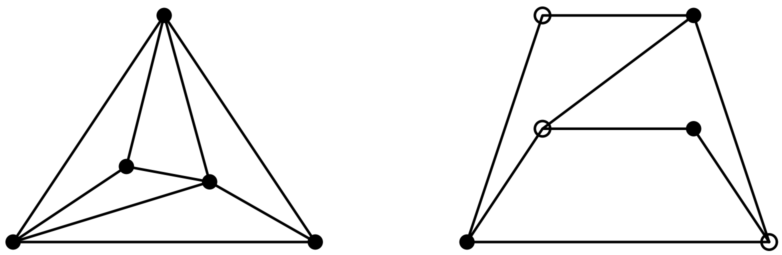


Fig. B.3 Plane realizations of $K_5 \setminus e$ and $K_{3,3} \setminus e$

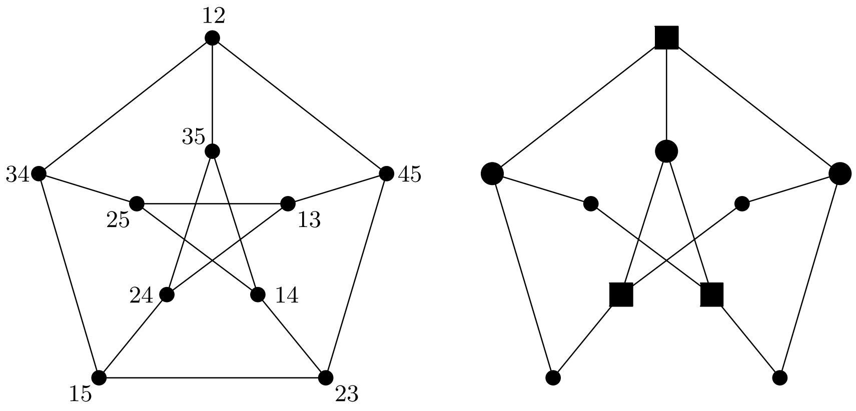


Fig. B.4 The Petersen graph

moves from a red square always have to be moves to a green square, so that red and green squares alternate in the knight's cycle. But white and black squares also occur alternately; as the two colorings of the board are obviously distinct, this is impossible.

1.5.6 See Fig. B.3.

1.5.7 Any subdivision of a graph increases the number of vertices by the same value as the number of edges.

1.5.10 The Petersen graph G has girth $g = 5$. As G contains more than $40/3$ edges, G cannot be planar by Theorem 1.5.3. Figure B.4 shows G and a subgraph homeomorphic to $K_{3,3}$, where the vertices of $K_{3,3}$ are indicated by fat circles and squares, whereas the vertices obtained by subdivision are drawn as small circles. Thus Result 1.5.8 applies. Contracting each outer vertex of G with its adjacent inner vertex shows that G can be contracted to K_5, so that Result 1.5.9 likewise applies.

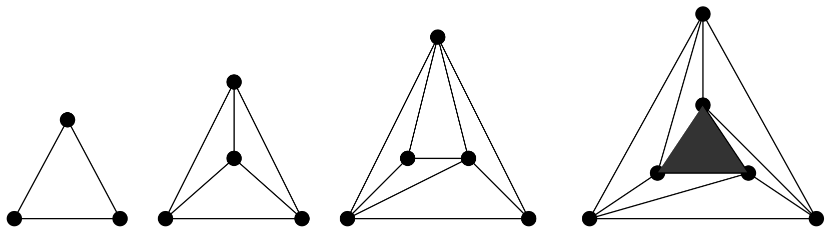


Fig. B.5 Maximal planar graphs

1.5.11 We write the 2-subsets $\{x, y\}$ of $\{1, \ldots, 5\}$ simply as xy. Then the vertices of T_5 are the xy, and xy and zw are adjacent in $\overline{T_5}$ if and only if x, y, z, w are four distinct elements. Now it is easy to give the desired isomorphism using the labelling of the vertices shown in Fig. B.4.

1.5.12 Each permutation $\alpha \in S_5$ induces an automorphism of T_5—and hence, by Exercise 1.5.11, an automorphism of the Petersen graph—by mapping each 2-subset xy to $x^\alpha y^\alpha$. Actually, S_5 already yields *all* automorphisms of the Petersen graph; however, proving this requires a little more effort. (Hint: Try to show that there are at most 120 automorphisms of the Petersen graph.)

1.5.13 For $n = 1, \ldots, 4$, K_n is already planar. For $n \geq 5$, K_n cannot be planar, since a planar graph on n vertices has at most $3n - 6$ edges by Corollary 1.5.4. Thus we have to remove at least $\frac{1}{2}n(n-1) - (3n-6) = \frac{1}{2}(n-2)(n-5) + 1$ edges. Using induction, it can be shown that there exists a planar graph with $3n-6$ edges for each n; in fact, this graph can even be assumed to have a triangle as its outer border. The induction basis $(n = 3, 4, 5)$ and the recursive construction of placing a planar graph with n vertices and $3n - 6$ edges inside a triangle are sketched in Fig. B.5.

1.5.14 As $n - n_d$ vertices have degree at least $d + 1$, Corollary 1.5.4 implies $(n - n_d)(d + 1) \leq \sum_v \deg v = 2m \leq 6n - 12$ and hence the assertion. In particular, $n_5 \geq 2$ and $n_6 \geq n/7$; thus more than 14% of the vertices of a planar graph have degree at most 6.

The given formula can be strengthened as follows: any planar graph can be embedded in a planar graph (by adding appropriate edges) whose vertices have degree at least 3. For these planar graphs, the left hand side of the inequality can be increased by $3n_d$, and we obtain

$$n_d \geq \frac{n(d-5) + 12}{d-2};$$

in particular, $n_5 \geq 4$ and $n_6 \geq n/4$.

1.6.1 Let G be pseudosymmetric. Choose an arbitrary edge $e_1 = v_0v_1$, then some edge $e_2 = v_1v_2$ and so on, always selecting edges which have not occurred before. Whenever we reach a vertex $v_i \neq v_0$ via an edge e_i, there is an unused edge e_{i+1} available for leaving v_i, since G is pseudosymmetric. Hence our construction yields a directed cycle C. Removing C from G results in a pseudosymmetric graph H, and the assertion follows by induction.

1.6.4 Obviously, an edge contained in a cycle cannot be a bridge. Conversely, let $e = uv$ be an edge which is not a bridge. Then the connected component containing u and v is still connected after removing e, so that there exists a path P from u to v not containing e. Appending e to P yields the desired cycle.

1.6.5 G is Eulerian by Theorem 1.3.1. Let $(v_0, \dots, v_m = v_0)$ be the sequence of vertices in an Euler tour $(e_1, \dots, e_m)$ of G. Orienting each edge e_i from v_{i-1} towards v_i, we obtain an orientation of G, and $(e_1, \dots, e_m)$ is a directed Euler tour for this orientation. Hence this orientation is pseudosymmetric and strongly connected.

1.6.6 First let $W = (v_0, \dots, v_n)$ be a closed directed walk which is not a cycle. Suppose there exists some index $i \neq 0, n$ such that $v_0 = v_i = v_n$. Then each of the closed walks $(v_0, \dots, v_i)$ and $(v_i, \dots, v_n)$ has shorter length, and the assertion follows by induction. In the general case, there are indices $i, j \neq 0, n$ with $i \neq j$ and $v_i = v_j$; again, the assertion follows by induction.

Now let $W = (v_0, \dots, v_n)$ be a directed walk with start vertex $a = v_0$ and end vertex $b = v_n$. If W is not a path, it contains repeated vertices. Let $x = v_i$ be the first such vertex, and let v_j be the next occurrence of x on W. Then the subwalk W' of W from v_i to v_j is a directed closed walk and hence contains a directed cycle C. Omitting C from W yields a shorter directed walk W' from a to b, and the assertion follows by induction on the length of W.

B.2 Solutions for Chap. 2

2.1.3 Let G have the $n!$ permutations of $\{1, \dots, n\}$ as vertices, and let two permutations be adjacent if and only if they differ by only a transposition. The case $n = 3$ is shown in Fig. B.6, where we denote the permutation (x, y, z) of $\{1, 2, 3\}$ by xyz; here the sequence $(123, 132, 312, 321, 231, 213)$ provides a solution.

2.1.4 (a) First assume that $G \setminus v_0$ is acyclic, and let C be a maximal path starting at v_0. Then C is a cycle. If C were not an Euler tour, we could find a cycle C' in $G \setminus C$ as in Example 2.1.2. By hypothesis, C' would have to contain v_0, so that C would not be maximal. Hence G is arbitrarily traceable

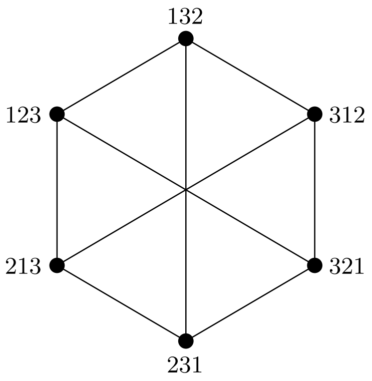


Fig. B.6 Transposition graph for S_3

from v_0. Conversely, suppose that G is arbitrarily traceable from v_0. If there exists a cycle C in $G \setminus v_0$, we can choose an Euler tour K of the connected component of v_0 in $G \setminus C$, so that K is a maximal trail starting in v_0, a contradiction.

(b) Let w be a vertex of maximal degree $2k$ in G, and let C be an Euler tour for G. Then C can be divided into k cycles $C_1, \dots, C_k$, each of which contains w only once. As $G \setminus v_0$ is acyclic by (a), v_0 has to occur in each of these cycles, and hence also $\deg v_0 = 2k$.

(c) Suppose G is arbitrarily traceable from u, v, and w. By part (a), each of these three vertices has to occur in all cycles of G. Suppose that G contains at least two cycles (which intersect in u, v, and w); then it is easy to construct a third cycle which contains only two of these vertices, a contradiction. Hence G contains at most one cycle and thus is itself a cycle.

(d) By part (b), both vertices have to be vertices of maximal degree, say $2k$. Choose two vertices u and v and connect them by $2k$ parallel edges. Then all subdivisions of this multigraph are arbitrarily traceable from both u and v.

2.2.5 Use induction on h; the case $h = 1$ is clear. With $B = A^h$, the (i,k)-entry of A^{h+1} is the sum of all terms $b_{ij}a_{jk}$ over $j = 1, \dots, n$. By the induction hypothesis, b_{ij} is the number of walks of length h from i to j. Moreover, $a_{jk} = 1$ if (j,k) is an edge, and $a_{jk} = 0$ otherwise. Observe that a walk of length $h+1$ from i to k consists of a walk of length h (from i to some vertex j) followed by a last edge (j,k). This proves the assertion for graphs; the same argument works in the directed case, if we restrict attention to directed walks.

2.2.6 By Exercise 2.2.5, the (i,j)-entry of the matrix A^2 is the number of walks of length 2 from i to j; note that this reduces to the degree of i whenever $i = j$. Denote the matrix with all entries equal to 1 by J. Using the defining properties of a strongly regular graph yields the desired quadratic equation: $A^2 = aI + cA + d(J - I - A)$.

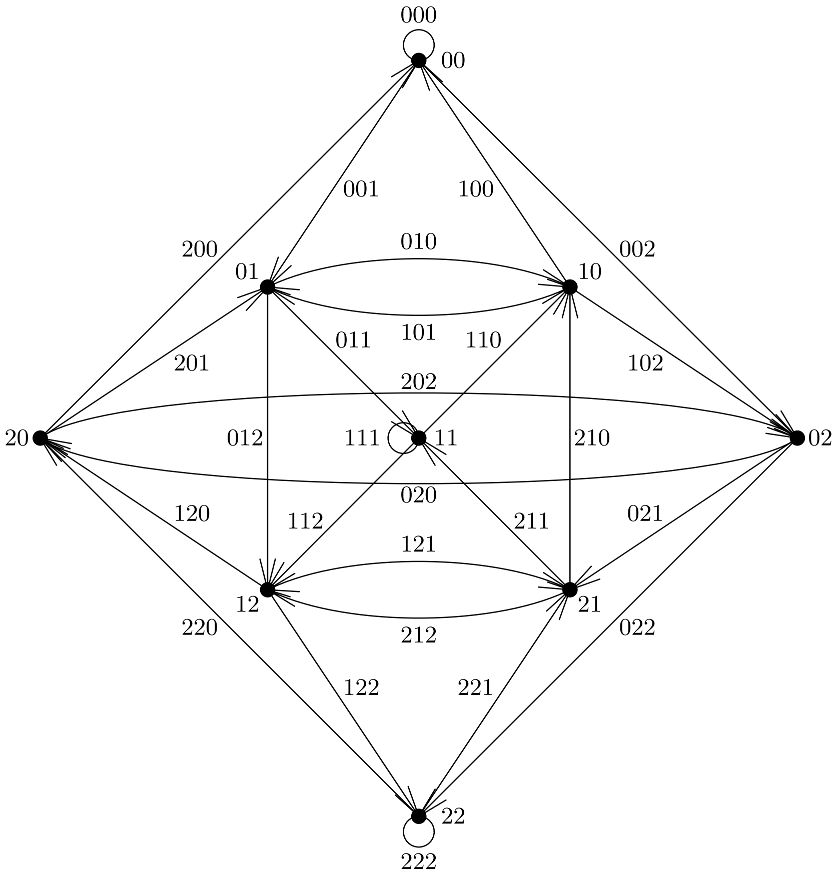


Fig. B.7 The digraph $G_{3,3}$

2.3.2 Note that a word $w = a_i \dots a_{i+n-1}$ is the immediate predecessor of a word $v = a_{i+1} \dots a_{i+n}$ in a de Bruijn sequence if and only if the edge v has the end vertex of w as start vertex; thus the de Bruijn sequences correspond to Euler tours in $G_{s,n}$. It remains to show that $G_{s,n}$ satisfies the two conditions of Theorem 1.6.1. First, $G_{s,n}$ is strongly connected: two vertices $b_1 \dots b_{n-1}$ and $c_1 \dots c_{n-1}$ are connected by the directed path

$$(b_1 \dots b_{n-1}c_1, b_2 \dots b_{n-1}c_1c_2, \dots, b_{n-1}c_1 \dots c_{n-1}).$$

$G_{s,n}$ is also pseudosymmetric: $d_{\text{in}}(x) = d_{\text{out}}(x) = s$ for each vertex x.

2.3.3 The digraph $G_{3,3}$ is shown in Fig. B.7.

Using $s = 00$, the procedure TRACE$(s, \text{new}; K)$ yields the cycle

$$K = (000, 001, 010, 100, 002, 020, 200).$$

Then all edges with start vertex 00 have been used, and $L = (00, 01, 10, 02, 20)$. In step (5) of EULER, the vertex $u = 20$ is removed from L; then step (7) calls TRACE$(u, \text{new}; C)$, which yields the cycle

$$C = (201, 011, 110, 101, 012, 120, 202, 021, 210, 102, 022, 220).$$

This cycle is inserted in front of the edge 200 into K according to step (8) of EULER; we then have $K = (000, 001, \ldots, 020, 201, 011, \ldots, 220, 200)$ and $L = (00, 01, 10, 02, 11, 12, 21, 22)$. Next, $u = 22$ is removed from L in step (5), and the cycle

$$C = (221, 211, 111, 112, 121, 212, 122, 222)$$

is constructed and inserted into K in front of the edge 220. After this, EULER discovers that all edges have been used (by investigating all vertices in L). The de Bruijn sequence corresponding to this Euler tour is

$$0\ 0\ 0\ 1\ 0\ 0\ 2\ 0\ 1\ 1\ 0\ 1\ 2\ 0\ 2\ 1\ 0\ 2\ 2\ 1\ 1\ 1\ 2\ 1\ 2\ 2\ 2.$$

2.6.8 Let $G = (V, E)$ be the empty digraph with n vertices: $E = \emptyset$. Then any algorithm using the adjacency matrix has to check at least one of the two entries a_{ij} and a_{ji} for each pair (i, j) with $i \neq j$: otherwise, we could add the edges (i, j) and (j, i) to G and the algorithm would not realize that the digraph is no longer acyclic. Thus the algorithm has to check at least $n(n-1)/2 = \Omega(n^2)$ entries.

2.6.9 The algorithm first calculates $\text{ind}(1) = 2$, $\text{ind}(2) = 0$, $\text{ind}(3) = 3$, $\text{ind}(4) = 1$, $\text{ind}(5) = 2$, $\text{ind}(6) = 4$, $\text{ind}(7) = 3$, and $L = (2)$. Then 2 is removed from L and the function *ind* is updated as follows: $\text{ind}(1) = 1$, $\text{ind}(3) = 2$, $\text{ind}(4) = 0$, $\text{ind}(7) = 2$. Now 4 is appended to L. During the next iteration, 4 is removed from L, and the following updates are performed: $\text{ind}(1) = 0$, $\text{ind}(3) = 1$, $\text{ind}(5) = 1$, $\text{ind}(7) = 1$. Then 1 is appended to L and immediately removed again during the next iteration. Continuing in this way yields the topological sorting $(2, 4, 1, 3, 5, 7, 6)$ for G; see Fig. B.8, where indeed all edges are oriented from left to right.

2.7.6 DHC contains HC as a special case; this follows by considering the complete orientation of a given graph.

2.7.7 We transform HC to HP. Let $G = (V, E)$ be a connected graph. We choose a fixed vertex v_0. Then we adjoin three new vertices u, u', and w to G, and add the following edges: uu', wv_0, and an edge uv for each vertex v adjacent to v_0; see Fig. B.9. The resulting graph G' has a Hamiltonian path if and only if G admits a Hamiltonian cycle; this follows by noting that every Hamiltonian path of G' has to start with the edge uu' and to end with the edge v_0w.

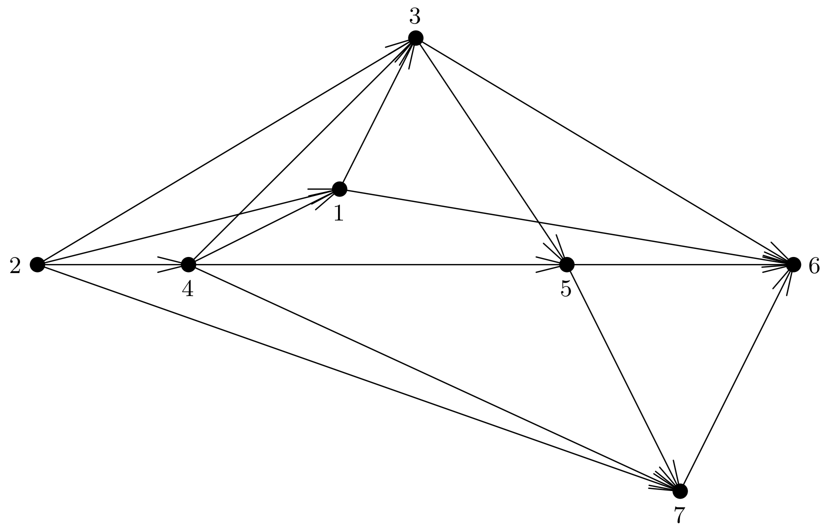


Fig. B.8 Solution to Exercise 2.6.9

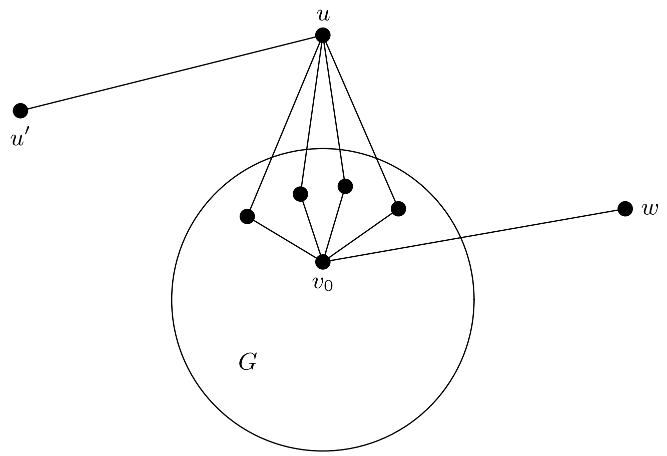


Fig. B.9 Construction for Exercise 2.7.7

2.7.8 Note that HP is a special case of **Longest path**: given a graph G with n vertices, we apply **Longest path** with $k = n$.

Now assume that we also have to specify the end vertices of the path. If we had a polynomial algorithm for this modified problem, we could just invoke the algorithm for all pairs of vertices to get a polynomial algorithm for the unrestricted problem.

The corresponding result holds for longest cycles: the question "Does a given graph G admit a cycle consisting of at least k edges?" contains HC.

2.8.4 Independent sets are precisely the complements of vertex covers. As VC is NP-complete, it follows immediately that **Independent set** is NP-complete as well.

The cliques in a graph G are precisely the independent sets of the complementary graph $\overline{G}$. Therefore, **Clique** is likewise NP-complete.

2.8.7 One proceeds exactly as in the proof of Theorem 2.8.6, with the following modification: in H, we in addition also introduce all edges of the form uv, where u and v are non-adjacent vertices of G. In other words, we make sure that the induced subgraph $H|V$ is a complete graph. As before, any vertex cover W of G is also a dominating set for H, and as W is a subset of V, the subgraph $H|W$ is trivially connected. Conversely, as before, we can construct from any dominating set D for H a dominating set D' which consists of vertices in V only. Hence D' is also a vertex cover for G of size at most $|D|$, and the subgraph $H|W'$ is connected.

B.3 Solutions for Chap. 3

3.1.3 Let all pairs (j,k) with $j=1,\dots,n$ and $k=0,\dots,b$ be vertices of G. We choose all pairs $((j-1,k),(j,k))$ as edges of length 0 (for $j=2,\dots,n$; $k=0,\dots,b$), and all pairs $((j-1,k_j-a_j),(j,k_j))$ as edges with length c_j (for $j=2,\dots,n$ and $k_j=a_j,\dots,b$). We also adjoin a start vertex s to G, and add the edges $(s,(1,0))$ with length 0 and $(s,(1,a_1))$ with length c_1. Then the paths from s to (j,k) correspond to those subsets of $\{1,\dots,j\}$ whose total weight is k (and whose total value is the length of the associated path). Finally, we add an end vertex t and edges $((n,k),t)$ of length 0 (for $k=0,\dots,b$). Then paths from s to t correspond to subsets whose weight is at most b, and the length of a longest path from s to t is the value of an optimal solution for the given knapsack problem.

3.2.3 The distances in the metric space have to be integral; moreover, $d(x,y)\geq 2$ always has to imply that a point z with $d(x,y)=d(x,z)+d(z,y)$ exists. It is clear that this condition is necessary. In order to show that it is also sufficient, choose all pairs $\{x,y\}$ with $d(x,y)=1$ as edges.

3.3.4 The connected components can be determined as follows, where p denotes the number of connected components and where $c(v)$ is the component of G containing $v\in V$.

Procedure COMP$(G;p,c)$

```
(1) $i \leftarrow 1$;
(2) while $V \neq \emptyset$ do
(3)         choose a vertex $s \in V$;
```

```
(4)         BFS($G, s; d$);
(5)         $L \leftarrow \{v \in V : d(v)$ is defined$\}$; $V \leftarrow V \setminus L$;
(6)         for $v \in L$ do $c(v) \leftarrow i$ od
(7)         $i \leftarrow i + 1$
(8)     od
```

3.3.8 Let G be a graph containing cycles. Obviously, G contains a cycle which is accessible from some vertex s if and only if a BFS with start vertex s reaches a vertex w (when searching from the vertex v, say) such that $d(w)$ is already defined. Considering the point where such a vertex w occurs for the first time, we obtain a bound g for the length of a shortest cycle accessible from s:[1]

$$g \leq \begin{cases} 2d(v) + 2 & \text{if } d(w) = d(v) + 1; \\ 2d(v) + 1 & \text{if } d(w) = d(v). \end{cases}$$

If $d(w) = d(v)$, the bound cannot be improved by continuing the BFS. However, if $d(w) = d(v) + 1$, the BFS should be continued until all vertices which are in the same layer as v have been examined, because l might still be decreased by one; after this, the BFS may be terminated.

If we execute this procedure for all possible start vertices, the final value of g clearly equals the girth of G. If we also store a vertex s for which the BFS did yield the best value for g, it is easy to actually determine a cycle C of shortest length using a final modified BFS with start vertex s: we always store the vertex v from which w is reached when it is labelled with $d(w)$; that is, we add the instruction $p(w) \leftarrow v$ in step (7) of BFS. The final BFS can be terminated as soon as an edge vw which closes a cycle C occurs. Then we use the predecessor function p to construct the paths (in the BFS-tree T_s) from v and w to the root, and define C as the union of these two paths and the edge vw. We leave it to the reader to write down such a procedure explicitly.[2] As BFS has complexity $O(|E|)$, we achieve a complexity of $O(|V||E|)$ by this approach.

3.4.5 As the distances $d(s, v)$ are known by assumption, one may determine with complexity $O(|E|)$ the set E'' of *all* edges of G satisfying condition (3.2). It follows from the proof of Theorem 3.4.4 that E'' contains an SP-tree; more precisely, any spanning arborescence with root s of $G'' = (V, E'')$ is an SP-tree. Hence a BFS on G'' with start vertex s will determine the desired SP-tree. In view of Theorem 3.3.2, this proves the assertion.

[1]Note that this is indeed just a bound; the precise length can be determined by backtracking the paths from v and w to s in the BFS-tree T_s up to the first vertex they have in common. Obviously, this vertex does not have to be s.

[2]If we want to check first whether G actually contains cycles, we may use the procedure COMP of Exercise 3.3.4 to determine the connected components, and then check the numbers of edges of the components using Theorem 1.2.8.

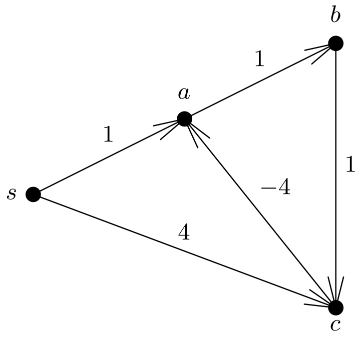


Fig. B.10 A network with a negative cycle

3.4.6 First let T be an SP-tree and uv an edge of G. By definition, the path from s to v in T is a shortest path from s to v in G. On the other hand, appending the edge uv to the path from s to u in T also yields a path from s to v in G. Therefore

$$d_T(s,v) = d(s,v) \le d_T(s,u) + w(uv),$$

which is the desired inequality. Conversely, suppose that

$$(*) \quad d_T(s,v) \le d_T(s,u) + w(uv)$$

holds for all edges uv of G. If P is a shortest path from s to v in G (for $v \neq s$) and $e = uv$ is the last edge of P, then $P' = P \setminus e$ is a shortest path from s to u in G, by Lemma 3.4.1. Using induction on the number of edges of P, we may assume $d(s,u) = w(P') = d_T(s,u)$. Then $(*)$ implies

$$d(s,v) = d(s,u) + w(uv) = d_T(s,u) + w(uv) \ge d_T(s,v),$$

so that $d_T(s,v) = d(s,v)$. Thus T is indeed a shortest path tree.

3.4.7 Consider the network (G,w) displayed in Fig. B.10. Then

$$P = s - a - b - c$$

is the unique shortest path from s to c (with length 3), and $P' = s - c - a$ is the unique shortest path from s to a (with length 0). Hence any SP-tree would have to contain the union of these two paths; but this union is already all of G and contains, for instance, the directed cycle $C = a - b - c - a$, a contradiction.

Now change the value $w(ca)$ from -4 to -3, so that C is still a directed cycle of negative length. But now $d(s,a) = 1$, and both P' and $s - a$ are shortest paths from s to a. Thus the path P is an SP-tree for the modified network.

3.5.2 Let us again consider the network (G,w) used for the solution of Exercise 3.4.7; see Fig. B.10. This time, we change the value $w(ca)$ from -4

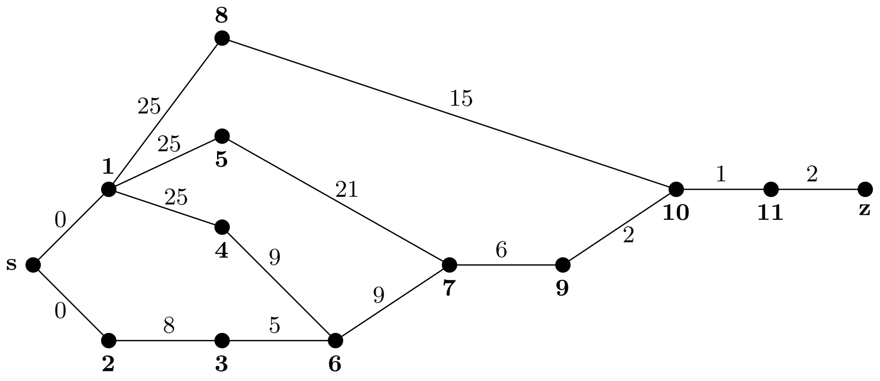


Fig. B.11 Digraph for the project *New production facility*

to -2, so that $C = a - b - c - a$ becomes a directed cycle of length 0. Then $d(s,s) = 0$, $d(s,a) = 1$, $d(s,b) = 2$ and $d(s,c) = 3$ in the modified network, giving one solution of Bellman's equations (B). However, it is easily checked that $u_s = u_a = 0$, $u_b = 1$ and $u_c = 2$ also gives a solution of (B).

It is easy to generalize this example: Let (G, w) be any network containing an induced cycle C of length 0, and assume that no edges are leading from C to another vertex of G. By subtracting a suitable constant from the distances of all vertices on C, one may obtain a second solution of system (B).

3.5.5 Let u_i denote the length of a longest path from 1 to i. Then the following analogue of the Bellman equations has to be satisfied:

$$(\mathrm{B}') \quad u_1 = 0 \quad \text{and} \quad u_i = \max\{u_k + w_{ki} : i \neq k\} \quad (i = 2, \dots, n),$$

where we put $w_{ki} = -\infty$ if (k,i) is not an edge of G. Then the results of Sect. 3.5 carry over to this case: replace w by $-w$ and apply the original theorems to $(G, -w)$. If we do not want to require G to be acyclic, it suffices to assume that G contains cycles of negative length only.

The digraph corresponding to the knapsack problem of Exercise 3.1.3 is acyclic, so that it is possible to determine a longest path from s to t—that is, a solution of the knapsack problem—with complexity $O(|E|)$. However, this does not yield an efficient algorithm, because the number of edges of G has order of magnitude $O(nb)$, so that it depends not only on n but also on b. Restricting the values of b yields a polynomial algorithm, whereas the general knapsack problem is NP-hard; see [Kar72] and [GarJo79].

3.6.2 We obtain the network shown in Fig. B.11 and the values $t_s = 0$, $t_1 = 0$, $t_2 = 0$, $t_3 = 8$, $t_4 = 25$, $t_5 = 25$, $t_8 = 25$, $t_6 = 34$, $t_7 = 46$, $t_9 = 52$, $t_{10} = 54$, $t_{11} = 55$, $t_z = 57$ and $T_z = 57$, $m_z = 0$; $T_{11} = 55$, $m_{11} = 0$; $T_{10} = 54$, $m_{10} = 0$; $T_9 = 52$, $m_9 = 0$; $T_7 = 46$, $m_7 = 0$; $T_6 = 37$, $m_6 = 3$; $T_8 = 39$, $m_8 = 14$; $T_5 = 25$,

$m_5 = 0$; $T_4 = 28$, $m_4 = 3$; $T_3 = 32$, $m_3 = 24$; $T_2 = 24$, $m_2 = 24$; $T_1 = 0$, $m_1 = 0$; $T_s = 0$, $m_s = 0$. The critical path is $(s, 1, 5, 7, 9, 10, 11, z)$.

3.6.3 Consider the network on G where all edges have length 1. As G is acyclic, we may use TOPSORT to determine a topological sorting for G. Then the length of a longest path from s to v can be determined as in Sect. 3.6 or as explained in the solution to Exercise 3.5.5, by recursively solving the equations (B′) or (CPM), respectively. The entire method has complexity $O(|E|)$.

3.6.4 For the time being, we denote the rank function on G by r'. Thus we have to show that, at the end of RANK, $r(v) = r'(v)$ holds for all v. This can be done using induction on the order in which r is defined. Note that $p(w)$ is the predecessor of w on a longest path from s to w; this function can also be used to find such a path: in reverse order, we get the path $(w, p(w), p(p(w)), \dots, s)$. The values $d(v) = d_{\text{in}}(v)$ needed in step (3) can be determined from the adjacency lists (as in TOPSORT). Ordering the vertices of G by increasing rank yields a topological sorting of G; the order of vertices of the same rank is arbitrary. As each edge is examined exactly twice during RANK (once when d is determined in (3), and once in step (7)), this algorithm has complexity $O(|E|)$.

3.7.4 We introduce a variable $p(v)$ which will yield the predecessor of v on a shortest path from s to v (at the end of the algorithm): $p(v)$ is initialized to be 0, and step (6) is changed as follows:

```
(6′)  for $v \in T \cap A_u$ do if $d(u) + w(uv) < d(v)$
                    then $d(v) \leftarrow d(u) + w(uv)$; $p(v) \leftarrow u$ fi od
```

At the end of the algorithm, all edges of the form $p(v)v$ constitute an SP-tree.

3.7.6 One obtains in turn $d(1) = 0$, $d(5) = 1$, $d(3) = 2$, $d(4) = 6$, $d(2) = 9$, $d(8) = 13$, $d(6) = d(7) = 14$.

3.7.9 We may assume the given network to be connected; then planarity implies $|E| = \Theta(|V|)$; see Example 2.5.1. Thus the modified algorithm of Dijkstra has complexity $O(|V| \log |V|)$.

3.7.10 Let us denote the values defined in (1) and (2) by $d_0(v)$, and the values defined during the k-th iteration of the **repeat**-loop by $d_k(v)$. Using induction, one shows that $d_k(v)$ is the length of a shortest path from s to v which has at most k edges. As (G, w) does not contain any cycles of negative length, a shortest path from s to v consists of at most $|V| - 1$ edges. Thus the condition in (7) holds for $k = |V|$ at the latest. As one iteration of the **repeat**-loop requires $O(|E|)$ steps (using backward adjacency lists), we obtain a complexity of $O(|V||E|)$.

3.8.1 Determine the least common multiple T of all time cycles and replace each line L with time cycle $T_L = T/m_L$, where $m_L \neq 1$, by m_L lines with time cycle T and times of departure s_L, $s_L + T_L$, $s_L + 2T_L, \dots$

3.9.3 Proceed as in the solution to Exercise 3.7.4.

3.9.5 The final matrix is

$$D_7 = \begin{pmatrix} 0 & 4 & 5 & 7 & 12 & 10 & 12 \\ \infty & 0 & 6 & 3 & 8 & 6 & 8 \\ \infty & \infty & 0 & 4 & 9 & 7 & 9 \\ \infty & \infty & 3 & 0 & 5 & 3 & 3 \\ \infty & \infty & 7 & 4 & 0 & 3 & 2 \\ \infty & \infty & 9 & 6 & 2 & 0 & 2 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}.$$

3.9.6 Replace the length function w in the procedure FLOYD by the *adjacency function* of G: put $d(i,j) = 1$ in (3) if ij is an edge, and $d(i,j) = 0$ otherwise. Then change step (9) to

(9′) $d(i,j) \leftarrow \max(d(i,j), \min(d(i,k), d(k,j)))$;

alternatively, *max* could be interpreted as the Boolean operation *or*, and *min* as *and*.

3.9.7 Let G be an acyclic digraph, and consider the network on G having all lengths equal to 1. As in the solution to Exercise 3.9.6, we replace the length function w in the procedure FLOYD by the adjacency function of G; moreover, we calculate $\max(d(i,j), d(i,k) + d(k,j))$ in step (9) of that procedure instead of the minimum given there. As G is acyclic, the revised procedure will compute longest paths between all pairs of vertices: at the end of the algorithm, $d(i,j) = 0$ if and only if j is not accessible from i; otherwise, $d(i,j)$ is the maximal length of a path from i to j. (This can be shown by analogy to the proof of Theorem 3.9.2.) Then G_{red} consists of all the edges ij with $d(i,j) = 1$.

3.10.3 Define the values of the variables $d_k(v)$ as in the solution to Exercise 3.7.10. Then G contains a directed cycle of negative length which is accessible from s if and only if $d_{n-1} \neq d_n$. (The reader should prove this claim in detail.) Since s is a root of G, the algorithm of Bellman-Ford can be used to find cycles of negative length by replacing the **repeat-until**-loop used in BELLFORD with a **for-do**-loop. If we also introduce a predecessor function $p(v)$, we can find either a directed cycle of negative length or an SP-tree with root s. We give such a procedure below, using backward adjacency lists A'_v.

Procedure SPTREE($G, w, s; d, p, \mathrm{neg}, T$)

```
(1)  $d(s) \leftarrow 0$;
(2)  $T \leftarrow \emptyset$;
(3)  for $v \in V \setminus \{s\}$ do $d(v) \leftarrow \infty$ od
(4)  for $i = 1$ to $n$ do
(5)      for $v \in V$ do $d'(v) \leftarrow d(v)$ od
(6)      for $v \in V$ do
(7)          for $u \in A'_v$ do
(8)              if $d'(v) > d'(u) + w_{uv}$
(9)              then $d(v) \leftarrow d'(u) + w_{uv}$; $p(v) \leftarrow u$
(10)             fi
(11)         od
(12)     od
(13) od
(14) if $d(v) = d'(v)$ for all $v \in V$
(15) then neg $\leftarrow$ false;
(16)     for $v \in V \setminus \{s\}$ do $T \leftarrow T \cup \{p(v)v\}$ od
(17) else neg $\leftarrow$ true
(18) fi
```

3.10.4 Replace the initial values $d(i,i) = 0$ in step (3) of procedure FLOYD by $d(i,i) = \infty$. Then, at the end of the algorithm, $d(i,i)$ equals the shortest length of a directed cycle through i.

3.11.2 Note that $a = a \oplus o$ shows that $\succeq$ is reflexive. Also $a = b \oplus b'$ and $b = c \oplus c'$ imply $a = c \oplus (b' + c')$, so that $\succeq$ is transitive as well. Suppose that $\oplus$ is idempotent. Then $a = b \oplus c$ and $b = a \oplus d$ imply

$$a = b \oplus c = b \oplus (b \oplus c) = b \oplus a = a \oplus b = a \oplus (a \oplus d) = a \oplus d = b;$$

it follows that $\succeq$ is antisymmetric.

3.11.3 Let E be the matrix with diagonal entries 0 and all other entries ∞. Then $D = D * W \oplus E$.

3.11.5 We have $(A')^k = \sum_{i=0}^{k} \binom{k}{i} A^i = \sum_{i=0}^{k} A^i = A^{(k)}$. Thus $A^{(n)}$ can be calculated for $n = 2^a$ using a matrix multiplications:

$$A^{(1)} = A' = A \oplus E, \qquad A^{(2)} = \left(A'\right)^2, \qquad A^{(4)} = \left(A^{(2)}\right)^2, \quad \text{etc.}$$

If we assume that the operations $\oplus$ and $*$ in R take one step each, we obtain a complexity of $O(n^3 \log n)$ for this method of calculating $A^{(n)}$. For the special case $(\overline{\mathbb{R}}, \oplus, *)$, we get—as explained in Lemma 3.11.4—an alternative to the algorithm of Floyd-Warshall, as $D = W^{(n-1)}$. However, the complexity of this technique is inferior to the one achieved in Theorem 3.9.2.

3.11.6 It is routine to verify that the matrices form a path algebra. For any solution Y of (3.8), we have

$$Y = W * (W * Y \oplus B) \oplus B = W^2 * Y \oplus W^{(1)} * B;$$

hence, by induction,

$$Y = W^{k+1} * Y \oplus W^{(k)} * B \quad \text{for all } k.$$

In particular, for $k = p$,

$$Y = W^{p+1} * Y \oplus W^* * B; \quad \text{that is, } Y \succeq W^* * B.$$

If the addition $\oplus$ on R is idempotent, then addition of matrices is likewise idempotent; in this case, the corresponding preordering on the set of matrices is even a partial ordering by Exercise 3.11.2. Then the minimal solution $W^* * B$ of $(*)$ is unique.

3.11.10 Choose $R = \{a : 0 \le a \le 1\}$, $\oplus = \max$, and $* = \cdot$.

3.11.11 Note that A is stable if and only if $A^r = 0$ for some $r \in \mathbb{N}$, since $A^{(r-1)} = A^{(r)} = A^{(r-1)} + A^r$ holds if and only if $A^r = 0$. Lemma 3.11.4 implies that this condition is satisfied in the acyclic case: then each walk contains at most $r-1$ edges, where r is the number of vertices of G. In this case, A is a solution of the equation $A^* = A^*A + E$. As K is a field, this means $A^*(E - A) = E$; that is, $A^* = (E - A)^{-1}$.

More generally, it is possible to show that A is stable if all cycles in G have weight 0 with respect to w. The converse is false in general: it is easy to find an example with weights 1 and -1 such that A is stable, but G contains cycles of weight $\neq 0$. However, the converse does hold for $K = \mathbb{R}$ and positive weights.

B.4 Solutions for Chap. 4

4.1.2

(1) ⇒ (2): Let e be an edge contained in the unique cycle C of G. Then $G \setminus e$ is connected and acyclic, so that $G \setminus e$ is a tree.

(2) ⇒ (3): As every tree on n vertices has $n-1$ edges and is connected, the claim in (3) follows.

(3) ⇒ (4): As G is not a tree (since it has one more edge than a tree would have), there must be edges in G which are not bridges; see Lemma 4.1.1. Removing some edge e which is not a bridge yields a tree, so that e has to be contained in each cycle of G. Thus the set of all edges which are not bridges forms a cycle.

(4) $\Rightarrow$ (1): An edge e is not a bridge if and only if it lies in a cycle; see Exercise 1.6.4. Thus G contains a unique cycle, which consists of those edges which are not bridges.

4.1.3 The claim concerning the number of centers is clear for the trees K_1 and K_2. For every other tree T, remove all leaves of T; then the resulting tree T' has the same centers as T, and the assertion follows by induction.

Denote the diameter of a tree T by d and the eccentricity of a center by e. Then either $d = 2e$ or $d = 2e - 1$, and $d = 2e$ holds if and only if T has a unique center. For a formal proof, proceed again by induction.

4.1.4 Let W be a trail of maximal length in G. As G is acyclic, W has to be a path, and as W is maximal, the end vertices of W have degree 1. Thus $G \setminus W$ is a forest containing $2k-2$ vertices of odd degree. Now use induction.

4.1.5 By hypothesis, $\overline{T}$ has at least two connected components. Let x and y be two arbitrary vertices in distinct connected components of $\overline{T}$. In particular, x and y are not adjacent in $\overline{T}$, so that T contains the edge xy. Thus any two points in distinct components of $\overline{T}$ have to be adjacent in T.

The preceding observation shows that there cannot be three distinct connected components: otherwise, we would obtain a cycle of length 3 in T. Moreover, one of the two components must be an isolated point of $\overline{T}$: otherwise, we would obtain a cycle of length 4 in T. Hence T contains a vertex x which is adjacent to all other vertices, so that T is a star. The final assertion follows from Exercise 1.2.4 and Theorem 1.2.6.

4.1.6 There are precisely six isomorphism types of trees on 6 vertices; representatives for these types were given in Fig. 1.6; we will denote these representatives by $T_1, \dots, T_6$. Now let T be any tree on $\{1, \dots, 6\}$. Then the image of T under an arbitrary permutation $\sigma \in S_6$ is a tree isomorphic to T. By a well-known equation for permutation groups, the number of trees isomorphic to T is equal to the order of S_6 (that is, $6! = 720$) divided by the order of the automorphism group of T. We obtain:

T_1: cyclic group of order 2 (rotate the tree by 180°), 360 isomorphic trees;
T_2: cyclic group of order 2 (exchange the two lower leaves of the tree), 360 isomorphic trees;
T_3: symmetric group S_3 (acting on the three lower leaves of the tree), 120 isomorphic trees;
T_4: cyclic group of order 2 (reflect the tree, exchanging the two branches), 360 isomorphic trees;
T_5: direct product of 3 cyclic groups of order 2 (reflect the tree, exchanging the two centers and the two pairs of leaves; or switch the two leaves of one of the two pairs), 90 isomorphic trees;
T_6: symmetric group S_5 (acting on the five leaves), 6 isomorphic trees.

This gives a total of $360+360+120+360+90+6=1296=6^4$ trees, which agrees with the result of Corollary 1.2.11.

4.2.11 By Theorem 4.2.9, the number of spanning trees of the complete bipartite graph $K_{m,n}$ is equal to the absolute value of the determinant of the matrix

$$A' = \begin{pmatrix} nI_m & -J_{m,n-1} \\ -J_{n-1,m} & mI_{n-1} \end{pmatrix},$$

where the indices give the numbers of rows and columns of the respective submatrices (and where I denotes an identity matrix and J a matrix having all entries 1, as usual). Now it is just a matter of some linear algebra to show $\det A' = n^{m-1}m^{n-1}$: using appropriate row and column transformations, one can transform A' into a triangular matrix with diagonal entries $1, n, \ldots, n, m, \ldots, m$.

4.2.12 The proofs of the results in question carry over: just take into account that now $1+1=0$, and hence $-1=+1$.

4.2.13 First assume that G' is bipartite, with respect to the partition $V = S \dot\cup T$. Let M' be a square submatrix of M of order k, say. The case $k=1$ is trivial, so let $k \neq 1$. First consider the case where each column of M' contains two entries 1. The k vertices corresponding to the rows of M' can be divided into two sets $S' \subset S$ and $T' \subset T$. Each column of M' corresponds to an edge of G which has both end vertices in $S' \cup T'$ (by hypothesis). As G is bipartite, each column of M has one entry 1 in a row corresponding to S', and the other entry 1 in a row corresponding to T'. Hence the sum of the rows corresponding to S' equals the sum of the rows corresponding to T', so that the rows of M' are linearly dependent, and hence $\det M' = 0$. It remains to consider the case where M' contains a column with at most one entry 1. Then the claim follows by developing $\det M'$ with respect to this column (and using induction).

Conversely, let M be totally unimodular, and suppose that G is not bipartite. By Theorem 3.3.5, G contains a cycle C of odd length, say

$$C: \quad v_0 \overset{e_1}{\text{———}} v_1 \cdots v_{2n-1} \overset{e_{2n}}{\text{———}} v_{2n} \overset{e_{2n+1}}{\text{———}} v_0.$$

But then the determinant of the submatrix M corresponding to the $2n+1$ vertices and the $2n+1$ edges of C is 2, a contradiction.

4.2.14 By Corollary 1.2.11, the graph K_n has precisely n^{n-2} spanning trees. Note that each spanning tree of K_n has $n-1$ edges and that each edge e has to be contained in the same number x of spanning trees, which implies $x = 2n^{n-3}$. Hence the number of spanning trees of $K_n \setminus e$ is $n^{n-2} - 2n^{n-3} = (n-2)n^{n-3}$.

4.2.15 G has $p = n - m$ connected components.

4.2.16 We may assume that F consists of the edges $\{2i-1, 2i\}$, where $i = 1, \dots, n$. Then, by Theorem 4.2.9, the number of spanning trees of $G \setminus F$ equals the determinant of the $(2n-1) \times (2n-1)$-matrix

$$M = \begin{pmatrix} 2n-2 & 0 & * & * & * & \dots & * & * & * & * \\ 0 & 2n-2 & * & * & * & \dots & * & * & * & * \\ * & * & 2n-2 & 0 & * & \dots & * & * & * & * \\ * & * & 0 & 2n-2 & * & \dots & * & * & * & * \\ & & & & \ddots & & & & & \\ * & * & * & * & * & \dots & * & 2n-2 & 0 & * \\ * & * & * & * & * & \dots & * & 0 & 2n-2 & * \\ * & * & * & * & * & \dots & * & * & * & 2n-2 \end{pmatrix},$$

where we have used the abbreviation $*$ to indicate entries equal to -1. We may compute the determinant of M as the product of its eigenvalues. From the form of M, one easily sees the following $2n-3$ eigenvalues:

- an $(n-1)$-fold eigenvalue $2n-2$, belonging to the pairwise orthogonal eigenvectors $(1 * 00 \dots 0)^T; (001 * 00 \dots 0)^T; \dots; (00 \dots 001 * 0)^T$;
- an $(n-2)$-fold eigenvalue $2n$, belonging to the pairwise orthogonal eigenvectors $(11 * * 00 \dots 0)^T; (1100 * * 00 \dots 0)^T; \dots; (110 \dots 00 * * 0)^T$.

The remaining two eigenvalues are not obvious. However, the orthogonal complement of the $(2n-3)$-dimensional vector space generated by the eigenvalues constructed so far is clearly spanned by the two vectors $(1, \dots, 1, 0)^T$ and $(0, \dots, 0, 1)^T$, which are not eigenvectors. Thus the further eigenvectors have to have the form $(1, \dots, 1, x)^T$. Then the corresponding eigenvalue is necessarily $2 - x$, which leads to the condition

$$(2n-2)(x-1) = x(2-x).$$

We get two solutions for x, namely $-n + 2 \pm \sqrt{n^2 - 2n + 2}$. Hence the missing two eigenvalues are $n \pm \sqrt{n^2 - 2n + 2}$, with product $2n-2$. Therefore the desired number equals

$$(2n-2)^n (2n)^{n-2}.$$

4.3.4 Let e be an edge incident with v which has smallest weight among all such edges, and suppose that e is not contained in a given minimal spanning tree T for G. The cycle $C_T(e)$ which arises by adding e to T has to contain a second edge incident with v, say f; by Theorem 4.3.1, $w(e) \geq w(f)$. In view of our choice of e, we conclude $w(f) = w(e)$, so that f is an edge of T having the required property.

4.3.5 The assertion is an immediate consequence of Exercise 4.3.6. A direct proof of the special case in question could proceed as follows. Suppose that G contains two distinct minimal spanning trees T and T'. Order the edges of T and T' according to increasing weight and assume that both trees have their first $k-1$ edges in common, whereas they differ in their respective k^{th} edges:

$$T = \{e_1, \dots, e_{k-1}, e_k, \dots, e_{n-1}\} \quad \text{and} \quad T' = \{e_1, \dots, e_{k-1}, e'_k, \dots, e'_{n-1}\},$$

where (without loss of generality)

$$w(e_1) < \dots < w(e_{n-1}) \quad \text{and} \quad w(e_k) < w(e'_k) < \dots < w(e'_{n-1}).$$

Adding the edge e_k to T' yields a cycle $C_{T'}(e_k)$; by Theorem 4.3.1, $w(e_k) \geq w(f)$ for all edges $f \in C_{T'}(e_k)$. As the weights of the edges are distinct, all edges $f \neq e_k$ of $C_{T'}(e_k)$ have to be contained among the first $k-1$ edges $e_1, \dots, e_{k-1}$ of T'. Hence T contains the cycle $C_{T'}(e_k)$, a contradiction.

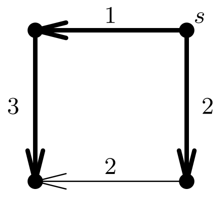
4.3.6 Let T be any minimal spanning tree, so that T satisfies condition (4.1). Then T can be transformed into any fixed minimal spanning tree T' by a sequence of edge exchanges as in the proof of Theorem 4.3.1, where we obtained a third minimal spanning tree $T'' = (T' \setminus \{e'\}) \cup \{e\}$. Moreover, we had $w(e) = w(e')$, so that such an exchange always transforms T' into a tree T'' with the same weight sequence. Hence the induction argument given in the proof of Theorem 4.3.1 actually proves the additional assertion of the present exercise.

4.4.6 Perturb the weight function w by adding small constants to the weights of edges having the same weight under w. Clearly, this may be done in such a way that the resulting weight function w' assigns distinct weights to distinct edges, and that $w'(e) < w'(e')$ holds if and only if either $w(e) < w(e')$ or $w(e) = w(e')$, but e precedes e' under the specified tiebreaking rule. By Exercise 4.3.5, there is a unique minimal spanning tree with respect to w', which implies the assertion.

4.4.11 Order the edges of G according to increasing weight. As the algorithm of Kruskal constructs a minimal spanning tree T by selecting edges in this order (as far as possible), no spanning tree having a smaller maximum edge weight can exist. Moreover, any two minimal spanning trees have the same weight sequence by Exercise 4.3.6. Hence any spanning tree T' satisfies $W(T') \geq W(T)$.

4.4.15 Assign weight 1 to all the edges, and apply the algorithm of Boruvka in this situation. Then we could choose an arbitrary edge e_u leaving a given connected component $U \in M$. In general, there will exist two connected components $U, U' \in M$ which can be connected by two different edges of G; then choosing these two edges as e_u and $e_{u'}$ would create a cycle.

Fig. B.12 A digraph



4.4.16 A minimal spanning tree has weight $2+13+21+35+51=122$.

4.4.17 The proof of Theorem 4.3.1 shows that the subgraph of the minimal spanning trees (for a given weight function w) is connected. If we assign weight $w(e)=1$ to all edges e, we see that this implies that the whole tree graph is connected.

4.5.5 The edges $e_{15}, e_{14}, e_{13}, e_{12}, e_{11}, e_{10}$ and e_8 form a maximal spanning tree (of weight $28+27+26+24+10+9+8=132$). The edges are given in the order in which the algorithm of Kruskal would find them.

4.5.6 The following characterization of maximal spanning trees follows from Theorem 4.3.1 by replacing w by $-w$: a spanning tree T is maximal if and only if the condition

$$(*) \quad w(e) \le w(f) \quad \text{for all edges } f \text{ in } C_T(e)$$

holds for each edge $e \notin T$. Now let $e=uv$ be an edge of G not contained in T. By hypothesis, the unique path P from u to v in T has capacity $w(P) \ge w(e)$; this implies $(*)$ in view of $C_T(e)=P \cup \{e\}$, which proves the assertion.

4.5.8 The digraph shown in Fig. B.12 provides an example.

4.7.9 Let T be an arbitrary spanning tree for G, and let x be a center of T. Denote the eccentricity of x in T by $e_T(x)$; then T has diameter either $d_T=2e_T(x)$ or $d_T=2e_T(x)-1$ by Exercise 4.1.3. Clearly, x has eccentricity at most $e_T(x)$ in G. Thus it is an obvious approach to look for spanning trees whose centers are centers of G as well.

Now let z be a center of G, and let T_z be a spanning tree for G determined by a BFS starting at z. Note that T_z is an SP-tree for G with root z. It is easy to see that z is also a center of T_z. Therefore T_z has diameter $d=2e$ or $d=2e-1$, where e denotes the eccentricity of z in G. Moreover, every other spanning tree has diameter at least $2e-1$. Hence the tree T_z solves our problem; note that a center z (and then a tree T_z) can be determined with complexity $O(|V|^3)$ by Theorem 3.9.8.

We mention that it is easy to find examples where a BFS starting at z could either find a tree of diameter $2e$ or a tree of diameter $2e-1$, depending on the order in which adjacent vertices are examined.

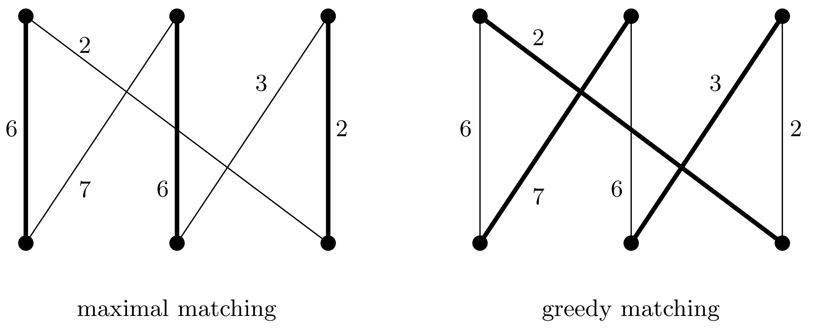
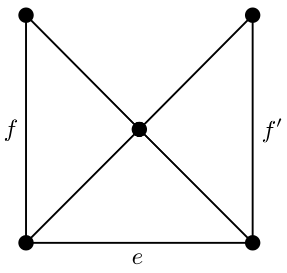


Fig. B.13 Two matchings

Fig. B.14 A counter-example for the case $k = 2$



B.5 Solutions for Chap. 5

5.1.5 The network in Fig. B.13 has a maximal matching of weight 14, but the greedy algorithm constructs a matching of weight 12.

5.2.3 Let N be the incidence matrix of the graph $G = (V, E)$. Identify E with the set of columns of N and apply Theorem 4.2.3.

5.2.4 Let $A \subseteq E$. As the forests of G form the graphic matroid $M(G) = M_0(G)$ on E, A has a well-defined rank $\varrho(A)$ in $M(G)$, namely the maximal cardinality of a forest contained in A. We use this fact to verify condition (3) in Theorem 5.2.1 for $M = M_k(G)$, which will establish that M is likewise a matroid. We distinguish two cases: if $|A| - \varrho(A) \leq k$, then A itself is independent in M (and thus the only maximal independent set contained in A); and if $|A| - \varrho(A) > k$, then the maximal independent sets of M contained in A are the maximal forests in A enlarged by any k further edges from A (and thus all have cardinality $\varrho(A) + k$).

5.2.5 The assertion is a special case of Exercise 5.2.3, as the 1-forests of $G = (V, E)$ are just those subsets of E which contain at most one cycle. The analogous statement for $k = 2$ does *not* hold, as Fig. B.14 shows.

Here removing either e or both of f and f' results in maximal subgraphs with at most two cycles, so that condition (3) in Theorem (5.2.1) is violated. Note that adding two edges to the star spanning G may create either two or three cycles, depending on whether or not e is one of these edges.

5.2.6 Conditions (1) and (2) are clear. To show that (3) holds, let J be a maximal independent subset of $A \cap B$, and choose a maximal independent subset K of $A \cup B$ containing J. Write $K = J \,\dot\cup\, X \,\dot\cup\, Y$ with $X \subset A$ and $Y \subset B$. Then $J \cup X$ and $J \cup Y$ are independent subsets of A and B, respectively, so that

$$\rho(A \cup B) + \rho(A \cap B) = 2|J| + |X| + |Y| = |J \cup X| + |J \cup Y| \le \rho(A) + \rho(B).$$

5.2.10 By Theorem 5.2.9, $\sigma(X) = \{e \in E : \rho(X \cup \{e\}) = \rho(X)\}$; therefore condition (1) is clear. To prove (2), let J be a maximal independent subset of Y, and choose a maximal independent subset K of X containing J. If $e \in \sigma(Y)$, then $e \in \sigma(X)$: otherwise $K \cup \{e\}$ would be independent, so that $J \cup \{e\}$ would be independent as well, contradicting $\rho(J \cup \{e\}) = \rho(J)$).

By Theorem 5.2.9, $\sigma(X)$ is the unique maximal set containing X such that $\rho(\sigma(X)) = \rho(X)$; now (3) is clear. To show (4), let J be a maximal independent subset of X (and hence of $\sigma(X)$). As $y \notin \sigma(X)$ and $y \in \sigma(X \cup \{x\})$, $J \cup \{x\}$ and $J \cup \{y\}$ have to be independent sets. Moreover, $\rho(X \cup \{x\}) = \rho(X \cup \{y\}) = \rho(X \cup \{x, y\})$. But this implies $x \in \sigma(X \cup \{y\})$.

5.2.11 Let B be a basis of the matroid $M = (E, \mathbf{S})$. As $\rho(B) = \rho(E)$, Theorem 5.2.9 yields $\rho(B) = E$, so that B is a generating set for M. Suppose that B is not minimal. Then there exists a proper subset C of B such that $B \subset E = \sigma(C)$. But then $\rho(E) = |C| < |B|$, which contradicts the fact that B is independent.

Conversely, let D be a minimal generating set and A a maximal independent subset of D. Then $\rho(D) = |A|$ implies $D \subset \sigma(A)$ and (using Exercise 5.2.10) $E = \sigma(D) \subset \sigma(\sigma(A)) = \sigma(A)$. Hence A is a generating set of M, and the minimality of D implies $A = D$. Thus D is independent. Now $\sigma(D) = E$, so that $|D| = \rho(E)$; therefore D is a basis of M.

5.2.12 Let A and B be two closed sets in M. Then

$$\sigma(A \cap B) \subset \sigma(A) \cap \sigma(B) = A \cap B \subset \sigma(A \cap B),$$

so that $A \cap B$ is closed as well; this establishes (a).

To prove (b), let A be a closed set containing X. Then $\sigma(X) \subset \sigma(A) = A$. Thus $\sigma(X)$ is contained in the intersection of all closed sets containing X. Now (a) implies that $\sigma(X)$ coincides with this intersection.

Finally, suppose that the condition in (c) is violated for some $x \in E \setminus X$, so that $\rho(X \cup \{x\}) = \rho(X)$. Then $x \in \sigma(X)$, and X cannot be closed. The converse is similar.

5.2.13 Let $\{x_1, \dots, x_r\}$ be a basis of $(E, \mathbf{S})$. The 2^r subsets of this basis have 2^r distinct spans.

5.2.17 Suppose that condition (2′) does not hold. Choose two cycles C and D and elements $x \in C \cap D$ and $y \in C \setminus D$ violating (2′), so that $|C \cup D|$ is minimal among all counterexamples. In view of Theorem 5.2.16, there exists a cycle $F_1 \subset (C \cup D) \setminus \{x\}$ with $y \notin F_1$. Note that the set $F_1 \cap (D \setminus C)$ cannot be empty: otherwise F_1 would be a proper subset of C. Hence we may choose an element $z \in F_1 \cap (D \setminus C)$.

Now consider the cycles D and F_1 and the elements $z \in D \cap F_1$ and $x \in D \setminus F_1$. Note that $D \cup F_1$ is a proper subset of $C \cup D$, since $y \notin D \cup F_1$. By the minimality of our counterexample, there exists a cycle F_2 such that $x \in F_2 \subset (D \cup F_1) \setminus \{z\}$. Consider C, F_2, $x \in C \cap F_2$, and $y \in C \setminus F_2$. Again, the minimality of our counterexample applies: there exists a cycle F_3 such that $y \in F_3 \subset (C \cup F_2) \setminus \{x\}$. As $C \cup F_2$ is contained in $C \cup D$, we have obtained a contradiction.

5.3.4 We use Exercise 5.2.11 to show that $E \setminus C$ is a closed set of M^*. Thus let c be any element of C. We need to check that adding c to $E \setminus C$ increases the rank:

$$\begin{aligned}\varrho^*\big((E \setminus C) \cup \{c\}\big) &= \big|(E \setminus C) \cup \{c\}\big| + \varrho\big(C \setminus \{c\}\big) - \varrho(E) \\ &= |E \setminus C| + 1 + \varrho(C) - \varrho(E) = \varrho^*(E \setminus C) + 1.\end{aligned}$$

It remains to show $\varrho^*(E \setminus C) = \varrho^*(M^*) - 1$. As C is a circuit, $C \setminus \{c\}$ is an independent set and hence contained in a basis B of M. Then $B^* = E \setminus B$ is a basis for M^* and therefore has rank $\varrho^*(M^*)$. But $B^* \subseteq (E \setminus C) \cup \{c\}$, and thus $\varrho^*((E \setminus C) \cup \{c\}) = \varrho^*(M^*)$, which proves the assertion.

5.3.7 By Theorem 5.3.1, $\rho(E \setminus A^*) = \rho^*(A^*) - |A^*| + \rho(E) = \rho(E)$, since A^* is independent in M^*. As A is an independent subset of $E \setminus A^*$, A can be extended to a maximal independent subset (in M) of $E \setminus A^*$; we denote this subset by B. Then $\rho(B) = \rho(E)$, so that B is a basis of M. Hence $B^* = E \setminus B$ is a basis of M^* containing A^*.

5.3.8 First let B be a basis of M. Suppose that C^* is a cocircuit which is disjoint to B. Then $E \setminus B$ contains the cocircuit C^* and is dependent in M^*, which contradicts Corollary 5.3.2. Now suppose that a subset X of B intersects each cocircuit. Then $E \setminus X$ cannot contain any circuit of M^*, so that $E \setminus X$ must be independent in M^*. As $E \setminus X$ contains the basis $E \setminus B$ of M^*, we conclude $X = B$. Thus the bases are the minimal sets intersecting each cocircuit. The converse is shown in a similar manner.

5.3.9 Suppose $C \cap C^* = \{e\}$. Then the disjoint sets $A = C \setminus \{e\}$ and $A^* = C^* \setminus \{e\}$ are independent in M and in M^*, respectively. By Exercise 5.3.7, A

and A^* can be extended to bases B and B^* of M and M^*, respectively, and these bases are disjoint. Hence $E = B \cup B^*$. As C and C^* are dependent, e can be contained neither in B nor in B^*, a contradiction.

5.3.10 Let B be a basis of M containing $C \setminus \{x\}$. As $B^* = E \setminus B$ is a basis of M^*, $B^* \cup \{y\}$ has to contain a unique cocircuit C^* of M by Theorem 5.2.14. Obviously, y must be contained in C^*. Now $x \notin C^*$ would imply $|C \cap C^*| = 1$, contradicting Exercise 5.3.9. Thus $x, y \in C \cap C^*$, so that $C \cap C^* = \{x, y\}$.

5.4.6 It suffices to find a subset A of E and two maximal independent subsets D and D' of A such that $2|D'| = n|D|$. We may assume $V = \{1, \dots, n\}$. Then $D = \{(i, i+1) : i = 1, \dots, n-1\}$, $D' = \{(i,j) : i, j = 1, \dots, n \text{ and } i > j\}$ and $A = D \cup D'$ have the required property.

5.4.10 M is the intersection of the graphic matroid $M(G)$, the head-partition matroid of G, and the tail-partition matroid of G.

5.5.7 Suppose w does not have to satisfy the triangle inequality. Then we may, for instance, increase the weight of the edge of maximal weight in Example 5.5.6 by an arbitrary value and thus make the solution determined by the greedy algorithm arbitrarily poor.

5.6.3 Suppose that (CC) is violated by some $A \in \mathbf{S}$, elements $x, y \in \operatorname{ext}(A)$, and a set $X \subset E \setminus (A \cup \operatorname{ext}(A))$. Thus there exists a basis B such that $A \cup X \cup \{x\} \subset B$, whereas $A \cup X \cup \{y\}$ is not contained in any basis. Consider the following weight function for E:

$$w(z) = \begin{cases} 3 & \text{if } z \in A, \\ 2 & \text{if } z \in X, \\ 1 & \text{if } z = y, \\ 0 & \text{otherwise.} \end{cases}$$

Then the basis B has weight $w(B) = 3|A| + 2|X|$. The greedy algorithm begins by constructing (in some order) the feasible set A and then adds y; note that the elements of X have larger weight than y, but are not contained in $\operatorname{ext}(A)$. After that, the algorithm can add at most $|X| - 1$ of the elements of X, because we assumed that $A \cup X \cup \{y\}$ is not contained in any feasible set. Thus the solution generated by the greedy algorithm has weight at most

$$3|A| + 1 + 2\big(|X| - 1\big) < w(B),$$

a contradiction.

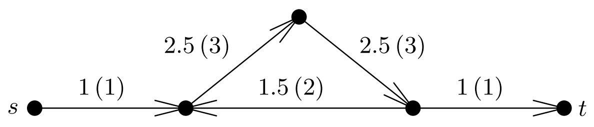


Fig. B.15 A flow

B.6 Solutions for Chap. 6

6.1.9 Replace each vertex v by a pair (v, v') of vertices, and each edge vw by $v'w$. Furthermore, add all edges of the form vv', and put $c(v'w) = c(vw)$ and $c(vv') = d(v)$. It is easily checked that a flow f' on the new network corresponds to a flow f on N satisfying (F3).

Now let (S,T) be a cut in the new network, and denote the set of edges e with $e^- \in S$ and $e^+ \in T$ by E'. Each edge of type $v'w$ corresponds to an edge vw in N, and each edge of type vv' corresponds to a vertex v of N. Thus the set E' of edges of the cut (S,T) corresponds to a cut in N in the following sense: a (generalized) *cut* is a set of edges and vertices (distinct from s and t) of G so that every directed path from s to t contains at least one of these edges and vertices. The *capacity* of such a cut is the sum of all $c(e)$ and $d(v)$ for edges e and vertices v, respectively, which are contained in the cut. Then the generalization of Theorem 6.1.6 states that he minimal capacity of a generalized cut equals the maximal value of a flow satisfying (F3). This theorem is easily derived by applying Theorem 6.1.6 to the network defined above.

6.1.10 If we require k vertices $s_1, \dots, s_k$ as sources (so that (F2) does not have to be satisfied for these vertices, and as much flow as possible should originate there), we can add a new source s and all edges ss_i $(i = 1, \dots, k)$ with sufficiently large capacity.

6.1.11 Let W be the maximal value of a flow on N, and let (S,T) be a minimal cut; by hypothesis, $c(S,T) = W \neq 0$. If we remove an edge e with $e^- \in S$ and $e^+ \in T$ and $c(e) \neq 0$ from G, the capacity $c(S,T)$ and hence the value of a maximal flow is decreased by $c(e)$. This suggests to choose e as an edge of maximal capacity in a minimal cut. However, these edges do not have to be most vital, as the example of the network given in Fig. 6.12 shows: here the edge sa is obviously most vital, but it is not contained in a minimal cut.

6.1.12 No: the flow in the flow network of Fig. B.15 provides a counter-example.

6.1.13 The capacities in the flow network of Fig. B.15 actually define an integral flow, which is obviously maximal but not the sum of elementary flows.

6.1.14 First, in step (3) of Algorithm 6.1.7, we set $d(v) = 0$ for $v \neq s$. During the following labelling process, the labels are not permanent; similarly to the algorithm of Dijkstra, the label of the vertex v which is chosen in step (5) is made permanent at this point. As we want to construct augmenting paths of maximal capacity from s to all the other vertices, we choose in step (5)—among all labelled vertices v with $u(v) =$ false (that is, v is not yet permanent)—the vertex v for which $d(v)$ is maximal; initially, this is s.

Moreover, we do not change the flow as soon as t is reached, but wait until t is chosen in step (5) (and thus made permanent). For this purpose, we insert an **if** clause after step (5): if $v = t$, we may change the flow as in steps (16) to (28) of Algorithm 6.1.7; of course, we have to set $d(v) = 0$ for $v \neq s$ in step (27). Otherwise (if $v \neq t$), the labelling process is continued from v. As in steps (6) to (9), we first consider all edges of the form $e = vw$. If $u(w) =$ false (that is, w is not yet labelled permanently) and $d(w) < \min\{d(v), c(e) - f(e)\}$, then $d(w)$ is replaced by this minimum and w is labelled with $(v, +, d(w))$, so that the former label is also replaced. Steps (10) to (13) (for edges of the form $e = wv$) are changed in an analogous manner. Next v is made permanent in step (14). We leave the details and the task of writing down a formal version of this method to the reader.

6.1.15 Let us write $P = S \cap S'$, $Q = T \cap S'$, $R = S \cap T'$, and $U = T \cap T'$; see Fig. B.16. Denote the maximal flow value on N by W. By hypothesis,

$$W = c(S,T) = c(P,Q) + c(P,U) + c(R,Q) + c(R,U)$$

and

$$W = c\big(S',T'\big) = c(P,R) + c(P,U) + c(Q,R) + c(Q,U).$$

On the other hand, using Lemma 6.1.2,

$$W \le c\big(S \cap S', T \cup T'\big) = c(P,Q) + c(P,R) + c(P,U)$$

and

$$W \le c\big(S \cup S', T \cap T'\big) = c(P,U) + c(Q,U) + c(R,U).$$

Hence

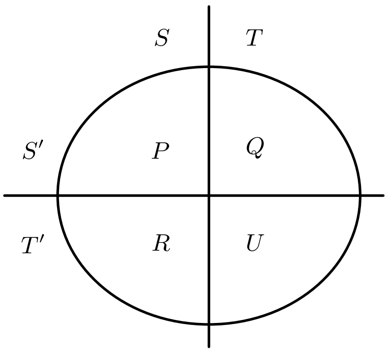
$$2W = c(S,T) + c\big(S',T'\big) \ge c\big(S \cap S', T \cup T'\big) + c\big(S \cup S', T \cap T'\big) \ge 2W.$$

Thus we have equality throughout, implying

$$c\big(S \cap S', T \cup T'\big) = c\big(S \cup S', T \cap T'\big) = W.$$

6.1.16 Let (S,T) be any minimal cut, and assume that some vertex $v \in T$ is accessible from s on an augmenting path P with respect to the given maximal flow f. Clearly, P has to contain an edge e which lies in the cocycle $E(S,T)$. If e is a forward edge, it cannot be saturated; and if e is a backward edge, it cannot be void. But this contradicts the characterization of minimal cuts

Fig. B.16 The cuts in Exercise 6.1.15



given in Lemma 6.1.2, and we conclude $S_f \subseteq S$ for each minimal cut (S,T). Hence the intersection S_0 of all such S contains S_f. By Exercise 6.1.15, S_0 is itself the s-part of a minimal cut, which proves the assertion: $S_f = S_0$.

6.2.4 A second maximal flow g can be obtained from the flow f_9 in Fig. B.15 by letting the edge ct not carry any flow, and enlarging the value of the flow on the edges cf and ft accordingly: $g(ct) = 0$, $g(cf) = 15$, $g(ft) = 17$. Actually, there are further maximal flows, as there are several ways of distributing the flow emanating from c.

In contrast, (S_f, T_f) is the unique minimal cut, as can be seen using Exercise 6.1.16 and the criterion in Lemma 6.1.2. For instance, if we wanted to move the vertex b from the t-part T_f to the s-part S_f of the cut, we would have to include also c into the s-part, as the edge bc is not saturated. Conversely, if we wanted to include c, we also would have to include b, as bc is not void either. Using this type of argument shows that the s-part S_f cannot be enlarged at all.

6.2.5 We use the algorithm described in the solution to Exercise 6.1.14. During the first iteration, the vertices chosen in step (5) are s with $d(s) = \infty$, a with $d(a) = 38$, d with $d(d) = 13$, c with $d(c) = 10$, f with $d(f) = 10$, and t with $d(t) = 10$ (in this order). This yields an augmenting path with capacity 10; we obtain the flow f_1 of value 10 shown in Fig. B.17, which also gives the labels determined by the first iteration.

During the next iteration, the vertices s with $d(s) = \infty$, a with $d(a) = 28$, d with $d(d) = 13$, b with $d(b) = 8$, c with $d(c) = 8$, f with $d(f) = 8$, and t with $d(t) = 8$ are chosen in step (5). The corresponding augmenting path with capacity 8 yields the flow f_2 shown in Fig. B.18.

During the following iteration, the vertices chosen in step (5) are s with $d(s) = \infty$, a with $d(a) = 20$, d with $d(d) = 13$, and t with $d(t) = 7$. We obtain an augmenting path with capacity 7 and the flow f_3 shown in Fig. B.19.

Four more iterations are needed; the augmenting paths constructed are

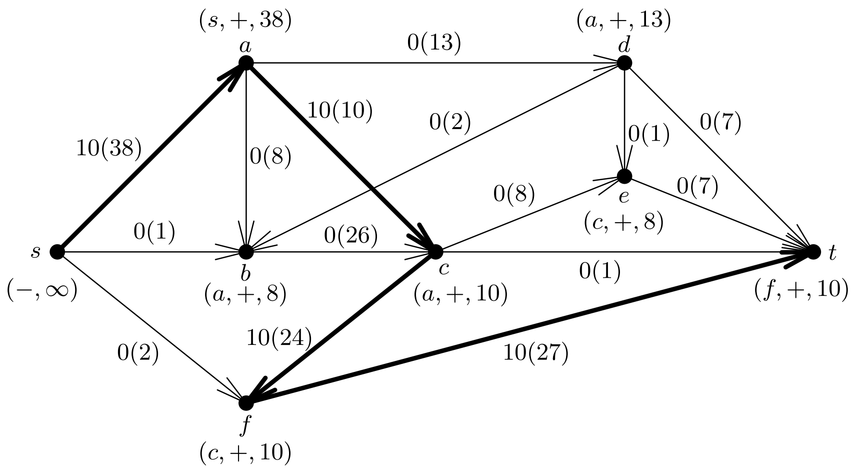


Fig. B.17 $w(f_1) = 10$

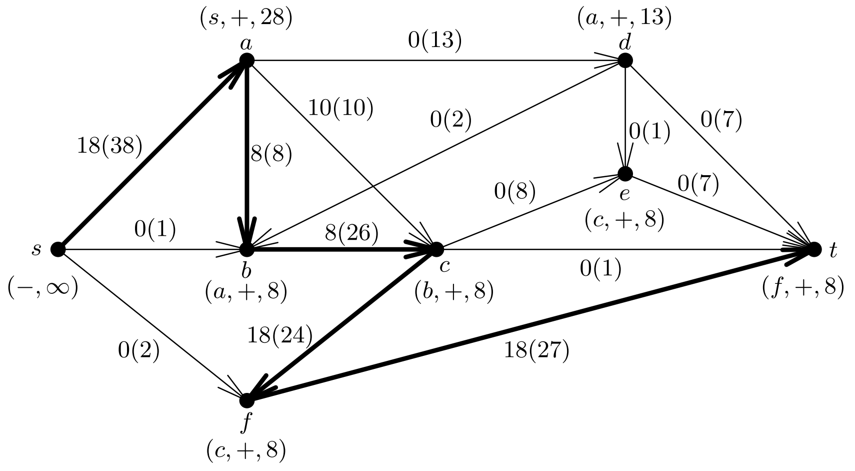


Fig. B.18 $w(f_2) = 18$

- $s — f — t$ with capacity 2,
- $s — a — d — b — c — f — t$ with capacity 2,
- $s — b — c — t$ with capacity 1,
- and $s — a — d — e — t$ with capacity 1.

The resulting flow f with $w(f) = 31$ is shown in Fig. B.20.

Thus this algorithm needs seven flow changes, whereas the algorithm of Edmonds and Karp used in Example 6.2.3 made nine changes. However, in

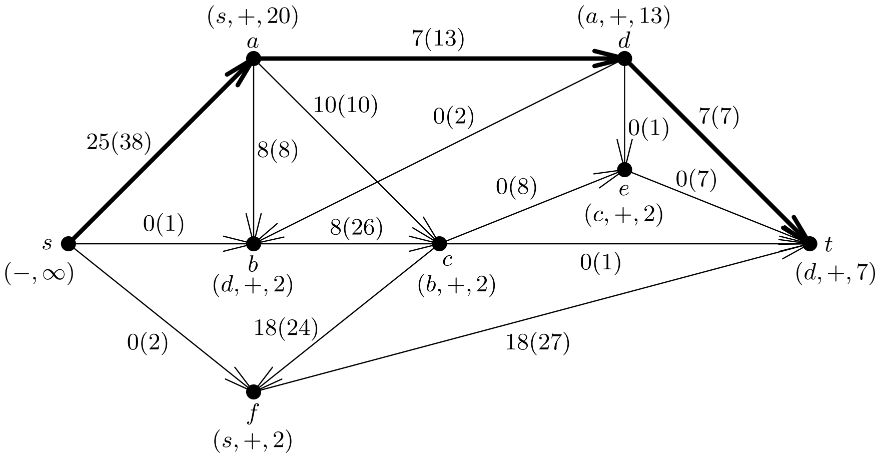


Fig. B.19 $w(f_3) = 25$

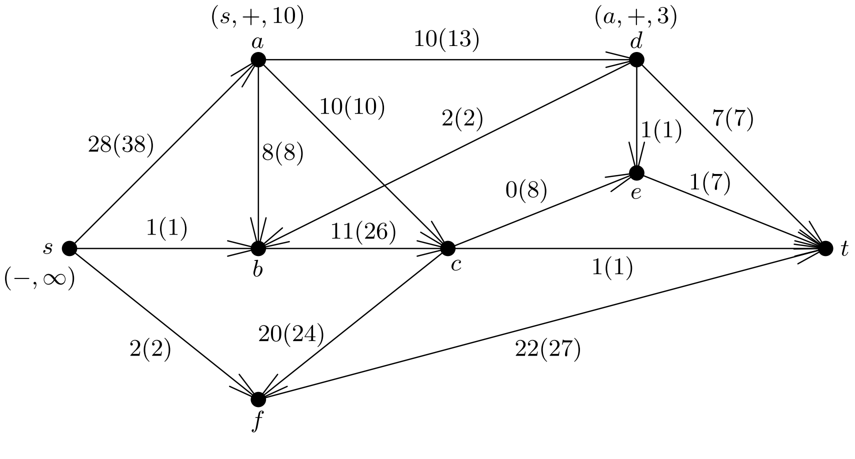


Fig. B.20 $w(f) = 31$

the algorithm used here, the labelling process is somewhat more involved. Note that the maximal flows of Figs. 6.12 and B.20 are not identical.

6.2.6 The maximal value of a flow is 5; Fig. B.21 shows a flow f with $w(f) = 5$ and a cut having this capacity.

6.2.7 Let f be the flow of value $W = w(f)$ which was found for the incorrect capacity $d(e)$, let (S,T) be a minimal cut, and denote the correct capacity by $c(e)$. The results for the incorrect input data can be used when calculating a flow for the correct capacity as follows, where we distinguish two cases.

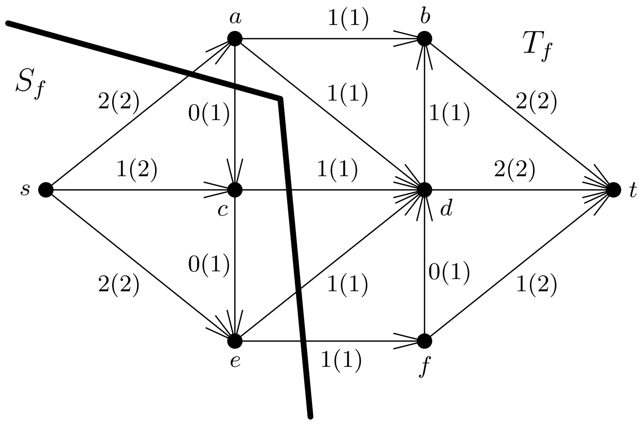


Fig. B.21 Solution to Exercise 6.2.5

Case 1. $c(e) < d(e)$. It is clear that (S,T) is still a minimal cut, if e is contained in (S,T) (that is, $e^- \in S$ and $e^+ \in T$). In the corrected network, (S,T) has capacity $c(S,T) - (d(e) - c(e))$, so that the maximal value of a flow is $W' = W - (d(e) - c(e))$. To find a flow of value W', consider all the augmenting paths (constructed before) containing e and decrease the value of the corresponding flow by $d(e) - c(e)$.

If e is not contained in (S,T) and $f(e) \leq c(e)$, there is obviously nothing to change. If $f(e) > c(e)$, we decrease the flow by $f(e) - c(e)$ (as before) and run the algorithm again, using the decreased flow as the initial flow.

Case 2. $c(e) > d(e)$: If e is not contained in (S,T), then (S,T) is still a minimal cut and there is nothing to change. Otherwise, we run the algorithm again, using f as the initial flow.

6.2.8 Note that the edge $e = ac$ is contained in the minimal cut (S,T) shown in Fig. 6.12. If $c(e) = 8$, (S,T) is still a minimal cut, so that the value of the flow has to be decreased to 29. A maximal flow of this value can be constructed from the flow of Fig. 6.12 by decreasing the flow values of all edges in the augmenting path shown in Fig. 6.7 by 2. For $c(e) = 12$, the same augmenting path can be used for increasing the value of the flow to 33.

6.2.9 First, the capacity of ac is increased to 12, so that the value of the flow can be increased to 33 (by increasing $f(e)$ by 2 for each of the edges $e = sa, ac, cf, ft$); see Exercise 6.2.8. Since the edge ad is not contained in the minimal cut (S,T), increasing the capacity of this edge does not affect the maximal flow. Now we delete the edge de. As this edge is contained in (S,T), the value of the flow has to be decreased by 1, say along the path $s — a — d — e — t$. Finally, ct is removed. The value of a maximal flow is not changed, because the unit of flow carried by ct can be moved along the

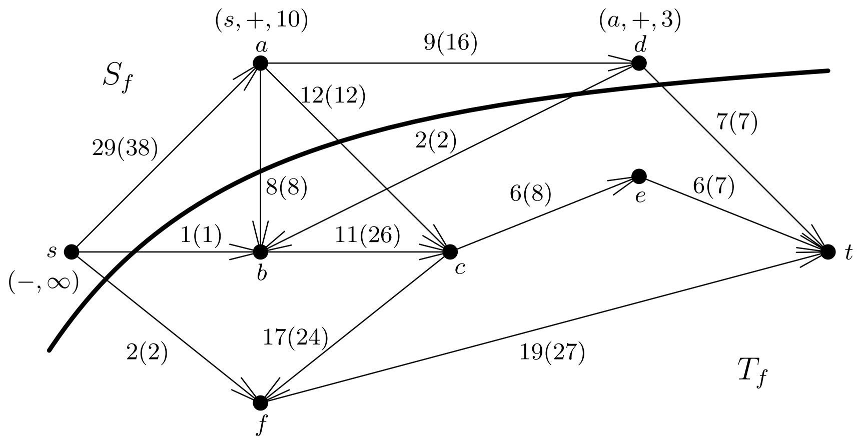


Fig. B.22 $w(f)=32=c(S,T)$

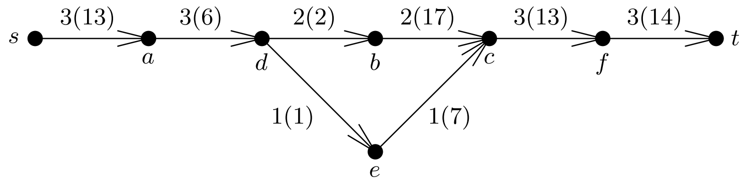


Fig. B.23 A blocking flow

path c — f — t instead. We obtain the flow of value 32 shown in Fig. B.22; note that (S_f, T_f) is still a minimal cut, as it should be according to Exercise 6.1.16.

6.3.5 By definition, $c'(S,T)$ is the sum of all $c'(x)$ for $x^- \in S$ and $x^+ \in T$. If $x = e'$ corresponds to a forward edge e, we have $c'(x) = c(e) - f(e)$. Otherwise (if $x = e''$ corresponds to a backward edge e), $c'(x) = f(e)$. Thus

$$c'(S,T) = \sum_{e^- \in S, e^+ \in T} c(e) - \sum_{e^- \in S, e^+ \in T} f(e) + \sum_{e^- \in T, e^+ \in S} f(e);$$

hence, using Lemma 6.1.2, $c'(S,T) = c(S,T) - w(f)$. In particular, this holds for minimal cuts, and the assertion follows by applying Theorem 6.1.6 to both networks.

6.3.8 Execute a BFS starting at t on the digraph with opposite orientation, and remove all vertices which are not reached during the algorithm.

6.3.9 The network N'' and a blocking flow are shown in Fig. B.23.

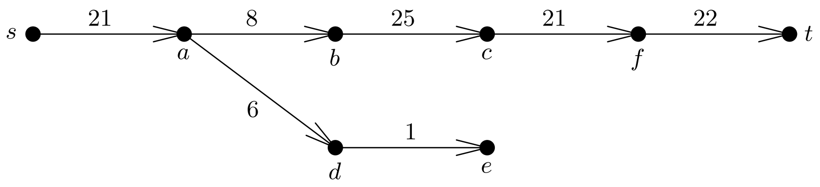


Fig. B.24 Layered auxiliary network for $N'(f)$

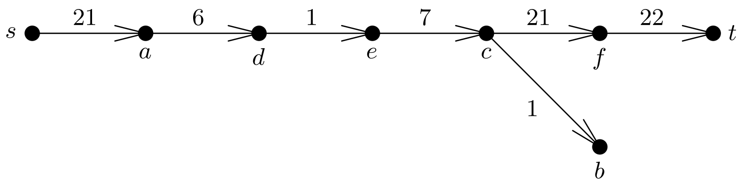


Fig. B.25 Layered auxiliary network for $N''(f)$

6.3.10 Consider Example 6.3.7, and note that the blocking flow g on $N''(f)$ of value 10 leads to a maximal flow g' of value 11 on $N''(f)$. The underlying flow f has value 10, whereas the maximal value of a flow on N is $31 \neq 10+11$; see Example 6.2.3.

6.3.13 The layered auxiliary network with respect to g on $N'(f)$ is shown in Fig. B.24, and the layered auxiliary network with respect to g on $N''(f)$ is shown in Fig. B.25. The flow determined on N by f and g is the flow $h = f_6$ shown in Fig. 6.9. Thus $N''(h)$ is equal to the network of Fig. B.24.

6.3.19 Replace step (17) of procedure AUXNET (Algorithm 6.3.14) by

```
(17′)   if $t \in V''$ then max $\leftarrow$ false; $d \leftarrow i$ else max $\leftarrow$ true;
        $S \leftarrow V''$; $T \leftarrow V \setminus S$ fi
```

6.4.5 A blocking flow determined by Algorithm 6.4.1 is shown in Fig. B.26. The paths corresponding to the sequences (s,a,d,f,t), (s,b,d,f,t), (s,c,d,f,t), (s,c,d,g,t), (s,a,e,h,t), (s,a,e,k,t) were constructed in this order (as usual, if there were several possible ways of choosing the edge $e = uv$ in step (5), we have proceeded according to the alphabetical order of the vertices); their capacities are 3, 2, 4, 3, 1, and 10, respectively. Thus the total value of the flow is 23.

6.4.10 Algorithm 6.4.6 needs four iterations, where the vertices of minimal potential are h with $p(h) = 4$, c with $p(c) = 7$, d with $p(d) = 2$, and e with $p(e) = 10$, respectively. The resulting blocking flow of value 23 is shown in Fig. B.27. Note that it is not identical with the one given in Fig. B.26.

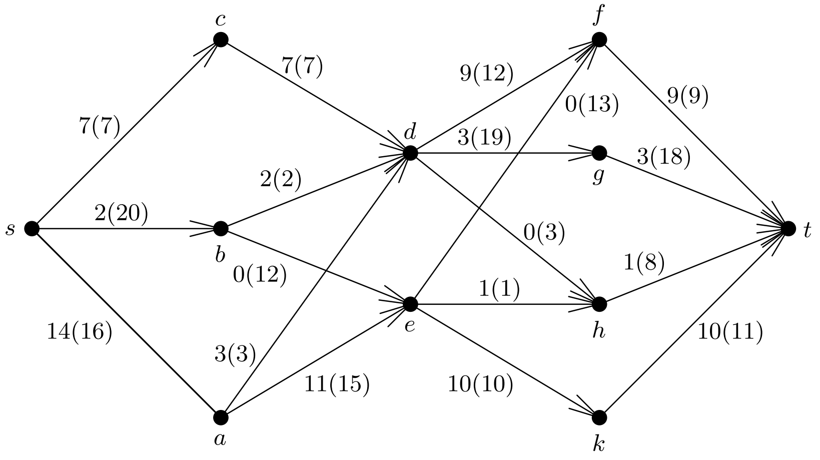


Fig. B.26 Solution to Exercise 6.4.5

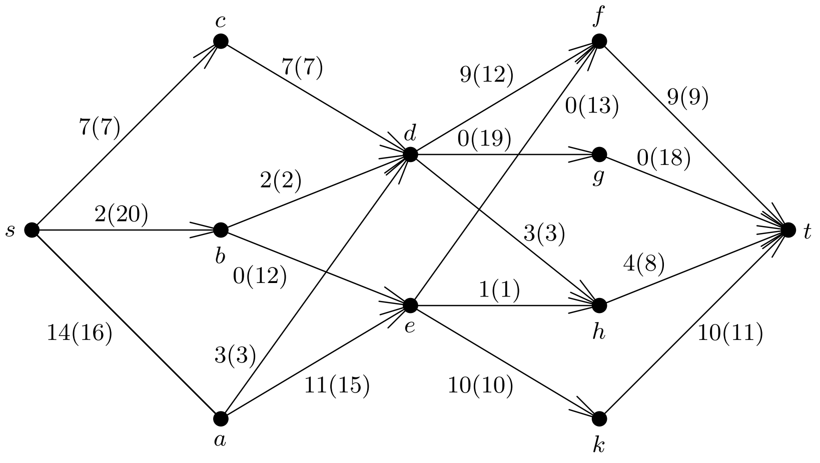


Fig. B.27 Solution to Exercise 6.4.10

6.5.6 Define a bipartite graph G on $S \dot\cup T$, where $S = \{1, \dots, m\}$ and $T = \{1', \dots, n'\}$, and let $\{i, j'\}$ be an edge if and only if girl i and boy j' know each other. Then the desired arrangement for a dance obviously corresponds to a matching of maximal cardinality in G; a solution can be determined using Example 6.5.5.

6.5.7 Let $A \subset S$, and let $X \subset A$ be an independent subset of maximal cardinality of A, say $|X| = k$. Consider the network N constructed from G in Example 6.5.5. Remove all vertices of $S \setminus A$ together with all edges incident with them from N, and denote the resulting network by N_A. Moreover, let

M be a matching of G with $X = \{e^- : e \in M\}$. As we saw in Example 6.5.5, M induces a flow of value k on N_A.

Now let Y be a maximal independent subset of A, say $Y = \{e^- : e \in M'\}$ for some matching M'; by hypothesis, $|Y| \leq k$. Suppose $|Y| < k$. Then the flow on N_A corresponding to M' cannot be maximal, and a maximal flow f can be obtained by constructing $k - |Y|$ augmenting paths in N_A. It is easy to see that there always is a matching corresponding to an independent subset of A containing Y (for each change of the flow). Thus Y cannot have been maximal either, a contradiction. Hence any two maximal independent subsets of A have the same cardinality k, so that $(S, \mathbf{S})$ satisfies condition (3) of Theorem 5.2.1 and therefore is a matroid.

Such matroids are called *transversal matroids*; they are considered in Sect. 7.3. We have given an algorithmic proof for the fact that $(S, \mathbf{S})$ is a matroid, by showing in a constructive way that condition (3) of Theorem 5.2.1 is satisfied. In a similar manner, the validity of condition (3) can be proved also—in the language of transversal theory—by using the algorithm of [Hal56]; see Sect. 7.3.

6.6.13 As there are only $n-2$ vertices distinct from s and t, at least one of the numbers i in the range $1 \leq i \leq n-1$ does not occur as a label $d(v)$. Choose such an i, and consider $S = \{v \in V : d(v) > i\}$ and $T = \{w \in V : d(w) < i\}$. Note $s \in S$ and $t \in T$. Our selection of i implies $d(v) \geq d(w) + 2$ for all choices of $v \in S$ and $w \in T$. Thus no edge $e = vw$ with $v \in S$ and $w \in T$ can belong to the residual graph G_f, since it violates the condition $d(v) \leq d(w) + 1$. As explained at the beginning of Sect. 6.6, G_f corresponds to the auxiliary network $N'(f)$ used in the classical algorithms. Using similar arguments, it is easily seen that the fact that no edge $e = vw$ with $v \in S$ and $w \in T$ belongs to G_f translates into the statement that each edge e with $e^- \in S$ and $e^+ \in T$ is saturated, whereas each edge e with $e^- \in T$ and $e^+ \in S$ is void. Now Lemma 6.1.2 shows that (S, T) is a minimal cut.

6.6.19 The algorithm FIFOFLOW determines (after nine phases) the maximal flow shown in Fig. 6.12. It needs 14 RELABEL and 23 PUSH operations, that is three more RELABEL and five more PUSH operations than HLFLOW.

B.7 Solutions for Chap. 7

7.1.3 We use the algorithm of Edmonds and Karp. Suppose that there exists an augmenting path containing a backward edge, and let P be the first such path and $e = uv$ be the last backward edge in P. When P is constructed, we must have $f(e) \neq 0$. Let Q be the last augmenting path constructed before

P for which $f(e)$ was changed (and actually increased). Then P and Q have the form

$$P: \quad s \overset{P'}{\text{———}} v \text{———} u \overset{P''}{\text{———}} t$$

and

$$Q: \quad s \overset{Q'}{\text{———}} u \text{———} v \overset{Q''}{\text{———}} t.$$

Denote the capacities of P and Q by γ and δ, respectively. Suppose first that $\gamma \le \delta$. Then we may replace Q and P by the following three paths:

$$s \overset{Q'}{\text{———}} u \text{———} v \overset{Q''}{\text{———}} t \qquad \text{(with capacity } \delta - \gamma\text{);}$$

$$s \overset{Q'}{\text{———}} u \overset{P''}{\text{———}} t \qquad \text{(with capacity } \gamma\text{);}$$

$$s \overset{P'}{\text{———}} v \overset{Q''}{\text{———}} t \qquad \text{(with capacity } \gamma\text{).}$$

Then P'', Q', and Q'' contain only forward edges, and the sum of the capacities of these three paths is $\gamma + \delta$, so that we have removed the backward edge e from P.

For $\gamma > \delta$, we use similar arguments to replace P and Q by three paths whose capacities sum to $\gamma + \delta$. However, the backward edge e is not removed in this case, since we need the path P with capacity $\gamma - \delta$. Nevertheless, the capacity of P is decreased, so that this method has to terminate. As the algorithm of Edmonds and Karp is finite, we get a finite method for constructing a maximal flow which uses only augmenting paths consisting exclusively of forward edges.

In Example 6.2.3, the only backward edge occurs in the last augmenting path, which has capacity 1 (see Fig. 6.11):

$$P: \quad s \text{ — } a \text{ — } d \text{ — } e \text{ — } c \text{ — } f \text{ — } t;$$

the backward edge is ce. Here Q is the following augmenting path:

$$Q: \quad s \text{ — } a \text{ — } c \text{ — } e \text{ — } t,$$

which has capacity 7; see Fig. 6.6. As described above, we may replace P and Q by the following three augmenting paths:

- s — a — c — e — t (with capacity 6);
- s — a — c — f — t (with capacity 1);
- s — a — d — e — t (with capacity 1).

7.1.7 Clearly, the proposed criterion is sufficient. Now let G be k-connected. By Menger's theorem, any two non-adjacent vertices of G are connected by k vertex disjoint paths. It remains to consider adjacent vertices s and t. Let H be the graph obtained by removing the edge st from G. Obviously, H is at least $(k-1)$-connected. Again by Menger's theorem, s and t are connected in H by $k-1$ vertex disjoint paths. Then st is the k-th path from s to t in G.

7.1.8 By Exercise 7.1.7, any two vertices of a k-connected graph are connected by k vertex disjoint paths, so that every vertex must have degree at least k. On the other hand, Exercise 1.5.14 shows that a planar graph has to contain vertices of degree at most 5. This proves the first assertion.

The graph with six vertices shown in Fig. B.5 is 4-connected. If G is 5-connected, every vertex must have degree at least 5. As in the solution to Exercise 1.5.14, we get the following bound on the number n_5 of vertices of degree at most (and hence equal to) 5:

$$6(n-n_5)+5n_5 \leq 12n-6;$$

thus $n \geq n_5 \geq 12$. The icosahedral graph provides an example with twelve vertices; see Fig. 9.1.

7.1.9 Let G be the given graph, and s and t the specified vertices. Consider the graph H whose vertices are the edges of G together with s and t and define adjacency as follows: two edges of G are adjacent in H if and only if they share a common vertex $v \neq s,t$ in G; any edge e of the form sv is adjacent to s; and any edge e of the form vt is adjacent to t. Note that s and t are not adjacent in H.

It is clear that edge disjoint paths from s to t in G are transformed into vertex disjoint paths in H by the preceding construction. In the converse direction, one has to be a bit careful, as a path in H corresponds to a trail in G, but not necessarily to a path. However, this difficulty can be overcome by appealing to Exercise 1.2.1, which guarantees that we may select a path contained in a given trail.

Finally, it is again clear that edge separators for s and t in G correspond to vertex separators for s and t in H. Hence the undirected case of Theorem 7.1.1 indeed reduces to the undirected case of Theorem 7.1.4. Note that the same approach also works in the directed case; here one needs to use Exercise 1.6.6 instead of Exercise 1.2.1.

7.1.10 Add two vertices s and t and all edges sx for $x \in S$ as well as all edges yt for $y \in T$ to G. Then the assertion follows from Theorem 7.1.4.

7.2.2 An unextendable matching M' has at least $k/2$ edges: otherwise, at least one of the k edges of a maximal matching M could be added to M'. It is easy to construct examples which show that this bound is best possible.

7.2.4 As explained in Example 6.5.5, we may run the labelling algorithm (or, more efficiently, Dinic' algorithm) to determine a maximal 0-1-flow f and hence a maximal matching M. Let us denote the associated minimal cut by (X,Y), where X consists of all vertices which are accessible from s on an augmenting path. We write S_X for $S \cap X$, and S_Y for $S \cap Y$; the analogous subsets of T will be denoted by T_X and T_Y.

We claim that $W = S_Y \cup T_X$ is a minimum cardinality vertex cover for G. The only edges which might not be covered by W are the edges of the form vw with $v \in S_X$ and $w \in T_Y$. If such an edge vw does not belong to M, it does not carry any flow, and hence could be used to extend an augmenting path from s to v (which exists, as $v \in S_X$) on to w, contradicting $w \in T_Y$. It remains to consider the case where $vw \in M$. But then the edge sv is saturated, so that v can only be reached via an augmenting path from s whose final edge is vw, used as a backward edge; again, this gives the contradiction $w \in T_X$. Thus W is indeed a vertex cover.

Finally, we show that W and M have the same cardinality. By the max-flow min-cut theorem, it suffices to check $|W| = c(X,Y)$, since $|M|$ equals the value of the maximal flow f. As all capacities are 1, we simply need to count the edges in the cocycle $C = E(X,Y)$. Trivially, an edge sv belongs to C if and only if $v \notin S_X$, giving $|S_Y|$ edges in C with start vertex s. Similarly, we get $|T_X|$ edges in C with start vertex in T_X (and end vertex t). Altogether, we now already have $|W|$ edges in C, so that C should not contain any further edges; indeed, any edge vw with start vertex $v \in S_X$ necessarily has $w \in T_X$, as we have already seen when proving that W is a vertex cover.

7.2.8 The Petersen graph (see Fig. 1.12) is 3-regular, but does not have a 1-factorization. Assume otherwise. Then at least one of the three 1-factors involved, say M, has to contain two edges of the outer cycle, say the two edges drawn as dashed lines in Fig. B.28. But this already determines M uniquely: for instance, the fifth point of the outer circle forces M to contain the spoke edge through that point. Hence M is the 1-factor consisting of the five dashed edges in Fig. B.28. But the complement of M is the union of two vertex disjoint 5-cycles, and thus cannot split into two 1-factors. (This argument is taken from [Vol04].)

7.2.10 Let us choose the disjoint union of the three $2n$-sets $R = \{r_1, \dots, r_{2n}\}$, $S = \{s_1, \dots, s_{2n}\}$, and $T = \{t_1, \dots, t_{2n}\}$ as the vertex set of K_{6n}. Moreover denote the complete bipartite graph on $S \mathbin{\dot\cup} T$ by K_{ST}, and the 1-factor $\{s_i t_i : i = 1, \dots, 2n\}$ of K_{ST} by F_{ST}.

By Corollary 7.2.7 and Exercise 1.1.2, both $G_{ST} = K_{ST} \setminus F_{ST}$ and the complete graph K_R on R can be decomposed into $2n-1$ 1-factors. By choosing an arbitrary bijection between these two sets of 1-factors and by merging all the corresponding factors, we obtain $2n-1$ 1-factors of K_{6n}; altogether, these factors contain precisely all the edges of one of the types $s_i t_j$ and $r_i r_j$ (for $i \neq j$).

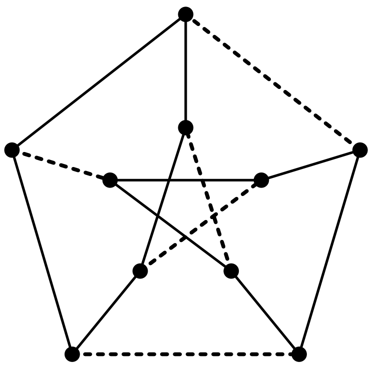


Fig. B.28 A 1-factor of the Petersen graph

The same method yields (for the two cyclic permutations of the sets R, S, and T) $4n-2$ further 1-factors of K_{6n}. The remaining edges which do not occur in one of these $6n-3$ 1-factors are of the form r_is_i, r_it_i, and s_it_i (for $i=1,\dots,2n$); obviously, these edges form a Δ-factor.

7.2.11 Denote the nine vertices by ij, where $i,j=0,1,2$. Then the edges where i is constant form three triangles which yield a first Δ-factor; similarly, we obtain a second Δ-factor for constant j; then the remaining two Δ-factors are uniquely determined. This unique decomposition of K_9 into Δ-factors is—using geometric terminology—just the affine plane of order 3; see, for instance, [BetJL99].

7.2.12 Choose $2n-1$ factors of a 1-factorization of K_{6n-2} (see Exercise 1.1.2) and denote the graph formed by these factors by G. Then $\overline{G}$ is regular with degree $(4n-2)$ and, hence, can be decomposed into 2-factors by Theorem 7.2.9. Now choose a bijection between these two sets of $2n-1$ factors and merge corresponding factors.

7.3.2 The assertion is clear for $n=1$. Thus let $n>1$. Choose $x_1 \in A_1$ and put

$$\mathbf{B} = (B_2,\dots,B_n) \quad \text{with } B_i = A_i \setminus \{x_1\}.$$

Assume first that $\mathbf{A}$ does not contain a critical subfamily. Then the union of any k sets in $\mathbf{A}$ contains at least $k+1$ elements; thus $\mathbf{B}$ clearly satisfies (H$'$). Hence $\mathbf{B}$ contains a transversal T, so that $T \cup \{x_1\}$ is a transversal of $\mathbf{A}$.

It remains to consider the case where $\mathbf{A}$ contains a critical subfamily, say $\mathbf{A}' = (A_1,\dots,A_m)$. By the induction hypothesis, $\mathbf{A}'$ contains a transversal T'. Put $\mathbf{C} = (C_{m+1},\dots,C_n)$, where $C_i = A_i \setminus T'$. Now one checks that $\mathbf{C}$ likewise satisfies condition (H$'$), so that $\mathbf{C}$ has a transversal T''. Then $T' \cup T''$ is a transversal of $\mathbf{A}$.

7.3.3 It is obvious that the maximal cardinality of a matching of G cannot exceed the minimal cardinality of a vertex cover of G. Now suppose that

$X = S' \cup T'$ (where $S' \subset S$ and $T' \subset T$) is a minimal vertex cover. We will apply Theorem 7.3.1 in the terminology used in Theorem 7.2.5.

Consider the bipartite graph G' induced on the set $(S \setminus S') \dot{\cup} T'$. We want to show that G' satisfies condition (H). Suppose otherwise. Then there exists a subset J of T' with $|\Gamma(J)| < |J|$, so that the set $S' \cup \Gamma(J) \cup (T' \setminus J)$ is a vertex cover for G which has smaller cardinality than $|X|$. This contradicts our assumption above and proves that G' satisfies (H). By Theorem 7.2.5, G' has a matching of cardinality $|T'|$.

Similarly, the bipartite graph G'' induced on the set $S \cup (T \setminus T')$ contains a matching of cardinality $|S'|$. Then the union of these two matchings of G' and G'' forms a matching of cardinality $|X|$ of G.

7.3.6 The maximal cardinality of a matching in a bipartite graph (with vertex set $S \dot{\cup} T$) is $|T| - \max\{|J| - |\Gamma(J)| : J \subset T\}$.

7.3.10 Consider the family $\mathbf{A}$ which consists of d_i copies of A_i for $i = 1, \dots, k$. Then $\mathbf{S}$ is precisely the set of partial transversals of $\mathbf{A}$, so that the assertion follows from Theorem 7.3.8.

7.3.13 Trivially, (1) follows from (2). So suppose that (1) holds. Write $m = |S|$ and assume $\mathbf{A}' = (A_1, \dots, A_k)$. Let D be an arbitrary set of cardinality n which is disjoint to S, and consider the family $\mathbf{B}$ consisting of the sets

$$A_1, \dots, A_k, A_{k+1} \cup D, \dots, A_n \cup D \text{ and } m \text{ times the set } \big(S \setminus S'\big) \cup D.$$

Now suppose that $\mathbf{B}$ has a transversal. As $\mathbf{B}$ consists of $m+n$ subsets of the set $S \cup D$ having $m+n$ elements, this transversal has to be $S \cup D$ itself. Thus, S is a transversal of a subfamily of $\mathbf{B}$ which contains all the sets $A_1, \dots, A_k$, some of the sets $A_{k+1} \cup D, \dots, A_n \cup D$, and some copies of $(S \setminus S') \cup D$. If we delete all those elements representing copies of $(S \setminus S') \cup D$ from S, we obtain a subset S'' of S which contains S' and is a transversal for a subfamily of $\mathbf{A}$ containing $\mathbf{A}'$.

It remains to show that the family $\mathbf{B}$ defined above satisfies condition (H′) of the marriage theorem. This condition is

$$\left| \left(\bigcup_{j \in J} A_j \right) \cup \left(\bigcup_{j \in K} A_j \cup D \right) \cup \left(\bigcup_{i=1}^{c} (S \setminus S') \cup D \right) \right| \geq |J| + |K| + c \qquad \text{(B.1)}$$

for all $J \subset \{1, \dots, k\}$, $K \subset \{k+1, \dots, n\}$ and $c \in \{0, \dots, m\}$. First consider the case $c = 0$. If $K = \emptyset$, (B.1) follows from condition (H′) for $\mathbf{A}'$, which holds as $\mathbf{A}'$ has a transversal. If $K \neq \emptyset$, the union on the left hand side contains the n-set D, so that (B.1) is satisfied because of $n \geq |J| + |K|$. Now let $c \neq 0$; it suffices to consider the case $c = m$. As D and S are disjoint, (B.1) becomes

$$\left| \bigcup_{j \in J} A_j \cup \big(S \setminus S'\big) \right| \geq |J| + m - n \quad \text{for } J \subset \{1, \dots, n\}. \qquad \text{(B.2)}$$

But

$$\left|\bigcup_{j\in J} A_j \cup (S \setminus S')\right| = m - |S'| + \left|\left(\bigcup_{j\in J} A_j\right) \cap S'\right|,$$

so that (B.2) is equivalent to

$$\left|\left(\bigcup_{j\in J} A_j\right) \cap S'\right| \geq |J| + |S'| - n.$$

This condition holds by Theorem 7.3.7, as S' is a partial transversal of $\mathbf{A}$.

7.3.14 Let G be the bipartite graph with vertex set $S \,\dot\cup\, T$ corresponding to $\mathbf{A}$. As in Exercise 6.5.7, one sees that there is also a matroid induced on T. Using the terminology of set families, the independent sets of this matroid are precisely those subsets of the index set T for which the corresponding subfamily of $\mathbf{A}$ has a transversal.

7.3.18 Let $\mathbf{B}$ be the family consisting of p_i copies of A_i for $i = 1, \dots, n$. Then the existence of sets X_i with the desired properties is equivalent to the existence of a transversal of $\mathbf{B}$. Now condition (H$'$) for $\mathbf{B}$ is precisely the condition given in the exercise, so that the assertion follows from the marriage theorem.

7.4.13 The assertions of Corollaries 7.4.6 and 7.2.7 are equivalent.

7.4.14 Let D be a diagonal with entries $d_1, \dots, d_n$ satisfying $d_1 \dots d_n \geq n^{-n}$. The inequality between the arithmetic and the geometric mean[3] implies

$$(d_1 \dots d_n)^{1/n} \leq \frac{d_1 + \dots + d_n}{n},$$

so that $d_1 + \dots + d_n \geq 1$.

7.4.15 Let $\mathbf{T}$ be the set family as described in the hint. Then $\mathbf{T}$ satisfies condition (H$'$), since the ktr entries 1 in any given k rows of A have to be contained in at least kt columns of A (note that A has column sums $\leq r$). Therefore $\mathbf{T}$ has a transversal, so that there exist pairwise disjoint t-subsets S_i of T_i for $i = 1, \dots, m$. Then the matrix P with entries $p_{ij} = 1$ for $i \in S_j$ and $p_{ij} = 0$ otherwise has row sums t and column sums ≤ 1. Moreover, the matrix $\mathbf{A}' = A - P$ has row sums $t(r-1)$.

[3]For a proof of the inequality mentioned above and of a more general inequality due to Muirhead [Mui03] using the methods of transversal theory, we refer the reader to [Mir71b, Theorem 4.3.3].

As we want to use induction on r, we still have to make sure that the set X of all those indices for which column j of A has sum r is contained in $S_1 \cup \dots \cup S_m$ (so that $\mathbf{A}'$ has column sums $\leq r-1$). By Corollary 7.3.9, it is sufficient to show that X is a partial transversal of $\mathbf{T}$. However, any k columns having sum r together contain precisely kr entries 1, and these entries have to be contained in at least k/t rows of A. As each T_i occurs precisely t times in $\mathbf{T}$, any k elements of X correspond to at least k sets in $\mathbf{T}$. Now Theorem 7.2.5 implies that X is a partial transversal.

7.4.16 Using the equivalence of 0-1-matrices and bipartite graphs discussed at the beginning of Sect. 7.4, the assertion amounts to showing that a bipartite graph of maximal degree r can be decomposed into r matchings. Let $S \mathbin{\dot\cup} T$ be the vertex set of G, and denote the set of vertices of degree r in S and T by S' and T', respectively. By Theorem 7.2.5, there exist matchings M' and M'' of G which meet S' and T', respectively. By Corollary 7.3.12, there also exists a matching M meeting $S' \cup T'$. Then $G \setminus M$ has maximal degree $r-1$, and the assertion follows by induction.

7.4.17 We may assume $n \geq 3$. We show first that the subspace W of $\mathbb{R}^{(n,n)}$ spanned by the permutation matrices consists precisely of those matrices for which all row and column sums are equal. Obviously, any linear combination of permutation matrices is contained in W and has constant row and column sum. Conversely, let A be a matrix with constant row and column sum. If A does not contain any negative entries, A is contained in W by Theorem 7.4.7. Otherwise, put $b = \max\{-a_{ij} : i,j = 1,\dots,n\}$. Then the matrix $B = A + bJ$ (where J is the matrix with all entries 1) has nonnegative entries and constant row and column sum. Therefore J and B (and A as well) are linear combinations of permutation matrices.

Now let W' be the subspace spanned by the $2n-2$ matrices S_i and Z_i (for $i = 1,\dots,n-1$) which have entry 1 in cell (n,i) and in cell (i,n), respectively, and all other entries 0. Obviously, W and W' have only the zero matrix in common. Thus $\dim W = n^2 - 2n + 2$ follows if we can show that W and W' together generate $\mathbb{R}^{(n,n)}$. Let A be an arbitrary matrix in $\mathbb{R}^{(n,n)}$. By adding appropriate multiples of S_i or of Z_i to A, we can obtain a matrix C for which the first $n-1$ rows and the first $n-1$ columns have a fixed sum s. Then the last row and the last column of C must have identical sum, say x. Adding aS_i and aZ_i to C, the sum s can be changed to $s' = s + a$; simultaneously, x is changed to $x' = x + (n-1)a$. As $n \neq 2$, we can determine a so that $x' = s'$; that is, the resulting matrix C' has constant row and column sum. Thus C' is contained in W, so that A is contained in $W + W'$.

7.5.4 Suppose G is a minimal counterexample to the assertion, and let $\mathbf{D}$ be a dissection of G consisting of as few paths as possible. Then $\mathbf{D}$ contains at least $\alpha + 1$ paths. Suppose we have $|\mathbf{D}| \geq \alpha + 2$. We omit a path W from $\mathbf{D}$. As G is minimal, $G \setminus W$ has a dissection into at most α paths, say $\mathbf{D}'$.

But then $\mathbf{D}' \cup \{W\}$ is a dissection of G into $\alpha+1$ paths contradicting our assumption.

Hence $|\mathbf{D}| = \alpha + 1$, say $\mathbf{D} = \{W_1, \dots, W_{\alpha+1}\}$. Denote the start vertex of W_i by p_i. By definition of α, the $\alpha+1$ vertices p_i cannot form an independent set; we may assume that p_1p_2 is an edge. If W_1 consists of p_1 only, we may omit W_1 and replace W_2 by $(p_1p_2)W_2$, so that G would be decomposable into α paths. Thus W_1 cannot be trivial.

Let W_1' be the path obtained by omitting the first edge p_1p_1' from W_1. As G is a minimal counterexample, the graph $H = G \setminus p_1$ satisfies the assertion. Now $\{W_1', W_2, \dots, W_{\alpha+1}\}$ is a dissection of H, so that we can find a dissection $\{Z_1, \dots, Z_k\}$ of H into $k \le \alpha$ paths such that the start vertices of these paths are contained in $\{p_1', p_2, \dots, p_{\alpha+1}\}$.

If p_1' is the start vertex of one of the paths Z_i, Z_i can be replaced by $(p_1p_1')Z_i$, which yields a dissection of G into at most α paths. If $k < \alpha$, we may add the trivial path $\{p_1'\}$ to the Z_i. If neither of these two conditions holds, we must have $k = \alpha$, and the start vertices of the Z_i are precisely the vertices $p_2, \dots, p_{\alpha+1}$. Thus p_2 is the start vertex of some Z_h. Replacing Z_h by $(p_1p_2)Z_h$ again yields a dissection of G into at most α paths. Therefore G cannot be a counterexample, and the assertion holds in general.

7.5.5 As a tournament is an orientation of a complete graph, the maximal independent sets have only one element in this case. Thus the assertion follows immediately from Exercise 7.5.4.

Let us also give a very easy direct proof (not using Exercise 7.5.4). Choose a directed path of maximal length in G, say $W\colon\ v_1 — v_2 — \cdots — v_k$. Suppose that W is not a Hamiltonian path; then there exists a vertex v not on W. As W is maximal, G contains neither an edge vv_1 nor an edge v_kv, so that G has to contain the edges v_1v and vv_k. Hence there must be some index i $(1 < i < k)$ such that G contains the edges v_iv and vv_{i+1}. Then we can replace the edge v_iv_{i+1} in W by these two edges, so that W is not maximal, a contradiction.

7.5.9 Let k be the maximal cardinality of a chain in M. Moreover, let A denote the antichain of the maximal elements of M. Then the maximal cardinality of a chain in $M \setminus A$ is $k-1$, and the assertion follows by induction.

7.5.10 Let $\mathbf{A} = (A_1, \dots, A_n)$ be a family of subsets of $\{x_1, \dots, x_m\}$ satisfying (H$'$). We define a partial ordering on $M = \{x_1, \dots, x_m, A_1, \dots, A_n\}$ by

$$u \prec v \quad \Longleftrightarrow \quad u = x_i, \qquad v = A_j \quad \text{and} \quad x_i \in A_j \quad \text{(for suitable } i, j).$$

Let $\{x_1, \dots, x_h, A_1, \dots, A_k\}$ be an antichain of maximal cardinality $s = h + k$. Then $k \le |A_1 \cup \dots \cup A_k| \le m - h$, so that $s = h + k \le m$. By Dilworth's theorem, $(M, \preceq)$ can be decomposed into s chains, say (after renumbering)

$$\{x_1, A_1\}, \dots, \{x_i, A_i\}, \quad \{A_{i+1}\}, \dots, \{A_n\}, \quad \{x_{i+1}\}, \dots, \{x_m\}.$$

Then $s = m + n - i$, and hence $n = s - m + i \leq i$; this forces $n = i$, so that $\{x_1, \dots, x_n\}$ is a transversal of $\mathbf{A}$.

7.7.3 Use Theorem 7.7.4.

7.7.2 We have derived Theorem 7.7.1 from Theorem 6.1.6 by constructing an appropriate flow network N. If c, a, and d are integral, the capacity function of N is likewise integral. Thus Theorem 6.1.5 implies that there exists an integral solution (provided that there are feasible flows).

B.8 Solutions for Chap. 8

8.1.2 Note that each vertex has to have degree at least k if G is k-connected.

8.1.3 Add a new vertex t and all edges xt with $x \in T$ to G. It is easy to show that the resulting graph H is again k-connected: clearly, there is no vertex separator for H consisting of $k - 1$ vertices. By Theorem 8.1.1, there are k vertex disjoint paths from s to t; these paths have to contain all the k edges xt with $x \in T$. Deleting these edges, we obtain the desired paths in G.

8.1.6 The graph $K_{m,m+1}$ has connectivity $\kappa = m$ and independence number $\alpha = m + 1$. It cannot be Hamiltonian, since a Hamiltonian cycle would have length $2m + 1$; by Theorem 3.3.5, bipartite graphs do not contain cycles of odd length.

8.1.7 Using the procedure BLOCK01FLOW of Lemma 6.5.2, we can determine a maximal 0-1-flow as follows (by analogy with Algorithm 6.3.17). Here G is a digraph with two special vertices s and t, and *val* denotes the value of a maximal flow.

Procedure MAX01FLOW($G, s, t; f$, val)

```
(1) for $e \in E$ do $c(e) \leftarrow 1$; $f(e) \leftarrow 0$ od
(2) val $\leftarrow 0$; $N \leftarrow (G, c, s, t)$;
(3) repeat
(4)     AUXNET $(N, f; N''$, max, $d)$;
(5)     if max = false then BLOCK01FLOW($N''; g$); AUGMENT $(f, g; f)$ fi
(6) until max = true;
(7) for $e \in A_s$ do
(8)     if $f(e) = 1$ then val $\leftarrow$ val +1 fi
(9) od
```

The proofs of Theorems 7.1.1 and 7.1.4 imply that the maximal number of vertex disjoint paths from s to t in G equals the maximal value of a 0-1-flow on the 0-1-network with underlying digraph H defined during the following procedure.

Procedure PATHNR$(G, s, t; k)$

```
(1) $V' \leftarrow \{s,t\}$; $E' \leftarrow \emptyset$;
(2) for $v \in V \setminus \{s,t\}$ do $V' \leftarrow V' \cup \{v', v''\}$; $E' \leftarrow E' \cup \{v'v''\}$ od
(3) for $e \in E$ do
(4)     if $e = sv$ with $v \neq t$ then $E' \leftarrow E' \cup \{sv'\}$ fi
(5)     if $e = tv$ with $v \neq s$ then $E' \leftarrow E' \cup \{v''t\}$ fi
(6)     if $e = uv$ with $u, v \neq s, t$ then $E' \leftarrow E' \cup \{u''v', v''u'\}$ fi
(7) od
(8) $H \leftarrow (V', E')$; MAX01FLOW $(H, s, t; f, \text{val})$;
(9) if $st \in E$ then $k \leftarrow \text{val} + 1$ else $k \leftarrow \text{val}$ fi
```

Theorems 7.1.1 and 7.1.4 show that this procedure is correct; note that s and t are not adjacent in H. If s and t should be adjacent in G, we have to add one further path from s to t, namely the edge st itself. By Corollary 7.1.5, PATHNR has complexity $O(|V|^{1/2}|E|)$. Finally, if G is an undirected graph, we can replace G by its complete orientation (as in the proof of Theorem 7.1.1).

8.2.6 Define a graph G which has a vertex for each junction of the maze, where also the entrance, the exit, and dead ends are viewed as junctions. The edges of G correspond to those paths in the maze which connect two consecutive junctions: the end vertices of an edge are the respective junctions. Figures B.29 and B.30 show the graph G which corresponds to the maze given in Fig. 8.3. The labels of the vertices in Fig. B.30 indicate one possible course for a DFS on G which starts at the entrance of the maze (which is represented by the vertex labelled 1); the algorithm terminates when the exit is reached (that is, at the vertex labelled 64). The corresponding path through the maze is drawn in Fig. B.31; for the sake of simplicity, we have not included dead ends occurring during the DFS (which have, of course, to be traversed and then necessitate corresponding backtracking).

Of course, when we designed the above solution, we had a bird's-eye view of the maze (and used this knowledge). However, it is not hard to find a rule which allows us to apply a DFS to a maze without knowing it in its entirety, provided that it is possible to label junctions and paths when we pass them. We leave it to the reader to formulate such a rule.[4]

[4] In this context, the following quotation from Umberto Eco's *The Name of the Rose* is of some interest; see [Eco83, p. 176]:

> At every new junction, never seen before, the path we have taken will be marked with three signs. If, because of previous signs on some of the paths of the junction, you see that the junction has already been visited, you will make only one mark on the path you have taken. If all the apertures of the junction are still without signs, you will choose any one, making two signs on it. Proceeding through an aperture that bears only one sign, you will make two more, so that now the aperture bears three. All the parts of the labyrinth must have been visited if, arriving at a junction,

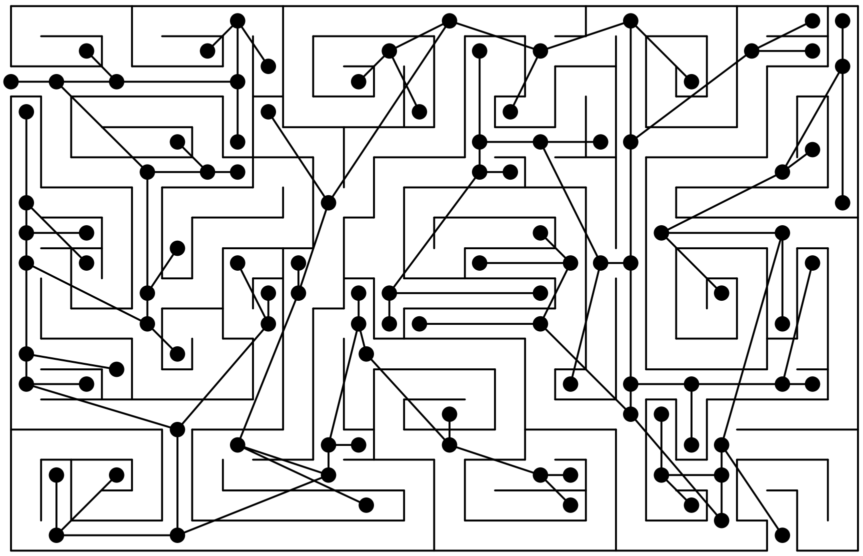


Fig. B.29 A maze with corresponding graph G

8.3.2 Consider two vertices u and v for which $d(u,v)$ is maximal. If v were a cut point, then $G \setminus v$ would consist of two components, so that we could choose a vertex w which is not contained in the component of u. Then every path from u to w would have to contain v, so that the distance from w to u would have to be at least $d(u,v)+1$, a contradiction. Therefore v and u cannot be cut points. On the other hand, a path of length n contains precisely $n-2$ cut points.

8.3.3 Suppose that $bc(G)$ contains a cycle $(B_1, c_1, B_2, c_2, \dots, B_k, c_k, B_1)$. Then we can remove c_k and still reach vertices in B_1 from vertices in B_k, a contradiction. This proves that $bc(G)$ is always acyclic. If G is connected, also $bc(G)$ is connected, so that $bc(G)$ is a tree. This proves (a).

For claim (b), we may assume that G is connected, so that $p = 1$. Then $bc(G)$ is a tree and, hence, contains precisely $b(G) + c(G) - 1$ edges. Each edge connects a cut point with a block, so that the number of edges equals the sum of all the $b(c)$ (over all cut points c). Therefore

$$b(G) + c(G) - 1 = \sum_c b(c) = \sum_c 1 + \sum_v \big(b(v) - 1\big) = c(G) + \sum_v \big(b(v) - 1\big),$$

you never take a passage with three signs, unless none of the other passages is now without signs.

This somewhat chaotic rule contains the basic idea of a depth first search, even though the hero of the tale, William of Baskerville (who admits that he just recites 'an ancient text I once read'), obviously confused the labelling rules a bit.

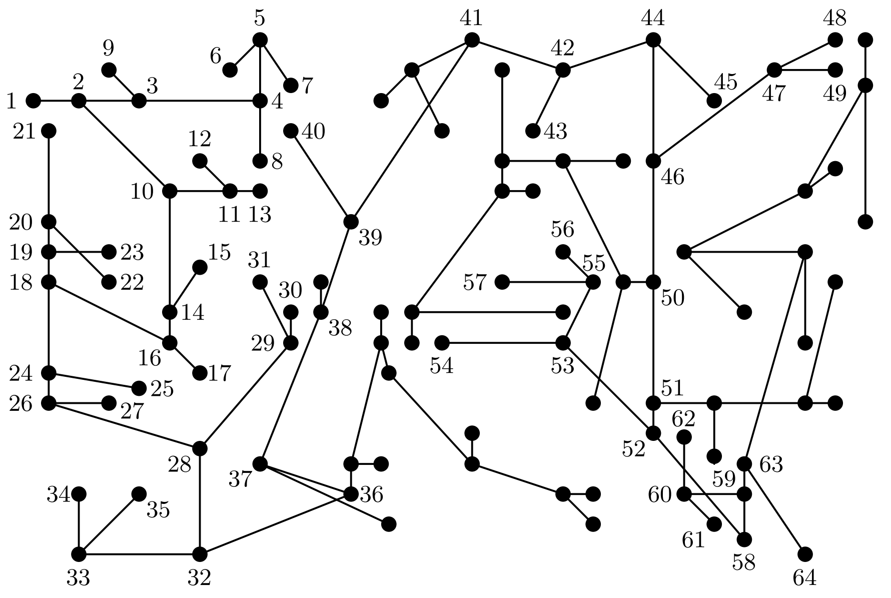


Fig. B.30 A partial DFS on G

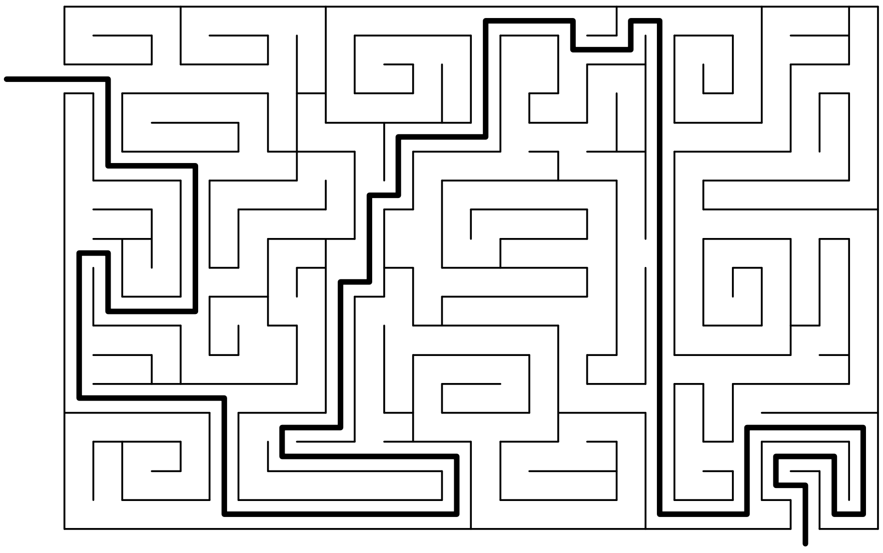


Fig. B.31 A path through the maze

since each vertex which is not a cut point is contained in precisely one block. Assertion (c) can be proved in a similar manner.

For (d), we use induction on the number $c(G)$ of cut points. The case $c(G) = 1$ is clear. Now assume $c(G) > 1$. Then $bc(G)$ contains a leaf, and every leaf B has to be a block; note that the unique edge incident with B has a cut point c as its other end vertex. Removing B from the graph G corresponds to removing c and B from $bc(G)$. Now the assertion follows by induction.

8.3.4 Let $b(G) = k$. We denote the cardinalities of the blocks by $n_1, \dots, n_k$ and the number of vertices of G by n. By Exercise 8.3.3 (b), $n_1 + \dots + n_k = k + n - 1$. By Exercise 8.3.3 (d), a graph with r cut points has to have at least $r + 1$ blocks; also, G will have the maximum possible number of edges if and only if each block is a complete graph on at least two vertices. Thus this number is given by

$$\max\left\{\sum_{i=1}^{k}\binom{n_i}{2} : n_1 + \dots + n_k = n + k - 1; n_1, \dots, n_k \geq 2; k \geq r + 1\right\}$$

$$= \max\left\{k - 1 + \binom{n + k - 1 - (2k - 2)}{2} : k \geq r + 1\right\} = \binom{n - r}{2} + r,$$

which is realized by a graph consisting of K_{n-r} with a path of length r appended.

8.3.10 We obtain the graph shown in Fig. B.32, where each vertex v is labelled with its DFS-number $nr(v)$ and with $L(v)$. Algorithm 8.3.8 yields the cut points i, e, s, and h in this order. The blocks are $\{k, j, i\}$; $\{i, e\}$; $\{e, f, b, a, s\}$; $\{l, h\}$; and $\{h, d, g, c, s\}$. The fat edges are the edges of the DFS tree, and cut points are indicated by a circle.

8.4.3 As u is reached later than v during the DFS, the examination of u has to take place during the examination of v.

8.4.4 If a back edge $e = vu$ occurs during the DFS, we obtain a directed cycle in G, as u is an ancestor of v. Conversely, suppose that G contains a directed cycle. Let v be the first vertex of G examined during the DFS which is contained in a directed cycle, and let $e = uv$ be an edge on such a cycle C. By our choice of v, u is examined later than v during the DFS, so that e is neither a forward edge nor a tree edge. As u is accessible from v (using C), u has to be a descendant of v. Thus e cannot be a cross edge either, so that e must be a back edge.

8.5.2 Choose G to be a directed cycle or the complete orientation of a path.

8.5.3 Let C and C' be two distinct strong components of G. As G is connected, there exists an edge e connecting a vertex in C and a vertex in C'.

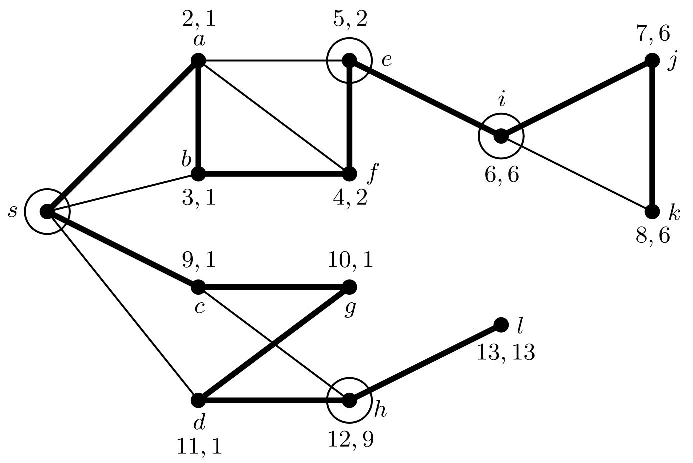
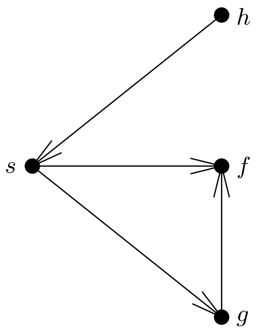


Fig. B.32 DFS-tree, blocks, and cut points

Fig. B.33 Condensed digraph for the digraph of Fig. 3.3



Then e cannot be contained in a directed cycle, because that would imply $C = C'$. Thus G has to be strongly connected provided that every edge of G is contained in a directed cycle. The converse holds by Theorem 8.5.1.

8.5.8 The vertices h, f, and g each form a strong component with only one element; the remaining vertices together form a further strong component.

8.5.9 Suppose the strong components $C_1, \dots, C_m$ are contained in a cycle of G'. Then there are edges $v_i v_i'$ with $v_i \in C_i$ and $v_i' \in C_{i+1}$ (where $m+1$ is interpreted as 1). As C_i contains a directed path from v_{i-1}' to v_i, we obtain a directed cycle, so that $C_1, \dots, C_m$ have to be contained in a common strong component, a contradiction. Therefore G' has to be acyclic. Figure B.33 shows G' for the digraph G of Fig. 3.3.

8.5.10 Define a digraph to be strongly k-connected if it is the complete orientation of K_{k+1}, or if each set S of vertices for which $G \setminus S$ is not strongly

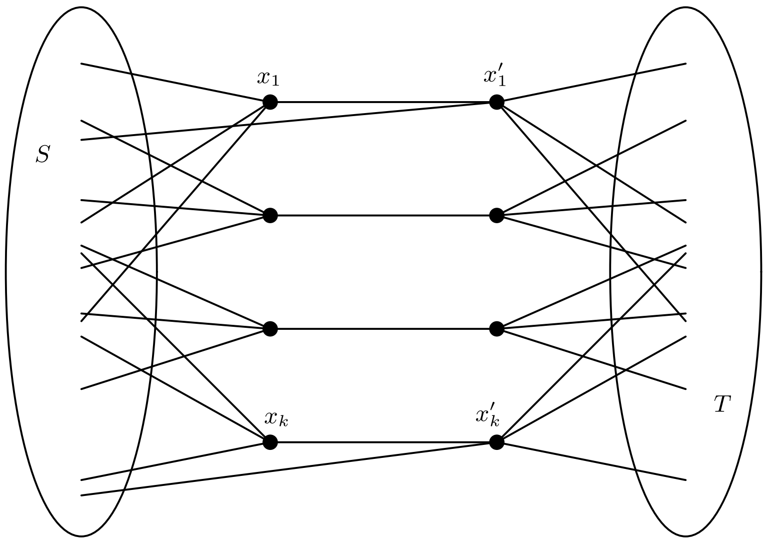


Fig. B.34 Solution to Exercise 8.6.2

connected contains at least k vertices. Then the analogues of Theorems 8.1.1 and 8.1.9 hold; in both cases, $\kappa(v_i, w)$ as well as $\kappa(w, v_i)$ have to be calculated.

8.6.2 For $k = m = d$, we can choose $G = K_{d+1}$. For $k \neq d$, we use two copies of the complete graph K_{d+1} on two disjoint vertex sets S and T, together with $2k$ further vertices $x_1, \dots, x_k, x_1', \dots, x_k'$ and all the edges $x_i x_i'$. Moreover, we connect each of the x_i to $d-1$ vertices in S, and each of the x_i' to $d-1$ vertices in T. Finally, we add $m-k$ further edges connecting the vertices in $S \cup \{x_2, \dots, x_k\}$ to some of the x_i'. See Fig. B.34.

8.6.3 Let E' be a minimal edge separator of G. Then $G \setminus E'$ has two connected components S and T, and E' is the cocycle determined by the cut (S, T). We may assume $x = |S| \leq n/2$. Then E' has to contain at least $x\delta - x(x-1) = x(\delta - x + 1)$ edges. It is easy to check $x(\delta - x + 1) \geq \delta$ if $\delta \geq n/2$ (for $x = 1, \dots, n/2$). The graph consisting of two disjoint copies of K_d (connected by at most $d-1$ edges) shows that nothing can be said for the case $\delta < n/2$.

B.9 Solutions for Chap. 9

9.1.2 If we want to color the icosahedral graph of Fig. 9.1, the three vertices of the outer triangle have to get different colors. As any two vertices of this triangle have a further common neighbor, the colors for these three neighbors are now forced (assuming that it is possible to use only three colors);

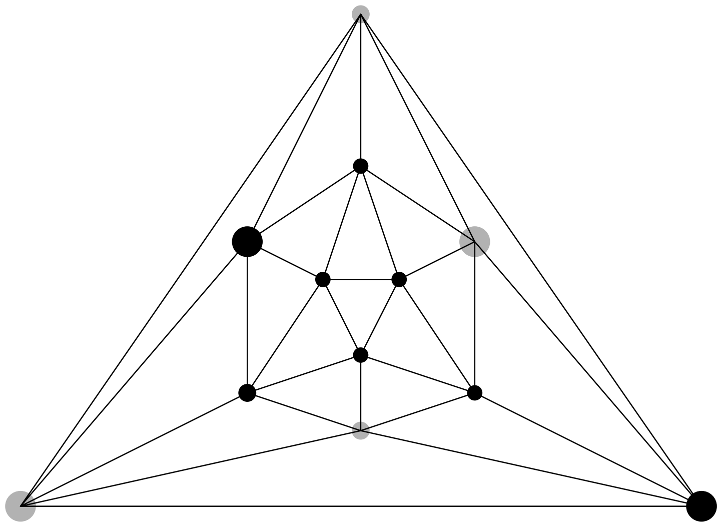


Fig. B.35 A partial 3-coloring of the icosahedral graph

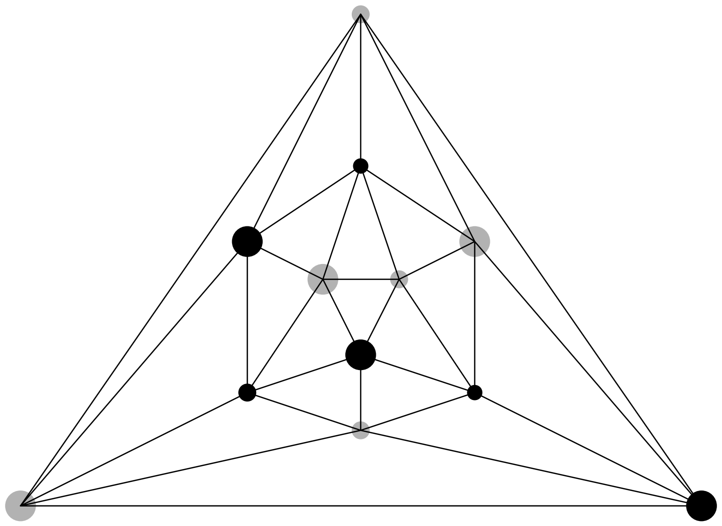


Fig. B.36 A 4-coloring of the icosahedral graph

see Fig. B.35, where the three colors used are indicated by small gray circles, big gray circles, and big black circles. But now there are vertices of degree 5 for which three neighbors already use up all three colors, so that the coloring cannot be completed. If we allow a fourth color, the partial coloring of Fig. B.35 can be completed; see Fig. B.36.

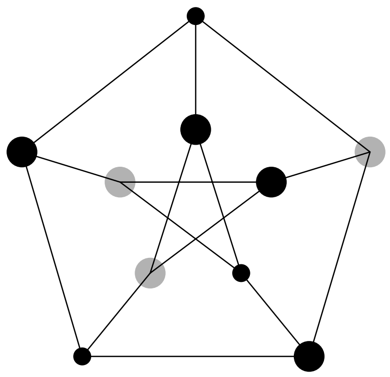


Fig. B.37 A 3-coloring of the Petersen graph

9.1.9 Note that the Petersen graph is 3-regular; hence $\chi(G) \le 3$, by Brook's theorem. Now $\chi(G) = 2$ is impossible, as the Petersen graph is not bipartite. Hence $\chi(G) = 3$; see Fig. B.37 for an explicit 3-coloring.

9.2.4 Let $(M_0, M_1, \dots, M_k = M_0)$ be a sequence of vertices defining a cycle of length $k \ge 4$ in an interval graph, where $M_i = (x_i, y_i)$ for $i = 0, \dots, k-1$. (The case of closed intervals is similar.) We may assume $x_0 < x_1$. If $M_2 M_0$ is an edge, we have found a chord of the cycle. Otherwise, $M_2 \cap M_0 = \emptyset$, $M_1 \cap M_0 \ne \emptyset$, and $M_2 \cap M_1 \ne \emptyset$ imply $x_1 < y_0 \le x_2 < y_1$. Thus, if the cycle does not have a chord, the lower bounds of the intervals M_i have to form a monotonically increasing sequence. But then $M_{k-1} M_0$ cannot be an edge, a contradiction. Hence G must be chordal.

9.2.9 As induced subgraphs of a bipartite graph are likewise bipartite, it suffices to prove $\alpha(G) = \theta(G)$ for every bipartite graph G. In the bipartite case, $\theta(G) = |V| - \alpha'(G)$, where $\alpha'(G)$ denotes the maximal cardinality of a matching. Moreover, $\alpha(G) = |V| - \beta(G)$; see Lemma 7.5.1. Therefore Theorem 7.2.3 yields $\alpha(G) = \theta(G)$. (Verifying $\chi(G) = \omega(G)$ is even easier: both parameters are 2 in the bipartite case; see Example 9.1.1.)

9.3.1 Clearly, $\chi'(G)$ is the minimal number of matchings into which G can be decomposed. Thus the assertion amounts to showing that a bipartite graph of maximal degree r can always be decomposed into r matchings, which was proved in the solution to Exercise 7.4.16.

9.3.4 By Exercise 7.2.8, the Petersen graph does not admit a 1-factorization, and hence Corollary 10.3.3 implies $\chi'(G) = 4$. An explicit 4-coloring may be obtained as follows: take the broken edges in Fig. B.28 as one color class. As noted in the solution to Exercise 7.2.8, the complement of M is the union of two vertex disjoint 5-cycles. Trivially, we may color the edges of these two cycles using three further colors.

9.4.6 Clearly, $G = G(H,S)$ is regular of degree k, where k is the cardinality of S. Given any two elements x and y of H, we have to determine the number of elements $z \in H$ which are adjacent to both x and y. As H acts regularly on G, we may assume $y = 1$. By definition, z is adjacent to both 1 and x if and only if $z^{-1}, xz^{-1} \in S$. If we put $d = z^{-1}$ and $c = xz^{-1}$, we may use (9.1) to re-write the preceding condition as

$$x = cd^{-1}, \qquad z = d^{-1} \quad \text{with } c, d \in S.$$

Hence the number of elements z of H which are adjacent to both 1 and x equals the number of *quotient representations* of x from S. Noting that x is adjacent to 1 if and only if $x \in S$, one sees that condition (2) in the assertion holds if and only if G is strongly regular with parameters λ and μ.

9.5.3 Denote the two parts of the bipartition of $K_{3,3}$ by S and T, and consider an arbitrary choice of color lists of cardinality ≥ 3 each. If two vertices in the same part, say in S, admit the same color c, we may color them with c, and color the third vertex in S in any admissible way. As this forbids at most two colors and as T is an independent set, we may certainly also color the vertices in the other part T correctly.

Hence we may assume that the color lists for the three vertices in a given part are all disjoint. Thus we have 9 colors available for S, and also for T. Now we just have to pick a color from each of the three lists for S in such a way that the resulting 3-set of colors does not coincide with one of the three color lists for T. Obviously, this is possible (indeed, in many ways).

B.10 Solutions for Chap. 10

10.1.6 Put $b(e) = c(e) = 1$ for each directed edge e of G, and replace each undirected edge $e = \{u, v\}$ by two directed edges $e' = uv$ and $e'' = vu$ with $b(e') = b(e'') = 0$ and $c(e') = c(e'') = 1$; this defines a directed multigraph H with capacity constraints b and c. Obviously, every Euler tour of G yields a feasible circulation on H.

Conversely, let f be a feasible circulation on H. Let e be an undirected edge of G. If either $f(e') = 1$, $f(e'') = 0$ or $f(e'') = 1$, $f(e') = 0$, we replace e by e' or by e'', respectively. Performing this operation for all undirected edges yields a mixed multigraph G' for which the number of directed edges with start vertex v always equals the number of directed edges with end vertex v. Then an Euler tour can be constructed using the methods of Chap. 1; cf. Theorems 1.3.1 and 1.6.1.

10.1.7 We introduce the following vertices:

- a source s and a sink t;
- a vertex 0 which represents the person selling the napkins;

- vertices $1, \dots, N$ corresponding to the dirty napkins which are sent off for cleaning (we assume that all napkins are washed for $i \le N - n$);
- vertices $1', \dots, N'$ which represent the supply of clean napkins needed for the N days.

We also add the following edges (with respective capacity constraints):

- $e = s0$ with $b(e) = 0$, $c(e) = \infty$, $\gamma(e) = 0$;
- all si with $b(si) = c(si) = r_i$, $\gamma(si) = 0$;
- all $0i'$ with $b(0i') = 0$, $c(0i') = \infty$, $\gamma(0i') = \alpha$;
- all $e = i(i+m)'$ with $b(e) = 0$, $c(e) = r_i$, $\gamma(e) = \beta$ (for $i + m > N$, the edge $i(i+m)'$ has to be interpreted as it, so that the cost of this edge has to be changed to 0);
- all $e = i(i+n)'$ with $b(e) = 0$, $c(e) = r_i$, $\gamma(e) = \delta$ (for $i + n > N$, the edge $i(i+n)'$ has to be interpreted as it, so that the cost of this edge has to be changed to 0);
- all $i't$ with $b(i't) = c(i't) = r_i$, $\gamma(i't) = 0$;
- all edges $e = i'(i+1)'$ with $b(e) = 0$, $c(e) = \infty$, $\gamma(e) = 0$; these edges represent the possibility of saving unused napkins for the next day.

10.2.3 As before, we define $c'(e) = c(e) - b(e)$. Moreover, put

$$c'(sv) = \sum_{\substack{e^+ = v \\ b(e) > 0}} b(e) - \sum_{\substack{e^- = v \\ b(e) < 0}} b(e); \qquad c'(vt) = \sum_{\substack{e^- = v \\ b(e) > 0}} b(e) - \sum_{\substack{e^+ = v \\ b(e) < 0}} b(e).$$

Then Theorem 10.2.1 remains valid without changes: a feasible circulation on G exists if and only if the maximal value of a flow on N is given by $W = \sum_e b(e)$.

10.2.6 First determine—if possible—a feasible flow as in Example 10.2.2. Next, a maximal flow can be found as in Chap. 6. To make sure that this flow is still feasible, we have to replace the condition $f(e) \ne 0$ in step (10) of Algorithm 6.3.14 (when constructing the auxiliary network) by $f(e) > b(e)$ and replace the assignment in step (11) by $c''(e) \leftarrow f(e) - b(e)$. Let us denote the resulting procedure by LEGAUXNET. We may now proceed as in Algorithm 6.3.17. (Note that the algorithm of Ford and Fulkerson with similar changes would serve the same purpose.) We obtain the following algorithm of complexity $O(|V|^3)$, where we put $N = (G, b, c, s, t)$ and use the FIFO preflow push algorithm for determining a blocking flow.

```
Procedure MAXLEGFLOW($N$; legal, $f$)
(1) Add the edge $r = ts$ to $G$; $b(r) \leftarrow 0$; $c(r) \leftarrow \infty$;
(2) LEGCIRC($G, b, c; f$, legal);
(3) if legal = true then
(4)       remove the edge $r = ts$ from $G$;
```

```
(5)    repeat
(6)        LEGAUXNET($N, f; N''$, max, $d$);
(7)        if max = false then BLOCKMKM($N''; g$); AUGMENT($f, g; f$) fi
(8)    until max = true
(9) fi
```

10.2.9 Apply the criterion of Theorem 10.2.7 to the directed multigraph H defined in the solution to Exercise 10.1.6 (using the capacity functions b and c given there); this yields the following theorem.

Let G be a connected mixed multigraph. Then G has an Euler tour if and only if the following two conditions hold:

(i) *Each vertex of G is incident with an even number of edges.*
(ii) *For each subset X of V, the difference between the number of directed edges e with $e^- \in X$ and $e^+ \in V \setminus X$ and the number of directed edges e with $e^+ \in X$ and $e^- \in V \setminus X$ is at most as large as the number of undirected edges connecting X and $V \setminus X$.*

10.2.10 Similarly to the proof of Theorem 10.2.8, one sees that the minimal value of a feasible flow is given by

$$\max\left\{ \sum_{e^- \in S, e^+ \in T} b(e) - \sum_{e^+ \in S, e^- \in T} c(e) : (S,T) \text{ is a cut on } N \right\},$$

where has $c(r) = v$ and $b(r) = -\infty$ for the return arc $r = ts$.

To determine a minimal flow, an arbitrary feasible flow can be changed applying methods similar to those used in Chap. 6. To find a path along which the value of the flow can be decreased, we admit forward edges e in the auxiliary network if and only if $b(e) < f(e)$, and backward edges if and only if $f(e) < c(e)$. Then the bounds on the complexity are the same as in Chap. 6. We leave the details to the reader.

10.2.11 By Exercise 8.5.3, G is strongly connected if and only if every edge is contained in a directed cycle. Hence we may show that this criterion is satisfied if and only if G has a feasible circulation.

First assume that G has a feasible circulation. Let $e = uv$ be an edge of G, and let S be the set of all vertices s from which u is accessible. If $v \notin S$, then $(S, V \setminus S)$ is a cut for which all edges of the corresponding cocycle are oriented from S to $V \setminus S$. Such a cut would violate the condition of Theorem 10.2.7, as $b(e) > 0$. Therefore there exists a directed path W from v to u. Then e is contained in the directed cycle $u \overset{e}{\text{---}} v \overset{W}{\text{---}} u$.

Conversely, assume that every edge of G is contained in a directed cycle. Then all cocycles contain edges in both possible directions, so that the condition of Theorem 10.2.7 is satisfied, since $c(e) = \infty$ for all e.

Finally, let N be a flow network with $c(e)=\infty$ and $b(e)>0$ for all edges e. Removing the return arc $r=ts$, we see that a feasible flow exists if and only if each edge is contained either in a directed cycle or in a directed path from s to t.

10.3.4 The first assertion is an immediate consequence of condition (Z1):

$$\begin{aligned}
f(S,T) &= \sum_{e^-\in S, e^+\in T} f(e) = \sum_{v\in S}\sum_{e^-=v, e^+\in T} f(e)\\
&= \sum_{v\in S}\Big(\sum_{e^-=v} f(e) - \sum_{e^-=v, e^+\in S} f(e)\Big)\\
&= \sum_{v\in S}\Big(\sum_{e^+=v} f(e) - \sum_{e^-=v, e^+\in S} f(e)\Big)\\
&= \sum_{v\in S}\Big(\sum_{e^+=v, e^-\in T} f(e) + \sum_{e^+=v, e^-\in S} f(e) - \sum_{e^-=v, e^+\in S} f(e)\Big)\\
&= \sum_{e^-\in T, e^+\in S} f(e) = f(T,S).
\end{aligned}$$

For the second assertion, let $f\neq 0$ be a circulation and consider an edge $e=uv$ in the support of f. Suppose that e is a bridge. Then $G\setminus e$ has at least two connected components S and T, say $u\in S$ and $v\in T$. Now e is the only edge in the cocycle $E(S,T)$, so that $f(S,T)=f(e)\neq 0$ and $f(T,S)=0$, contradicting the first assertion.

10.3.7 We may assume that each of the elementary circulations f_e in the proof of Theorem 10.3.6 satisfies the condition $f_e(e)=1$ (otherwise we multiply f_e by -1). Given an arbitrary circulation f, put

$$g = f - \sum_{e\in G\setminus T} f(e) f_e.$$

Then the support of g is contained in T, and Corollary 10.3.3 yields $g=0$.

10.3.8 Denote the vector in $\mathbb{R}^m$ corresponding to $\delta q : E\to\mathbb{R}$ by $\boldsymbol{\delta} q$. Then

$$\boldsymbol{\delta}\mathbf{q} = \sum_{i=1}^{n} q(v_i)\mathbf{a}_i,$$

where $\mathbf{a}_i$ is the i-th row of M (corresponding to the vertex i). Now P corresponds to the row space of M, and Theorem 4.2.4 yields $\dim P = \operatorname{rank} M = n-p$. This shows part (a).

For part (b), let (S,T) be a cut of G. We put $q(v)=1$ for $v \in S$, and $q(v)=0$ for $v \in T$. Then $\delta q(e)=+1$ or $=-1$ for all edges e contained in the cocycle corresponding to (S,T), and $\delta q(e)=0$ for all other edges.

Finally, let T be a spanning tree and T' the corresponding cotree: $T' = E \setminus T$. Consider any edge $e \in T$. By Lemma 4.3.2, there exists a unique cut for which the corresponding cocycle C_e contains only edges in T', except for e. By part (b), there is a potential difference δq_e for C_e whose support consists precisely of the edges of C_e. Thus the δq_e with $e \in T$ are $n-1$ linearly independent potential differences; in view of part (a), they have to form a basis of P: as G is connected, $p=1$.

10.3.16 First assume the existence of a cycle K as described in the first case of the painting lemma. As no edge of K is uncolored, we have $b(e) < c(e)$ for all $e \in K$. Now every black edge of K has the same orientation as e_0 and satisfies $f(e) < c(e)$; every green edge of K has the opposite orientation as e_0 and satisfies $b(e) < c(e)$; and every red edge satisfies both of the previous conditions, that is, $b(e) < f(e) < c(e)$. If we traverse K in the direction given by e_0, black edges are forward edges, while green edges are backward edges; red edges may be either forward or backward edges. The previous remarks show that we may define a new circulation f' by increasing f on all forward edges and decreasing f on all backward edges (by a sufficiently small amount δ), without increasing any of the deviation values $d(e)$. On the contrary, at least one deviation will become strictly smaller: $d(e_0)$ is replaced by $d(e_0)-\delta$. Thus f' is indeed a better circulation than f: it satisfies $D(f') \le D(f) - \delta$.

Now assume the existence of a cocycle C as described in the second case of the painting lemma. Let (S,T) be the cut defining C; we may assume that e_0 is oriented from T to S. Similarly to the first case, every forward edge of C (that is, every edge with the same orientation as e_0 in C) satisfies $f(e) \le b(e)$, while every backward edge satisfies $c(e) \le f(e)$. Moreover, we have strict inequality for the forward edge e_0, namely $f(e_0) < b(e_0)$. Hence, using Exercise 10.3.4,

$$c(S,T) \le f(S,T) = f(T,S) < b(T,S).$$

On the other hand, the circulation theorem 10.2.7 gives the necessary condition $b(T,S) \le c(S,T)$ for the existence of a feasible circulation. These inequalities are inconsistent, and hence no feasible circulation can exist in the second case.

Finally, note that the value δ in the first case may always be chosen as an integer, since the capacities are integral by hypothesis. Using induction, the value $D = D(f)$ likewise is integral throughout the entire procedure. As D strictly decreases and is bounded from below by 0, the procedure terminates with either a feasible circulation or a cocycle certifying the nonexistence of such a circulation.

10.3.17 Note that the necessity of the criterion given in the circulation theorem is rather trivial: in view of Exercise 10.3.4, any feasible circulation f satisfies

$$c(S,T) \geq f(S,T) = f(T,S) \geq b(T,S).$$

Now assume that this criterion is satisfied and that the capacities are integral. Then the second case in the algorithm of Herz cannot occur (as shown in the solution of Exercise 10.3.16), and hence the algorithm terminates with a feasible circulation.

10.4.4 For any feasible circulation, the integer

$$M = \sum_{\substack{e \\ \gamma(e)>0}} \gamma(e)c(e) + \sum_{\substack{e \\ \gamma(e)<0}} \gamma(e)b(e)$$

is an upper bound for the cost. Defining m as in the proof of Lemma 10.4.2, $M - m$ is an upper bound for the number of iterations needed.

10.5.4 We first consider the problem of determining the optimal cost $\gamma(v)$, where M is the maximal value of a flow on N and $v \leq M$ a real number. Denote the largest integer $\leq v$ by w, and let f be an optimal flow of value w constructed by Algorithm 10.5.2. Moreover, let W be an augmenting path of least possible cost from s to t in the auxiliary network $N'(f)$ with respect to the cost function γ'. As W has integral capacity, f can be augmented along W by $\delta = v - w < 1$ with cost $\delta\gamma'(W)$. It can be shown that the resulting flow of value v is optimal (proceed as in the proof of Lemma 10.5.1).

Thus the cost function is linear between any two integers w and $w+1$. As the cost of an augmenting path is always nonnegative, the cost function is also monotonically increasing. Finally, for any two feasible flows f and f' of values v and v', respectively, and for each λ with $0 \leq \lambda \leq 1$, the linear combination $\lambda f + (1-\lambda)f'$ is a feasible flow with value $\lambda v + (1-\lambda)v'$, so that

$$\gamma\big(\lambda v + (1-\lambda)v'\big) \leq \lambda\gamma(v) + (1-\lambda)\gamma\big(v'\big).$$

Hence the cost function is a monotonically increasing, piecewise linear, convex function.

10.5.5 By Example 10.1.4, the assignment problem can be reduced to the determination of an optimal flow of value n on a flow network with $2n+2$ vertices. As all capacities are integral (actually, they are always 1) and as the cost function is nonnegative, the algorithm of Busacker and Gowen can be used for determining an optimal flow with complexity $O(|V|^2 n) = O(n^3)$, by Theorem 10.5.3. Hence the assignment problem has complexity at most $O(n^3)$; it will be studied more thoroughly in Chap. 14.

10.6.2 The following procedure provides a possible solution:

Procedure RESIDUAL($G, c, f; H$)

```
(1) $E' \leftarrow \emptyset$;
(2) for $e \in E$ do
(3)     if $c(e) > f(e)$ then $E' \leftarrow E' \cup \{e\}$ fi
(4) od
(5) $H \leftarrow (V, E')$
```

10.6.8 Let f be an ε-optimal pseudoflow on (G, c) with respect to the cost function γ. We construct the auxiliary graph H_f described in the proof of Theorem 10.6.6 with cost function $\gamma^{(\varepsilon)}$, and proceed by determining an SP-tree for $(H_f, \gamma^{(\varepsilon)})$ using the procedure SPTREE given in Exercise 3.10.3; then the desired potential is just the distance function in this network, by Corollary 10.6.7. The procedure below does the job; here s is a vertex not contained in G.

Procedure POTENTIAL($G, c, \gamma, f, \varepsilon; p$)

```
(1) RESIDUAL($G, c, f; H$);
(2) $V^* \leftarrow V \cup \{s\}$; $E^* \leftarrow E'$;
(3) for $e \in E$ do $\gamma^*(e) \leftarrow \gamma^*(e) + \varepsilon$ od
(4) for $v \in V$ do $E^* \leftarrow E^* \cup \{sv\}$; $\gamma^*(sv) \leftarrow 0$ od
(5) $H^* \leftarrow (V^*, E^*)$;
(6) SPTREE ($H^*, \gamma^*, s; p, q, \mathrm{neg}, T$)
```

Note that p is the required distance function d_T in the arborescence T; the remaining output variables (that is, the predecessor function q for the SP-tree T and the Boolean variable neg) are not actually needed here. We could, of course, use the condition neg = false to check whether the given pseudoflow is indeed ε-optimal; see Theorem 10.6.6.

10.6.13 In the following procedure, s is a vertex not contained in H, and $n - 1$ denotes the number of vertices of H.

Procedure MEANCYCLE($H, w; \mu, C$)

```
(1) TOPSORT ($H$; topnr, acyclic);
(2) if acyclic = true
(3) then $\mu \leftarrow \infty$
(4) else $V^* \leftarrow V \cup \{s\}$; $E^* \leftarrow E$; $F(0, s) \leftarrow 0$;
(5)     for $v \in V$ do
(6)         $E^* \leftarrow E^* \cup \{sv\}$;
(7)         $w(sv) \leftarrow 0$; $F(0, v) \leftarrow \infty$
(8)     od
```

```
(9)       for $k = 1$ to $n$ do
(10)          for $v \in V^*$ do
(11)              $F(k,v) \leftarrow \min\{F(k-1,u) + w(uv) : uv \in E^*\}$;
(12)              $q(k,v) \leftarrow u$, where $u \in V$ is an element such that
                  $F(k-1,u) + w(uv) = \min\{F(k-1,x) + w(xv) : xv \in E^*\}$
(13)          od
(14)      od
(15)      for $v \in V$ do
(16)          $M(v) \leftarrow \max\{\frac{F(n,v)-F(k,v)}{n-k} : k = 0,\dots,n-1\}$
(17)      od
(18)      choose $v$ with $M(v) = \min\{M(x) : x \in V\}$;
(19)      $\mu \leftarrow M(v)$;
(20)      determine a walk $W$ of length $F(n,v)$ from $s$ to $v$ which consists
             of $n$ edges;
(21)      determine a cycle $C$ contained in $W$
(22)  fi
```

To prove that this procedure is correct, we use the proofs of Theorem 10.6.11 and Corollary 10.6.12. The procedure TOPSORT checks—according to Theorem 2.6.6—whether H^* (and hence H) is acyclic; in this case, μ is set to ∞. Otherwise, H contains directed cycles, and the **for**-loop in steps (9) to (14) determines the minimal length $F(k,v)$ of a directed walk from s to v consisting of precisely k edges (for all k and v); this is done recursively. Then, in steps (15) to (19), the minimum cycle mean μ of a directed cycle in H is calculated in accordance with Theorem 10.6.11. Now consider—as in the proof of Theorem 10.6.11—the changed weight function w' defined by $w'(e) = w(e) - \mu$ for all $e \in E$. The second part of the proof of Theorem 10.6.11 shows that the corresponding values $F'(k,v)$ and the vertex v chosen in step (18) satisfy the condition

$$\max\left\{\frac{F'(n,v) - F'(k,v)}{n-k} : k = 0,\dots,n-1\right\} = 0.$$

Thus the network (H,w') has minimum cycle mean 0. Now the first part of the proof of Theorem 10.6.11 shows that $F'(n,v) = F(n,v) - n\mu$ is the shortest length of a directed walk from s to v (and therefore the distance from s to v) in (H^*,w'). In step (20), a directed walk W from s to v having this length and consisting of n edges is determined; this is done recursively using the function $q(k,v)$ defined in step (12): the last edge of W is uv, where $u = q(n,v)$; the edge before the last is $u'u$, where $u' = q(n,u)$ and so on.

As W consists of precisely n edges, W has to contain a directed cycle C which is determined in step (21): this can be implemented, for example, by a labelling process while W is traced from s to v. Then $W \setminus C$ is a directed walk from s to v as well, which must have length at least $F'(u,v)$ in (H^*,w'). Therefore $w'(C)$ has to be 0; otherwise, $w'(C)$ would be positive because of $\mu' = 0$, so that $w'(W \setminus C) < w'(W)$. Hence $w(C) = \mu$.

10.6.15 Using Exercise 10.6.13 and Theorem 10.6.14, we obtain the following procedure:

```
Procedure TIGHT($G, c, \gamma, f; \varepsilon$)
(1) RESIDUAL($G, c, f; H$);
(2) MEANCYCLE($H, \gamma; \mu, C$);
(3) if $\mu \geq 0$ then $\varepsilon \leftarrow 0$ else $\varepsilon \leftarrow -\mu$ fi
```

10.8.9 Define the function Φ as given in the hint. At the beginning of Algorithm 10.8.1, $\Phi \leq |V|$, since the admissible graph G_A does not contain any edges at this point, so that $\Phi(v) = 1$ holds trivially for all vertices.

A saturating PUSH-operation, say PUSH(u, v), can increase Φ by at most $\Phi(v) \leq |V|$ (if v becomes active by this operation), so that all the saturating PUSH-operations together can increase Φ by at most $O(|V|^2|E|)$, by Lemma 10.8.7. A RELABEL(v)-operation might add new edges of the form vu to G_A, so that Φ is increased by at most $|V|$. Note that RELABEL(v) does not change the values $\Phi(w)$ for $w \neq v$: as we saw in the proof of Lemma 10.8.8, G_A does not contain any edges with end vertex v after this operation. By Lemma 10.8.6, all the RELABEL-operations together can increase Φ by at most $O(|V|^3)$; this value is dominated by $O(|V|^2|E|)$.

It remains to consider the non-saturating PUSH-operations. Such a PUSH(u, v) makes u inactive, whereas v might become active; thus it decreases Φ by $\Phi(u)$, and possibly increases Φ by $\Phi(v)$. However, $\Phi(u) \geq \Phi(v) + 1$, since each vertex in G_A which is accessible from v is accessible from u as well, and since u is not accessible from v (as G_A is acyclic by Lemma 10.8.8). Note that a PUSH-operation does not add any edges to G_A according to the proof of Lemma 10.8.8. Thus each non-saturating PUSH decreases Φ by at least 1. It follows that the total number of non-saturating PUSH-operations is bounded by the total increase of Φ during the algorithm, which is $O(|V|^2|E|)$.

10.9.6 The circulation f constructed during the initialization of Algorithm 10.9.1 is clearly C-optimal, so that $\varepsilon(f_0) \leq C$. By Lemma 10.9.3, $|E|$ consecutive iterations decrease $\varepsilon(f)$ by at least a factor of $1 - 1/|V|$. Theorem 10.6.5 guarantees that the algorithm terminates with an optimal circulation f as soon as $\varepsilon(f)$ becomes smaller than $1/|V|$; hence it suffices to decrease $\varepsilon(f)$ by a total factor of value $< 1/C|V|$. By Theorem 10.6.4, $|E||V|$ consecutive iterations always decrease $\varepsilon(f)$ by at least a factor of $1/2$, so that the algorithm has to terminate with an optimal circulation after at most $O(|V||E| \log C|V|)$ iterations.

10.11.2 Axioms (MS1) and (MS2) for a metric space hold trivially. We need to check the triangle inequality (MS3). Let $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ be three words in S^n. Denote by X the set of indices for which $\mathbf{x}$ and $\mathbf{y}$ disagree, so that

$d(\mathbf{x},\mathbf{y}) = |X|$; similarly, let Z be the set of indices for which $\mathbf{z}$ and $\mathbf{y}$ disagree. Then all three words agree for all indices not in $X \cup Z$, and hence

$$d(\mathbf{x},\mathbf{z}) \le |X \cup Z| = |X| + |Z| - |X \cap Z| \le d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z}).$$

10.11.4 By Exercise 4.2.14, the incidence matrix M of G has rank $n-1$, also when considered as a binary matrix. Note that a binary vector $\mathbf{f}$ satisfies $M\mathbf{f} = 0$ over $\mathbb{Z}_2$ if and only if $\mathbf{f}$ is the incidence vector of an even subgraph of G. Hence $C_E(G)$ has dimension $m - \operatorname{rank} A = m - n + 1$, as in the case of circulations. (This again shows that the even subgraphs of G may be viewed as the *binary circulations* on G.)

10.11.17 Let $a \ge 2$. The even graphical code of $K_{p,p'}$ with $p = 2^{a+1}$ and $p' = 2^a$ has parameters $[2^{2a+1}, 2^{2a} - 2^{a+1} - 2^a + 1, 4]$. We apply Theorem 10.11.15 with $c = 2$, $n_1 = 2^{a+1}$, and $n_2 = 2^a$. Then we may use for O_1 the extended binary Hamming code with parameters $[2^{a+1}, 2^{a+1} - (a+1) - 1, 4]$, and for O_2 the extended binary Hamming code with parameters $[2^a, 2^a - a - 1, 4]$. This results in a graphical code C^* with parameters $[2^{2a+1}, 2^{2a+1} - (2a+1) - 1, 4]$. By the remarks preceding Example 10.11.16, C^* is in fact the extended binary Hamming code with these parameters.

10.11.18 We have to show that the extended binary Hamming codes with parameters $[2^h, 2^h - h - 1, 4]$ can be constructed recursively as purely graphical codes. The case $h = 2$ is realized by the even graphical code of a cycle of length 4, which indeed has parameters $[4, 1, 4]$. The case $h = 3$ can be obtained from the even graphical code belonging to $K_{4,2}$, an $[8, 3, 4]$ code, using an augmentation according to Lemma 10.11.10. Now we can use induction on h, applying the constructions in Example 10.11.16 (for even values of h) and Exercise 10.11.17 (for odd values of h).

B.11 Solutions for Chap. 11

11.3.3 First assume $xv \in E'$, that is, $d(v) + \sum_{e^+=v} b(e) - \sum_{e^-=v} b(e) < 0$. With $g(e) = b(e)$ for all $e \in E$, the demand restriction for v yields

$$\begin{aligned} d(v) = d'(v) &= g(xv) + \sum_{e^+=v} g(e) - \sum_{e^-=v} g(e) \\ &= g(xv) + \sum_{e^+=v} b(e) - \sum_{e^-=v} b(e), \end{aligned}$$

and hence

$$g(xv) = d(v) - \sum_{e^+=v} b(e) + \sum_{e^-=v} b(e) = c'(xv) - 1.$$

Thus indeed $g(xv) = h(xv)$, where h is the admissible flow defined in the first part of the proof of Theorem 11.3.1. Similarly, one checks $g(vx) = h(vx)$ whenever $vx \in E'$.

B.12 Solutions for Chap. 12

12.1.4 As the proof of Theorem 12.1.1 shows, every maximal spanning tree for (G,w) is also an equivalent flow tree for $N = (G,c)$. Conversely, let T be an equivalent flow tree for N. Note that the flow value $w_T(x,y)$ between x and y in the network $(T,w|T)$ equals the capacity $w(P_{xy})$ of the unique path P_{xy} from x to y in T. By hypothesis, $w_T(x,y) = w(x,y)$ for all $x,y \in V$, which implies that P_{xy} is a path of maximal capacity from x to y in the network (G,w). By Exercise 4.5.6, T is a maximal spanning tree for (G,w).

12.1.5 We use induction on the number n of vertices. The case $n = 2$ is trivial. Thus let $n \geq 3$. Choose a pair (x,y) of vertices such that $w(x,y)$ is maximal, and remove one of these vertices, say x. By the induction hypothesis, the smaller flow network on $G \setminus x$ can be realized on a path P, say

$$P: \quad x_1 \text{ --- } x_2 \text{ --- } \cdots \text{ --- } x_{n-1},$$

where $y = x_i$. We insert x after y in P and denote the resulting path by P'. As $w(x,y)$ is the largest flow value on N, the flow values realized before on $G \setminus x$ are not changed by this operation. Clearly, we also obtain the correct flow value $w(x,y)$ between x and y.

It remains to consider $w(x,z)$ for a vertex z with $z \neq x,y$. Then $w(x,z) = w(y,z)$: the inequality (12.1) of Theorem 12.1.1 shows

$$w(x,z) \geq \min\bigl\{w(x,y), w(y,z)\bigr\} = w(y,z);$$

similarly, $w(y,z) \geq w(x,z)$. As P realizes all flow values $w(y,z)$ correctly, P' yields the correct values $w(x,z)$.

Applying this technique recursively, we obtain from the network of Fig. 12.1 the flow networks on smaller trees shown in Fig. B.38 (in the order shown there). These smaller flow networks can be realized (beginning with the trivial path on two vertices) on the paths shown below the corresponding tree.

12.3.5 For the graph in Example 12.3.1, $u(a) = 13$, $u(b) = 13$, $u(c) = 12$, $u(d) = 12$, $u(h) = 13$, $u(g) = 15$, $u(f) = 15$, and $u(e) = 11$. As shown in the proof of Theorem 12.3.4, the increased flow requirements which can be realized with the minimal capacity of 52 given by r (see Fig. 12.11) are $s(x,y) = \min\{u(x), u(y)\}$. Using this weight function on K yields the same dominating tree T as in Example 12.3.1; only the weights differ, see Fig. B.39.

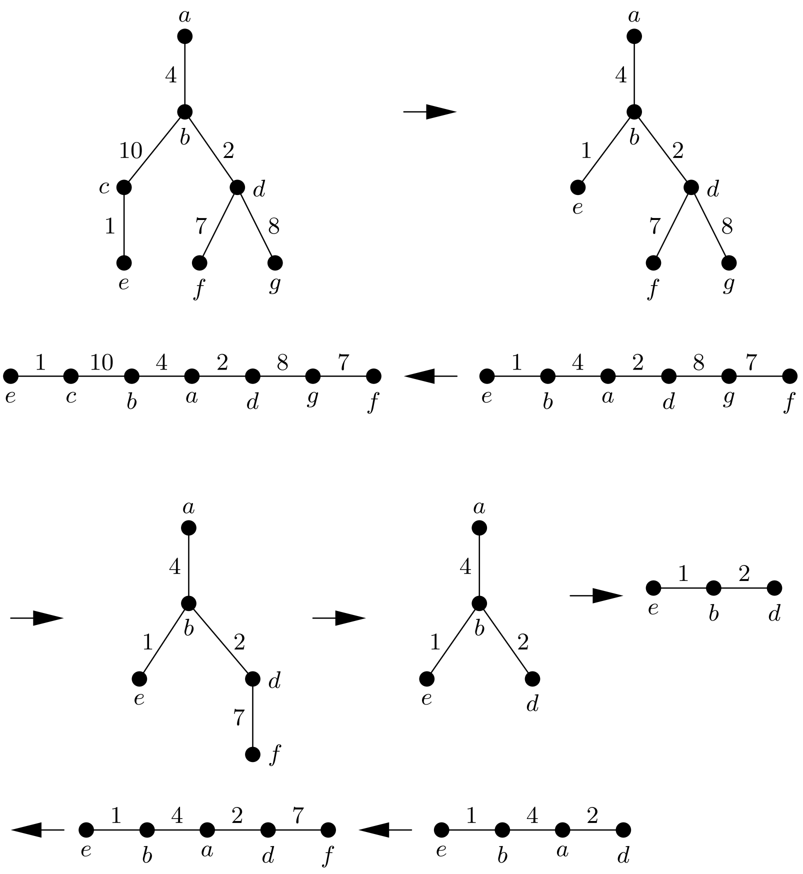


Fig. B.38 Recursive realization of a flow network on a path

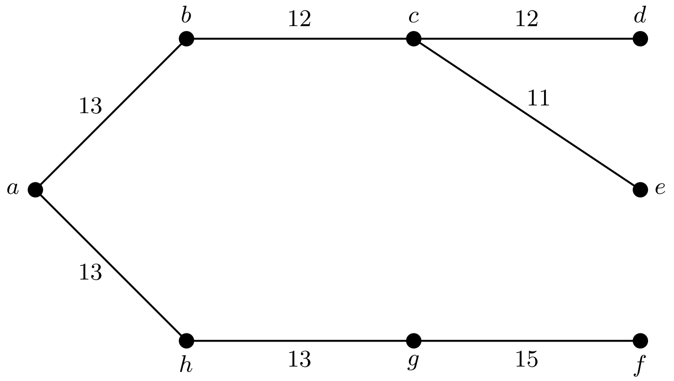


Fig. B.39 Dominating tree T

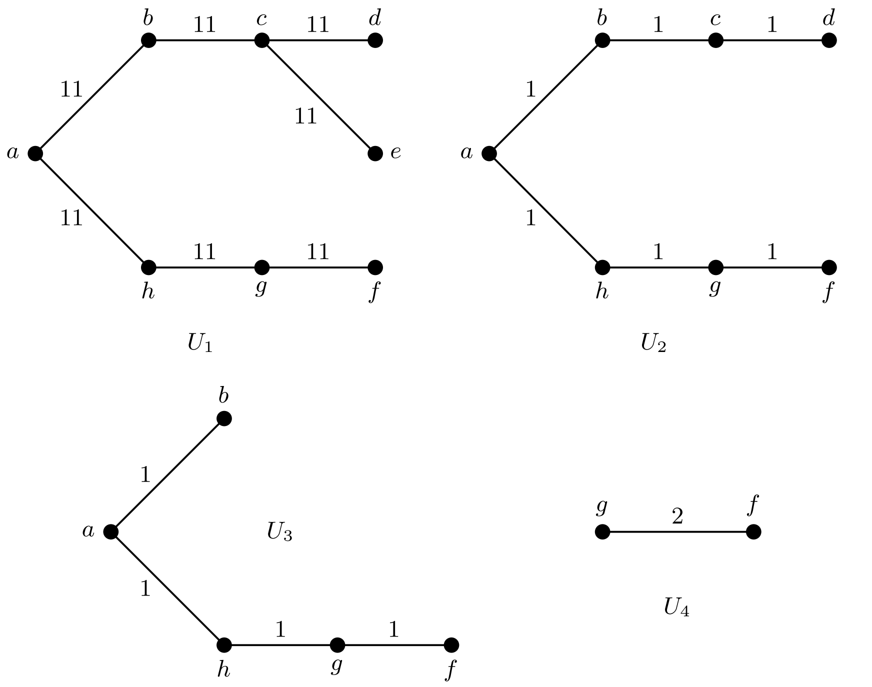


Fig. B.40 Partitioning T into uniform trees

Now we decompose T into the uniform trees $U_1, \dots, U_4$ shown in Fig. B.40 and construct corresponding cycles, say $C_1 = (a,b,c,d,e,f,g,h,a)$ with weight $11/2$, $C_2 = (a,b,c,d,f,g,h,a)$ with weight $1/2$, $C_3 = (a,b,f,g,h,a)$ with weight $1/2$, and the edge $C_4 = (g,f)$ with weight 2 (recall the order of the vertices is arbitrary); this yields the dominating network N shown in Fig. B.41. Note that N indeed allows higher flow values: for example, $w(a,c) = 12$, whereas the network of Fig. 12.11 gives a flow between a and c of value 8 only.

12.4.9 The network for the given values of the request function is shown in Fig. B.42. By Theorem 12.4.6, we have to determine a cut tree T for (G,r); this is done using Algorithm 12.4.2. After initializing T as a star with center 1, we obtain $s = 2$, $t = 1$, $w = 18$, and $s = \{2,3,4,5,6\}$, so that $f(s) = 2$. The vertices $3,4,5$, and 6 are then cut off from 1 and connected to 2 instead.

Next we have $s = 3$, $t = 2$, $w = 13$, and $S = \{3,4,5\}$. We set $f(3) = 13$, cut off the vertices 4 and 5 from 2, and connect them to 3 instead. For $s = 4$, we get $t = 3$, $w = 14$, and $S = \{4\}$. The tree T is not changed during this iteration, we just set $f(4) = 14$.

Next $s = 5$, $t = 3$, $w = 15$, and $S = \{4,5,6\}$. The vertices 3, 4, and 5 are removed from T, $s = 5$ is then connected to $p(t) = p(3) = 3$, and 3 and 4 are

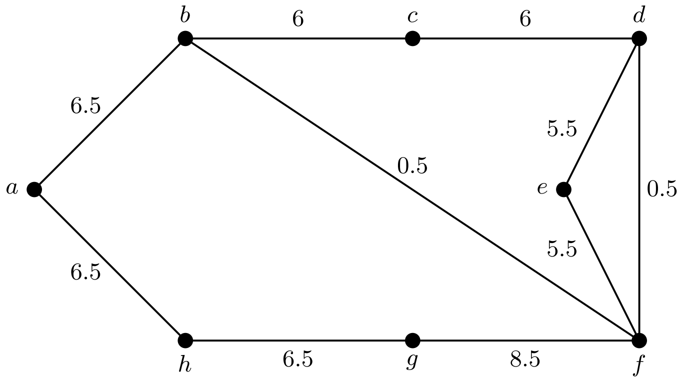


Fig. B.41 A dominating network

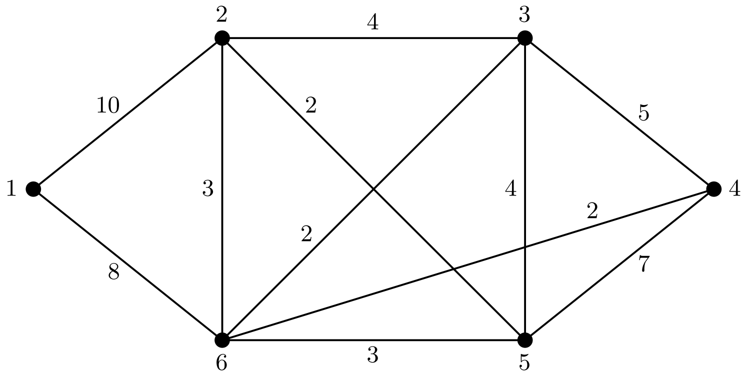


Fig. B.42 Network for Exercise 12.4.9

connected to 5. Also, $f(5)$ is now given the value $f(t) = f(3) = 13$, and $f(3)$ is changed to $w = 15$.

In the final iteration, $s = 6$, $t = 2$, $w = 17$, and $S = \{3,4,5,6\}$. We set $f(6) = 17$, and cut off 5 from 2 and re-connect it to 6. The resulting tree with weight 77 solves Problem 12.4.4 for the given request function r. Figure B.43 illustrates how the algorithm works.

12.5.2 The relevant part of the auxiliary network corresponding to the flow g of Fig. 12.16 is drawn in Fig. B.44; the fat edges form an augmenting path with cost 3 and capacity 15. Increasing the capacity of both sb and ct by θ (for $\theta = 1,\dots,15$), we obtain a flow of value $v = 41 + \theta$. The total cost for the corresponding increase of the capacity is $20 + 3\theta$. In particular, we obtain the flow h of value 56 and cost 65 shown in Fig. B.45.

The next step yields the auxiliary network shown in Fig. B.46; again, the fat edges form an augmenting path, now with cost 4 and unlimited capacity.

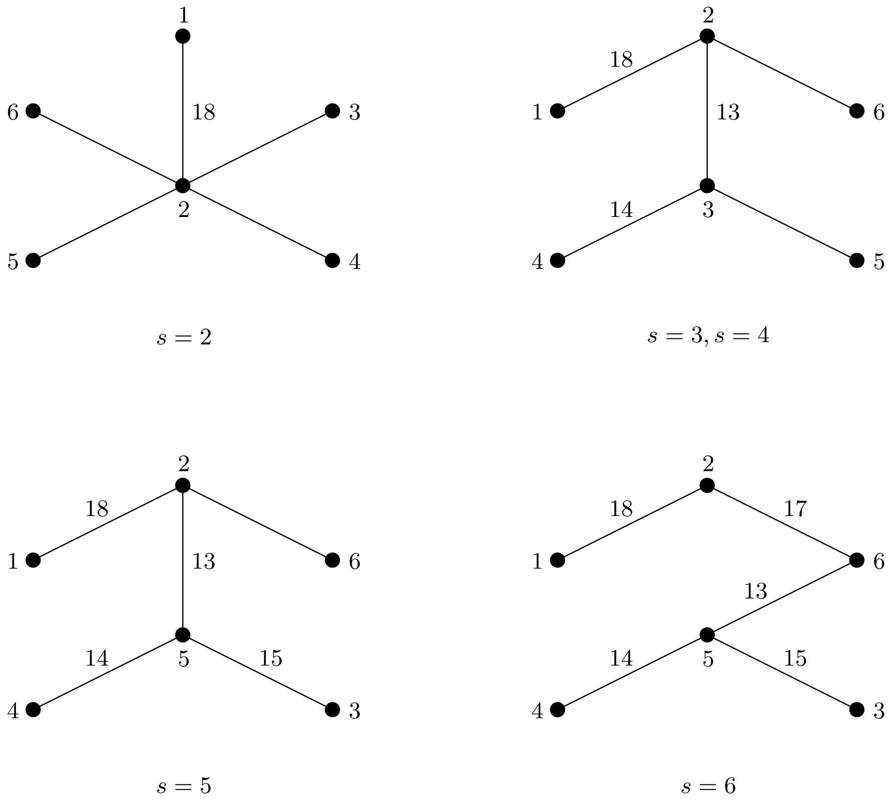


Fig. B.43 Determining a cut tree for N

Thus we can now realize any flow value $v = 56 + \tau$ with total cost $65 + 4\tau$ by increasing the capacity of each of the edges sb, bc, and ct by τ. Note that there are other paths of cost 4 in the auxiliary network of Fig. B.46, but the capacity of these paths is limited. We have now determined the cost function $z(v)$ completely (by executing the iteration step of the algorithm of Busacker and Gowen three times).

B.13 Solutions for Chap. 13

13.1.3 Define an auxiliary graph H as follows. Adjoin a d-element set D with $D \cap V = \emptyset$ to the vertex set V of G, and add all edges of the form vw with $v \in V$ and $w \in D$ to E. It is now easy to see that G has a matching with precisely d exposed vertices if and only if H has a perfect matching. Thus we have to show that condition (13.4) is equivalent to the existence of a perfect matching of H. For each subset X of $V \cup D$, let $o'(X)$ denote the number of

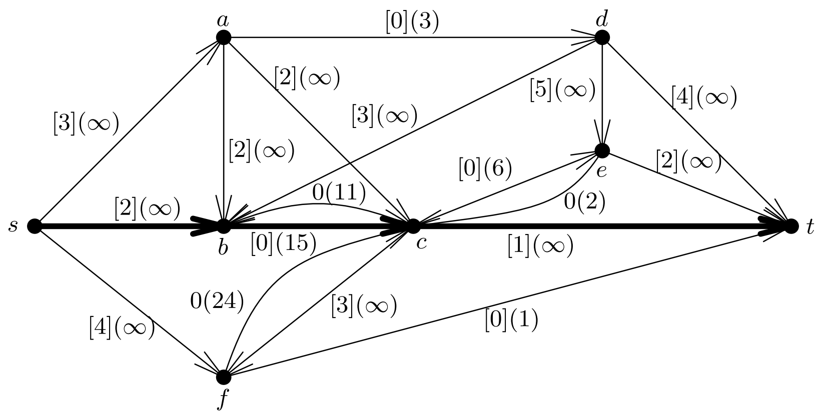


Fig. B.44 Auxiliary network for g

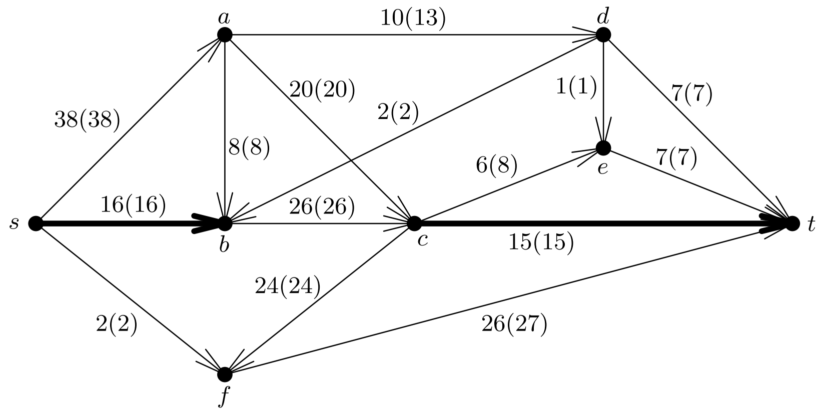


Fig. B.45 Flow h of value 56

odd components of $H \setminus X$.

$$o(S) = o'(S \cup D) \leq |S \cup D| = |S| + d \quad \text{for all } S \subset V.$$

Moreover, if H has a perfect matching, $|V| + d$ has to be even, so that (13.4) is necessary.

Conversely, suppose that (13.4) is satisfied. By Theorem 13.1.1, we have to show that the following condition holds:

$$o'(X) \leq |X| \quad \text{for all } X \subset V \cup D. \tag{B.3}$$

Assume first that D is not contained in X. Then, by the construction of H, the graph $H \setminus X$ is connected so that (B.3) is clearly satisfied for $X \neq \emptyset$. For

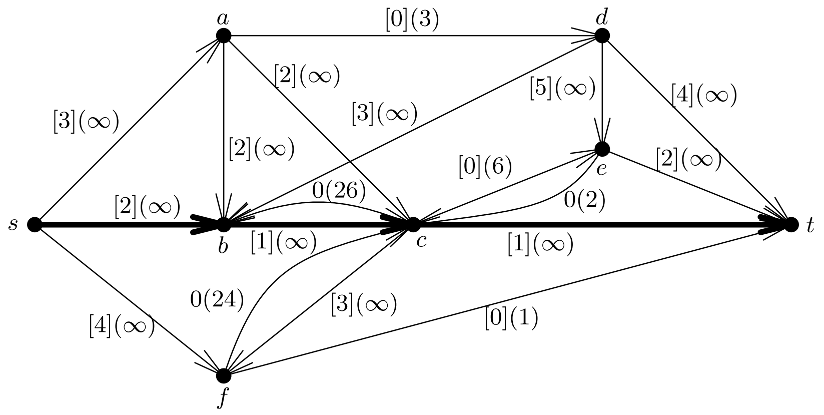


Fig. B.46 Auxiliary network for h

$X = \emptyset$, (B.3) holds as $|V \cup D| = |V| + d$ is even by hypothesis. Now assume $D \subset X$, say $X = S \dot\cup D$ for some $S \subset V$. Then (13.4) implies

$$o'(X) = o(S) \le |S| + d = |X|,$$

so that (B.3) is satisfied for this case as well.

13.1.4 Let H be the graph which results from adding the edges of the complete graph on T to G. Clearly, G has a perfect matching if and only if H does. Thus it suffices to show that condition (H) holds for G if and only if (T) holds for H. Put $n = |S| = |T|$.

First assume the validity of (H) for G. Given any subset X of V, we have to show $o(X) \le |X|$. This is clear for $X = \emptyset$, as H is connected and contains precisely $2n$ vertices.

Next we consider the case where $X \subset T$ and $X \ne \emptyset$, and put $J = T \setminus X$. Then the components of $H \setminus X$ are the set $Y = J \cup \Gamma(J)$ and the singletons corresponding to the elements of $S \setminus \Gamma(J)$. If $|Y|$ is even, we have $o(X) = n - |\Gamma(J)|$ and $|X| = n - |J|$. Now (H) implies $|\Gamma(J)| \ge |J|$, so that $o(X) \le |X|$ holds, as desired. If $|Y|$ is odd, $|\Gamma(J)| \ge |J|$ actually forces $|\Gamma(J)| \ge |J| + 1$, and the assertion follows in the same manner.

It remains to consider the case where X is not a subset of T. If $T \subset X$, the assertion holds trivially. Otherwise, let $X' = T \cap X$ and put $J = T \setminus X$, so that the components of $H \setminus X$ are the set $J \cup \Gamma(J)$ and the singletons corresponding to the elements of $S \setminus \Gamma(J)$. This implies

$$o(X) \le o\big(X'\big) + 1 \le \big|X'\big| + 1 \le |X|,$$

as required.

Conversely, assume the validity of condition (T) for H. Then one may check that (H) holds for G using a similar—actually easier—argument.

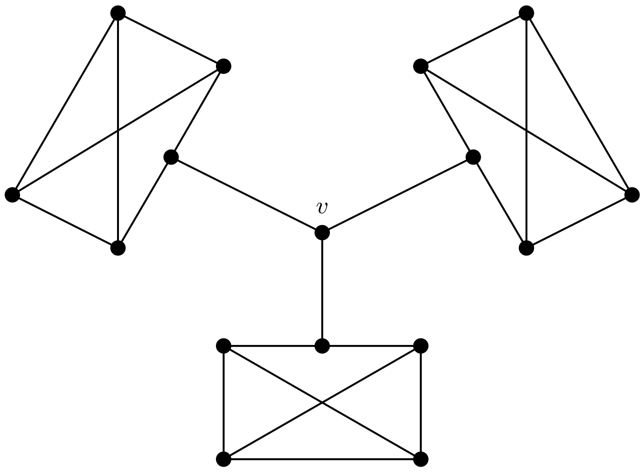


Fig. B.47 A 3-regular graph without a perfect matching

13.1.5 Let S be a subset of V, and denote the odd components of $G \setminus S$ by $V_1, \dots, V_k$. Moreover, let m_i be the number of edges connecting a vertex in V_i to a vertex in S (for $i = 1, \dots, k$). Since G does not contain any bridges, always $m_i \neq 1$. As G is 3-regular, $\sum_{v \in V_i} \deg v = 3|V_i|$ for $i = 1, \dots, k$, so that

$$m_i = \sum_{v \in V_i} \deg v - 2|E_i|$$

is an odd number (where E_i denotes the edge set of the graph G_i induced on V_i). Hence always $m_i \geq 3$, which yields

$$o(S) = k \leq \frac{1}{3}(m_1 + \cdots + m_k) \leq \frac{1}{3}\left(\sum_{v \in S} \deg v\right) = |S|.$$

Thus condition (T) is satisfied, and the assertion follows from Theorem 13.1.1.

Figure B.47 shows a 3-regular graph containing bridges; this graph cannot have a perfect matching, as $o(\{v\}) = 3$. Finally, the Petersen graph defined in Exercise 1.5.10 is a 3-regular graph without bridges which does not admit a 1-factorization.

13.2.5 Let C be a Hamiltonian cycle in a graph G on $2n$ vertices. Choosing every other edge of C, we obtain a perfect matching of G. Thus every Hamiltonian graph having an even number of vertices admits a perfect matching. Hence the two proposed criteria are indeed sufficient for the existence of a perfect matching of G, by Corollary 1.4.3 and Exercise 1.4.4.

13.2.6 We show first that G is 2-connected. Suppose otherwise. Then there exists a cut point v, so that $G \setminus v$ has two components X and Y. Choose an edge of the form vx with $x \in X$ and extend it to a perfect matching K. Then $|Y|$ has to be even and X has to be odd. However, the same argument also shows that $|Y|$ is odd and $|X|$ is even, a contradiction.

Assume first that G is bipartite, say $V = S \,\dot\cup\, T$. Suppose there are non-adjacent vertices $s \in S$ and $t \in T$, and let P be a path from s to t. As P has odd length, choosing the first, the third, ..., and the last edge of P gives a matching M. By hypothesis, M can be extended to a perfect matching M'. Then $M' \oplus P$ is a matching whose only exposed vertices are s and t. As s and t are not adjacent, this matching cannot be extended, a contradiction. Thus necessarily $G = K_{n,n}$ in the bipartite case.

It remains to consider the case where G is not bipartite. We show first that each vertex is contained in a cycle of odd length. Let v be a vertex of G, and let C be an arbitrary cycle of odd length; note that such a cycle exists by Theorem 3.3.5. We may assume that v is not contained in C. As G is 2-connected, Exercise 8.1.3 guarantees the existence of two paths P and P' with start vertex v and end vertex some vertex of C, which share only the vertex v. Thus these two paths together with the appropriate path in C which connects the two end vertices of P and P' form the desired cycle of odd length through v.

Now suppose that G contains two vertices u and v which are not adjacent. We claim that u and v are connected by a path of odd length. To see this, choose a cycle C of odd length containing v. If u is contained in C, the claim is clear. Otherwise, choose a path P with start vertex u and end vertex $w \neq v$ on C which does not contain any further vertices of C. Then P together with the appropriate path in C which connects w and v gives the required path of odd length from u to v. Now we obtain a contradiction just as in the bipartite case, and hence necessarily $G = K_{2n}$.

13.4.2 To simplify matters, we will make use of the inherent symmetry of the graph shown in Fig. 13.9 and consider only its right half: we restrict attention to the subgraph G induced on the set $\{r, s, 1, 2, 3, 4, 5, 6\}$. The left half—which is isomorphic to the right half—can be treated in the same way, and a final augmenting path arises by joining the two individual augmenting paths via the matching edge ss'.

Beginning the procedure at r yields the alternating tree T shown in Fig. B.48. When examining the edge 26, the blossom $B = \{2, 5, 6\}$ with base 2 is discovered and contracted. We obtain the graph $G' = G/B$ and the corresponding contracted tree $T' = T/B$ shown in Fig. B.49.

Next we examine the pseudovertex b and find the blossom $B' = \{r, 1, 3, 4, b\}$ with base r (because of edge 64). Contracting this blossom yields the graph $G'' = G'/B'$ which consists of the edge $b's$ only. This edge forms a trivial augmenting path P''. Expanding this path starting at s gives the augmenting path

$$P': \quad s — 1 — b — 4 — 3 — r$$

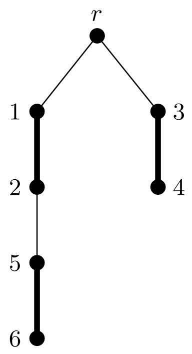


Fig. B.48 Alternating tree T for G

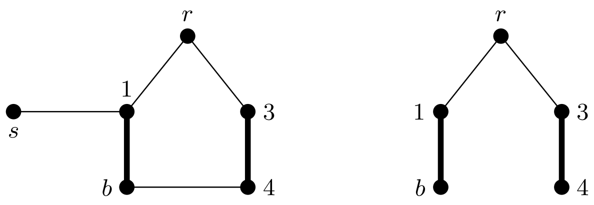


Fig. B.49 Contracted graph G/B with corresponding tree T/B

in G' and finally the augmenting path

$$P: \quad s — 1 — 2 — 5 — 6 — 4 — 3 — r$$

in G.

13.4.5 The graph G of Fig. 13.19 is drawn again in Fig. B.50. Obviously, $1 — 2 — 3 — 5$ is an augmenting path in G with respect to M. Contracting the blossom $B = \{2, 3, 4\}$, we obtain the graph $G' = G/B$ shown also in Fig. B.50; this graph has the matching $M' = \{b6\}$. Clearly, G' does not contain an augmenting path with respect to M'.

13.4.8 Let H be the graph with vertex set $V = \{1, \ldots, n\}$ which has an edge ij if and only if ij' (and then also ji') is an edge of G (for $i \neq j$). Then matchings in H consisting of k edges correspond to symmetric matchings in G with $2k$ edges. Thus a maximal symmetric matching of G can be determined using Algorithm 13.4.6 with complexity $O(n^3)$.[5]

13.5.5 If the maximal matching M constructed via the algorithm of Edmonds should be perfect, we simply have $D(G) = A(G) = \emptyset$ and $C(G) = V$.

[5]The work of Kocay and Stone and Fremuth-Paeger and Jungnickel mentioned at the beginning of this chapter uses the reverse approach: a symmetric bipartite graph G and an associated network are used for constructing a maximal matching in the corresponding graph H.

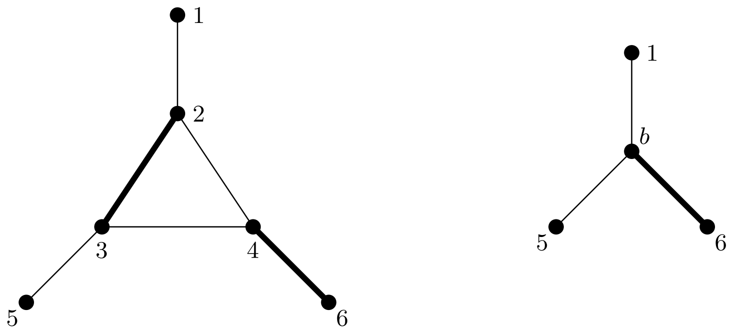


Fig. B.50 Graph G and contracted graph G/B

Thus assume that M is not perfect, so that the set X of exposed vertices is not empty. As noted before, $X \subseteq D(G)$.

Now let G_0 be the final graph obtained from G (by repeated shrinking of blossoms) in the last iteration of the algorithm, searching from the exposed vertex x, say; also, denote the maximal matching constructed in G_0 (and expanded to M) by M_0. Note that all vertices in a blossom B become even when the associated pseudovertex b is expanded again, as the stem of B is an alternating path of even length and as each vertex in B different from the base w of B can be reached from w on an even length path in B.

One also checks that any alternating path involved with a blossom B is expanded to an alternating path with the same parity when b is expanded, as the path will enter B through its base. This implies that the classification of those vertices of G which still belong to G_0 with respect to M_0 is the same as with respect to M. Thus $D(G) = \mathcal{E}$ consists of all vertices included in a blossom and of the original vertices of G included in $D(G_0) = \mathcal{E}_0$. Also, $A(G_0) = \mathcal{O}_0$. Finally, note that the classification of vertices in G_0 can be read from the final alternating tree T_0.

13.5.6 As mentioned in Example 13.5.2, the initial matching of G drawn in Fig. 13.10 is a near-perfect matching of $H = G \setminus 18$, leaving the vertex 17 exposed. Applying the search procedure in the algorithm of Edmonds to H, everything runs exactly as explained in detail in Sect. 13.4: we just need to leave out the discarded vertex 18 and its two incident edges. Therefore we do not find 18 when we search from b' in the tree $T'' = T'/B'$ in Fig. 13.14. Instead, when we examine the edge $\{12, 14\}$, we close another blossom, namely $B'' = \{b', 11, 12, 13, 14\}$. Hence we have to perform a further shrinking, and the final graph $G_0 = G''/B''$ agrees with the final tree T_0: it is just the alternating path 17—1—b'' of length 2. Applying the method outlined in Exercise 13.5.5 now confirms the result already stated in Example 13.5.2:

$$A(H) = \{1\}, \qquad C(H) = \emptyset \quad \text{and} \quad D(H) = \{2, \ldots, 17\}.$$

13.6.3 Let G be the bipartite graph on $V = S \dot{\cup} T$ corresponding to $\mathbf{A} = (A_1, \dots, A_n)$ (as defined in Sect. 7.3). Obviously, the partial transversals of $\mathbf{A}$ are precisely those subsets of S which are met by a matching of G. Therefore the partial transversals of $\mathbf{A}$ form a matroid by Corollary 13.6.2.

13.6.4 As the maximal matchings of G induce the bases of the matching matroid $(V, \mathbf{S})$, the assertion follows from Theorem 5.2.7. Alternatively, we may use Theorem 13.2.2: extending a matching using an augmenting path (as in the proof of Theorem 13.2.2) leaves any saturated vertex saturated, so that the assertion follows by induction.

B.14 Solutions for Chap. 14

14.1.2 Proceeding as outlined in Sect. 14.1, we obtain:

Procedure OPTMATCH$(n, w; M, D)$

```
(1)  $W \leftarrow \max\{w_{ij} : i, j = 1, \dots, n\}$;
(2)  $V \leftarrow \{1, \dots, n\} \cup \{1', \dots, n'\} \cup \{s, t\}$;
(3)  $E \leftarrow \{ij' : i, j = 1, \dots, n\} \cup \{si : i = 1, \dots, n\} \cup \{j't : j = 1, \dots, n\}$;
(4)  $G \leftarrow (V, E)$;
(5)  for $i = 1$ to $n$ do
(6)       $\gamma(si) \leftarrow 0$; $\gamma(i't) \leftarrow 0$; for $j = 1$ to $n$ do $\gamma(ij') \leftarrow W - w_{ij}$ od
(7)  od
(8)  for $e \in E$ do $c(e) \leftarrow 1$ od
(9)  OPTFLOW$(G, c, s, t, \gamma, n; f, \mathrm{sol})$;
(10) $M \leftarrow \{ij' : f(ij') = 1\}$; $D \leftarrow \sum_{e \in M} w(e)$
```

To achieve a complexity of $O(n^3)$, we have to use the algorithm of Dijkstra for determining the shortest paths in step (7) of OPTFLOW, as explained in Sect. 10.5).

14.2.6 During the first four phases, we obtain (without any changes) the edges $\{1, 4'\}$, $\{2, 9'\}$, $\{3, 6'\}$, and $\{4, 1'\}$ (in this order). Even the feasible node weighting $(\mathbf{u}, \mathbf{v})$ remains unchanged.

During the fifth phase (where $i = 5$), the only vertex j' with $\delta_j = 0$ is $9'$, which is saturated already. Nothing is changed by $i = 2$, because mate$(9') = 2$ is the smallest vertex in Q. Next, for $i = 6$, we find the edge $\{6, 3'\}$. Similarly, during the phases 6 and 7, the edges $\{7, 8'\}$ and $\{8, 7'\}$ are constructed. Up to this point, $(\mathbf{u}, \mathbf{v})$ was not changed.

During phase 8, we have $i = 5$, $i = 2$ (because of $\delta_9 = 0$, mate$(9') = 2$), $i = 9$, and $i = 7$ (because of $\delta_8 = 0$, mate$(8') = 7$). Now $J = \{2, 5, 7, 9\}$, $K = \{8', 9'\}$, and $\delta = 1$, so that the u_i and v_j have to be changed. We obtain the exposed vertex $5'$ with $\delta_5 = 0$, and the edge $\{9, 5'\}$ is added to the matching constructed so far.

The ninth (and last) phase is the most involved one. Again, we first have $i = 5$, $i = 2$, and $i = 7$. Then $(\mathbf{u}, \mathbf{v})$ has to be changed according to $J = \{2, 5, 7\}$, $K = \{8', 9'\}$, and $\delta = 2$. Then $\delta_4 = 0$ and mate $(4') = 1$, so that $i = 1$. Again, $(\mathbf{u}, \mathbf{v})$ has to be changed, this time for $J = \{1, 2, 5, 7\}$, $K = \{4', 8', 9'\}$ and $\delta = 3$. Three more changes of $(\mathbf{u}, \mathbf{v})$ follow: for $J = \{1, 2, 4, 5, 7\}$, $K = \{1', 4', 8', 9'\}$, $\delta = 1$; $J = \{1, 2, 4, 5, 7, 9\}$, $K = \{1', 4', 5', 8', 9'\}$, $\delta = 2$; and $J = \{1, 2, 4, 5, 7, 8, 9\}$, $K = \{1', 4', 5', 7', 8', 9'\}$, $\delta = 5$. Now $2'$ is exposed and we can complete the matching by adding the edge $\{5, 2'\}$.

We show the values for $(\mathbf{u}, \mathbf{v})$ below; the entries corresponding to edges used in the construction are in bold type. Note that indeed $w(M) = \sum(u_i + v_i)$ $(= 603)$ holds.

$$\begin{array}{cc}
\begin{pmatrix}
0 & 31 & 24 & \mathbf{80} & 62 & 39 & 24 & 41 & 42 \\
31 & 0 & 0 & 34 & 54 & 5 & 51 & 45 & \mathbf{61} \\
24 & 0 & 0 & 31 & 32 & \mathbf{59} & 28 & 44 & 25 \\
\mathbf{80} & 34 & 31 & 0 & 65 & 45 & 25 & 44 & 47 \\
62 & \mathbf{54} & 32 & 65 & 0 & 38 & 48 & 66 & 68 \\
39 & 5 & \mathbf{59} & 45 & 38 & 0 & 8 & 25 & 18 \\
24 & 51 & 28 & 25 & 48 & 8 & 0 & \mathbf{71} & 66 \\
41 & 45 & 44 & 44 & 66 & 25 & \mathbf{71} & 0 & 69 \\
42 & 61 & 25 & 47 & \mathbf{68} & 18 & 66 & 69 & 0
\end{pmatrix}
&
\begin{matrix} 69 \\ 47 \\ 59 \\ 72 \\ 54 \\ 59 \\ 57 \\ 66 \\ 61 \end{matrix}
\\[1em]
\begin{matrix} 8 & 0 & 0 & 11 & 7 & 0 & 5 & 14 & 14 \end{matrix}
&
\mathbf{v} \backslash \mathbf{u}
\end{array}$$

Note that the matching consisting of the edges $\{1, 4'\}$, $\{2, 5'\}$, $\{3, 6'\}$, $\{4, 1'\}$, $\{5, 9'\}$, $\{6, 3'\}$, $\{7, 8'\}$, $\{8, 7'\}$, and $\{9, 2'\}$ is optimal as well.

14.2.7 The algebraic assignment problem for the ordered semigroup $(\mathbb{R}_0^+, \min)$ yields the bottleneck assignment problem.

14.2.8 During the first two phases, the edges $\{1, 3'\}$ and $\{2, 4'\}$ are found. In phase 3, first $i = 3$ and $\delta_4 = 1$; as $\text{mate}(4') = 2$, then $i = 2$, and we find the exposed vertex $5'$ with $\delta_5 = 1$. Thus the present matching is changed using $p(5) = 2$, $\text{mate}(2) = 4'$, and $p(4) = 3$; we obtain the edges $\{1, 3'\}$, $\{2, 5'\}$, and $\{3, 4'\}$.

In phase 4, the current matching is enlarged by the edge $\{5, 2'\}$. During the final phase, $(\mathbf{u}, \mathbf{v})$ has to be changed twice: first with $J = \{2, 3, 4\}$, $K = \{4', 5'\}$, and $\delta = 5/4$; and then with $J = \{2, 3, 4, 5\}$, $K = \{2', 4', 5'\}$, and $\delta = 36/35$. The matching is changed once again; we get the solution

$$M = \{\{1, 3'\}, \{2, 5'\}, \{3, 1'\}, \{4, 4'\}, \{5, 2'\}\}.$$

The corresponding entries are set bold in the matrix below. We also check our calculations: $w(M) = \prod(u_i, v_i) = 15120$.

$$\begin{array}{ccccc|c} \begin{pmatrix} 3 & 8 & \mathbf{9} & 1 & 6 \\ 1 & 4 & 1 & 5 & \mathbf{5} \\ \mathbf{7} & 2 & 7 & 9 & 2 \\ 3 & 1 & 6 & \mathbf{8} & 8 \\ 2 & \mathbf{6} & 3 & 6 & 2 \end{pmatrix} & \begin{matrix} 9 \\ 35/9 \\ 7 \\ 56/9 \\ 35/6 \end{matrix} \\ \begin{matrix} 1 & \frac{36}{35} & 1 & \frac{9}{7} & \frac{9}{7} \end{matrix} & \mathbf{v}\backslash\mathbf{u} \end{array}$$

Note that this product-optimal matching accidentally coincides with the optimal matching of Example 14.2.5; as Exercise 14.2.10 shows, this really is exceptional.

14.2.9 Denote the given weight matrix by $W = (w_{ij})$ and put $W' = (\log w_{ij})$. Then the product-optimal matchings with respect to W are precisely the optimal matchings with respect to W'.

However, this transformation is not of practical interest. When executing calculations with W' using a computer, errors occur because of rounding (logarithms are irrational in general), and this means we cannot check our solution by comparing $w'(M)$ with $\sum(u'_i + v'_i)$.

Alternatively, we might consider doing all calculations *symbolically*, so that we perform operations such as replacing $\log p + \log q$ with $\log pq$. But then we may as well use the version of the Hungarian algorithm modified for $(\mathbb{R}^+, \cdot)$. Nevertheless, the above transformation at least yields an immediate proof for the correctness of this approach.

14.2.10 For the matrix

$$\begin{pmatrix} 3 & \mathbf{1} & 1 \\ 1 & 4 & \mathbf{5} \\ \mathbf{6} & 1 & 4 \end{pmatrix},$$

the matching given by the bold entries has weight $12 = 1 + 5 + 6$ and is obviously optimal. However, it is not product-optimal. On the other hand, the matching corresponding to the entries in the main diagonal is product-optimal but not optimal.

14.3.2 First let A be the incidence matrix of a digraph G. By Lemma 10.3.1, the vector $\mathbf{f} = (f_e)_{e\in E}$ gives a circulation if and only if $A\mathbf{f}^T = 0$. Therefore we get the ILP

$$\text{minimize} \quad \boldsymbol{\gamma}\mathbf{x}^T \quad \text{subject to} \quad A\mathbf{x}^T = \mathbf{0}^T, \quad \mathbf{b} \le \mathbf{x} \le \mathbf{c}.$$

For the second part, let T be a spanning tree of a graph G on n vertices. As T contains $n-1$ edges and is acyclic, we can use the following ZOLP:

$$\text{maximize} \quad \mathbf{w}\mathbf{x}^T \quad \text{subject to} \quad \mathbf{1}\mathbf{x}^T = n-1 \quad \text{and} \quad \sum_{e\in C} x_e \le |C|-1,$$

where $\mathbf{x} = (x_e)_{e\in E}$ and where C runs over all cycles in G. This approach is not interesting in practice, as the ZOLP will (in general) contain far too many inequalities.

14.4.6 Let G be a regular bipartite multigraph with vertex set $V = S \,\dot\cup\, T$, where $|S| = |T| = n$. We define the *adjacency matrix* $A = (a_{ij})_{i,j=1,\dots,n}$ of the multigraph G as follows: a_{ij} is the number of edges of G with end vertices i and j', where we assume $S = \{1,\dots,n\}$ and $T = \{1',\dots,n'\}$.

Thus A is a matrix with nonnegative integral entries, and its row and column sums are constant. By Theorem 7.4.5, we can write A as a sum of permutation matrices. As each permutation matrix corresponds to a 1-factor of G, the decomposition of A yields a 1-factorization of G.

14.4.7 Obviously, $\mathbf{L}(G) \subset \mathbf{H}(G) \cap \mathbb{Z}^E$. Now let $\mathbf{x}$ be a vector in $\mathbf{H}(G) \cap \mathbb{Z}^E$, and choose some positive integer k which is larger than the absolute value of $\mathbf{x}$. Then $\mathbf{x}' = \mathbf{x} + \sum k\mathbf{m}$ is likewise an element of $\mathbf{H}(G) \cap \mathbb{Z}^E$, where $\mathbf{m}$ runs over the incidence vectors of the perfect matchings of G. Moreover, $\mathbf{x}' \ge \mathbf{0}$.

We now define a regular bipartite multigraph G' by replacing each edge e of G with x'_e parallel edges. Note that G' is indeed regular, since $\mathbf{x}'$ is contained in $\mathbf{H}(G)$. By Exercise 14.4.6, G' can be decomposed into 1-factors. As each 1-factor of G' induces a 1-factor of G, we see that $\mathbf{x}'$ has to be a linear combination of incidence vectors of perfect matchings of G with nonnegative integral coefficients, so that also $\mathbf{x} = \mathbf{x}' - \sum k\mathbf{m}$ is contained in $\mathbf{L}(G)$.

14.5.6 Every closed walk of G which contains each edge at least once induces a circulation $\mathbf{f}$ on G: define f_e as the number of times e occurs in the given walk. Note $\mathbf{f} \ge \mathbf{1}$.

Conversely, every circulation $\mathbf{f}$ with $\mathbf{f} \ge \mathbf{1}$ induces a closed walk on G which contains all edges: replace each edge e with f_e parallel edges. By Theorem 1.6.1, the resulting pseudosymmetric digraph contains an Euler tour, which induces the desired walk.

Note that G is obviously connected. Thus a shortest directed closed walk corresponds to an optimal circulation with respect to the capacity constraints $b(e) = 1$ and $c(e) = \infty$ and the cost $\gamma(e) = w(e)$ (for all $e \in E$). Thus the directed CPP can be solved using the algorithm OPTCIRC from Sect. 10.7.

14.6.5 $W: s \,\rule{1em}{0.4pt}\, c \,\rule{1em}{0.4pt}\, t$ is a shortest $\{s,t\}$-path (of length -1). The corresponding f-factors are

$$F = \big\{\{a,a\},\{b,b\},sc,ct\big\} \quad \text{and} \quad F' = \big\{\{a,a\},\{b,b\},sc,ct,c_g b_g,a_e b_e\big\},$$

and the corresponding perfect matching is $M = \{a'a'', b'b'', sc', c''t, a_e b_e, c_g b_g\}$.

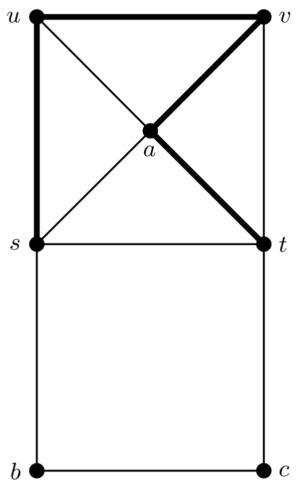


Fig. B.51 A path of even length in G

14.6.6 If (G, w) contains cycles of negative length, the method is not applicable, as the proof of Lemma 14.6.1 shows. (The construction would yield a path from s to t together with—possibly negative—cycles.) This is not surprising, since the problem of finding a shortest path is then NP-hard; see Appendix A or [GarJo79].

14.6.9 Use the following modification of the transformation described in Theorem 14.6.7: each edge of the form tv is now replaced by an edge tv'' (instead of tv'), and an eventual edge st is removed.

For the graph G of Example 14.6.8, the even $\{s,t\}$-path

$$W: \quad s - u - v - a - t$$

corresponds to the perfect matching

$$M = \{su', u''v'', v'a', a''t, b'b'', c'c''\}$$

in the auxiliary graph G''; see Figs. B.51 and B.52.

14.7.2 Define the weight function w on H as follows:

$$w_e = \begin{cases} 0 & \text{for } e \notin E, \\ c-1 & \text{for } e \in E_1, \\ c & \text{for } e \in E \setminus E_1. \end{cases}$$

Now let M be an optimal matching of H with respect to this weight function. Then M consists exclusively of edges of G if and only if $w(M) \geq n(c-1)$, where $|V| = 2n$. In this case, the number of edges of M contained in E_1 is $cn - w(M)$. Thus the problem RPM1 has a solution if and only if $cn - w(M) \leq b_1$.

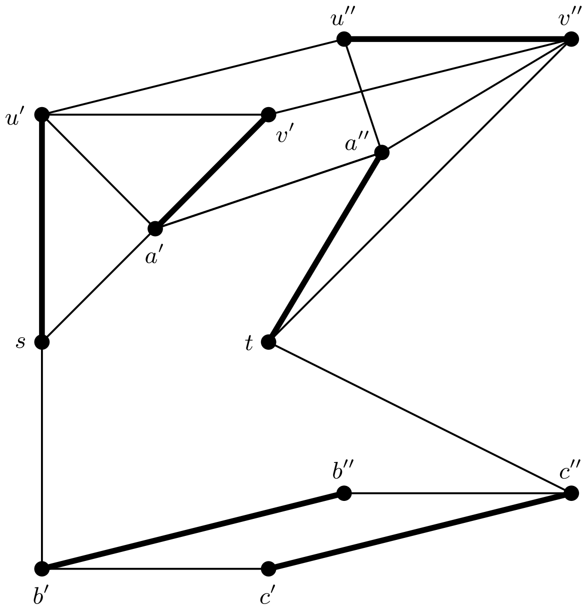


Fig. B.52 The corresponding perfect matching in G''

By Result 14.4.5, $w(M)$ can be determined with complexity $O(n^3)$, and the assertion follows.

14.7.4 Put $E_1 = R$, $E_2 = E \setminus R$, $b_1 = b$, and $b_2 = n - b$ (where $|V| = 2n$). The perfect matchings of G which satisfy condition (14.10) of Problem 14.7.1 for these values are precisely the desired solutions of EPM, since every perfect matching contains exactly n edges.

14.8.6 We use the edge labelling of the Petersen graph given in Fig. 14.15 for indexing the coordinate positions, so that the edge labelled i corresponds to the coordinate position i (for $i = 1, \dots, 15$). As noted in the hint, it clearly suffices to check that the cyclic shifts of the codewords in some basis of P^* are again in P^*.

Thus we begin by determining a basis of the Petersen code P, using the method given in the proof of Theorem 10.3.6. We first have to select a spanning tree T for the Petersen graph; let us choose the five spoke edges 3, 6, 9, 12, and 15 together with four edges of the outer cycle, say 2, 5, 8, and 11. Then each of the six edges $e \notin T$ determines a unique cycle contained in $T \cup \{e\}$, namely:

- $C_1 = \{1, 15, 2, 11, 5, 12\}$, belonging to $1 \notin T$;
- $C_4 = \{4, 15, 2, 11, 3\}$, belonging to $4 \notin T$;
- $C_7 = \{7, 6, 8, 2, 11, 3\}$, belonging to $7 \notin T$;
- $C_{10} = \{10, 9, 2, 8, 6\}$, belonging to $10 \notin T$;
- $C_{13} = \{13, 9, 11, 5, 12\}$, belonging to $13 \notin T$;
- $C_{14} = \{14, 8, 2, 11, 5\}$, belonging to $14 \notin T$.

Then we obtain the augmented Petersen code P^* by adjoining an odd subgraph, for instance, the matching formed by the five spoke edges as a further basis element, see Example 10.11.11:

- $S = \{3, 6, 9, 12, 15\}$.

Now it suffices to check that the cyclic shifts of these seven codewords are again in P^*. We obtain the following images, where τ denotes the cyclic shift:

- $C_1^\tau = \{1, 2, 3, 6, 12, 13\}$, an odd subgraph;
- $C_4^\tau = \{1, 3, 4, 5, 12\}$, a cycle;
- $C_7^\tau = \{3, 4, 7, 8, 9, 12\}$, an odd subgraph;
- $C_{10}^\tau = \{3, 7, 9, 10, 11\}$, a cycle;
- $C_{13}^\tau = \{6, 10, 12, 13, 14\}$, a cycle;
- $C_{14}^\tau = \{3, 6, 9, 12, 15\}$, the spoke matching;
- $S^\tau = \{1, 4, 7, 10, 13\}$, the inner pentagram.

All these words indeed belong to P^*.

B.15 Solutions for Chap. 15

15.2.2 Consider an edge e of smallest weight in a perfect matching M of minimal weight, say $e = ij$ with weight w_{ij}. In the symmetric case, $w_{ij} = w_{ji}$. If the remaining weights are much larger, also the edge ji will belong to M. Hence it is more likely that a pair of antiparallel edges occurs in the symmetric case than in the asymmetric case.

15.2.5 Let T be the minimal spanning tree associated with the given TSP which is shown in Fig. 15.2. Note that we obtain a minimal s-tree for $s = Aa$ by adding the edge $AaFr$, so that the weight is $186 + 26 = 212$. Similarly, we obtain weights of $186 + 34 = 220$ and $186 + 22 = 208$ for $s = Ba$ and $s = Mu$, respectively.

For $s = Du$, a minimal spanning tree on the remaining eight vertices consists of the edges $BeHa$, $HaAa$, $AaFr$, $FrSt$, $StBa$, $StNu$ and $NuMu$, so that the weight of a minimal s-tree is $187 + 8 + 23 = 218$.

For the remaining four choices of s, we just list the weight of a minimal s-tree: $184 + 20 + 22 = 226$ for $s = Fr$; $158 + 29 + 43 = 230$ for $s = Ha$; $172 + 17 + 19 = 208$ for $s = Nu$; and $176 + 19 + 20 = 215$ for $s = St$.

15.2.6 Let T be a minimal spanning tree associated with the given TSP, and assume that s is a leaf of T. Then we obtain a minimal s-tree by adding an edge of smallest weight among all edges not in T and incident with s to T. This follows by observing that $T \setminus e$, where e is the unique edge of T incident with s, has to be a minimal spanning tree for the complete graph induced on the remaining points.

In general, however, matters cannot be as simple. For instance, if s has degree at least 3 in T, we cannot obtain an s-tree containing T for trivial reasons. This observation also suggests examples showing that the strategy for selecting s which we have used in Example 15.2.4 may fail badly.

For instance, let T be a star for which all edges have weight a, and assume that all remaining edges have weight $b > a$; note that T is the unique minimal spanning tree for TSP-instances of this type. Our strategy does not allow us to select s as the center of T, which would lead to a minimal s-tree of weight $(n-2)b + 2a$. Any other choice of s is permissible and would result in a minimal s-tree of weight $(n-2)a + 2b$; in general, this will be a considerably smaller—and hence inferior—bound. Thus our strategy can prevent the optimal choice for s, and the deviation between the resulting bounds can even be made arbitrarily large.

15.2.7 Any s-tree is a spanning 1-forest, but not conversely: if we add an edge joining two non-adjacent vertices of degree at least two in a spanning tree, we obtain a 1-forest which is clearly not an s-tree for any choice of s. Thus the weight of a minimal spanning 1-forest is a lower bound for the weight of a minimal s-tree (for all choices of s). Clearly, it is also a lower bound for the weight of a given TSP, as any two is obviously a spanning 1-forest.

A minimal spanning 1-forest is easily found using the greedy algorithm, as the 1-forests form a matroid on the edge set, by Exercise 5.2.6. It can also be obtained by adding an edge of smallest weight to any minimal spanning tree; this follows from Exercise 4.3.6. Adding the edge $StMu$ of weight 22 to the minimal spanning tree in Fig. 15.2 gives a minimal 1-forest with weight 186+22 = 208. Note that this happens to be the minimal s-tree for $s = Mu$ discussed in the solution to Example 15.2.5.

15.2.8 By Corollary 1.2.11, there are $(n-1)^{n-3}$ distinct spanning trees on the remaining $n-1$ vertices. To each of these trees, we have to add one of the $(n-1)(n-2)/2$ pairs of edges incident with s, so that the total number of s-trees of K_n is $\frac{1}{2}(n-2)(n-1)^{n-2}$.

15.2.9 In every tour, each vertex i is incident with two edges whose weight is at least $s(i) + s'(i)$. This leads to the desired inequality and yields a lower bound of 214 for the TSP of Example 15.1.2; note that w is integral.

15.3.1 Let B be an s-tree. Then

$$\sum_i p_i = c \times \sum_i (\deg_B i - 2) = c \times \left(\sum_i \deg_B i - 2n \right) = c(2n - 2n) = 0.$$

15.5.4 As in Example 15.5.3, we begin with $s = Fr$. In the first two steps, we obtain the partial tour (Fr, St, Fr) with length 40 and then (Fr, St, Nu, Fr) with length 61.

Now Mu is inserted, and we get (Fr, St, Mu, Nu, Fr) with length 81. The next iteration yields (Fr, Du, St, Mu, Nu, Fr) with length 125. Inserting Aa between Du and St yields a partial tour with length 138.

We proceed with $(Fr, Du, Aa, Ba, St, Mu, Nu, Fr)$ with length 176; after this, we insert Ha between Fr and Du, which yields a partial tour with length 246. Finally, we obtain the tour $(Fr, Be, Ha, Du, Aa, Ba, St, Mu, Nu, Fr)$ with length 281.

15.6.4 First, the edges $AaMu$ and $FrBe$ are replaced with $MuBe$ and $AaFr$. This reduces the weight of the tour shown in Fig. 15.7 by $34 = (64 + 56) - (60 + 26)$; the resulting tour of weight $307 - 34 = 273$ is $(Aa, Du, Ha, Be, Mu, Nu, Ba, St, Fr, Aa)$.

Next, the edges $NuBa$ and $FrSt$ are replaced with $BaFr$ and $StNu$. This yields the tour $(Aa, Du, Ha, Be, Mu, Nu, St, Ba, Fr, Aa)$ and reduces the weight by $(43 + 20) - (34 + 19) = 10$ to 263.

Finally, we replace the edges $StNu$ and $BeMu$ with $StMu$ and $NuBe$, which yields the (optimal) tour of length 250 shown in Fig. 15.9; indeed, this step reduces the weight by $(19 + 60) - (22 + 44) = 13$.

15.7.7 As $\mathbf{A}'$ is assumed to be an ε-approximative algorithm for TSP, we can use it to solve the problem HC as described in the proof of Theorem 15.4.1. Note that each iteration of the algorithm $\mathbf{A}'$ (that is, each application of $\mathbf{A}$ to a neighborhood $N(f)$) decreases the weight of the current tour—with the exception of the final application of $\mathbf{A}$, which merely discovers that the current tour is now locally optimal.

As the weight function defined in the proof of Theorem 15.4.1 takes only two values, there can be only $n + 1$ distinct lengths of tours. Therefore $\mathbf{A}'$ cannot need more than $O(n)$ iterations of $\mathbf{A}$. Since $\mathbf{A}$ is polynomial, $\mathbf{A}'$ would be a polynomial algorithm for HC, so that P = NP by Result 13.2.2.

15.7.8 Suppose that the given problem could be solved polynomially. We show that this implies that we can find even an optimal tour in polynomial time. We may assume that all weights are ≥ 2. (If necessary, we add a constant to all weights.)

Now we check whether some specified edge e_1 is contained in an optimal tour. If the answer is yes, we reduce $w(e_1)$ to 1; this ensures that e_1 has to be contained in *every* optimal tour for the modified problem. More precisely, the optimal tours for the new problem are precisely those optimal tours for the old problem which contain the edge e_1. Continuing in this manner, we obtain an optimal tour for the original problem after $O(n^2)$ calls of the decision problem which we assumed to be polynomial.

15.7.9 Note that TSP suboptimality does not actually *produce* a better tour: it only tells us if such a tour *exists*. Hence it is not possible to apply a hypothetical polynomial algorithm $\mathbf{A}$ for this problem repeatedly, which would be necessary if we were to use $\mathbf{A}$ to construct a polynomial algorithm for HC.

Appendix C
List of Symbols

Now pray, what did he mean by that?
Richard Brinsley Sheridan

C.1 General Symbols

This first part of the list contains some general symbols which are more or less standard. The special symbols of graph theory will be covered in the next section.

Sets

$A \cup B$	union of the sets A and B
$A \dot{\cup} B$	disjoint union of the sets A and B
$A \cap B$	intersection of the sets A and B
$A \times B$	Cartesian product of the sets A and B
$A \setminus B$	A without B: $A \cap \overline{B}$
$A \oplus B$	symmetric difference of A and B: $(A \setminus B) \cup (B \setminus A)$
2^A	power set of A
$\overline{A}$	complement of A (with respect to a given universal set)
A^t	set of ordered t-tuples with elements from A
$\binom{A}{t}$	set of t-subsets of A
$\|A\|$	cardinality of A
$\emptyset$	empty set
$A \subset B$	A is a subset of B

Mappings

$f: A \to B$	f is a mapping from A to B
$f(x)$	image of x under the mapping f
$f: x \mapsto y$	f maps x to y: $f(x) = y$
$f(X)$	$\{f(x) : x \in X\}$ for $f: A \to B$ and $X \subset A$
supp f	support of f

Numbers

$\sum_{i=1}^{n} a_i$	$a_1 + \cdots + a_n$
$\prod_{i=1}^{n} a_i$	$a_1 a_2 \ldots a_n$
$\lceil x \rceil$	smallest integer $\geq x$ (for $x \in \mathbb{R}$)
$\lfloor x \rfloor$	largest integer $\leq x$ (for $x \in \mathbb{R}$)
$n!$	$n(n-1)(n-2)\ldots 1$ (for $n \in \mathbb{N}$)
$\binom{n}{t}$	number of t-subsets of an n-set
e	base of the natural logarithm

Matrices

A^T	transpose of the matrix A
J	matrix with all entries 1
I	identity matrix
$\mathrm{diag}(a_1, \ldots, a_n)$	diagonal matrix with entries $a_{11} = a_1, \ldots, a_{nn} = a_n$
(a_{ij})	matrix with entries a_{ij}
$\det A$, $\lvert A \rvert$	determinant of the matrix A
$\mathrm{per} A$	permanent of the matrix A

Sets of numbers and algebraic structures

$\mathbb{N}$ or $\mathbb{Z}^+$	set of natural numbers (not including 0)
$\mathbb{N}_0$	set of natural numbers including 0
$\mathbb{Z}$	ring of integers
$\mathbb{Z}_n$	ring of integers modulo n
$\mathbb{Q}$	field of rational numbers
$\mathbb{Q}^+$	set of positive rational numbers
$\mathbb{Q}_0^+$	set of non-negative rational numbers
$\mathbb{R}$	field of real numbers
$\mathbb{R}^+$	set of positive real numbers
$\mathbb{R}_0^+$	set of non-negative real numbers
K^*	multiplicative group of the field K
K^n	n-dimensional vector space over the field K
$K^{(n,n)}$	ring of $(n \times n)$-matrices over the field K
S_n	symmetric group acting on n elements

Miscellaneous

$x := y$, $y =: x$	x is defined to be y
$x \leftarrow y$	x is assigned the value of y

C.2 Special Symbols

This second part of the list contains symbols from graph theory and the symbols introduced in this book.

Graphs and networks

$\overline{G}$	complementary graph of the graph G
G_{red}	reduced digraph of the acyclic digraph G
$G\vert U$	subgraph of G induced on the vertex set U
$G \setminus e$	G with the edge e discarded
$G \setminus T$	subgraph of G induced on the set $V \setminus T$
$G \setminus v$	subgraph of G induced on the set $V \setminus \{v\}$ (for $v \in V$)
G/B	contraction of the graph G with respect to the blossom B
G/e	contraction of the graph G with respect to the edge e
$\vert G\vert$	multigraph underlying the directed multigraph G
(G)	underlying graph for the multigraph G
$\overrightarrow{G}$	complete orientation of the graph G
$[G]$	closure of the graph G
$bc(G)$	block-cutpoint graph of G
$G(H,S)$	Cayley graph defined b the group H and the set S
$G_{s,n}$	de Bruijn graph
$H_{\mathbf{u},\mathbf{v}}$	equality subgraph for G with respect to $(\mathbf{u},\mathbf{v})$
K_n	complete graph on n vertices
$K_{m,n}$	complete bipartite graph on $m+n$ vertices
$L(G)$	line graph of G
N', $N'(f)$	auxiliary network for N (with respect to the flow f)
N'', $N''(f)$	layered auxiliary network for N (with respect to the flow f)
$T(G)$	tree graph of the connected graph G
T_n	triangular graph on $\binom{n}{2}$ vertices

Objects in graphs

$C_T(e)$	cycle determined by the spanning tree T and the edge $e \notin T$
$a \overset{e}{—} b$	edge $e = ab$
e^-	the start vertex (*tail*) of the edge e
e^+	the end vertex (*head*) of the edge e
$E(S)$, $E(X,Y)$	edge set corresponding to the cut $S = (X,Y)$
$E\vert V'$	edge set induced on the vertex set V'
F_ε	set of ε-fixed edges (with respect to a given circulation)
$S_T(e)$	cut determined by the spanning tree T and the edge $e \in T$
$\Gamma(J)$	neighborhood of the vertex set J
$\Gamma(v)$	neighborhood of the vertex v

Parameters for graphs

$\text{ch}(G)$	choosability of G
$\deg v$	degree of the vertex v
$d_{\text{in}}(v)$	indegree of the vertex v
$d_{\text{out}}(v)$	outdegree of the vertex v
g	girth of G

n_d	number of vertices of degree d
$\alpha(G)$	independence number of G
$\alpha'(G)$	maximal cardinality of a matching of G
$\beta(G)$	minimal cardinality of a vertex cover of G
$\delta(G)$	minimal degree of a vertex of G
$\Delta(G)$	maximal degree of a vertex of G
Δ	minimal number of paths in a dissection of G (*Dilworth number*)
$\theta(G)$	clique partition number of G
$\kappa(G)$	connectivity of G
$\lambda(G)$	edge connectivity of G
$\nu(G)$	cyclomatic number of G
$\chi(G)$	chromatic number of G
$\chi'(G)$	chromatic index of G
$\omega(G)$	maximal cardinality of a clique in G

Mappings on graphs and networks

$a(x)$	supply at the vertex x
$b(e)$	lower capacity constraint for the edge e
$c(e)$	capacity of the edge e
$c(W)$	capacity of the path W
$c(S,T)$	capacity of the cut (S,T)
$d(x)$	demand at the vertex x
$d(a,b)$	distance between the vertices a and b
$d_H(a,b)$	distance between vertices a and b in the graph H
$f(S,T)$	flow value for the cut (S,T) with respect to the flow f
$m(K)$	mean weight of the cycle K
$o(S)$	number of odd components of $G \setminus S$
$p(v)$	flow potential at the vertex v
$r(v)$	rank of the vertex v in an acyclic digraph
$w(e)$	weight (or length) of the edge e
$w(f)$	value of the flow f
$w(P)$	weight of the optimal solution for the problem P
$w(X)$	weight (or length) of a set X of edges
$w(\pi)$	weight of the tour π
$w_A(P)$	weight of the solution for problem P determined by algorithm A
$w(s,t)$	value of a maximal flow between s and t (in a symmetric network)
$\gamma(e)$	cost of the edge e
$\gamma^{(\varepsilon)}(f)$	cost of the edge e increased by ε
$\gamma_p(e)$	reduced cost of the edge e (with respect to the potential p)
$\gamma(f)$	cost of the circulation or the flow f
$\gamma(M)$	cost of the perfect matching M
$\gamma(v)$	cost of an optimal flow with value v

δq	potential difference
$\varepsilon(f)$	optimality parameter for the circulation f
$\epsilon(v)$	excentricity of the vertex v
$\kappa(s,t)$	maximal number of vertex disjoint paths from s to t
$\lambda(s,t)$	maximal number of edge disjoint paths from s to t
$\mu(G,w)$	minimum cycle mean in the network (G,w)
$\pi_V(T)$	Prüfer code of the tree T on the vertex set V

Matroids and independence systems

$\mathrm{lr}(A)$	lower rank of the set A
$M(G)$	graphic matroid corresponding to G
M^*	dual matroid of the matroid M
$\overline{M}$	hereditary closure of the set system M
$\mathrm{rq}\ (M)$	rank quotient of the independence system M
$\mathrm{ur}\ (A)$	upper rank of the set A
$\mathbf{S}\|N$	restriction of the set system $\mathbf{S}$ to the set N
$\rho(A)$	rank of the set A
$\sigma(A)$	span of the set A

Matrices

A	adjacency matrix of a graph
A'	degree matrix of a graph
M	incidence matrix of a graph or a digraph
$\rho(A)$	term rank of the matrix A

Codes

$C_E(G)$	even graphical code based on G
$d(\mathbf{x},\mathbf{y})$	Hamming distance of $\mathbf{x}$ and $\mathbf{y}$
d	minimum distance of a code
k	dimension of a linear of a code
n	length of a code
$w(\mathbf{x})$	weight of $\mathbf{x}$

Miscellaneous

A_v	adjacency list for the vertex v
A'_v	reverse adjacency list for the vertex v
$d(\mathbf{a})$	deficiency of the set family $\mathbf{A}$
$t(\mathbf{A})$	transversal index of the set family $\mathbf{A}$
$O(f(n))$	upper bound on the complexity
$\Omega(f(n))$	lower bound on the complexity
$\Theta(f(n))$	rate of growth

References

Round up the usual suspects.
From 'Casablanca'

[AarLe97] Aarts, E., Lenstra, J.K.: Local Search in Combinatorial Optimization. Wiley, New York (1997)

[Abu90] Abu-Sbeih, M.Z.: On the number of spanning trees of K_n and $K_{m,n}$. Discrete Math. **84**, 205–207 (1990)

[AhoHU74] Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison Wesley, Reading (1974)

[AhoHU83] Aho, A.V., Hopcroft, J.E., Ullman, J.D.: Data Structures and Algorithms. Addison Wesley, Reading (1983)

[AhuOr89] Ahuja, R.K., Orlin, J.B.: A fast and simple algorithm for the maximum flow problem. Oper. Res. **37**, 748–759 (1989)

[AhuOr92] Ahuja, R.K., Orlin, J.B.: The scaling network simplex algorithm. Oper. Res. **40**(Suppl. 1), S5–S13 (1992)

[AhuMO89] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows. In: Nemhauser, G.L., Rinnooy Kan, A.H.G., Todd, M.J. (eds.) Handbooks in Operations Research and Management Science. Optimization, vol. 1, pp. 211–369. North Holland, Amsterdam (1989)

[AhuOT89] Ahuja, R.K., Orlin, J.B., Tarjan, R.E.: Improved time bounds for the maximum flow problem. SIAM J. Comput. **18**, 939–954 (1989)

[AhuMOT90] Ahuja, R.K., Mehlhorn, K., Orlin, J.B., Tarjan, R.E.: Faster algorithms for the shortest path problem. J. Assoc. Comput. Mach. **37**, 213–223 (1990)

[AhuMO91] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Some recent advances in network flows. SIAM Rev. **33**, 175–219 (1991)

[AhuGOT92] Ahuja, R.K., Goldberg, A.V., Orlin, J.B., Tarjan, R.E.: Finding minimum-cost flows by double scaling. Math. Program. **53**, 243–266 (1992)

[AhuKMO92] Ahuja, R.K., Kodialam, M., Mishra, A.K., Orlin, J.B.: Computational testing of maximum flow algorithms. Sloan working paper, Sloan School of Management, MIT (1992)

[AhuMO93] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs (1993)

[AhuOST94] Ahuja, R.K., Orlin, J.B., Stein, C., Tarjan, R.E.: Improved algorithm for bipartite network flow. SIAM J. Comput. **23**, 906–933 (1994)

[Aig84] Aigner, M.: Graphentheorie. Eine Entwicklung aus dem 4-Farben-Problem. Teubner, Stuttgart (1984)

[Aig97] Aigner, M.: Combinatorial Theory. Springer, New York (1997)

[Alo90] Alon, N.: Generating pseudo-random permutations and maximum flow algorithms. Inf. Process. Lett. **35**, 201–204 (1990)

D. Jungnickel, *Graphs, Networks and Algorithms*,
Algorithms and Computation in Mathematics 5,
DOI 10.1007/978-3-642-32278-5, © Springer-Verlag Berlin Heidelberg 2013

[Alt88] Althöfer, I.: On optimal realizations of finite metric spaces by graphs. Discrete Comput. Geom. **3**, 103–122 (1988)

[And71] Anderson, I.: Perfect matchings of a graph. J. Comb. Theory **10**, 183–186 (1971)

[And90] Anderson, I.: Combinatorial Designs: Construction Methods. Ellis Horwood, Chichester (1990)

[And97] Anderson, I.: Combinatorial Designs and Tournaments. Oxford University Press, Oxford (1997)

[And77] Anderson, L.D.: On edge-colorings of graphs. Math. Scand. **40**, 161–175 (1977)

[AndHa67] Anderson, S.S., Harary, F.: Trees and unicyclic graphs. Math. Teach. **60**, 345–348 (1967)

[Ans85] Anstee, R.P.: An algorithmic proof of Tutte's f-factor theorem. J. Algorithms **6**, 112–131 (1985)

[AppHa77] Appel, K., Haken, W.: Every planar map is 4-colorable. I. Discharging. Ill. J. Math. **21**, 429–490 (1977)

[AppHa89] Appel, K., Haken, W.: Every Planar Map is Four Colorable. American Mathematical Society, Providence (1989)

[AppHK77] Appel, K., Haken, W., Koch, J.: Every planar map is 4-colorable: II. Reducibility. Ill. J. Math. **21**, 491–567 (1989)

[AppCo93] Applegate, D., Cook, W.: Solving large-scale matching problems. In: Johnson, D.S., McGeoch, C.C. (eds.) Network Flows and Matching, pp. 557–576. Am. Math. Soc., Providence (1993)

[AppBCC95] Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Finding cuts in the TSP (A preliminary report). DIMACS Technical Report 95-05 (1995)

[AppBCC98] Applegate, D., Bixby, R., Chvátal, V., Cook, W.: On the solution of traveling salesman problems. Doc. Math. **III**, 645–656 (1998) (Extra Volume ICM 1998)

[AppBCC01] Applegate, D., Bixby, R., Chvátal, V., Cook, W.: TSP cuts which do not conform to the template paradigm. In: Jünger, M., Naddef, D. (eds.) Computational Combinatorial Optimization, pp. 261–304. Springer, Heidelberg (2001)

[AppBCC03] Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. Math. Program. **97B**, 91–153 (2003)

[AppBCC04] Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Concorde (2004). Available at www.tsp.gatech.edu

[AppBCC06] Applegate, D., Bixby, R., Chvátal, V., Cook, W.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2006)

[ArkPa86] Arkin, E.M., Papadimitriou, C.H.: On the complexity of circulations. J. Algorithms **7**, 134–145 (1986)

[Aro98] Arora, S.: Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. J. Assoc. Comput. Mach. **45**, 753–782 (1998)

[AroBa09] Arora, S., Barak, B.: Computational Complexity. A Modern Approach. Cambridge University Press, Cambridge (2009)

[AroKa06] Arora, S., Karakostas, G.: A $2+\varepsilon$ approximation algorithm for the k-MST problem. Math. Program. **107**, 491–504 (2006)

[AroSa02] Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. In: Proc. 33rd IEEE Symp. on Foundations of Computer Science, pp. 2–13 (1992)

[AroLMS92] Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and hardness of approximation problems. In: Proc. 33th IEEE Symp. on Foundations of Computer Science, pp. 14–23 (1992)

[AusIMN91] Ausiello, G., Italiano, G.F., Marchetti Spaccamela, A., Nanni, U.: Incremental algorithms for minimal length paths. J. Algorithms **12**, 615–638 (1991)

[Avi78] Avis, D.: Two greedy heuristics for the weighted matching problem. Congr. Numer. **21**, 65–76 (1978)

[Avi83] Avis, D.: A survey of heuristics for the weighted matching problem. Networks **13**, 475–493 (1983)

[BabFLS91] Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking computations in polylogarithmic time. In: Proc. 23rd ACM Symp. on Theory of Computing, pp. 21–31 (1991)

[BabFL91] Babai, L., Fortnow, L., Lund, C.: Nondeterministic exponential time has two-prover interactive protocols. Comput. Complex. **1**, 3–40 (1991)

[Bae53] Bäbler, F.: Über eine spezielle Klasse Eulerscher Graphen. Comment. Math. Helv. **21**, 81–100 (1953)

[BacKe92] Bachem, A., Kern, W.: Linear Programming Duality. An Introduction to Oriented Matroids. Springer, Berlin (1992)

[Bac89] Bachmann, F.: Ebene Spiegelungsgeometrie. B.I. Wissenschaftsverlag, Mannheim (1989)

[BakWi77] Baker, R.D., Wilson, R.M.: Nearly Kirkman triple systems. Util. Math. **11**, 289–296 (1977)

[BalFi93] Balas, E., Fischetti, M.: A lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets. Math. Program. **58**, 325–352 (1993)

[BalXu91] Balas, E., Xue, J.: Minimum weighted coloring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs. SIAM J. Comput. **20**, 209–221 (1991)

[BalYu86] Balas, E., Yu, C.S.: Finding a maximum clique in an arbitrary graph. SIAM J. Comput. **15**, 1054–1068 (1986)

[BalRa97] Balinski, M., Ratier, G.: On stable marriages and graphs, and strategy and polytopes. SIAM Rev. **39**, 575–604 (1997)

[BalGo91] Balinsky, M.L., Gonzales, J.: Maximum matchings in bipartite graphs via strong spanning trees. Networks **21**, 165–179 (1991)

[Bal85] Ball, M.O.: Polynomial algorithms for matching problems with side constraints. Research report CORR 85–21, University of Waterloo (1985)

[BalCo87] Ball, W.W.R., Coxeter, H.S.M.: Mathematical Recreations and Essays, 13th edn. Dover, New York (1987)

[BalDe83] Ball, M.O., Derigs, U.: An analysis of alternate strategies for implementing matching algorithms. Networks **13**, 517–549 (1983)

[Ban90] Bandelt, H.-J.: Recognition of tree matrices. SIAM J. Discrete Math. **3**, 1–6 (1990)

[BanGu09] Bang-Jensen, J., Gutin, G.Z.: Digraphs, 2nd edn. Springer, London (2009)

[BarKP93] Bar-Ilan, J., Kortsarz, G., Peleg, D.: How to allocate network centers. J. Algorithms **15**, 385–415 (1993)

[Bar90] Barahona, F.: On some applications of the Chinese postman problem. In: Korte, B., Lovász, L., Prömel, H.J., Schrijver, A. (eds.) Paths, Flows and VLSI-Layout, pp. 1–16. Springer, Berlin (1990)

[BarPu87] Barahona, F., Pulleyblank, W.R.: Exact arborescences, matchings and cycles. Discrete Appl. Math. **16**, 91–99 (1987)

[BarTa89] Barahona, F., Tardos, E.: Note on Weintraub's minimum-cost circulation algorithm. SIAM J. Comput. **18**, 579–583 (1989)

[Bar75] Baranyai, Z.: On the factorization of the complete uniform hypergraph. In: Proc. Erdős-Koll., Keszthely, 1973, pp. 91–108. North Holland, Amsterdam (1975)

[BarSa95] Barnes, T.M., Savage, C.D.: A recurrence for counting graphical partitions. Electron. J. Comb. **2**, # R 11 (1995)

[BauWo82] Bauer, F.L., Wössner, H.: Algorithmic Language and Program Development. Springer, Berlin (1982)

[BazSS06] Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms, 3rd edn. Wiley, New York (2006)

[BazJS10] Bazaraa, M.S., Jarvis, J.J., Sherali, H.D.: Linear Programming and Network Flows, 4th edn. Wiley, New York (2010)

[Bel58] Bellman, R.E.: On a routing problem. Q. Appl. Math. **16**, 87–90 (1958)

[Ben90] Bentley, J.L.: Experiments on traveling salesman heuristics. In: Proc. First SIAM Symp. on Discr. Algorithms, pp. 91–99 (1990)

[Ber57] Berge, C.: Two theorems in graph theory. Proc. Natl. Acad. Sci. USA **43**, 842–844 (1957)

[Ber58] Berge, C.: Sur le couplage maximum d'un graphe. C. R. Math. Acad. Sci. Paris **247**, 258–259 (1958)

[Ber61] Berge, C.: Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind (Zusammenfassung). Wiss. Z., Martin-Luther-Univ. Halle-Wittenb., Math.-Nat.wiss. Reihe **10**, 114–115 (1961)

[Ber73] Berge, C.: Graphs and Hypergraphs. North Holland, Amsterdam (1973)

[BerCh84] Berge, C., Chvátal, V.: Topics in Perfect Graphs. North Holland, Amsterdam (1984)

[BerFo91] Berge, C., Fournier, J.C.: A short proof for a generalization of Vizing's theorem. J. Graph Theory **15**, 333–336 (1991)

[BerGh62] Berge, C., Ghouila-Houri, A.: Programmes, Jeux et Réseaux de Transport. Dunod, Paris (1991)

[BerMT94] Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.A.: On the inherent intractability of certain coding problems. IEEE Trans. Inf. Theory **24**, 384–386 (1978)

[BerRa94] Berman, P., Ramaiyer, V.: Improved approximations for the Steiner tree problem. J. Algorithms **17**, 381–408 (1994)

[Ber78] Bermond, J.C.: Hamiltonian graphs. In: Beineke, L.W., Wilson, R.J. (eds.) Selected Topics in Graph Theory, pp. 127–167. Academic Press, New York (1978)

[Ber92] Bermond, J.C. (ed.): Interconnection Networks. North Holland, Amsterdam (1992)

[Ber93] Bertsekas, D.P.: A simple and fast label correcting algorithm for shortest paths. Networks **23**, 703–709 (1993)

[Bet74] Beth, T.: Algebraische Auflösungsalgorithmen für einige unendliche Familien von 3–Designs. Matematiche **29**, 105–135 (1974)

[BetJL99] Beth, T., Jungnickel, D., Lenz, H.: Design Theory, 2nd edn. vols. 1 and 2. Cambridge University Press, Cambridge (1999)

[Bie89] Bien, F.: Constructions of telephone networks by group representations. Not. Am. Math. Soc. **36**, 5–22 (1989)

[BieBM90] Bienstock, D., Brickell, E.F., Monma, C.N.: On the structure of minimum-weight k-connected spanning networks. SIAM J. Discrete Math. **3**, 320–329 (1990)

[Bie05] Bierbrauer, J.: Introduction to Coding Theory. Chapman & Hall/CRC, Boca Raton (2005)

[Big93] Biggs, N.L.: Algebraic Graph Theory, 2nd edn. Cambridge University Press, Cambridge (1993)

[BigLW76] Biggs, N.L., Lloyd, E.K., Wilson, R.J.: Graph Theory 1736–1936. Oxford University Press, Oxford (1976)

[Bir46] Birkhoff, G.: Tres observaciones sobre el algebra lineal. Rev. Univ. Nac. Tucumán Ser. A **5**, 147–151 (1946)

[BjoZi92] Bjørner, A., Ziegler, G.M.: Introduction to greedoids. In: White, N. (ed.) Matroid Applications, pp. 284–357. Cambridge University Press, Cambridge (1992)

[BjoLSW92] Bjørner, A., Las Vergnas, M., Sturmfels, B., White, N., Ziegler, G.M.: Oriented Matroids. Cambridge University Press, Cambridge (1992)

[Bla83] Blahut, R.E.: Theory and Practice of Error Control Codes. Addison-Wesley, Reading (1983)

[BlaSh89] Bland, R.G., Shallcross, D.F.: Large traveling salesman problems arising from experiments in X-ray crystallography: A preliminary report on computation. Oper. Res. Lett. **8**, 125–128 (1989)

[BobHa71] Bobrow, L.S., Hakimi, S.L.: Graph theoretic q-ary codes. IEEE Trans. Inf. Theory **17**, 215–218 (1971)

[BoeFW07] Böckenhauer, H.-J., Forlizzi, L., Hromkovič, J., Kneis, J., Kupke, J., Proietti, G., Widmayer, P.: On the approximability of TSP on local modifications of optimally solved instances. Algorithmic Oper. Res. **2**, 83–93 (2007)

[BoeHS11] Böckenhauer, H.-J., Hromkovič, J., Sprock, A.: Knowing all optimal solutions does not help for TSP reoptimization. In: Springer Lecture Notes in Computer Science, vol. 6610, pp. 7–15 (2011)

[BoeTi80] Boesch, F., Tindell, R.: Robbins theorem for mixed multigraphs. Am. Math. Mon. **87**, 716–719 (1980)

[Bol78] Bollobás, B.: Extremal Graph Theory. Academic Press, New York (1978)

[BonCh76] Bondy, J.A., Chvátal, V.: A method in graph theory. Discrete Math. **15**, 111–135 (1976)

[BonMu76] Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. North Holland, Amsterdam (1976)

[BonZi11] Bonsma, P., Zickfeld, F.: A 3/2-approximation algorithm for finding spanning trees with many leaves in cubic graphs. SIAM J. Discrete Math. **25**, 1652–1666 (2011)

[Boo94] Book, R.V.: Relativizations of the P =? NP and other problems: developments in structural complexity theory. SIAM Rev. **36**, 157–175 (1994)

[BooLu76] Booth, S., Lueker, S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**, 335–379 (1976)

[Bor60] Borchardt, C.W.: Über eine der Interpolation entsprechende Darstellung der Eliminationsresultante. J. Reine Angew. Math. **57**, 111–121 (1860)

[Bor87] Borgwardt, K.H.: The Simplex Method. A Probabilistic Analysis. Springer, Berlin (1987)

[Bor26a] Boruvka, O.: O jistém problému minimálním. Acta Soc. Sci. Nat. Morav. **3**, 37–58 (1926)

[Bor26b] Boruvka, O.: Príspevek k resení otázky ekonomické stavby elektrovodních sítí. Elektrotech. Obz. **15**, 153–154 (1926)

[Bos90] Bosák, J.: Decompositions of Graphs. Kluwer Academic, Dordrecht (1990)

[BoyFa90] Boyd, E.A., Faigle, U.: An algorithmic characterization of antimatroids. Discrete Appl. Math. **28**, 197–205 (1990)

[BreHa67] Bredeson, J.G., Hakimi, S.L.: Decoding of graph theoretic codes. IEEE Trans. Inf. Theory **13**, 348–349 (1967)

[Bro41] Brooks, R.L.: On colouring the nodes of a network. Proc. Camb. Philos. Soc. **37**, 194–197 (1941)

[BruBH90] Brucker, P., Burkard, R.E., Hurink, J.: Cyclic schedules for r irregularly occurring events. J. Comput. Appl. Math. **30**, 173–189 (1990)

[BruBr92] Bryant, V., Brooksbank, P.: Greedy algorithm compatibility and heavy-set structures. Eur. J. Comb. **13**, 81–86 (1992)

[Bun74] Bunemann, P.: A note on the metric properties of trees. J. Comb. Theory, Ser. B **17**, 48–50 (1974)

[Bur86] Burkard, R.E.: Optimal schedules for periodically recurring events. Discrete Appl. Math. **15**, 167–180 (1986)

[Bur97] Burkard, R.E.: Efficiently solvable special cases of hard combinatorial optimization problems. Math. Program. **79**, 55–69 (1997)

[BurHZ77] Burkard, R.E., Hahn, W., Zimmermann, U.: An algebraic approach to assignment problems. Math. Program. **12**, 318–327 (1977)

[BurDDW98] Burkard, R.E., Deineko, V.G., van Dal, R., van der Veen, J.A.A., Woeginger, G.J.: Well-solvable special cases of the traveling salesman problem: a survey. SIAM Rev. **40**, 496–546 (1998)

[BurGo61] Busacker, R.G., Gowen, P.J.: A procedure for determining a family of minimum cost flow networks. ORO Techn. Report 15, John Hopkins University (1961)

[BusPa06] Busygin, S., Pasechnik, D.V.: On NP-hardness of the clique partition—independence number gap recognition and related problems. Discrete Math. **306**, 460–463 (2006)

[CamFM79] Camerini, P.M., Fratta, L., Maffioli, F.: A note on finding optimum branchings. Networks **9**, 309–312 (1979)

[CamMMT85] Camerini, P.M., Maffioli, F., Martello, S., Toth, P.: Most and least uniform spanning trees. Discrete Appl. Math. **15**, 181–197 (1986)

[Cam76] Cameron, P.J.: Parallelisms of Complete Designs. Cambridge University Press, Cambridge (1976)

[CamLi91] Cameron, P.J., van Lint, J.H.: Designs, Graphs, Codes and Their Links. Cambridge University Press, Cambridge (1991)

[CamRa91] Campbell, D.M., Radford, D.: Tree isomorphism algorithms: speed versus clarity. Math. Mag. **64**, 252–261 (1991)

[CamFT89] Carpaneto, G., Fischetti, M., Toth, P.: New lower bounds for the symmetric travelling salesman problem. Math. Program. **45**, 233–254 (1989)

[Car71] Carré, P.A.: An algebra for network routing problems. J. Inst. Math. Appl. **7**, 273–294 (1971)

[Car79] Carré, P.A.: Graphs and Networks. Oxford University Press, Oxford (1979)

[Cat79] Catlin, P.A.: Hajós' graph coloring conjecture: variations and counterexamples. J. Comb. Theory, Ser. B **26**, 268–274 (1979)

[Cay89] Cayley, A.: A theorem on trees. Q. J. Math. **23**, 376–378 (1889)

[ChaHo09] Chandran, B.G., Hochbaum, D.S.: A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. Oper. Res. **57**, 358–376 (2009)

[ChaTa84] Chandrasekaran, R., Tamir, A.: Polynomial testing of the query 'Is $a^b \geq c^d$' with application to finding a minimal cost reliability ratio spanning tree. Discrete Appl. Math. **9**, 117–123 (1984)

[ChaAN81] Chandrasekaran, R., Aneja, Y.P., Nair, K.P.K.: Minimal cost reliability ratio spanning tree. Ann. Discrete Math. **11**, 53–60 (1981)

[ChaGK06] Charikar, M., Hoemans, M.X., Karloff, H.: On the integrality ratio for the asymmetric traveling salesman problem. Math. Oper. Res. **31**, 245–252 (2006)

[Cha66] Chartrand, G.: A graph-theoretic approach to communication problems. SIAM J. Appl. Math. **14**, 778–781 (1966)

[ChaHa68] Chartrand, G., Harary, F.: Graphs with described connectivities. In: Erdős, P., Katona, G. (eds.) Theory of Graphs, pp. 61–63. Academic Press, New York (1968)

[ChaWh70] Chartrand, G., White, A.T.: Randomly transversable graphs. Elem. Math. **25**, 101–107 (1970)

[CheHu92] Cheng, C.K., Hu, T.C.: Maximum concurrent flows and minimum cuts. Algorithmica **8**, 233–249 (1992)

[CheTa76] Cheriton, D., Tarjan, R.E.: Finding minimum spanning trees. SIAM J. Comput. **5**, 724–742 (1976)

[CheHa89] Cheriyan, J., Hagerup, T.: A randomized maximum flow algorithm. In: Proc. 30th IEEE Conf. on Foundations of Computer Science, pp. 118–123 (1989)

[CheHa95] Cheriyan, J., Hagerup, T.: A randomized maximum flow algorithm. SIAM J. Comput. **24**, 203–226 (1995)

[CheMa89] Cheriyan, J., Maheshwari, S.N.: Analysis of preflow push algorithms for maximum network flow. SIAM J. Comput. **18**, 1057–1086 (1989)

[CheHM96] Cheriyan, J., Hagerup, T., Mehlhorn, K.: A $O(n^3)$-time maximum flow algorithm. SIAM J. Comput. **25**, 1144–1170 (1996)

[CheGo95] Cherkassky, B.V., Goldberg, A.V.: On implementing Push-Relabel method for the maximum flow problem. In: Balas, E., Clausen, J. (eds.) Integer Programming and Combinatorial Optimization, pp. 157–171. Springer, Berlin (1995)

[Che80] Cheung, T.Y.: Computational comparison of eight methods for the maximum network flow problem. ACM Trans. Math. Softw. **6**, 1–16 (1980)

[Chr75] Christofides, N.: Graph Theory: An Algorithmic Approach. Academic Press, New York (1975)

[Chr76] Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388, Grad. School of Ind. Admin., Carnegie-Mellon University (1976)

[ChuLi65] Chu, Y.J., Liu, T.H.: On the shortest arborescence of a directed graph. Sci. Sin. **14**, 1396–1400 (1965)

[ChuRST02] Chudnovsky, M., Robertson, N., Seymour, P.D., Thomas, R.: The strong perfect graph theorem. Preprint (2002)

[ChuRST03] Chudnovsky, M., Robertson, N., Seymour, P.D., Thomas, R.: Progress on perfect graphs. Math. Program., Ser. B **97**, 405–422 (2003)

[Chu86] Chung, F.R.K.: Diameters of communication networks. Proc. Symp. Pure Appl. Math. **34**, 1–18 (1986)

[ChuGT85] Chung, F.R.K., Garey, M.R., Tarjan, R.E.: Strongly connected orientations of mixed multigraphs. Networks **15**, 477–484 (1985)

[Chv83] Chvátal, V.: Linear Programming. Freeman, New York (1983)

[Chv85] Chvátal, V.: Hamiltonian cycles. In: Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.) The Travelling Salesman Problem, pp. 403–429. Wiley, New York (1985)

[ChvEr72] Chvátal, V., Erdős, P.: A note on Hamiltonian circuits. Discrete Math. **2**, 111–113 (1972)

[ChvTh78] Chvátal, V., Thomasson, C.: Distances in orientations of graphs. J. Comb. Theory, Ser. B **24**, 61–75 (1978)

[Cie98] Cieslik, D.: Steiner Minimal Trees. Kluwer, Dordrecht (1998)

[Cie01] Cieslik, D.: The Steiner Ratio. Kluwer, Dordrecht (2001)

[Col87] Colbourn, C.J.: The Combinatorics of Network Reliability. Oxford University Press, Oxford (1987)

[ColDN89] Colbourn, C.J., Day, R.P.J., Nel, L.D.: Unranking and ranking spanning trees of a graph. J. Algorithms **10**, 271–286 (1989)

[ColHo82] Cole, R., Hopcroft, J.: On edge coloring bipartite graphs. SIAM J. Comput. **11**, 540–546 (1982)

[ConHMW92] Conrad, A., Hindrichs, T., Morsy, H., Wegener, I.: Wie es einem Springer gelingt, Schachbretter von beliebiger Größe zwischen beliebig vorgegebenen

Anfangs- und Endfeldern vollständig abzureiten. Spektrum der Wiss. 10–14 (1992)

[ConHMW94] Conrad, A., Hindrichs, T., Morsy, H., Wegener, I.: Solution of the knight's Hamiltonian path problem on chessboards. Discrete Appl. Math. **50**, 125–134 (1994)

[Coo71] Cook, S.A.: The complexity of theorem proving procedures. In: Proc. 3rd ACM Symp. on the Theory of Computing, pp. 151–158 (1971)

[Coo12] Cook, W.: In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press, Princeton (2011)

[CorLRS09] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)

[CorNe78] Cornuejols, G., Nemhauser, G.L.: Tight bounds for Christofides' traveling salesman heuristic. Math. Program. **14**, 116–121 (1978)

[CosLR05] Costa, M., Létocart, L., Roupin, F.: Minimal multicut and maximal integer multiflow: a survey. Eur. J. Oper. Res. **162**, 55–69 (2005)

[CouRo41] Courant, R., Robbins, H.: What is Mathematics? Oxford University Press, New York (1941)

[Cox61] Coxeter, H.M.S.: Introduction to Geometry. Wiley, New York (1961)

[Cox73] Coxeter, H.M.S.: Regular Polytopes, 3rd edn. Dover, New York (1973)

[Cro58] Croes, G.A.: A method for solving traveling-salesman problems. Oper. Res. **6**, 791–812 (1958)

[CroPa80] Crowder, H., Padberg, M.W.: Solving large-scale symmetric travelling salesman problems to optimality. Manag. Sci. **26**, 495–509 (1980)

[Cun86] Cunningham, W.H.: Improved bounds for matroid partition and intersection algorithms. SIAM J. Comput. **15**, 948–957 (1986)

[CunMa78] Cunningham, W.H., Marsh, A.B.: A primal algorithm for optimal matching. Math. Program. Stud. **8**, 50–72 (1978)

[CveDS80] Cvetkovic, D.M., Doob, M., Sachs, H.: Spectra of Graphs. Academic Press, New York (1980)

[CveDGT87] Cvetkovic, D.M., Doob, M., Gutman, I., Torgasev, A.: Recent Results in the Theory of Graph Spectra. North Holland, New York (1987)

[DanFJ54] Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solution of a large-scale traveling-salesman problem. Oper. Res. **2**, 393–410 (1954)

[DanFF56] Dantzig, G.B., Ford, L.R., Fulkerson, D.R.: A primal-dual algorithm for linear programs. In: Kuhn, H.W., Tucker, A.W. (eds.) Linear Inequalities and Related Systems, pp. 171–181. Princeton University Press, Princeton (1956)

[deB46] de Bruijn, N.G.: A combinatorial problem. Indag. Math. **8**, 461–467 (1946)

[deBA51] de Bruijn, N.G., van Aardenne-Ehrenfest, T.: Circuits and trees in oriented linear graphs. Simon Stevin **28**, 203–217 (1951)

[deW80] de Werra, D.: Geography, games and graphs. Discrete Appl. Math. **2**, 327–337 (1980)

[deW81] de Werra, D.: Scheduling in sports. Ann. Discrete Math. **11**, 381–395 (1981)

[deW82] de Werra, D.: Minimizing irregularities in sports schedules using graph theory. Discrete Appl. Math. **4**, 217–226 (1982)

[deW88] de Werra, D.: Some models of graphs for scheduling sports competitions. Discrete Appl. Math. **21**, 47–65 (1988)

[deWJM90] de Werra, D., Jacot-Descombes, L., Masson, P.: A constrained sports scheduling problem. Discrete Appl. Math. **26**, 41–49 (1990)

[DeiWo00] Deĭneko, V.G., Woeginger, G.: A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. Math. Program. **87**, 519–542 (2000)

[DeoPK82] Deo, N., Prabhu, G.M., Krishnamoorthy, M.S.: Algorithms for generating fundamental cycles in a graph. ACM Trans. Math. Softw. **8**, 26–42 (1982)

[Der88] Derigs, U.: Programming in Networks and Graphs. Springer, Berlin (1988)

[DerHe80] Derigs, U., Heske, A.: A computational study on some methods for solving the cardinality matching problem. Angew. Inform. **22**, 249–254 (1980)

[DerMe89] Derigs, U., Meier, W.: Implementing Goldberg's max-flow algorithm—A computational investigation. ZOR, Z. Oper.-Res. **33**, 383–403 (1989)

[DerMe91] Derigs, U., Metz, A.: Solving (large scale) matching problems combinatorially. Math. Program. **50**, 113–121 (1991)

[DeV12] DeVos, M.: The Gallai-Edmonds decomposition. Lecture Notes (2012). http://www.sfu.ca/~mdevos/notes/misc/gallai-edmonds.pdf

[Die10] Diestel, R.: Graph Theory, 4th edn. Springer, Berlin (2010)

[Dij59] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)

[Dil50] Dilworth, R.P.: A decomposition theorem for partially ordered sets. Ann. Math. **51**, 161–166 (1950)

[Din70] Dinic, E.A.: Algorithm for solution of a problem of maximum flow in networks with power estimation. Sov. Math. Dokl. **11**, 1277–1280 (1970)

[Dir52] Dirac, G.A.: Some theorems on abstract graphs. Proc. Lond. Math. Soc. **2**, 69–81 (1952)

[DixRT92] Dixon, B., Rauch, M., Tarjan, R.E.: Verification and sensitivity analysis of minimum spanning trees in linear time. SIAM J. Comput. **21**, 1184–1192 (1992)

[Dom89] Domschke, W.: Schedule synchronization for public transit networks. OR Spektrum **11**, 17–24 (1989)

[Dre84] Dress, A.: Trees, tight extensions of metric spaces, and the cohomological dimension of certain groups: A note on combinatorial properties of metric spaces. Adv. Math. **53**, 321–402 (1984)

[DuHw90a] Du, D.-Z., Hwang, F.: An approach for proving lower bounds: solution of Gilbert-Pollak's conjecture on Steiner ratio. In: Proc. 31st Annual Symp. on Foundations of Computer Science, Los Alamitos, CA, pp. 76–85. IEEE Computer Society, Los Alamitos (1990)

[DuHw90b] Du, D.-Z., Hwang, F.: The Steiner ratio conjecture of Gilbert and Pollak is true. Proc. Natl. Acad. Sci. USA **87**, 9464–9466 (1990)

[DuZh92] Du, D.-Z., Zhang, Y.: On better heuristics for Steiner minimum trees. Math. Program. **57**, 193–202 (1992)

[DuSR00] Du, D.-Z., Smith, J.M., Rubinstein, J.H.: Advances in Steiner Trees. Kluwer, Dordrecht (2000)

[Due93] Dueck, G.: New optimization heuristics: the great deluge algorithm and record-to-record travel. J. Comput. Phys. **104**, 86–92 (1993)

[DueSc90] Dueck, G., Scheuer, T.: Threshold accepting: A general purpose optimization algorithm appearing superior to simulating annealing. J. Comput. Phys. **90**, 161–175 (1990)

[DvoLS11] Dvořák, Z., Lidický, B., Škrekovski, R.: Graphs with two crossings are 5-choosable. SIAM J. Discrete Math. **25**, 1746–1753 (2011)

[Eco83] Eco, U.: The Name of the Rose. Harcourt Brace Jovanovich, San Diego (1983)

[Edm65a] Edmonds, J.: Maximum matching and a polytope with 0, 1-vertices. J. Res. Natl. Bur. Stand. B **69**, 125–130 (1965)

[Edm65b] Edmonds, J.: Paths, trees and flowers. Can. J. Math. **17**, 449–467 (1965)

[Edm67a] Edmonds, J.: An introduction to matching. Lecture Notes, Univ. of Michigan (1967)

[Edm67b] Edmonds, J.: Optimum branchings. J. Res. Natl. Bur. Stand. B **71**, 233–240 (1967)

[Edm70] Edmonds, J.: Submodular functions, matroids and certain polyhedra. In:

Guy, K. (ed.) Combinatorial Structures and Their Applications, pp. 69–87. Gordon & Breach, New York (1970)

[Edm71] Edmonds, J.: Matroids and the greedy algorithm. Math. Program. **1**, 127–136 (1971)

[Edm73] Edmonds, J.: Edge disjoint branchings. In: Rustin, R. (ed.) Combinatorial Algorithms, pp. 91–96. Algorithmics Press, New York (1973)

[Edm79] Edmonds, J.: Matroid intersection. Ann. Discrete Math. **4**, 39–49 (1979)

[EdmFu65] Edmonds, J., Fulkerson, D.R.: Transversals and matroid partition. J. Res. Natl. Bur. Stand. B **69**, 147–153 (1965)

[EdmGi77] Edmonds, J., Giles, R.: A min-max relation for submodular functions on graphs. Ann. Discrete Math. **1**, 185–204 (1977)

[EdmGi84] Edmonds, J., Giles, R.: Total dual integrality of linear systems. In: Pulleyblank, W.R. (ed.) Progress in Combinatorial Optimization, pp. 117–129. Academic Press Canada, Don Mills (1984)

[EdmJo73] Edmonds, J., Johnson, E.L.: Matching, Euler tours and the Chinese postman. Math. Program. **5**, 88–124 (1973)

[EdmKa72] Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. J. Assoc. Comput. Mach. **19**, 248–264 (1972)

[EdmLP82] Edmonds, J., Lovász, L., Pulleyblank, W.R.: Brick decompositions and the matching rank of graphs. Combinatorica **2**, 247–274 (1982)

[Ege31] Egerváry, E.: Matrixok kombinatorius tulajdonságairól. Mat. Fiz. Lapok **38**, 16–28 (1931)

[Ego81] Egoritsjev, G.E.: Solution of van der Waerden's permanent conjecture. Adv. Math. **42**, 299–305 (1981)

[EliFS56] Elias, P., Feinstein, A., Shannon, C.E.: Note on maximum flow through a network. IRE Trans. Inf. Theory **IT-12**, 117–119 (1956)

[Eng97] Engel, K.: Sperner Theory. Cambridge University Press, Cambridge (1997)

[Epp94] Eppstein, D.: Offline algorithms for dynamic minimum spanning tree problems. J. Algorithms **17**, 237–250 (1994)

[ErdSz35] Erdős, P., Szekeres, G.: A combinatorial problem in geometry. Compos. Math. **2**, 463–470 (1935)

[ErdRT80] Erdős, P., Rubin, A.L., Taylor, H.: Choosability in graphs. Congr. Numer. **26**, 125–157 (1980)

[ErvMcC93] Ervolina, T.R., McCormick, S.T.: Two strongly polynomial cut cancelling algorithms for minimum cost network flow. Discrete Appl. Math. **46**, 133–165 (1993)

[Etz86] Etzion, T.: An algorithm for constructing m-ary de Bruijn sequences. J. Algorithms **7**, 331–340 (1986)

[Eul36] Euler, L.: Solutio problematis ad geometriam situs pertinentis. Comment. Acad. Sci. Imp. Petropol. **8**, 128–140 (1736)

[Eul52/53] Euler, L.: Demonstratio nonnullorum insignium proprietatum quibus solida hadris planis inclusa sunt praedita. Novi Comment. Acad. Sci. Petropol. **4**, 140–160 (1752/1753)

[Eul66] Euler, L.: Solution d'une question curieuse qui ne paroit soumise à aucune analyse. Mém. Acad. R. Sci. Belles Lettres, Année 1759 **15**, 310–337 (1766)

[Eve73] Even, S.: Combinatorial Algorithms. Macmillan, New York (1973)

[Eve77] Even, S.: Algorithm for determining whether the connectivity of a graph is at least k. SIAM J. Comput. **6**, 393–396 (1977)

[Eve79] Even, S.: Graph Algorithms. Computer Science Press, Rockville (1979)

[EveTa75] Even, S., Tarjan, R.E.: Network flow and testing graph connectivity. SIAM J. Comput. **4**, 507–512 (1975)

[EveIS76] Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. **5**, 691–703 (1976)

[EveGT77] Even, S., Garey, M.R., Tarjan, R.E.: A note on connectivity and circuits in directed graphs. Unpublished manuscript (1977)

[Fai79] Faigle, U.: The greedy algorithm for partially ordered sets. Discrete Math. **28**, 153–159 (1979)

[Fai85] Faigle, U.: On ordered languages and the optimization of linear functions by greedy algorithms. J. Assoc. Comput. Mach. **32**, 861–870 (1985)

[Fal81] Falikman, D.I.: Proof of the van der Waerden conjecture regarding the permanent of a doubly stochastic matrix. Math. Notes - Ross. Akad. **29**, 475–479 (1981)

[FarMi60] Farahat, H.K., Mirsky, L.: Permutation endomorphisms and a refinement of a theorem of Birkhoff. Proc. Camb. Philos. Soc. **56**, 322–328 (1960)

[FedMo95] Feder, T., Motwani, R.: Clique partitions, graph compression and speeding up algorithms. J. Comput. Syst. Sci. **51**, 261–272 (1995)

[Fie94] Fiechter, C.-N.: A parallel tabu search algorithm for large traveling salesman problems. Discrete Appl. Math. **51**, 243–267 (1994)

[FieSe58] Fiedler, M., Sedlacek, J.: O W-basich orientovanych grafu. Čas. Pěst. Mat. **83**, 214–225 (1958)

[FisHJM94] Fischetti, M., Hamacher, H.W., Jørnsten, K.O., Maffioli, F.: Weighted k-cardinality trees: complexity and polyhedral structure. Networks **24**, 11–21 (1994)

[Fis85] Fishburn, P.C.: Interval Orders and Interval Graphs: A Study of Partially Ordered Sets. Wiley, New York (1985)

[Fis81] Fisher, M.L.: The Langrangian method for solving integer programming problems. Manag. Sci. **27**, 1–18 (1981)

[FleSk07] Fleischer, L., Skutella, M.: Quickest flows over time. SIAM J. Comput. **36**, 1600–1630 (2007)

[Fle83] Fleischner, H.: Eulerian graphs. In: Beineke, L.W., Wilson, R.J. (eds.) Selected Topics in Graph Theory 2, pp. 17–53. Academic Press, New York (1983)

[Fle90] Fleischner, H.: Eulerian Graphs and Related Topics, Part 1, vol. 1. North Holland, Amsterdam (1990)

[Fle91] Fleischner, H.: Eulerian Graphs and Related Topics, Part 1, vol. 2. North Holland, Amsterdam (1991)

[Flo62] Floyd, R.W.: Algorithm 97, shortest path. Commun. ACM **5**, 345 (1962)

[For56] Ford, L.R.: Network Flow Theory. Rand Corp., Santa Monica (1956)

[ForFu56] Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Can. J. Math. **8**, 399–404 (1956)

[ForFu57] Ford, L.R., Fulkerson, D.R.: A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. Can. J. Math. **9**, 210–218 (1957)

[ForFu58a] Ford, L.R., Fulkerson, D.R.: Constructing maximal dynamic flows from static flows. Oper. Res. **6**, 419–433 (1958)

[ForFu58b] Ford, L.R., Fulkerson, D.R.: Network flow and systems of representatives. Can. J. Math. **10**, 78–84 (1958)

[ForFu58c] Ford, L.R., Fulkerson, D.R.: A suggested computation for maximum multicommodity network flows. Manag. Sci. **5**, 97–101 (1958)

[ForFu62] Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, Princeton (1962)

[Fra92] Frank, A.: Augmenting graphs to meet edge-connectivity requirements. SIAM J. Discrete Math. **5**, 25–53 (1992)

[FraJo95] Frank, A., Jordán, T.: Minimal edge-coverings of pairs of sets. J. Comb. Theory, Ser. B **65**, 73–110 (1995)

[FraTa88] Frank, A., Tardos, E.: Generalized polymatroids and submodular flows. Math. Program. **42**, 489–563 (1988)

[Fre85] Frederickson, G.N.: Data structures for on-line updating of minimum spanning trees, with applications. SIAM J. Comput. **14**, 781–798 (1985)

[Fre87] Frederickson, G.N.: Fast algorithms for shortest paths in planar graphs. SIAM J. Comput. **16**, 1004–1022 (1987)

[FreTa87] Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses on improved network optimization algorithms. J. Assoc. Comput. Mach. **34**, 596–615 (1987)

[FreWi94] Fredman, M.L., Willard, D.E.: Trans-dichotomous algorithms for minimum spanning trees and shortest paths. J. Comput. Syst. Sci. **48**, 533–551 (1994)

[FreJu99a] Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows I. A unifying framework for design and analysis of matching algorithms. Networks **33**, 1–28 (1999)

[FreJu99b] Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows II. Simple augmentation algorithms. Networks **33**, 29–41 (1999)

[FreJu99c] Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows III. Strongly polynomial augmentation algorithms. Networks **33**, 43–56 (1999)

[FreJu01a] Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows IV. Duality and structure theory. Networks **37**, 194–201 (2001)

[FreJu01b] Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows V. Cycle canceling algorithms. Networks **37**, 202–209 (2001)

[FreJu01c] Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows VI. Polyhedral descriptions. Networks **37**, 210–218 (2001)

[FreJu02] Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows VII. Primal-dual algorithms. Networks **39**, 135–142 (2002)

[FreJu03] Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows VIII. A revised theory of phase ordered algorithms and the $O(\sqrt{n}m \log(n^2/m)/\log n)$ bound for the non-bipartite cardinality matching problem. Networks **41**, 137–142 (2003)

[Fro12] Frobenius, G.: Über Matrizen aus nicht negativen Elementen. Sitz.ber. Preuss. Akad. Wiss. **1912**, 456–477 (1912)

[Fuj86] Fujishige, S.: An $O(m^3 \log n)$ capacity-rounding algorithm for the minimum-cost circulation problem: A dual framework of Tardos' algorithm. Math. Program. **35**, 298–308 (1986)

[Fuj91] Fujishige, S.: Submodular Functions and Optimization. North Holland, Amsterdam (1991)

[Ful56] Fulkerson, D.R.: Note on Dilworth's decomposition theorem for partially ordered sets. Proc. Am. Math. Soc. **7**, 701–702 (1956)

[Ful59] Fulkerson, D.R.: Increasing the capacity of a network: the parametric budget problem. Manag. Sci. **5**, 472–483 (1959)

[Ful61] Fulkerson, D.R.: An out-of-kilter method for minimal cost flow problems. J. Soc. Ind. Appl. Math. **9**, 18–27 (1961)

[Gab76] Gabow, H.N.: An efficient implementation of Edmonds' algorithm for maximum matchings on graphs. J. Assoc. Comput. Mach. **23**, 221–234 (1976)

[Gab90] Gabow, H.N.: Data structures for weighted matching and nearest common ancestors with linking. In: Proc. First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 434–443. Soc. Ind. Appl. Math., Philadelphia (1990)

[GabKa82] Gabow, H.N., Kariv, O.: Algorithms for edge coloring bipartite graphs and multigraphs. SIAM J. Comput. **11**, 117–129 (1982)

[GabTa88] Gabow, H.N., Tarjan, R.E.: Algorithms for two bottleneck optimization problems. J. Algorithms **9**, 411–417 (1988)

[GabTa89] Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for network problems. SIAM J. Comput. **18**, 1013–1036 (1989)

[GabTa91] Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for general graph-matching problems. J. Assoc. Comput. Mach. **38**, 815–853 (1991)

[GabGST86] Gabow, H.N., Galil, Z., Spencer, T., Tarjan, R.E.: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica **6**, 109–122 (1986)

[Gal57] Gale, D.: A theorem on flows in networks. Pac. J. Math. **7**, 1073–1082 (1957)

[Gal68] Gale, D.: Optimal assignments in an ordered set: an application of matroid theory. J. Comb. Theory **4**, 176–180 (1968)

[GalSh62] Gale, D., Shepley, L.S.: College admissions and the stability of marriage. Am. Math. Mon. **69**, 9–15 (1962)

[Gal80] Galil, Z.: Finding the vertex connectivity of graphs. SIAM J. Comput. **9**, 197–199 (1980)

[Gal81] Galil, Z.: On the theoretical efficiency of various network flow algorithms. Theor. Comput. Sci. **14**, 103–111 (1981)

[GalSc88] Galil, Z., Schieber, B.: On funding most uniform spanning trees. Discrete Appl. Math. **20**, 173–175 (1988)

[GalTa88] Galil, Z., Tardos, E.: An $O(n^2(m + n \log n) \log n)$ min-cost flow algorithm. J. Assoc. Comput. Mach. **35**, 374–386 (1986)

[Gal64a] Gallai, T.: Elementare Relationen bezüglich der Glieder und trennenden Punkte eines Graphen. Magy. Tud. Akad. Mat. Kut. Intéz. K&quot;ozl. **9**, 235–236 (1964)

[Gal64b] Gallai, T.: Maximale Systeme unabhängiger Kanten. Publ. Math. Inst. Hung. Acad. Sci., Ser. A **9**, 401–413 (1964)

[Gal67] Gallai, T.: Transitiv orientierbare Graphen. Acta Math. Acad. Sci. Hung. **18**, 25–66 (1967)

[GaMi60] Gallai, T., Milgram, A.N.: Verallgemeinerung eines graphentheoretischen Satzes von Redéi. Acta Sci. Math. **21**, 181–186 (1960)

[GalGT89] Gallo, G., Grigoriades, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**, 30–55 (1989)

[GalPa88] Gallo, G., Pallottino, S.: Shortest path algorithms. Ann. Oper. Res. **13**, 3–79 (1988)

[GarJo76] Garey, M.R., Johnson, D.S.: The complexity of near-optimal graph coloring. J. Assoc. Comput. Mach. **23**, 43–49 (1976)

[GarJo79] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)

[GarJS76] Garey, M.R., Johnson, D.S., Stockmeyer, L.J.: Some simplified NP-complete graph problems. Theor. Comput. Sci. **1**, 237–267 (1976)

[GarJT76] Garey, M.R., Johnson, D.S., Tarjan, R.E.: The planar Hamiltonian circuit problem is NP-complete. SIAM J. Comput. **5**, 704–714 (1976)

[GarGJ77] Garey, M.R., Graham, R.L., Johnson, D.S.: The complexity of computing Steiner minimal trees. SIAM J. Appl. Math. **32**, 835–859 (1977)

[Gas96] Gasparyan, G.S.: Minimal imperfect graphs: A simple approach. Combinatorica **16**, 209–212 (1996)

[GhiJu90] Ghinelli, D., Jungnickel, D.: The Steinberg module of a graph. Arch. Math. **55**, 503–506 (1990)

[Gho62] Ghouila-Houri, A.: Caractérisation des graphes non orientés dont on peut orienter les arêtes de manière à obtenir le graphe d'une relation d'ordre. C. R. Math. Acad. Sci. Paris **254**, 1370–1371 (1962)

[GilPo68] Gilbert, E.N., Pollak, H.O.: Steiner minimal trees. SIAM J. Appl. Math. **16**, 1–29 (1968)

[GilHo64] Gilmore, P.C., Hoffman, A.J.: A characterization of comparability graphs and of interval graphs. Can. J. Math. **16**, 539–548 (1964)

[GlKP92] Glover, F., Klingman, D., Phillips, N.V.: Network Models in Optimization and Their Applications in Practice. Wiley, New York (1992)

[God93] Godsil, C.D.: Algebraic Combinatorics. Chapman and Hall, New York (1993)

[GodRo01] Godsil, C.D., Royle, G.: Algebraic Graph Theory. Springer, New York (2001)

[Goe88] Goecke, O.: A greedy algorithm for hereditary set systems and a generalization of the Rado-Edmonds characterization of matroids. Discrete Appl. Math. **20**, 39–49 (1988)

[GolKa96] Goldberg, A., Karzanov, A.V.: Path problems in skew-symmetric graphs. Combinatorica **16**, 353–382 (1996)

[GolKa04] Goldberg, A., Karzanov, A.V.: Maximum skew-symmetric flows and matchings. Math. Program. **100**, 537–568 (2004)

[GolRa99] Goldberg, A., Rao, S.: Flows in undirected unit capacity networks. SIAM J. Discrete Math. **12**, 1–5 (1999)

[GolTa88] Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum flow problem. J. Assoc. Comput. Mach. **35**, 921–940 (1988)

[GolTa89] Goldberg, A.V., Tarjan, R.E.: Finding minimum-cost circulations by canceling negative cycles. J. Assoc. Comput. Mach. **36**, 873–886 (1989)

[GolTa90] Goldberg, A.V., Tarjan, R.E.: Solving minimum cost-flow problems by successive approximation. Math. Oper. Res. **15**, 430–466 (1990)

[GolTT90] Goldberg, A.V., Tardos, E., Tarjan, R.E.: Network flow algorithms. In: Korte, B., Lovász, L., Prömel, H.J., Schrijver, A. (eds.) Paths, Flows and VLSI-layout, pp. 101–164. Springer, Berlin (1990)

[GolGT91] Goldberg, A.V., Grigoriadis, M.D., Tarjan, R.E.: Use of dynamic trees in a network simplex algorithm for the maximum flow problem. Math. Program. **50**, 277–290 (1991)

[GolPT91] Goldberg, A.V., Plotkin, S.A., Tardos, E.: Combinatorial algorithms for the generalized circulation problem. Math. Oper. Res. **16**, 351–381 (1991)

[GolGr88] Goldfarb, D., Grigoriadis, M.D.: A computational comparison of the Dinic and network simplex methods for maximum flow. Ann. Oper. Res. **13**, 83–123 (1988)

[GolHa90] Goldfarb, D., Hao, J.: A primal simplex algorithm that solves the maximum flow problem in at most nm pivots and $O(n^2m)$ time. Math. Program. **47**, 353–365 (1990)

[GolHa91] Goldfarb, D., Hao, J.: On strongly polynomial variants of the network simplex algorithm for the maximum flow problem. Oper. Res. Lett. **10**, 383–387 (1991)

[GolJL02] Goldfarb, D., Jin, Z., Lin, Y.: A polynomial dual simplex algorithm for the generalized circulation problem. Math. Program. **91**, 271–288 (2002)

[Gol67] Golomb, S.W.: Shift Register Sequences. Holden-Day, San Francisco (1967)

[Gol80] Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York (1980)

[GomHu61] Gomory, R.E., Hu, T.C.: Multi-terminal network flows. J. Soc. Ind. Appl. Math. **9**, 551–570 (1961)

[GomHu62] Gomory, R.E., Hu, T.C.: An application of generalized linear programming to network flows. J. Soc. Ind. Appl. Math. **10**, 260–283 (1962)

[GomHu64] Gomory, R.E., Hu, T.C.: Synthesis of a communication network. J. Soc. Ind. Appl. Math. **12**, 348–369 (1964)

[GonMi84] Gondran, M., Minoux, N.: Graphs and Algorithms. Wiley, New York (1984)

[Gon92] Gonzaga, C.C.: Path-following methods for linear programming. SIAM Rev. **34**, 167–224 (1992)

[GouJa83] Goulden, I.P., Jackson, D.M.: Combinatorial Enumeration. Wiley, New York (1983)

[GraHe85] Graham, R.L., Hell, P.: On the history of the minimum spanning tree problem. Ann. Hist. Comput. **7**, 43–57 (1985)

[GreKl78] Greene, R.C., Kleitman, D.J.: Proof techniques in the theory of finite sets. In: Rota, G.C. (ed.) Studies in Combinatorics, pp. 22–79. Math. Assoc. America, Washington (1978)

[Gri88] Griggs, J.R.: Saturated chains of subsets and a random walk. J. Comb. Theory, Ser. A **47**, 262–283 (1988)

[GriRo96] Griggs, T., Rosa, A.: A tour of European soccer schedules, or testing the popularity of GK_{2n}. Bull. Inst. Comb. Appl. **18**, 65–68 (1996)

[GriKa88] Grigoriadis, M.D., Kalantari, B.: A new class of heuristic algorithms for weighted perfect matching. J. Assoc. Comput. Mach. **35**, 769–776 (1988)

[Gro80] Grötschel, M.: On the symmetric travelling salesman problem: solution of a 120-city problem. Math. Program. Stud. **12**, 61–77 (1980)

[Gro84] Grötschel, M.: Developments in combinatorial optimization. In: Jäger, W., Moser, J., Remmert, R. (eds.) Perspectives in Mathematics: Anniversary of Oberwolfach 1984, pp. 249–294. Birkhäuser, Basel (1984)

[Gro85] Grötschel, M.: Operations Research I. Lecture Notes, Universität Augsburg (1985)

[GroHo85] Grötschel, M., Holland, G.: Solving matching problems with linear programming. Math. Program. **33**, 243–259 (1985)

[GroHo91] Grötschel, M., Holland, O.: Solution of large-scale symmetric travelling salesman problems. Math. Program. **51**, 141–202 (1991)

[GroLS84] Grötschel, M., Lovász, L., Schrijver, A.: Polynomial algorithms for perfect graphs. Ann. Discrete Math. **21**, 325–356 (1984)

[GroJR91] Grötschel, M., Jünger, M., Reinelt, G.: Optimal control of plotting and drilling machines: a case study. ZOR, Z. Oper.-Res. **35**, 61–84 (1991)

[GroLS93] Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, 2nd edn. Springer, Berlin (1993)

[Guc96] Guckert, M.: Anschlußoptimierung in öffentlichen Verkehrsnetzen—Graphentheoretische Grundlagen, objektorientierte Modellierung und Implementierung. Ph.D. Thesis, Universität Marburg (1996)

[Gul80] Guldan, F.: Maximization of distances of regular polygones on a circle. Appl. Math. **25**, 182–195 (1980)

[Gus87] Gusfield, D.: Three fast algorithms for four problems in stable marriage. SIAM J. Comput. **16**, 111–128 (1987)

[Gus88] Gusfield, D.: The structure of the stable roommate problem: efficient representation and enumeration of all stable assignments. SIAM J. Comput. **17**, 742–769 (1988)

[Gus90] Gusfield, D.: Very simple methods for all pairs network flow analysis. SIAM J. Comput. **19**, 143–155 (1990)

[GusIr89] Gusfield, D., Irving, R.W.: The Stable Marriage Problem. Structure and Algorithms. The MIT Press, Cambridge (1989)

[GusNa91] Gusfield, D., Naor, D.: Efficient algorithms for generalized cut-trees. Networks **21**, 505–520 (1991)

[GusMF87] Gusfield, D., Martel, C., Fernandez-Baca, D.: Fast algorithms for bipartite network flow. SIAM J. Comput. **16**, 237–251 (1987)

[GusPa02] Gutin, G., Punnen, A.: The Traveling Salesman Problem and Its Variations. Kluwer, Dordrecht (2002)

[Had61] Hadley, G.: Linear Algebra. Addison-Wesley, Reading (1961)

[Had75] Hadlock, F.O.: Finding a maximum cut of a planar graph in polynomial time. SIAM J. Comput. **4**, 221–225 (1975)

[Had43] Hadwiger, H.: Über eine Klassifikation der Streckenkomplexe. Vierteljahrsschr. Nat.forsch. Ges. Zür. **88**, 133–142 (1943)

[Haj61] Hajós, G.: Über eine Konstruktion nicht n-färbbarer Graphen. Wiss. Z., Martin-Luther-Univ. Halle-Wittenb., Math.-Nat.wiss. Reihe **10**, 116–117 (1961)

[Hak66] Hakimi, S.L.: Recent progress and new problems in applied graph theory. In: Proc. IEEE Region Six Annual Conf., pp. 635–643 (1966)

[HakBr68] Hakimi, S.L., Bredeson, J.G.: Graph theoretic error-correcting codes. IEEE Trans. Inf. Theory **14**, 584–591 (1968)

[HakBr69] Hakimi, S.L., Bredeson, J.G.: Ternary graph theoretic error-correcting codes. IEEE Trans. Inf. Theory **15**, 435–436 (1969)

[HakFr65] Hakimi, S.L., Frank, H.: Cut set matrices and linear codes. IEEE Trans. Inf. Theory **11**, 457–458 (1965)

[HakVa64] Hakimi, S.L., Yau, S.S.: Distance matrix of a graph and its realizability. Q. Appl. Math. **22**, 305–317 (1964)

[Hal56] Hall, M.: An algorithm for distinct representatives. Am. Math. Mon. **63**, 716–717 (1956)

[Hal86] Hall, M.: Combinatorial Theory, 2nd edn. Wiley, New York (1986)

[Hal35] Hall, P.: On representatives of subsets. J. Lond. Math. Soc. **10**, 26–30 (1935)

[HalVa50] Halmos, P.R., Vaughan, H.E.: The marriage problem. Am. J. Math. **72**, 214–215 (1950)

[HamRu94] Hamacher, H.W., Ruhe, G.: On spanning tree problems with multiple objectives. Ann. Oper. Res. **52**, 209–230 (1994)

[Har62] Harary, F.: The maximum connectivity of a graph. Proc. Natl. Acad. Sci. USA **48**, 1142–1146 (1962)

[Har69] Harary, F.: Graph Theory. Addison Wesley, Reading (1969)

[HarTu65] Harary, F., Tutte, W.T.: A dual form of Kuratowski's theorem. Can. Math. Bull. **8**, 17–20 and 173 (1965)

[HasJo85] Hassin, R., Johnson, D.B.: An $O(n \log^2 n)$ algorithm for maximum flow in undirected planar networks. SIAM J. Comput. **14**, 612–624 (1985)

[HauKo81] Hausmann, D., Korte, B.: Algorithmic versus axiomatic definitions of matroids. Math. Program. Stud. **14**, 98–111 (1981)

[Hea90] Heawood, P.J.: Map colour theorem. Q. J. Pure Appl. Math. **24**, 332–338 (1890)

[HelKa70] Held, M., Karp, R.: The travelling salesman problem and minimum spanning trees. Oper. Res. **18**, 1138–1162 (1970)

[HelKa71] Held, M., Karp, R.: The travelling salesman problem and minimum spanning trees II. Math. Program. **1**, 6–25 (1971)

[HelWC74] Held, M., Wolfe, P., Crowder, H.P.: Validation of subgradient optimization. Math. Program. **6**, 62–88 (1974)

[HelMS93] Helman, P., Mont, B.M.E., Shapiro, H.D.: An exact characterization of greedy structures. SIAM J. Discrete Math. **6**, 274–283 (1993)

[Hel00] Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur. J. Oper. Res. **126**, 106–130 (2000)

[Her67] Herz, J.C.: Cours de Théorie des Graphes. Faculté des Sciences de Lille (1967)

[Hie73] Hierholzer, C.: Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Math. Ann. **6**, 30–32 (1873)

[HilJa87] Hilton, A.J.W., Jackson, B.: A note concerning the chromatic index of multigraphs. J. Graph Theory **11**, 267–272 (1987)

[Hit41] Hitchcock, F.L.: The distribution of a product from several sources to numerous localities. J. Math. Phys. **20**, 224–230 (1941)

[HoLC91] Ho, J.-M., Lee, D.T., Chang, C.-H., Wong, C.K.: Minimum diameter spanning trees and related problems. SIAM J. Comput. **20**, 987–997 (1991)

[Hoc08] Hochbaum, D.S.: The pseudoflow algorithm: A new algorithm for the maximum-flow problem. Oper. Res. **56**, 992–1009 (2008)

[HocNS86] Hochbaum, D.S., Nishizeki, T., Shmoys, D.B.: A better than 'best possible' algorithm to edge color multigraphs. J. Algorithms **7**, 79–104 (1986)

[Hof60] Hoffman, A.J.: Some recent applications of the theory of linear inequalities to extremal combinatorial analysis. In: Bellman, R.E., Hall, M. (eds.) Combinatorial Analysis, pp. 113–127. Am. Math. Soc., Providence (1960)

[Hof74] Hoffman, A.J.: A generalization of max flow-min cut. Math. Program. **6**, 352–359 (1974)

[Hof79] Hoffman, A.J.: The role of unimodularity in applying linear inequalities to combinatorial theorems. Ann. Discrete Math. **4**, 73–84 (1979)

[HofKr56] Hoffman, A.J., Kruskal, J.B.: Integral boundary points of convex polyhedra. In: Kuhn, H.W., Tucker, A.W. (eds.) Linear Inequalities and Related Systems, pp. 233–246. Princeton University Press, Princeton (1956)

[HofKu56] Hoffman, A.J., Kuhn, H.W.: On systems of distinct representatives. Ann. Math. Stud. **38**, 199–206 (1956)

[HofMa64] Hoffman, A.J., Markowitz, H.M.: A note on shortest path, assignment and transportation problems. Nav. Res. Logist. Q. **10**, 375–379 (1963)

[Hol81] Holyer, I.J.: The NP-completeness of edge-coloring. SIAM J. Comput. **10**, 718–720 (1981)

[HopKa73] Hopcroft, J., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. SIAM J. Comput. **2**, 225–231 (1973)

[HopTa73] Hopcroft, J., Tarjan, R.E.: Dividing a graph into triconnected components. SIAM J. Comput. **2**, 135–158 (1973)

[HopUl79] Hopcroft, J., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison Wesley, Reading (1979)

[Hor87] Horton, J.D.: A polynomial time algorithm to find the shortest cycle basis of a graph. SIAM J. Comput. **16**, 358–366 (1987)

[Hu61] Hu, T.C.: The maximum capacity route problem. Oper. Res. **9**, 898–900 (1961)

[Hu74] Hu, T.C.: Optimum communication spanning trees. SIAM J. Comput. **3**, 188–195 (1974)

[HuMR82] Huang, C., Mendelsohn, E., Rosa, A.: On partially resolvable t-partitions. Ann. Discrete Math. **12**, 160–183 (1982)

[HuDi88] Hung, M.S., Divoky, J.J.: A computational study of efficient shortest path algorithms. Comput. Oper. Res. **15**, 567–576 (1988)

[Hup67] Huppert, B.: Endliche Gruppen I. Springer, Berlin (1967)

[HwaRW92] Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. North Holland, Amsterdam (1992)

[Ima83] Imai, H.: On the practical efficiency of various maximum flow algorithms. J. Oper. Res. Soc. Jpn. **26**, 61–82 (1983)

[ImrSZ84] Imrich, W., Simões-Pereira, J.M.S., Zamfirescu, C.M.: On optimal embeddings of metrics in graphs. J. Comb. Theory, Ser. B **36**, 1–15 (1984)

[Irn08] Irnich, S.: Solution of real-world postman problems. Eur. J. Oper. Res. **190**, 52–67 (2008)

[Irv85] Irving, R.W.: An efficient algorithm for the 'stable roommates' problem. J. Algorithms **6**, 577–595 (1985)

[IrvLe86] Irving, R.W., Leather, P.: The complexity of counting stable marriages. SIAM J. Comput. **15**, 655–667 (1986)

[IrvLG87] Irving, R.W., Leather, P., Gusfield, D.: An efficient algorithm for the 'optimal' stable marriage. J. Assoc. Comput. Mach. **34**, 532–543 (1987)

[ItaRo78] Itai, A., Rodeh, M.: Finding a minimum circuit in a graph. SIAM J. Comput. **7**, 413–423 (1978)

[ItaSh79] Itai, A., Shiloach, Y.: Maximum flow in planar networks. SIAM J. Comput. **8**, 135–150 (1979)

[ItaRT78] Itai, A., Rodeh, M., Tanimota, S.L.: Some matching problems in bipartite graphs. J. Assoc. Comput. Mach. **25**, 517–525 (1978)

[ItaPS82] Itai, A., Perl, Y., Shiloach, Y.: The complexity of finding maximum disjoint paths with length constraints. Networks **12**, 277–286 (1982)

[Jar30] Jarník, V.: O jistém problému minimálním. Acta Soc. Sci. Nat. Morav. **6**, 57–63 (1930)

[JarKR93] Jarrah, A.I.Z., Yu, G., Krishnamurthy, N., Rakshit, A.: A decision support framework for airline flight cancellations and delays. Transp. Sci. **27**, 266–280 (1993)

[Jel03] Jelliss, G.: Knight's Tour Notes (2003). http://www.ktn.freeuk.com/index.htm

[Jen76] Jenkyns, T.A.: The efficacy of the 'greedy' algorithm. In: Proc. 7th Southeastern Conf. Combinatorics, Graph Theory and Computing, pp. 341–350 (1976)

[JenTo95] Jensen, T.R., Toft, B.: Graph Coloring Problems. Wiley, New York (1995)

[JenWi85] Jensen, K., Wirth, N.: PASCAL User Manual and Report, 3rd edn. Springer, New York (1985)

[Joh75] Johnson, D.B.: Priority queues with update and minimum spanning trees. Inf. Process. Lett. **4**, 53–57 (1975)

[JohMcG93] Johnson, D.S., McGeoch, C.C. (eds.): Network Flows and Matching. American Mathematical Society, Providence (1993)

[JohVe82] Johnson, D.S., Venkatesan, S.M.: Using divide and conquer to find flows in directed planar networks in $O(n^{3/2} \log n)$ time. In: Proc. 20th Allerton Conf. on Communication, Control and Computing, pp. 898–905. Univ. of Illinois, Urbana (1982)

[JohLR78] Johnson, D.S., Lenstra, J.K., Rinnooy Kan, A.H.G.: The complexity of the network design problem. Networks **8**, 279–285 (1978)

[Jun79a] Jungnickel, D.: A construction of group divisible designs. J. Stat. Plan. Inference **3**, 273–278 (1979)

[Jun79b] Jungnickel, D.: Die Methode der Hilfsmatrizen. In: Tölke, J., Wills, J.M. (eds.) Contributions to Geometry, pp. 388–394. Birkhäuser, Basel (1979)

[Jun86] Jungnickel, D.: Transversaltheorie: Ein Überblick. Bayreuth. Math. Schr. **21**, 122–155 (1986)

[Jun93] Jungnickel, D.: Finite Fields. B.I. Wissenschaftsverlag, Mannheim (1993)

[JunLe88] Jungnickel, D., Leclerc, M.: A class of lattices. Ars Comb. **26**, 243–248 (1988)

[JunLe89] Jungnickel, D., Leclerc, M.: The 2–matching lattice of a graph. J. Comb. Theory, Ser. B **46**, 246–248 (1989)

[JunLe87] Jungnickel, D., Lenz, H.: Minimal linear spaces. J. Comb. Theory, Ser. A **44**, 229–240 (1987)

[JunVa95] Jungnickel, D., Vanstone, S.A.: An application of coding theory to a problem in graphical enumeration. Arch. Math. **65**, 461–464 (1995)

[JunVa96] Jungnickel, D., Vanstone, S.A.: Graphical codes—a tutorial. Bull. Inst. Comb. Appl. **18**, 45–64 (1996)

[JunVa97] Jungnickel, D., Vanstone, S.A.: Graphical codes revisited. IEEE Trans. Inf. Theory **43**, 136–146 (1997)

[JunVa99a] Jungnickel, D., Vanstone, S.A.: Ternary graphical codes. J. Comb. Math. Comb. Comput. **29**, 17–31 (1999)

[JunVa99b] Jungnickel, D., Vanstone, S.A.: q-ary graphical codes. Discrete Math. **208/209**, 375–386 (1999)

[KabYDN08] Kabadi, S.N., Yan, J., Du, D., Nair, K.P.K.: Integer exact network synthesis problem. SIAM J. Discrete Math. **23**, 136–154 (2008)

[Kah62] Kahn, A.B.: Topological sorting of large networks. Commun. ACM **5**, 558–562 (1962)

[Kal60] Kalaba, R.: On some communication network problems. In: Bellman, R.E., Hall, M. (eds.) Combinatorial Analysis, pp. 261–280. Am. Math. Soc., Providence (1960)

[Kar99] Karger, D.R.: Using random sampling to find maximum flows in uncapacitated undirected graphs. In: Proc. 29th. ACM Symp. on Theory of Computing, pp. 240–249. Assoc. Comput. Mach., New York (1999)

[Kar84] Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica **4**, 373–396 (1984)

[Kar72] Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)

[Kar75] Karp, R.M.: On the computational complexity of combinatorial problems. Networks **5**, 45–68 (1975)

[Kar78] Karp, R.M.: A characterization of the minimum cycle mean in a digraph. Discrete Math. **23**, 309–311 (1978)

[Kar74] Karzanov, A.V.: Determining the maximal flow in a network with the method of preflows. Sov. Math. Dokl. **15**, 434–437 (1974)

[Kas67] Kasteleyn, P.W.: Graph theory and crystal physics. In: Harary, F. (ed.) Graph Theory and Theoretical Physics, pp. 43–110. Academic Press, New York (1967)

[KawKR12] Kawarabayashi, K., Kobayashi, Y., Reed, B.: The disjoint paths problem in quadratic time. J. Comb. Theory, Ser. B **102**, 424–435 (2012)

[KayCh65] Kay, D.C., Chartrand, G.: A characterization of certain Ptolemaic graphs. Can. J. Math. **17**, 342–346 (1965)

[Kem79] Kempe, A.B.: On the geographical problem of the four colours. Am. J. Math. **2**, 193–200 (1879)

[Kha79] Khachiyan, L.G.: A polynomial algorithm in linear programming. Sov. Math. Dokl. **20**, 191–194 (1979)

[Khu12] Khuller, S.: Design and Analysis of Algorithms: Course Notes. Dept. of Computer Science, University of Maryland

[KinRT94] King, V., Rao, S., Tarjan, R.: A faster deterministic maximum flow algorithm. J. Algorithms **17**, 447–474 (1994)

[Kirh47] Kirchhoff, G.: Über die Auflösungen der Gleichungen, auf die man bei der Untersuchung der Verteilung galvanischer Ströme geführt wird. Ann. Phys. Chem. **72**, 497–508 (1847)

[Kir47] Kirkman, T.P.: On a problem in combinatorics. Camb. Dublin Math. J. **2**, 191–204 (1847)

[Kir50] Kirkman, T.P.: Query VI. Lady's Gentlem. Diary **147**, 48 (1850)

[Kle67] Klein, M.: A primal method for minimal cost flows, with applications to the assignment and transportation problems. Manag. Sci. **14**, 205–220 (1967)

[KleWe91] Kleitman, D.J., West, D.B.: Spanning trees with many leaves. SIAM J. Discrete Math. **4**, 99–106 (1991)

[Knu67] Knuth, D.E.: Oriented subtrees of an arc digraph. J. Comb. Theory **3**, 309–314 (1967)

[Knu81] Knuth, D.E.: A permanent inequality. Am. Math. Mon. **88**, 731–740 (1981)

[KocSt93] Kocay, W., Stone, D.: Balanced network flows. Bull. Inst. Comb. Appl. **7**, 17–32 (1993)

[KocSt95] Kocay, W., Stone, D.: An algorithm for balanced flows. J. Comb. Math. Comb. Comput. **19**, 3–31 (1995)

[Koe16] König, D.: Über Graphen und ihre Anwendungen auf Determinantentheorie und Mengenlehre. Math. Ann. **77**, 453–465 (1916)

[Koe31] König, D.: Graphen und Matrizen. Mat. Fiz. Lapok **38**, 116–119 (1931) (Hungarian with a summary in German)

[KorHa78] Korte, B., Hausmann, D.: An analysis of the greedy heuristic for independence systems. Ann. Discrete Math. **2**, 65–74 (1978)

[KorLo81] Korte, B., Lovász, L.: Mathematical structures underlying greedy algorithms. In: Gécseg, F. (ed.) Fundamentals of Computation Theory, pp. 205–209. Springer, Berlin (1981)

[KorMo89] Korte, N., Möhring, R.H.: An incremental linear-time algorithm for recognizing interval graphs. SIAM J. Comput. **18**, 68–81 (1989)

[KorVy12] Korte, B., Vygen, J.: Combinatorial Optimization. Theory and Algorithms, 5th edn. Springer, Berlin (2012)

[KorLP90] Korte, B., Lovász, L., Prömel, H.J., Schrijver, A. (eds.): Paths, Flows and VLSI-Layout. Springer, Berlin (1990)

[KorPS90] Korte, B., Prömel, H.J., Steger, A.: Steiner trees in VLSI-Layout. In: Korte, B., Lovász, L., Prömel, H.J., Schrijver, A. (eds.) Paths, Flows and VLSI-layout, pp. 185–214. Springer, Berlin (1990)

[KorLS91] Korte, B., Lovász, L., Schrader, R.: Greedoids. Springer, Berlin (1991)

[Kot00] Kotlov, A.: Short proof of the Gallai-Edmonds structure theorem (2000). arXiv:math/0011204v1

[Kra07] Kravitz, D.: Two comments on minimum spanning trees. Bull. Inst. Comb. Appl. **49**, 7–10 (2007)

[Kri75] Krishnamoorthy, M.S.: An NP-hard problem in bipartite graphs. SIGACT News **7**(1), 26 (1975)

[Kru56] Kruskal, J.B.: On the shortest spanning subtree of a graph and the travelling salesman problem. Proc. Am. Math. Soc. **7**, 48–50 (1956)

[Kuh55] Kuhn, H.W.: The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**, 83–97 (1955)

[Kuh56] Kuhn, H.W.: Variants of the Hungarian method for the assignment problem. Nav. Res. Logist. Q. **3**, 253–258 (1956)

[KuiSa86] Kuich, W., Salomaa, A.: Semirings, Automata, Languages. Springer, Berlin (1986)

[Kur30] Kuratowski, K.: Sur le problème des courbes gauches en topologie. Fundam. Math. **15**, 271–283 (1930)

[Kwa62] Kwan, M.-K.: Graphic programming using odd and even points. Chin. Math. **1**, 273–277 (1962)

[Lam87] Lamken, E.: A note on partitioned balanced tournament designs. Ars Comb. **24**, 5–16 (1987)

[LamVa87] Lamken, E., Vanstone, S.A.: The existence of partitioned balanced tournament designs. Ann. Discrete Math. **34**, 339–352 (1987)

[LamVa89] Lamken, E., Vanstone, S.A.: Balanced tournament designs and related topics. Discrete Math. **77**, 159–176 (1989)

[Las72] Las Vergnas, M.: Problèmes de couplages et problèmes hamiltoniens en théorie des graphes. Dissertation, Universitè de Paris VI (1972)

[Law75] Lawler, E.L.: Matroid intersection algorithms. Math. Program. **9**, 31–56 (1975)

[Law76] Lawler, E.L.: Combinatorial Optimization: Networks and Matriods. Holt, Rinehart and Winston, New York (1976)

[LawLRS85] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.): The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York (1985)

[LawLRS93] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: algorithms and complexity. In: Graves, S.C., Rinnooy Kan, A.H.G., Zipkin, P.H. (eds.) Logistics of Production and Inventory, pp. 445–522. Elsevier, Amsterdam (1993)

[Lec86] Leclerc, M.: Polynomial time algorithms for exact matching problems. M. Math. thesis, University of Waterloo, Dept. of Combinatorics and Optimization (1986)

[Lec87] Leclerc, M.: Algorithmen für kombinatorische Optimierungsprobleme mit Partitionsbeschränkungen. Dissertation, Universität Köln (1987)

[LecRe89] Leclerc, M., Rendl, F.: Constrained spanning trees and the travelling salesman problem. Eur. J. Oper. Res. **39**, 96–102 (1989)

[Leh64] Lehman, A.: A solution of the Shannon switching game. SIAM J. Appl. Math. **12**, 687–725 (1964)

[Lem88] Lemke, P.: The Maximum-Leaf Spanning Tree Problem in Cubic Graphs is NP-Complete. IMA Publication, vol. 428. University of Minnesota, Minneapolis (1988)

[Len90] Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. Wiley, New York (1990)

[LenRi75] Lenstra, J.K., Rinnooy Kan, A.H.G.: Some simple applications of the travelling salesman problem. Oper. Res. Q. **26**, 717–733 (1975)

[LesPP84] Lesk, M., Plummer, M.D., Pulleyblank, W.R.: Equi-matchable graphs. In: Bollobás, B. (ed.) Graph Theory and Combinatorics, pp. 239–254. Academic Press, New York (1984)

[LesOe86] Lesniak, L., Oellermann, O.R.: An Eulerian exposition. J. Graph Theory **10**, 277–297 (1986)

[LewLL86] Lewandowski, J.L., Liu, C.L., Liu, J.W.S.: An algorithmic proof of a generalization of the Birkhoff-Von Neumann theorem. J. Algorithms **7**, 323–330 (1986)

[LewPa81] Lewis, H.R., Papadimitriou, C.H.: Elements of the Theory of Computation. Prentice Hall, Englewood Cliffs (1981)

[LidNi94] Lidl, R., Niederreiter, H.: Introduction to Finite Fields and Their Applications, revised edn. Cambridge University Press, Cambridge (1994)

[Lin65] Lin, S.: Computer solutions of the travelling salesman problem. Bell Syst. Tech. J. **44**, 2245–2269 (1965)

[LinKe73] Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the travelling salesman problem. Oper. Res. **31**, 498–516 (1973)

[LinLW88] Linial, N., Lovász, L., Widgerson, A.: Rubber bands, convex embeddings and graph connectivity. Combinatorica **8**, 91–102 (1988)

[LitMSK63] Little, J.D.C., Murty, K.G., Sweeney, D.W., Karel, C.: An algorithm for the travelling salesman problem. Oper. Res. **11**, 972–989 (1963)

[Lom85] Lomonosov, M.V.: Combinatorial approaches to multiflow problems. Discrete Appl. Math. **11**, 1–93 (1985)

[Lov70a] Lovász, L.: Problem 11. In: Guy, R., Hanani, H., Sauer, N., Schönheim, J. (eds.) Combinatorial Structures and Their Applications, p. 497. Gordon and Breach, New York (1970)

[Lov70b] Lovász, L.: Subgraphs with prescribed valencies. J. Comb. Theory **8**, 391–416 (1970)

[Lov72] Lovász, L.: Normal hypergraphs and the perfect graph conjecture. Discrete Math. **2**, 253–267 (1972)

[Lov79] Lovász, L.: Graph theory and integer programming. Ann. Discrete Math. **4**, 141–158 (1979)

[Lov85] Lovász, L.: Some algorithmic problems on lattices. In: Lovász, L., Smerédi, E. (eds.) Theory of Algorithms, pp. 323–337. North Holland, Amsterdam (1985)

[Lov87] Lovász, L.: The matching structure and the matching lattice. J. Comb. Theory, Ser. B **43**, 187–222 (1987)

[LovPl86] Lovász, L., Plummer, M.D.: Matching Theory. North Holland, Amsterdam (1986)

[Luc82] Lucas, E.: Récréations Mathématiques. Paris (1882)

[Lue82] Lüneburg, H.: Programmbeispiele aus Algebra, Zahlentheorie und Kombinatorik. Report, Universität Kaiserslautern (1982)

[Lue89] Lüneburg, H.: Tools and Fundamental Constructions of Combinatorial Mathematics. Bibliographisches Institut, Mannheim (1989)

[Lyn75] Lynch, J.F.: The equivalence of theorem proving and the interconnection problem. SIGDA Newsl. **5**, 31–65 (1975)

[Ma94] Ma, S.L.: A survey of partial difference sets. Des. Codes Cryptogr. **4**, 221–261 (1994)

[Mac87] Maculan, N.: The Steiner problem in graphs. Ann. Discrete Math. **31**, 185–212 (1987)

[MacSl77] MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North Holland, Amsterdam (1977)

[MagWo94] Magnanti, T.L., Wong, R.T.: Network design and transportation planning: models and algorithms. Transp. Sci. **18**, 1–55 (1984)

[MalKM78] Malhotra, V.M., Kumar, M.P., Mahaswari, S.N.: An $O(|V|^3)$ algorithm for finding maximum flows in networks. Inf. Process. Lett. **7**, 277–278 (1978)

[ManSc89] Mansour, Y., Schieber, B.: Finding the edge connectivity of directed graphs. J. Algorithms **10**, 76–85 (1989)

[MarMi65] Marcus, M., Minc, H.: Diagonal products in doubly stochastic matrices. Q. J. Math. **16**, 32–34 (1965)

[MaRe59] Marcus, M., Ree, R.: Diagonals of doubly stochastic matrices. Q. J. Math. **10**, 296–302 (1959)

[Mar92] Martin, A.: Packen Von Steinerbäumen: Polyedrische Studien und Anwendung. Dissertation, Technische Universität Berlin (1992)

[MarOF91] Martin, O., Otto, S.W., Felten, E.W.: Large-step Markov chains for the traveling salesman problem. Complex Syst. **5**, 299–326 (1991)

[Mat95] Matsui, T.: The minimum spanning tree problem on a planar graph. Discrete Appl. Math. **58**, 91–94 (1995)

[Mat87] Matula, D.W.: Determining edge connectivity in $O(mn)$. In: Proc. 28th Symp. on Foundations of Computer Science, pp. 249–251 (1987)

[McEl87] McEliece, R.J.: Finite Fields for Computer Scientists and Engineers. Kluwer, Boston (1987)

[Meh84] Mehlhorn, K.: Data Structures and Algorithms. Springer, Berlin (1984)

[MehSa08] Mehlhorn, K., Sanders, P.: Algorithms and Data Structures. The Basic Toolbox. Springer, Berlin (2008)

[MehSc86] Mehlhorn, K., Schmidt, B.H.: On BF-orderable graphs. Discrete Appl. Math. **15**, 315–327 (1986)

[MenDu58] Mendelsohn, N.S., Dulmage, A.L.: Some generalizations of the problem of distinct representatives. Can. J. Math. **10**, 230–241 (1958)

[MenRo85] Mendelsohn, E., Rosa, A.: One-factorizations of the complete graph—a survey. J. Graph Theory **9**, 43–65 (1985)

[Men74] Meng, D.H.C.: Matchings and coverings for graphs. Ph.D. thesis, Michigan State University, East Lansing, Mich. (1974)

[Men27] Menger, K.: Zur allgemeinen Kurventheorie. Fundam. Math. **10**, 96–115 (1927)

[MicVa80] Micali, S., Vazirani, V.V.: An $O(\sqrt{|V||E|})$ algorithm for finding maximum matchings in general graphs. In: Proc. 21st IEEE Symp. on Foundations of Computer Science, pp. 17–27 (1980)

[Mic92] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin (1992)

[MicAK07] Michiels, W., Aarts, E., Korst, J.: Theoretical Aspects of Local Search. Springer, Berlin (2007)

[Mil10] Miller, G.A.: On a method due to Galois. Q. J. Pure Appl. Math. **41**, 382–384 (1910)

[Min78] Minc, H.: Permanents. Addison-Wesley, Reading (1978)

[Min88] Minc, H.: Nonnegative Matrices. Wiley, New York (1988)

[Min60] Minty, G.J.: Monotone networks. Proc. R. Soc. Lond. A **257**, 194–212 (1960)

[Min66] Minty, G.J.: On the axiomatic foundations of the theories of directed linear graphs, electrical networks and network programming. J. Math. Mech. **15**, 485–520 (1966)

[MirRo84] Mirkin, B.G., Rodin, N.S.: Genes and Graphs. Springer, New York (1984)

[Mir69a] Mirsky, L.: Hall's criterion as a 'self-refining' result. Monatshefte Math. **73**, 139–146 (1969)

[Mir69b] Mirsky, L.: Transversal theory and the study of abstract independence. J. Math. Anal. Appl. **25**, 209–217 (1969)

[Mir71a] Mirsky, L.: A dual of Dilworth's decomposition theorem. Am. Math. Mon. **78**, 876–877 (1971)

[Mir71b] Mirsky, L.: Transversal Theory. Academic Press, New York (1971)

[MirPe67] Mirsky, L., Perfect, H.: Applications of the notion of independence to problems of combinatorial analysis. J. Comb. Theory **2**, 327–357 (1967)

[Mit99] Mitchell, J.S.B.: Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for Euclidean TSP, k-MST, and related problems. SIAM J. Comput. **28**, 1298–1309 (1999)

[MohPo93] Mohar, B., Poljak, S.: Eigenvalues in combinatorial optimization. In: Brualdi, R., Friedland, S., Klee, V. (eds.) Combinatorial and Graph-Theoretic Problems in Linear Algebra, pp. 107–151. Springer, New York (1993)

[MohTh01] Mohar, B., Thomassen, C.: Graphs on Surfaces. John Hopkins University Press, Baltimore (2001)

[Mon83] Monien, B.: The complexity of determining a shortest cycle of even length. Computing **31**, 355–369 (1983)

[Moo59] Moore, E.F.: The shortest path through a maze. In: Proc. Int. Symp. on Theory of Switching Part II, pp. 285–292. Harvard University Press, Cambridge (1959)

[MuhGK81] Mühlenbein, H., Gorges-Schleuter, M., Krämer, O.: Evolution algorithms in combinatorial optimization. Parallel Comput. **7**, 65–85 (1988)

[Mui03] Muirhead, A.F.: Some methods applicable to identities and inequalities of symmetric algebraic functions of n letters. Proc. Edinb. Math. Soc. **21**, 144–157 (1903)

[Mul66] Müller-Merbach, H.: Die Anwendung des Gozinto-Graphs zur Berechnung des Roh- und Zwischenproduktbedarfs in chemischen Betrieben. Ablauf-Plan.forsch. **7**, 189–198 (1966)

[Mul69] Müller-Merbach, H.: Die Inversion von Gozinto-Matrizen mit einem graphen-orientierten Verfahren. Elektron. Datenverarb. **11**, 310–314 (1969)

[Mul73] Müller-Merbach, H.: Operations Research, 3rd edn. Franz Vahlen, München (1973)

[Nad90] Naddef, D.: Handles and teeth in the symmetric travelling salesman polytope. In: Cook, W., Seymour, P.D. (eds.) Polyhedral Combinatorics, pp. 61–74. Am. Math. Soc., Providence (1990)

[NemWo88] Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)

[NijWi78] Nijenhuis, A., Wilf, H.S.: Combinatorial Algorithms, 2nd edn. Academic Press, New York (1978)

[NisCh88] Nishizeki, T., Chiba, N.: Planar Graphs: Theory and Algorithms. North Holland, Amsterdam (1988)

[NobPi96] Nobert, Y., Picard, J.-C.: An optimal algorithm for the mixed Chinese Postman Problem. Networks **27**, 95–108 (1996)

[NtaHa81] Ntafos, S.C., Hakimi, S.L.: On the complexity of some coding problems. IEEE Trans. Inf. Theory **27**, 794–796 (1981)

[Or76] Or, I.: Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Ph.D. thesis, Northwestern University, Evanston, IL (1976)

[Ore51] Ore, O.: A problem regarding the tracing of graphs. Elem. Math. **6**, 49–53 (1951)

[Ore55] Ore, O.: Graphs and matching theorems. Duke Math. J. **22**, 625–639 (1955)

[Ore60] Ore, O.: Note on Hamiltonian circuits. Am. Math. Mon. **67**, 55 (1960)

[Orl93] Orlin, J.B.: A faster strongly polynomial minimum cost flow algorithm. Oper. Res. **41**, 338–350 (1993)

[Orl97] Orlin, J.B.: A polynomial time primal network simplex algorithm for minimum cost flows. Math. Program. **78**, 109–129 (1997)

[OrlAh92] Orlin, J.B., Ahuja, R.K.: New scaling algorithms for the assignment and minimum cycle mean problems. Math. Program. **54**, 41–56 (1992)

[OrlSh04] Orlin, J.B., Sharma, D.: Extended neighborhood: definition and characterization. Math. Program. **101**, 537–559 (2004)

[OrlPT93] Orlin, J.B., Plotkin, S.A., Tardos, E.: Polynomial dual network simplex algorithms. Math. Program. **60**, 255–276 (1993)

[Ott48] Otter, R.: The number of trees. Ann. Math. **49**, 583–599 (1948)

[Oxl92] Oxley, J.G.: Matroid Theory. Oxford University Press, Oxford (1992)

[Oza25] Ozanam, J.: Récréations Mathématiques et Physiques, vol. 1. Claude Jombert, Paris (1725)

[PadHo80] Padberg, M.W., Hong, S.: On the symmetric travelling salesman problem: A computational study. Math. Program. Stud. **12**, 78–107 (1980)

[PadRa74] Padberg, M.W., Rao, M.R.: The travelling salesman problem and a class of polyhedra of diameter two. Math. Program. **7**, 32–45 (1974)

[PadRi87] Padberg, M.W., Rinaldi, G.: Optimization of a 532-city symmetric travelling salesman problem. Oper. Res. Lett. **6**, 1–7 (1987)

[PadRi91] Padberg, M.W., Rinaldi, G.: A branch-and-cut algorithm for the resolution of large-scale travelling salesman problems. SIAM Rev. **33**, 60–100 (1991)

[PadSu91] Padberg, M.W., Sung, T.-Y.: An analytical comparison of different formulations of the travelling salesman problem. Math. Program. **52**, 315–357 (1991)

[Pap76] Papadimitriou, C.H.: On the complexity of edge traversing. J. Assoc. Comput. Mach. **23**, 544–554 (1976)

[Pap78] Papadimitriou, C.H.: The adjacency relation on the traveling salesman polytope is NP-complete. Math. Program. **14**, 312–324 (1978)

[Pap92] Papadimitriou, C.H.: The complexity of the Lin-Kernighan heuristic for the traveling salesman problem. SIAM J. Comp. **21**, 450–465 (1992)

[Pap94] Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading (1994)

[PapSt77] Papadimitriou, C.H., Steiglitz, K.: On the complexity of local search for the travelling salesman problem. SIAM J. Comput. **6**, 76–83 (1977)

[PapSt78] Papadimitriou, C.H., Steiglitz, K.: Some examples of difficult travelling salesman problems. Oper. Res. **26**, 434–443 (1978)

[PapSt82] Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice Hall, Englewood Cliffs (1982)

[PapVe06] Papadimitriou, C.H., Vempala, S.: On the approximability of the traveling salesman problem. Combinatorica **26**, 101–120 (2006)

[PapYa82] Papadimitriou, C.H., Yannakakis, M.: The complexity of restricted spanning tree problems. J. Assoc. Comput. Mach. **29**, 285–309 (1982)

[PapYa93] Papadimitriou, C.H., Yannakakis, M.: The traveling salesman problem with distances 1 and 2. Math. Oper. Res. **18**, 1–11 (1993)

[PapCo80] Pape, U., Conradt, D.: Maximales Matching in Graphen. In: Späth, H. (ed.)

Ausgewählte Operations Research Software in FORTRAN, pp. 103–114. Oldenbourg, München (1980)

[Pel36] Peltesohn, R.: Das Turnierproblem für Spiele zu je dreien. Dissertation, Universität Berlin (1936)

[Pet91] Petersen, J.: Die Theorie der regulären Graphen. Acta Math. **15**, 193–220 (1891)

[Pet98] Petersen, J.: Sur le théorème de Tait. L'Intermédiaire Math. **5**, 225–227 (1898)

[PetLo88] Peterson, P.A., Loui, M.C.: The general maximum matching algorithm of Micali and Vazirani. Algorithmica **3**, 511–533 (1988)

[Pos69] Posner, E.C.: Combinatorial structures in planetary reconnaissance. In: Mann, H.B. (ed.) Error Correcting Codes, pp. 15–46. Wiley, New York (1969)

[Pri57] Prim, R.C.: Shortest connection networks and some generalizations. Bell Syst. Tech. J. **36**, 1389–1401 (1957)

[Pri96] Prisner, E.: Line graphs and generalizations—a survey. Congr. Numer. **116**, 193–229 (1996)

[ProSt02] Prömel, H.-J., Steger, A.: The Steiner Tree Problem. Vieweg, Braunschweig (2002)

[Pro86] Provan, J.S.: The complexity of reliability computations in planar and acyclic graphs. SIAM J. Comput. **15**, 694–702 (1986)

[Pru18] Prüfer, H.: Neuer Beweis eines Satzes über Permutationen. Arch. Math. Phys. (3) **27**, 142–144 (1918)

[Pul83] Pulleyblank, W.R.: Polyhedral combinatorics. In: Bachem, A., Grötschel, M., Korte, B. (eds.) Mathematical Programming: The State of the Art, pp. 312–345. Springer, Berlin (1983)

[PymPe70] Pym, J.S., Perfect, H.: Submodular functions and independence structures. J. Math. Anal. Appl. **30**, 1–31 (1970)

[Qi88] Qi, L.: Directed submodularity, ditroids and directed submodular flows. Math. Program. **42**, 579–599 (1988)

[Rad42] Rado, R.: A theorem on independence relations. Q. J. Math. **13**, 83–89 (1942)

[Rad57] Rado, R.: Note on independence functions. Proc. Lond. Math. Soc. **7**, 300–320 (1957)

[RadGo91] Radzik, T., Goldberg, A.V.: Tight bounds on the number of minimum mean cycle cancellations and related results. In: Proc. 2nd ACM—SIAM Symp. on Discrete Algorithms, pp. 110–119 (1991)

[Ral81] Ralston, A.: A new memoryless algorithm for de Bruijn sequences. J. Algorithms **2**, 50–62 (1981)

[Ram68] Ramachandra Rao, A.: An extremal problem in graph theory. Isr. J. Math. **6**, 261–266 (1968)

[RayWi71] Ray-Chaudhuri, D.K., Wilson, R.M.: Solution of Kirkman's school girl problem. In: Proc. Symp. Pure Appl. Math., vol. 19, pp. 187–203. Am. Math. Soc., Providence (1971)

[Rea62] Read, R.C.: Euler graphs on labeled nodes. Can. J. Math. **14**, 482–486 (1962)

[Rec89] Recski, A.: Matroid Theory and Its Applications. Springer, Berlin (1989)

[Red34] Redéi, L.: Ein kombinatorischer Satz. Acta Univ. Szeged., Sect. Litt. **7**, 39–43 (1934)

[Ree87] Rees, R.: Uniformly resolvable pairwise balanced designs with block sizes two and three. J. Comb. Theory, Ser. A **45**, 207–225 (1987)

[Rei94] Reinelt, G.: The Traveling Salesman: Computational Solutions for TSP Applications. Springer, Berlin (1994)

[ReiTa81] Reingold, E.M., Tarjan, R.E.: On a greedy heuristic for complete matching. SIAM J. Comput. **10**, 676–681 (1981)

[Ren59] Rényi, A.: Some remarks on the theory of trees. Magy. Tud. Akad. Mat. Kut. Intéz. Közl. **4**, 73–85 (1959)

[ResWi07] Restrepo, M., Williamson, D.P.: A simple GAP-canceling algorithm for the generalized maximum flow problem. Math. Program. **118**, 47–74 (2009)

[Rie91] Rieder, J.: The lattices of matroid bases and exact matroid bases. Arch. Math. **56**, 616–623 (1991)

[Riz00] Rizzi, R.: A short proof of König's matching theorem. J. Graph Theory **33**, 138–139 (2000)

[Rob39] Robbins, H.: A theorem on graphs with an application to a problem of traffic control. Am. Math. Mon. **46**, 281–283 (1939)

[RobXu88] Roberts, F.S., Xu, Y.: On the optimal strongly connected orientations of city street graphs I: Large grids. SIAM J. Discrete Math. **1**, 199–222 (1988)

[RobSe95] Robertson, N., Seymour, P.: Graph minors. XIII. The disjoint paths problem. J. Comb. Theory, Ser. B **63**, 65–110 (1995)

[RobST93] Robertson, N., Seymour, P., Thomas, R.: Hadwiger's conjecture for K_6-free graphs. Combinatorica **13**, 279–361 (1993)

[RobSST97] Robertson, N., Sanders, D.P., Seymour, P., Thomas, R.: The four-colour theorem. J. Comb. Theory, Ser. B **70**, 2–44 (1997)

[Rob03a] Robinson, S.: Are medical students meeting their (best possible) match? SIAM News **36**(3), 8–9 (2003)

[Rob03b] Robinson, S.: How much can matching theory improve the lot of medical residents? SIAM News **36**(6), 4–5 (2003)

[RosSL77] Rosenkrantz, D.J., Stearns, E.A., Lewis, P.M.: An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput. **6**, 563–581 (1977)

[Ros77] Rosenthal, A.: Computing the reliability of complex networks. SIAM J. Appl. Math. **32**, 384–393 (1977)

[Rue86] Rueppel, R.: Analysis and Design of Stream Ciphers. Springer, New York (1986)

[Rys57] Ryser, H.J.: Combinatorial properties of matrices of zeros and ones. Can. J. Math. **9**, 371–377 (1957)

[SahGo76] Sahni, S., Gonzales, T.: P-complete approximation problems. J. Assoc. Comput. Mach. **23**, 555–565 (1976)

[Schn78] Schnorr, C.P.: An algorithm for transitive closure with linear expected time. SIAM J. Comput. **7**, 127–133 (1978)

[Schn79] Schnorr, C.P.: Bottlenecks and edge connectivity in unsymmetrical networks. SIAM J. Comput. **8**, 265–274 (1979)

[Schr80] Schreuder, J.A.M.: Constructing timetables for sport competitions. Math. Program. Stud. **13**, 58–67 (1980)

[Schr92] Schreuder, J.A.M.: Combinatorial aspects of construction of competition Dutch professional football leagues. Discrete Appl. Math. **35**, 301–312 (1992)

[Schr83a] Schrijver, A.: Min-max results in combinatorial optimization. In: Bachem, A., Grötschel, M., Korte, B. (eds.) Mathematical Programming: The State of the Art, pp. 439–500. Springer, Berlin (1983)

[Schr83b] Schrijver, A.: Short proofs on the matching polyhedron. J. Comb. Theory, Ser. B **34**, 104–108 (1983)

[Schr84] Schrijver, A.: Total dual integrality from directed graphs, crossing families, and sub- and supermodular functions. In: Pulleyblank, W.R. (ed.) Progress in Combinatorial Optimization, pp. 315–361. Academic Press, San Diego (1984)

[Schr86] Schrijver, A.: Theory of Integer and Linear Programming. Wiley, New York (1986)

[Schr03] Schrijver, A.: Combinatorial Optimization. Polyhedra and Efficiency. Springer, Berlin (2003) (in 3 volumes)

[Schw91] Schwenk, A.J.: Which rectangular chessboards have a knight's tour? Math. Mag. **64**, 325–332 (1991)

[SchwW78] Schwenk, A.J., Wilson, R.: On the eigenvalues of a graph. In: Beineke, L., Wilson, R. (eds.) Selected Topics in Graph Theory, pp. 307–336. Academic Press, London (1978)

[Sey79] Seymour, P.: Sums of circuits. In: Bondy, J.A., Murty, U.S.R. (eds.) Graph Theory and Related Topics, pp. 341–355. Academic Press, New York (1979)

[Sha48a] Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423 (1948) (Part I)

[Sha48b] Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 623–656 (1948) (Part II)

[Sha49a] Shannon, C.E.: A theorem on colouring lines of a network. J. Math. Phys. **28**, 148–151 (1949)

[Sha49b] Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**, 656–715 (1949)

[Sha79] Shapiro, J.F.: A survey of Langrangian techniques for discrete optimization. Ann. Discrete Math. **5**, 113–138 (1979)

[Shi75] Shimbel, A.: Structure in communication nets. In: Proc. Symp. Information Networks, pp. 199–203. Polytechnic Institute of Brooklyn, New York (1955)

[ShmWi90] Shmoys, D.B., Williamson, D.P.: Analyzing the Held-Karp-TSP bound: A monotonicity property with application. Inf. Process. Lett. **35**, 281–285 (1990)

[Sho85] Shor, N.Z.: Minimization Methods for Non-Differentiable Functions. Springer, Berlin (1985)

[SieHo91] Sierksma, G., Hoogeveen, H.: Seven criteria for integer sequences being graphic. J. Graph Theory **15**, 223–231 (1991)

[SikTu89] Siklóssy, L., Tulp, E.: Trains, an active time-table searcher. In: ECAI'89, pp. 170–175 (1989)

[Sim88] Simões-Pereira, J.M.S.: An optimality criterion for graph embeddings of metrics. SIAM J. Discrete Math. **1**, 223–229 (1988)

[Sip06] Sipser, K.: Introduction to the Theory of Computation, 2nd edn. Thompson, Boston (2006)

[Sle80] Sleator, D.D.: An $O(mn\log n)$ algorithm for maximum network flow. Ph.D. thesis, Stanford University (1980)

[SlaTa83] Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. J. Comput. Syst. Sci. **26**, 362–391 (1983)

[Spe28] Sperner, E.: Ein Satz über Untermengen einer endlichen Menge. Math. Z. **27**, 544–548 (1928)

[Spi85] Spinrad, J.: On comparability and permutation graphs. SIAM J. Comput. **14**, 658–670 (1985)

[SriCh92] Sridhar, S., Chandrasekaran, R.: Integer solution to synthesis of communication networks. Math. Oper. Res. **17**, 581–585 (1992)

[Sta86] Stanley, R.P.: Enumerative Combinatorics, vol. 1. Wadsworth & Brooks/Cole, Monterey (1986)

[Sta99] Stanley, R.P.: Enumerative Combinatorics, vol. 2. Cambridge University Press, Cambridge (1999)

[SteLe80] Stern, G., Lenz, H.: Steiner triple systems with given subspaces: another proof of the Doyen-Wilson theorem. Boll. UMI **17**, 109–114 (1980)

[Sto85] Stong, R.A.: On 1-factorizability of Cayley graphs. J. Comb. Theory, Ser. B **39**, 298–307 (1985)

[Str88] Strang, G.: Linear Algebra and Its Applications, 3rd edn. Harcourt Brace Jovanovich, San Diego (1993)

[Sum79] Sumner, D.P.: Randomly matchable graphs. J. Graph Theory **3**, 183–186 (1979)

[Suz82] Suzuki, M.: Group Theory I. Springer, Berlin (1982)

[SysDK83] Syslo, M.M., Deo, N., Kowalik, J.S.: Discrete Optimization Algorithms. Prentice Hall, Englewood Cliffs (1983)

[Sre98] Škrekovski, R.: Choosability of K_5-minor-free graphs. Discrete Math. **190**, 223–226 (1998)

[Tah92] Taha, H.A.: Operations Research, 5th edn. Macmillan, New York (1992)

[Tak90a] Takács, L.: On Cayley's formula for counting forests. J. Comb. Theory, Ser. A **53**, 321–323 (1990)

[Tak90b] Takács, L.: On the number of distinct forests. SIAM J. Discrete Math. **3**, 574–581 (1990)

[Tak92] Takaoka, T.: A new upper bound on the complexity of the all pairs shortest path problem. Inf. Process. Lett. **43**, 195–199 (1992)

[Tar85] Tardos, E.: A strongly polynomial minimum cost circulation algorithm. Combinatorica **5**, 247–255 (1985)

[Tar86] Tardos, E.: A strongly polynomial algorithm to solve combinatorial linear programs. Oper. Res. **34**, 250–256 (1986)

[Tar72] Tarjan, R.E.: Depth first search and linear graph algorithms. SIAM J. Comput. **1**, 146–160 (1972)

[Tar77] Tarjan, R.E.: Finding optimum branchings. Networks **7**, 25–35 (1977)

[Tar83] Tarjan, R.E.: In: Data Structures and Network Algorithms. Soc. Ind. Appl. Math., Philadelphia (1983)

[Tar84] Tarjan, R.E.: A simple version of Karzanov's blocking flow algorithm. Oper. Res. Lett. **2**, 265–268 (1984)

[Tar97] Tarjan, R.E.: Dynamic trees as search trees via Euler tours applied to the network simplex algorithm. Math. Program. **78**, 169–177 (1997)

[Tar95] Tarry, G.: Le problème des labyrinthes. Nouv. Ann. Math. **14**, 187 (1895)

[Ter96] Terlaky, T.: Interior Point Methods of Mathematical Programming. Kluwer, Dordrecht (1996)

[Tho98] Thomas, R.: An update on the four-color theorem. Not. Am. Math. Soc. **45**, 848–859 (1998)

[Tho81] Thomassen, C.: Kuratowski's theorem. J. Graph Theory **5**, 225–241 (1981)

[Tho94] Thomassen, C.: Every planar graph is 5-choosable. J. Comb. Theory, Ser. B **62**, 180–181 (1994)

[Tof96] Toft, B.: A survey of Hadwiger's conjecture. Congr. Numer. **115**, 249–283 (1996)

[TriHw90] Trietsch, D., Hwang, F.: An improved algorithm for Steiner trees. SIAM J. Appl. Math. **50**, 244–264 (1990)

[Tur36] Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proc. Lond. Math. Soc. **42**, 230–265 (1936)

[Tur88] Turner, J.S.: Almost all k-colorable graphs are easy to color. J. Algorithms **9**, 63–82 (1988)

[Tut47] Tutte, W.T.: The factorization of linear graphs. J. Lond. Math. Soc. **22**, 107–111 (1947)

[Tut48] Tutte, W.T.: The dissection of equilateral triangles into equilateral triangles. Proc. Camb. Philos. Soc. **44**, 203–217 (1948)

[Tut52] Tutte, W.T.: The factors of graphs. Can. J. Math. **4**, 314–328 (1952)

[Tut54] Tutte, W.T.: A short proof of the factor theorem for finite graphs. Can. J. Math. **6**, 347–352 (1952)

[Tut67] Tutte, W.T.: Antisymmetrical digraphs. Can. J. Math. **19**, 1101–1117 (1967)

[Tut71] Tutte, W.T.: Introduction to the Theory of Matroids. Elsevier, New York (1971)

[Tut84] Tutte, W.T.: Graph Theory. Cambridge University Press, Cambridge (1984)

[Vai89] Vaidya, P.M.: Geometry helps in matching. SIAM J. Comput. **19**, 1201–1225 (1989)

[Val79a] Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. **8**, 189–201 (1979)

[Val79b] Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. **8**, 410–421 (1979)

[vandW26] van der Waerden, B.L.: Aufgabe 45. Jahresber. DMV 117 (1926)

[vandW27] van der Waerden, B.L.: Ein Satz über Klasseneinteilungen von endlichen Mengen. Abh. Math. Semin. Univ. Hamb. **5**, 185–188 (1927)

[vandW37] van der Waerden, B.L.: Moderne Algebra, 2nd edn. Springer, Berlin (1937)

[vandW49] van der Waerden, B.L.: Modern Algebra, Vol. I. Frederick Ungar, New York (1949) (translated from the 2nd revised German edition by Fred Blum, with revisions and additions by the author)

[vanLi74] van Lint, J.H.: Combinatorial Theory Seminar Eindhoven University of Technology. Springer, Berlin (1974)

[vanLi99] van Lint, J.H.: Introduction to Coding Theory, 3rd edn. Springer, Berlin (1999)

[vanLiWi01] van Lint, J.H., Wilson, R.M.: A Course in Combinatorics, 2nd edn. Cambridge University Press, Cambridge (2001)

[VanOo89] Vanstone, S.A., van Oorschot, P.C.: An Introduction to Error Correcting Codes with Applications. Kluwer, Boston (1989)

[Vaz94] Vazirani, V.V.: A theory of alternating paths and blossoms for proving correctness of the $O(V^{1/2}E)$ general graph matching algorithm. Combinatorica **14**, 71–109 (1994)

[Veg11] Végh, L.: Augmenting undirected node-connectivity by one. SIAM J. Discrete Math. **25**, 695–718 (2011)

[Viz64] Vizing, V.G.: An estimate of the chromatic class of a p-graph. Diskretn. Anal. **3**, 25–30 (1964) (in Russian)

[Voi93] Voigt, M.: List colourings of planar graphs. Discrete Math. **120**, 215–219 (1993)

[VolJo82] Volgenant, T., Jonker, R.: A branch and bound algorithm for the symmetric travelling salesman problem based on the 1-tree relaxation. Eur. J. Oper. Res. **9**, 83–89 (1982)

[Vol04] Volkmann, L.: The Petersen graph is not 1-factorable: postscript to 'The Petersen graph is not 3-edge-colorable—a new proof'. Discrete Math. **287**, 193–194 (2004)

[Vos92] Voß, S.: Steiner's problem in graphs: heuristic methods. Discrete Appl. Math. **40**, 45–72 (1992)

[Wag36] Wagner, K.: Bemerkungen zum Vierfarbenproblem. Jahresber. DMV **46**, 26–32 (1936)

[Wag37] Wagner, K.: Über eine Eigenschaft der ebenen Komplexe. Math. Ann. **114**, 170–190 (1937)

[Wag60] Wagner, K.: Bemerkungen zu Hadwigers Vermutung. Math. Ann. **141**, 433–451 (1960)

[Wal92] Wallis, W.D.: One-factorizations of the complete graph. In: Dinitz, J.H., Stinson, D.R. (eds.) Contemporary Design Theory: A Collection of Surveys, pp. 593–639. Wiley, New York (1992)

[Wal97] Wallis, W.D.: One-Factorizations. Kluwer Academic, Dordrecht (1997)

[War62] Warshall, S.: A theorem on Boolean matrices. J. Assoc. Comput. Mach. **9**, 11–12 (1962)

[WatNa87] Watanabe, T., Nakamura, A.: Edge-connectivity augmentation problems. J. Comput. Syst. Sci. **35**, 96–144 (1987)

[Wei74] Weintraub, A.: A primal algorithm to solve network flow problems with convex costs. Manag. Sci. **21**, 87–97 (1974)

[Wel68] Welsh, D.J.A.: Kruskal's theorem for matroids. Proc. Camb. Philos. Soc. **64**, 3–4 (1968)

[Wel76] Welsh, D.J.A.: Matroid Theory. Academic Press, New York (1976)

[Wes11] West, D.B.: A short proof of the Berge-Tutte formula and the Gallai-Edmonds structure theorem. Eur. J. Comb. **32**, 674–676 (2011)

[Whi86] White, N. (ed.): Theory of Matroids. Cambridge University Press, Cambridge (1986)

[Whi87] White, N. (ed.): Combinatorial Geometries. Cambridge University Press, Cambridge (1987)

[Whi92] White, N. (ed.): Matroid Applications. Cambridge University Press, Cambridge (1992)

[Whi32a] Whitney, H.: Congruent graphs and the connectivity of graphs. Am. J. Math. **54**, 150–168 (1932)

[Whi32b] Whitney, H.: Non-separable and planar graphs. Trans. Am. Math. Soc. **54**, 339–362 (1932)

[Whi33] Whitney, H.: Planar graphs. Fundam. Math. **21**, 73–84 (1933)

[Whi35] Whitney, H.: On the abstract properties of linear dependence. Am. J. Math. **57**, 509–533 (1935)

[Wil72] Wilson, L.B.: An analysis of the stable marriage assignment problem. BIT Numer. Math. **12**, 569–575 (1972)

[Wil86] Wilson, R.J.: An Eulerian trail through Königsberg. J. Graph Theory **10**, 265–275 (1986)

[Wil89] Wilson, R.J.: A brief history of Hamiltonian graphs. Ann. Discrete Math. **41**, 487–496 (1989)

[Wil2002] Wilson, R.J.: Four Colors Suffice: How the Map Problem Was Solved. Princeton University Press, Princeton (2002)

[Win88] Winkler, P.: The complexity of metric realization. SIAM J. Discrete Math. **1**, 552–559 (1988)

[Wir76] Wirth, N.: Algorithms + Data Structures = Programs. Prentice Hall, Englewood Cliffs (1976)

[Woe03] Woeginger, G.J.: Exact algorithms for NP-hard problems: a survey. In: Springer Lect. Notes Comput. Sci., vol. 2570, pp. 185–207 (2003)

[Woe08] Woeginger, G.J.: Open problems around exact algorithms. Discrete Appl. Math. **156**, 397–405 (2008)

[Wol80] Wolsey, L.A.: Heuristic analysis, linear programming and branch and bound. Math. Program. Stud. **13**, 121–134 (1980)

[WooLi10] Wood, D.R., Linusson, S.: Thomassen's choosability argument revisited. SIAM J. Discrete Math. **24**, 1632–1637 (2010)

[Woo01] Woodall, D.R.: List colourings of graphs. In: Hirschfeld, J.W.P. (ed.) Combinatorial Surveys, pp. 269–301. Cambridge University Press, Cambridge (2001)

[Yan78] Yannakakis, M.: Node- and edge-deletion NP-complete problems. In: Proc. 10th ACM Symp. on Theory of Computing, pp. 253–264. Assoc. Comput. Mach., New York (1978)

[YanGa80] Yannakakis, M., Gavril, F.: Edge dominating sets in graphs. SIAM J. Appl. Math. **38**, 364–372 (1980)

[Yao75] Yao, A.C.: An $O(|E|\log\log|V|)$ algorithm for finding minimum spanning trees. Inf. Process. Lett. **4**, 21–23 (1975)

[Yap86] Yap, H.P.: Some Topics in Graph Theory. Cambridge University Press, Cambridge (1986)

[YouTO91] Young, N.E., Tarjan, R.E., Orlin, J.B.: Faster parametric shortest path and minimum-balance algorithms. Networks **21**, 205–221 (1991)

[Yu98] Yu, G. (ed.): Operations Research in the Airline Industry. Springer, New York (1998)

[YusZw97] Yuster, R., Zwick, U.: Finding even cycles even faster. SIAM J. Discrete Math. **10**, 209–222 (1997)

[Zad72] Zadeh, N.: Theoretical efficiency of the Edmonds-Karp algorithm for computing maximal flows. J. Assoc. Comput. Mach. **19**, 248–264 (1972)

[Zad73a] Zadeh, N.: A bad network problem for the simplex method and other minimum cost flow algorithms. Math. Program. **5**, 255–266 (1973)

[Zad73b] Zadeh, N.: More pathological examples for network flow problems. Math. Program. **5**, 217–224 (1973)

[Zim81] Zimmermann, U.: Linear and Combinatorial Optimization in Ordered Algebraic Structures. North Holland, Amsterdam (1981)

[Zuc96] Zuckerman, D.: On unapproximable versions of NP-complete problems. SIAM J. Comput. **25**, 1293–1304 (1996)

[Zwi95] Zwick, U.: The smallest networks on which the Ford-Fulkerson maximum flow procedure may fail to terminate. Theor. Comput. Sci. **148**, 165–170 (1995)

Index

Welcome back, my friends,
to the show that never ends....
Emerson, Lake & Palmer