

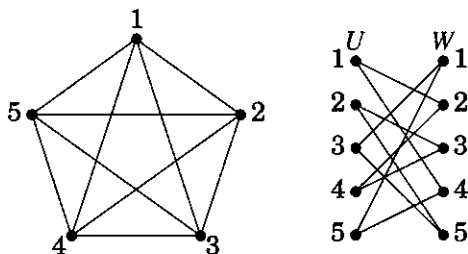
Petersen also proved a sufficient condition for 2-factors. A connected graph with even vertex degrees is Eulerian (Theorem 1.2.26) and decomposes into edge-disjoint cycles (Proposition 1.2.27). For regular graphs of even degree, the cycles in some decomposition can be grouped to form 2-factors.

3.3.9. Theorem. (Petersen [1891]) Every regular graph of even degree has a 2-factor.

Proof: Let G be a $2k$ -regular graph with vertices v_1, \dots, v_n . Every component of G is Eulerian, with some Eulerian circuit C . For each component, define a bipartite graph H with vertices u_1, \dots, u_n and w_1, \dots, w_n by putting $u_i \leftrightarrow w_j$ if v_j immediately follows v_i somewhere on C . Because C enters and exits each vertex k times, H is k -regular. (Actually, H is the split of the digraph obtained by orienting G in accordance with C —see Definition 1.4.20.)

Being a regular bipartite graph, H has a 1-factor M (Corollary 3.1.13). The edge incident to w_i in H corresponds to an edge entering v_i in C . The edge incident to u_i in H corresponds to an edge exiting v_i . Thus the 1-factor in H transforms into a 2-regular spanning subgraph of this component of G . Doing this for each component of G yields a 2-factor of G . ■

3.3.10. Example. *Construction of a 2-factor.* Consider the Eulerian circuit in $G = K_5$ that successively visits 1231425435. The corresponding bipartite graph H is on the right. For the 1-factor whose u, w -pairs are 12, 43, 25, 31, 54, the resulting 2-factor is the cycle (1, 2, 5, 4, 3). The remaining edges form another 1-factor, which corresponds to the 2-factor (1, 4, 2, 3, 5) that remains in G . ■



***f*-FACTORS OF GRAPHS (optional)**

A factor is a spanning subgraph of G ; we ask about existence of factors of special types. A k -factor is a k -regular factor; we have studied 1-factors and 2-factors. We can try to specify the degree at each vertex.

3.3.11. Definition. Given a function $f: V(G) \rightarrow \mathbb{N} \cup \{0\}$, an f -factor of a graph G is a subgraph H such that $d_H(v) = f(v)$ for all $v \in V(G)$.

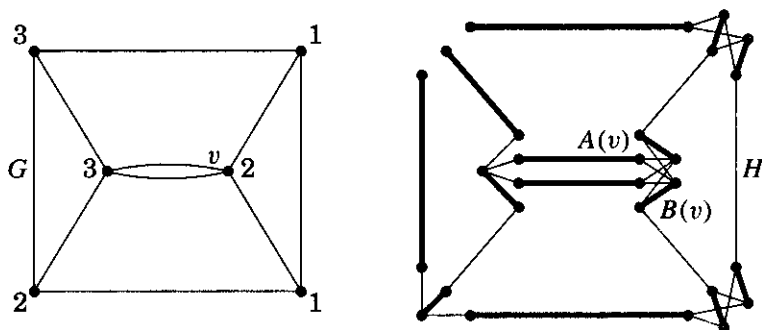
Tutte [1952] proved a necessary and sufficient condition for a graph G to have an f -factor (see Exercise 29). He later reduced the problem to checking for

a 1-factor in a related simple graph. We describe this reduction; it is a beautiful example of transforming a graph problem into a previously solved problem.

3.3.12. Example. A graph transformation (Tutte [1954a]). We assume that $f(w) \leq d(w)$ for all w ; otherwise G has too few edges at w to have an f -factor. We then construct a graph H that has a 1-factor if and only if G has an f -factor. Let $e(w) = d(w) - f(w)$; this is the *excess degree* at w and is nonnegative.

To construct H , replace each vertex v with a biclique $K_{d(v), e(v)}$ having partite sets $A(v)$ of size $d(v)$ and $B(v)$ of size $e(v)$. For each $vw \in E(G)$, add an edge joining one vertex of $A(v)$ to one vertex of $A(w)$. Each vertex of $A(v)$ participates in one such edge.

The figure below shows a graph G , vertex labels given by f , and the resulting simple graph H . The bold edges in H form a 1-factor that corresponds to an f -factor of G . In this example, the f -factor is not unique. ■



3.3.13. Theorem. A graph G has an f -factor if and only if the graph H constructed from G and f as in Example 3.3.12 has a 1-factor.

Proof: Necessity. If G has an f -factor, then the corresponding edges in H leave $e(v)$ vertices of $A(v)$ unmatched; match them arbitrarily to the vertices of $B(v)$ to obtain a 1-factor of H .

Sufficiency. From a 1-factor of H , deleting $B(v)$ and the vertices of $A(v)$ matched into $B(v)$ leaves $f(v)$ edges at v . Doing this for each v and merging the remaining $f(v)$ vertices of each $A(v)$ yields a subgraph of G with degree $f(v)$ at v . It is an f -factor of G . ■

Tutte's Condition for a 1-factor in the derived graph H of Example 3.3.12 transforms into a necessary and sufficient condition for an f -factor in G . Among the applications is a proof of the Erdős–Gallai [1960] characterization of degree sequences of simple graphs (Exercise 29).

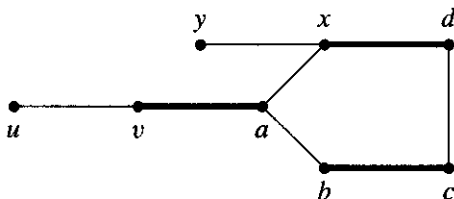
Given an algorithm to find a 1-factor, the correspondence in Theorem 3.3.13 provides an algorithmic test for an f -factor. Instead of just seeking a 1-factor (that is, a perfect matching), we next consider the more general problem of finding a maximum matching in a graph.

EDMONDS' BLOSSOM ALGORITHM (optional)

Berge's Theorem (Theorem 3.1.10) states that a matching M in G has maximum size if and only if G has no M -augmenting path. We can thus find a maximum matching using successive augmenting paths. Since we augment at most $n/2$ times, we obtain a good algorithm if the search for an augmenting path does not take too long. Edmonds [1965a] presented the first such algorithm in his famous paper "Paths, trees, and flowers".

In bipartite graphs, we can search quickly for augmenting paths (Algorithm 3.2.1) because we explore from each vertex at most once. An M -alternating path from u can reach a vertex x in the same partite set as u only along a saturated edge. Hence only once can we reach and explore x . This property fails in graphs with odd cycles, because M -alternating paths from an unsaturated vertex may reach x both along saturated and along unsaturated edges.

3.3.14. Example. In the graph below, with M indicated in bold, a search for shortest M -augmenting paths from u reaches x via the unsaturated edge ax . If we do not also consider a longer path reaching x via a saturated edge, then we miss the augmenting path u, v, a, b, c, d, x, y . ■



We describe Edmonds' solution to this difficulty. If an exploration of M -alternating paths from u reaches a vertex x by an unsaturated edge in one path and by a saturated edge in another path, then x belongs to an odd cycle. Alternating paths from u can diverge only when the next edge is unsaturated (leaving vertex a in Example 3.3.14); when the next edge is saturated there is only one choice for it. From the vertex where the paths diverge, the path reaching x on an unsaturated edge has odd length, and the path reaching it on a saturated edge has even length. Together, they form an odd cycle.

3.3.15. Definition. Let M be a matching in a graph G , and let u be an M -unsaturated vertex. A **flower** is the union of two M -alternating paths from u that reach a vertex x on steps of opposite parity (having not done so earlier). The **stem** of the flower is the maximal common initial path (of nonnegative even length). The **blossom** of the flower is the odd cycle obtained by deleting the stem.

In Example 3.3.14, the flower is the full graph except y , the stem is the path u, v, a , and the blossom is the 5-cycle. The horticultural terminology echoes the use of *tree* for the structures given by most search procedures.

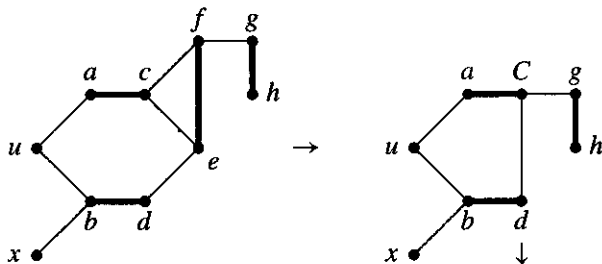
Blossoms do not impede our search. For each vertex z in a blossom, some M -alternating u, z -path reaches z on a saturated edge, found by traveling the proper direction around the blossom to reach z from the stem. We therefore can continue our search along any unsaturated edge from the blossom to a vertex not yet reached. Example 3.3.14 shows such an extension that immediately reaches an unsaturated vertex and completes an M -augmenting path.

Since each vertex of a blossom is saturated by an edge on these paths, no saturated edge emerges from a blossom (except the stem). The effect of these two observations is that we can view the entire blossom as a single “supervertex” reached along the saturated edge at the end of the stem. We search from all vertices of the supervertex blossom simultaneously along unsaturated edges.

We implement this consolidation by contracting the edges of a blossom B when we find it. The result is a new saturated vertex b incident to the last (saturated) edge of the stem. Its other incident edges are the unsaturated edges joining vertices of B to vertices outside B . We explore from b in the usual way to extend our search. We may later find another blossom containing b ; we then contract again. If we find an M -alternating path in the final graph from u to an unsaturated vertex x , then we can undo the contractions to obtain an M -augmenting path to x in the original graph.

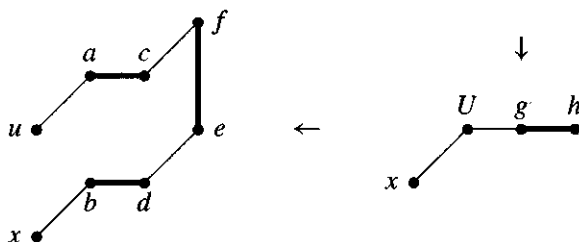
Except for the treatment of blossoms, the approach is that of Algorithm 3.2.1 for exploring M -alternating paths. In the corresponding phrasing, T is the set of vertices of the current graph reached along unsaturated edges, and S is the set of vertices reached along saturated edges. The vertices that arise by contracting blossoms belong to S .

3.3.16. Example. Let M be the bold matching in the graph on the left below. We search from the unsaturated vertex u for an M -augmenting path. We first explore the unsaturated edges incident to u , reaching a and b . Since a and b are saturated, we immediately extend the paths along the edges ac and bd . Now $S = \{u, c, d\}$. If we next explore from c , then we find its neighbors e and f along unsaturated edges. Since $ef \in M$, we discover the blossom with vertex set $\{c, e, f\}$. We contract the blossom to obtain the new vertex C , changing S to $\{u, C, d\}$. This yields the graph on the right.



Suppose we now explore from the vertex $C \in S$. Unsaturated edges take us to g and to d . Since g is saturated by the edge gh , we place h in S . Since d is already in S , we have found another blossom. The paths reaching d are u, b, d and u, a, C, d . We contract the blossom, obtaining the new vertex U and

the graph on the right below, with $S = \{U, h\}$. We next explore from h , finding nothing new (if we exhaust S without reaching an unsaturated vertex, then there is no M -augmenting path from u). Finally, we explore from U , reaching the unsaturated vertex x .



Having recorded the edge on which we reached each vertex, we can extract an M -augmenting u, x -path. We reached x from U , so we expand the blossom back into $\{u, a, C, d, b\}$ and find that x is reached from U along bx . The path in the blossom U that reaches b on a saturated edge ends with C, d, b . Since C is a blossom in the original graph, we expand C back into $\{c, f, e\}$. Note that d is reached from C by the unsaturated edge ed . The path from the “base” of C that reaches e along a saturated edge is c, f, e . Finally, c was reached from a and a from u , so we obtain the full M -augmenting path u, a, c, f, e, d, b, x . ■

We summarize the steps of the algorithm, glossing over the details of implementation, especially the treatment of contractions.

3.3.17. Algorithm. (Edmonds’ Blossom Algorithm [1965a]—sketch).

Input: A graph G , a matching M in G , an M -unsaturated vertex u .

Idea: Explore M -alternating paths from u , recording for each vertex the vertex from which it was reached, and contracting blossoms when found. Maintain sets S and T analogous to those in Algorithm 3.2.1, with S consisting of u and the vertices reached along saturated edges. Reaching an unsaturated vertex yields an augmentation.

Initialization: $S = \{u\}$ and $T = \emptyset$.

Iteration: If S has no unmarked vertex, stop; there is no M -augmenting path from u . Otherwise, select an unmarked $v \in S$. To explore from v , successively consider each $y \in N(v)$ such that $y \notin T$.

If y is unsaturated by M , then trace back from y (expanding blossoms as needed) to report an M -augmenting u, y -path.

If $y \in S$, then a blossom has been found. Suspend the exploration of v and contract the blossom, replacing its vertices in S and T by a single new vertex in S . Continue the search from this vertex in the smaller graph.

Otherwise, y is matched to some w by M . Include y in T (reached from v), and include w in S (reached from y).

After exploring all such neighbors of v , mark v and iterate. ■

We cannot explore all unsaturated vertices simultaneously as in Algorithm 3.2.1, because the membership of vertices in blossoms depends on the choice of

the initial unsaturated vertex. Nevertheless, if we find no M -augmenting path from u , then we can delete u from the graph and ignore it in the subsequent search for a maximum matching (Exercise 26).

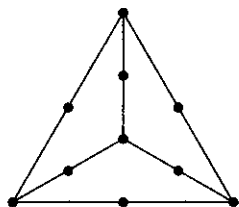
3.3.18. Remark. Edmonds' original algorithm runs in time $O(n^4)$. The implementation in Ahuja–Magnanti–Orlin [1993, p483–494] runs in time $O(n^3)$. This requires (1) appropriate data structures to represent the blossoms and to process contractions, and (2) careful analysis of the number of contractions that can be performed, the time spent exploring edges, and the time spent contracting and expanding blossoms.

The first algorithm solving the maximum matching problem in less than cubic time was the $O(n^{5/2})$ algorithm in Even–Kariv [1975]. The best algorithm now known runs in time $O(n^{1/2}m)$ for a graph with n vertices and m edges (this is faster than $O(n^{5/2})$ for sparse graphs). The algorithm is rather complicated and appears in Micali–Vazirani [1980], with a complete proof in Vazirani [1994].

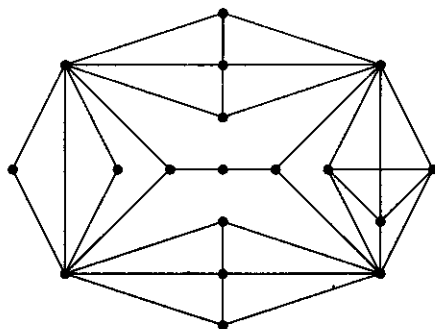
We have not discussed the weighted matching problem for general graphs. Edmonds [1965d] found an algorithm for this, which was implemented in time $O(n^3)$ by Gabow [1975] and by Lawler [1976]. Faster algorithms appear in Gabow [1990] and in Gabow–Tarjan [1989]. ■

EXERCISES

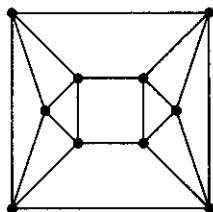
3.3.1. (–) Determine whether the graph below has a 1-factor.



3.3.2. (–) Exhibit a maximum matching in the graph below, and use a result in this section to give a short proof that it has no larger matching.



3.3.3. (–) In the graph drawn below, exhibit a k -factor for each k in $\{0, 1, 2, 3, 4\}$.



3.3.4. (–) Let G be a k -regular bipartite graph. Prove that G can be decomposed into r -factors if and only if r divides k .

3.3.5. (–) Given graphs G and H , determine the number of components and maximum degree of $G \vee H$ in terms of parameters of G and H .

• • • • •

3.3.6. (!) Prove that a tree T has a perfect matching if and only if $o(T - v) = 1$ for every $v \in V(T)$. (Chungphaisan)

3.3.7. (!) For each $k > 1$, construct a k -regular simple graph having no 1-factor.

3.3.8. Prove that if a graph G decomposes into 1-factors, then G has no cut-vertex. Draw a connected 3-regular simple graph that has a 1-factor and has a cut-vertex.

3.3.9. Prove that every graph G has a matching of size at least $n(G)/(1 + \Delta(G))$. (Hint: Apply induction on $e(G)$.) (Weinstein [1963])

3.3.10. (!) For every graph G , prove that $\beta(G) \leq 2\alpha'(G)$. For each $k \in \mathbb{N}$, construct a simple graph G with $\alpha'(G) = k$ and $\beta(G) = 2k$.

3.3.11. Let T be a set of vertices in a graph G . Prove that G has a matching that saturates T if and only if for all $S \subseteq V(G)$, the number of odd components of $G - S$ contained in $G[T]$ is at most $|S|$.

3.3.12. (!) *Extension of König–Egerváry Theorem to general graphs.* Given a graph G , let S_1, \dots, S_k and T be subsets of $V(G)$ such that each S_i has odd size. These sets form a **generalized cover** of G if every edge of G has one endpoint in T or both endpoints in some S_i . The **weight** of a generalized cover is $|T| + \sum |S_i|/2$. Let $\beta^*(G)$ be the minimum weight of a generalized cover. Prove that $\alpha'(G) = \beta^*(G)$. (Hint: Apply Corollary 3.3.7. Comment: every vertex cover is a generalized cover, and thus $\beta^*(G) \leq \beta(G)$.)

3.3.13. (+) *Tutte's Theorem from Hall's Theorem.* Let G be a graph such that $o(G - S) \leq |S|$ for all $S \subseteq V(G)$. Let T be a maximal vertex subset such that $o(G - T) = |T|$.

a) Prove that every component of $G - T$ is odd, and conclude that $T \neq \emptyset$.

b) Let C be a component of $G - T$. Prove that Tutte's Condition holds for every subgraph of C obtained by deleting one vertex. (Hint: Since $C - x$ has even order, a violation requires $o(C - x - S) \geq |S| + 2$.)

c) Let H be a bipartite graph with partite sets T and C , where C is the set of components of $G - T$. For $t \in T$ and $C \in \mathcal{C}$, put $tC \in E(H)$ if and only if $N_G(t)$ contains a vertex of C . Prove that H satisfies Hall's Condition for a matching that saturates C .

d) Use parts (a), (b), (c), and Hall's Theorem to prove Tutte's 1-factor Theorem by induction on $n(G)$. (Anderson [1971], Mader [1973])

3.3.14. For $k \in \mathbb{N}$, let G be a simple graph such that $\delta(G) \geq k$ and $n(G) \geq 2k$. Prove that $\alpha'(G) \geq k$. (Hint: Apply Corollary 3.3.7.) (Brandt [1994])

3.3.15. Let G be a 3-regular graph with at most two cut-edges. Prove that G has a 1-factor. (Petersen [1891])

3.3.16. (!) Let G be a k -regular graph of even order that remains connected when any $k - 2$ edges are deleted. Prove that G has a 1-factor.

3.3.17. With G as in Exercise 3.3.16, use Remark 3.3.5 to prove that every edge of G belongs to some 1-factor. (Comment: This strengthens Exercise 3.3.16.) (Schönberger [1934] for $k = 3$, Berge [1973, p162])

3.3.18. (+) For each odd k greater than 1, construct a graph G with no 1-factor that is k -regular and remains connected when any $k - 3$ edges are deleted. (Comment: Thus Exercise 3.3.16 is best possible.)

3.3.19. (!) Let G be a 3-regular simple graph with no cut-edge. Prove that G decomposes into copies of P_4 . (Hint: Use Theorem 3.3.9.)

3.3.20. (!) Prove that a 3-regular simple graph has a 1-factor if and only if it decomposes into copies of P_4 .

3.3.21. (+) Let G be a $2m$ -regular graph, and let T be a tree with m edges. Prove that if the diameter of T is less than the girth of G , then G decomposes into copies of T . (Hint: Use Theorem 3.3.9 to give an inductive proof of the stronger result that G has such a decomposition in which each vertex is used once as an image of each vertex of T .) (Häggkvist)

3.3.22. (!) Let G be an X, Y -bigraph. Let H be the graph obtained from G by adding one vertex to Y if $n(G)$ is odd and then adding edges to make Y a clique.

a) Prove that G has a matching of size $|X|$ if and only if H has a 1-factor.

b) Prove that if G satisfies Hall's Condition ($|N(S)| \geq |S|$ for all $S \subseteq X$), then H satisfies Tutte's Condition ($o(H - T) \leq |T|$ for all $T \subseteq V(H)$).

c) Use parts (a) and (b) to obtain Hall's Theorem from Tutte's Theorem.

3.3.23. Let G be a claw-free connected graph of even order.

a) Let T be a spanning tree of G generated by Breadth-First Search (Algorithm 2.3.8). Let x and y be vertices that have a common parent in T other than the root. Prove that x and y must be adjacent.

b) Use part (a) to prove that G has a 1-factor. (Comment: Without part (a), one can simply prove the stronger result that the last edge in a longest path belongs to a 1-factor.) (Sumner [1974a], Las Vergnas [1975])

3.3.24. (!) Let G be a simple graph of even order n having a set S of size k such that $o(G - S) > k$. Prove that G has at most $\binom{k}{2} + k(n - k) + \binom{n - 2k - 1}{2}$ edges, and that this bound is best possible. Use this to determine the maximum size of a simple n -vertex graph with no 1-factor. (Erdős-Gallai [1961])

3.3.25. A graph G is **factor-critical** if each subgraph $G - v$ obtained by deleting one vertex has a 1-factor. Prove that G is factor-critical if and only if $n(G)$ is odd and $o(G - S) \leq |S|$ for all nonempty $S \subseteq V(G)$. (Gallai [1963a])

3.3.26. (!) Let M be a matching in a graph G , and let u be an M -unsaturated vertex. Prove that if G has no M -augmenting path that starts at u , then u is unsaturated in some maximum matching in G .

3.3.27. (*) Assuming that Algorithm 3.3.17 is correct, we develop an algorithmic proof of Tutte's Theorem (Theorem 3.3.3).

a) Let G be a graph with no perfect matching, and let M be a maximum matching in G . Let S and T be the sets generated when running Algorithm 3.3.17 from u . Prove that $|T| < |S| \leq o(G - T)$.

b) Use part (a) to prove Theorem 3.3.3.

3.3.28. (*) Let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Given $f: V(G) \rightarrow \mathbb{N}_0$, the graph G is f -soluble if there exists $w: E(G) \rightarrow \mathbb{N}_0$ such that $\sum_{uv \in E(G)} w(uv) = f(v)$ for every $v \in V(G)$.

a) Prove that G has an f -factor if and only if the graph H obtained from G by subdividing each edge twice and defining f to be 1 on the new vertices is f -soluble. (This reduces testing for an f -factor to testing f -solubility.)

b) Given G and an $f: V(G) \rightarrow \mathbb{N}_0$, construct a graph H (with proof) such that G is f -soluble if and only if H has a 1-factor. (Tutte [1954a])

3.3.29. (*+) *Tutte's f -factor condition and graphic sequences.* Given $f: V(G) \rightarrow \mathbb{N}_0$, define $f(S) = \sum_{v \in S} f(v)$ for $S \subseteq V(G)$. For any disjoint subsets S, T of $V(G)$, let $q(S, T)$ denote the number of components Q of $G - S - T$ such that $e(Q, T) + f(V(Q))$ is odd, where $e(Q, T)$ is the number of edges from Q to T . Tutte [1952, 1954a] proved that G has an f -factor if and only if

$$q(S, T) + f(T) - \sum_{v \in T} d_{G-S}(v) \leq f(S)$$

for all choices of disjoint subsets $S, T \subset V$.

a) **The Parity Lemma.** Let $\delta(S, T) = f(S) - f(T) + \sum_{v \in T} d_{G-S}(v) - q(S, T)$. Prove that $\delta(S, T)$ has the same parity as $f(V)$ for disjoint sets $S, T \subseteq V(G)$. (Hint: Use induction on $|T|$.)

b) Suppose that $G = K_n$ and $f(v_i) = d_i$, where $\sum d_i$ is even and $d_1 \geq \dots \geq d_n$. Use the f -factor condition and part (a) to prove that G has an f -factor if and only if $\sum_{i=1}^k d_i \leq (n-1-s)k + \sum_{i=n+1-s}^n d_i$ for all k, s with $k+s \leq n$.

c) Conclude that $d_1, \dots, d_n \geq 0$ are the vertex degrees of a simple graph if and only if $\sum d_i$ is even and $\sum_{i=1}^k d_i \leq k(k-1) + \sum_{i=k+1}^n \min\{k, d_i\}$ for $1 \leq k \leq n$. (Erdős–Gallai [1960])

Chapter 4

Connectivity and Paths

4.1. Cuts and Connectivity

A good communication network is hard to disrupt. We want the graph (or digraph) of possible transmissions to remain connected even when some vertices or edges fail. When communication links are expensive, we want to achieve these goals with few edges. Loops are irrelevant for connection, so in this chapter we assume that our graphs and digraphs **have no loops**, especially when considering degree conditions.

CONNECTIVITY

How many vertices must be deleted to disconnect a graph?

4.1.1. Definition. A **separating set** or **vertex cut** of a graph G is a set $S \subseteq V(G)$ such that $G - S$ has more than one component. The **connectivity** of G , written $\kappa(G)$, is the minimum size of a vertex set S such that $G - S$ is disconnected or has only one vertex. A graph G is **k -connected** if its connectivity is at least k .

A graph other than a complete graph is k -connected if and only if every separating set has size at least k . We can view “ k -connected” as a structural condition, while “connectivity k ” is the solution of an optimization problem.

4.1.2. Example. *Connectivity of K_n and $K_{m,n}$.* Because a clique has no separating set, we need to adopt a convention for its connectivity. This explains the phrase “or has only one vertex” in Definition 4.1.1. We obtain $\kappa(K_n) = n - 1$, while $\kappa(G) \leq n(G) - 2$ when G is not a complete graph. With this convention, most general results about connectivity remain valid on complete graphs.

Consider a bipartition X, Y of $K_{m,n}$. Every induced subgraph that has at least one vertex from X and from Y is connected. Hence every separating set of $K_{m,n}$ contains X or Y . Since X and Y themselves are separating sets (or leave only one vertex), we have $\kappa(K_{m,n}) = \min\{m, n\}$. The connectivity of $K_{3,3}$ is 3; the graph is 1-connected, 2-connected, and 3-connected, but not 4-connected. ■

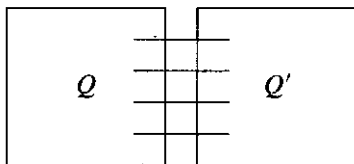
A graph with more than two vertices has connectivity 1 if and only if it is connected and has a cutvertex. A graph with more than one vertex has connectivity 0 if and only if it is disconnected. The 1-vertex graph K_1 is annoyingly inconsistent; it is connected, but for consistency in discussing connectivity we set $\kappa(K_1) = 0$.

4.1.3. Example. *The hypercube Q_k .* For $k \geq 2$, the neighbors of one vertex in Q_k form a separating set, so $\kappa(Q_k) \leq k$. To prove that $\kappa(Q_k) = k$, we show that every vertex cut has size at least k . We use induction on k .

Basis step: $k \in \{0, 1\}$. For $k \leq 1$, Q_k is a clique with $k + 1$ vertices and has connectivity k .

Induction step: $k \geq 2$. By the induction hypothesis, $\kappa(Q_{k-1}) = k - 1$. Consider the description of Q_k as two copies Q and Q' of Q_{k-1} plus a matching that joins corresponding vertices in Q and Q' (Example 1.3.8). Let S be a vertex cut in Q_k . If $Q - S$ is connected and $Q' - S$ is connected, then $Q_k - S$ is also connected unless S contains at least one endpoint of every matched pair. This requires $|S| \geq 2^{k-1}$, but $2^{k-1} \geq k$ for $k \geq 2$.

Hence we may assume that $Q - S$ is disconnected, which means that S has at least $k - 1$ vertices in Q , by the induction hypothesis. If S contains no vertices of Q' , then $Q' - S$ is connected and all vertices of $Q - S$ have neighbors in $Q' - S$, so $Q_k - S$ is connected. Hence S must also contain a vertex of Q' . This yields $|S| \geq k$, as desired. ■



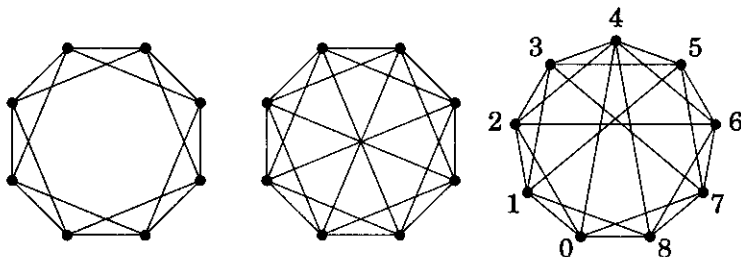
Deleting the neighbors of a vertex disconnects a graph (or leaves only one vertex), so $\kappa(G) \leq \delta(G)$. Equality need not hold; $2K_m$ has minimum degree $m - 1$ but connectivity 0.

Since connectivity k requires $\delta(G) \geq k$, it also requires at least $\lceil kn/2 \rceil$ edges. The k -dimensional cube achieves this bound, but only for the case $n = 2^k$. The bound is best possible whenever $k < n$, as shown by the next example.

4.1.4. Example. *Harary graphs.* Given $k < n$, place n vertices around a circle, equally spaced. If k is even, form $H_{k,n}$ by making each vertex adjacent to the nearest $k/2$ vertices in each direction around the circle. If k is odd and n is even, form $H_{k,n}$ by making each vertex adjacent to the nearest $(k - 1)/2$ vertices

in each direction and to the diametrically opposite vertex. In each case, $H_{k,n}$ is k -regular.

When k and n are both odd, index the vertices by the integers modulo n . Construct $H_{k,n}$ from $H_{k-1,n}$ by adding the edges $i \leftrightarrow i + (n-1)/2$ for $0 \leq i \leq (n-1)/2$. The graphs $H_{4,8}$, $H_{5,8}$, and $H_{5,9}$ appear below. ■



4.1.5. Theorem. (Harary [1962a]) $\kappa(H_{k,n}) = k$, and hence the minimum number of edges in a k -connected graph on n vertices is $\lceil kn/2 \rceil$.

Proof: We prove only the even case $k = 2r$, leaving the odd case as Exercise 12. Let $G = H_{k,n}$. Since $\delta(G) = k$, it suffices to prove $\kappa(G) \geq k$. For $S \subseteq V(G)$ with $|S| < k$, we prove that $G - S$ is connected. Consider $u, v \in V(G) - S$. The original circular arrangement has a clockwise u, v -path and a counterclockwise u, v -path along the circle; let A and B be the sets of internal vertices on these two paths.

Since $|S| < k$, the pigeonhole principle implies that in one of $\{A, B\}$, S has fewer than $k/2$ vertices. Since in G each vertex has edges to the next $k/2$ vertices in a particular direction, deleting fewer than $k/2$ consecutive vertices cannot block travel in that direction. Thus we can find a u, v -path in $G - S$ via the set A or B in which S has fewer than $k/2$ vertices. ■

Harary's construction determines the degree conditions that *allow* a graph to be k -connected. Exercise 22 determines the degree conditions that *force* a simple graph to be k -connected. Since it depends on vertex deletions, connectivity is not affected by deleting extra copies of multiple edges. Hence we state degree conditions for k -connectedness only in the context of simple graphs.

4.1.6. Remark. A direct proof of $\kappa(G) \geq k$ considers a vertex cut S and proves that $|S| \geq k$, or it considers a set S with fewer than k vertices and proves that $G - S$ is connected. The indirect approach assumes a cut of size less than k and obtains a contradiction. The indirect proof may be easier to find, but the direct proof may be clearer to state.

Note also that if $k < n(G)$ and G has a vertex cut of size less than k , then G has a vertex cut of size $k-1$ (first delete the cut, then continue deleting vertices until $k-1$ are gone, retaining a vertex in each of two components).

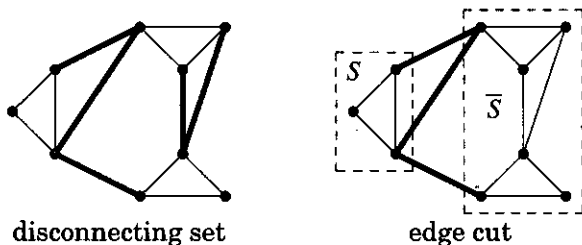
Finally, proving $\kappa(G) = k$ also requires presenting a vertex cut of size k ; this is usually the easy part. ■

EDGE-CONNECTIVITY

Perhaps our transmitters are secure and never fail, but our communication links are subject to noise or other disruptions. In this situation, we want to make it hard to disconnect our graph by deleting edges.

4.1.7. Definition. A **disconnecting set** of edges is a set $F \subseteq E(G)$ such that $G - F$ has more than one component. A graph is **k -edge-connected** if every disconnecting set has at least k edges. The **edge-connectivity** of G , written $\kappa'(G)$, is the minimum size of a disconnecting set (equivalently, the maximum k such that G is k -edge-connected).

Given $S, T \subseteq V(G)$, we write $[S, T]$ for the set of edges having one endpoint in S and the other in T . An **edge cut** is an edge set of the form $[S, \bar{S}]$, where S is a nonempty proper subset of $V(G)$ and \bar{S} denotes $V(G) - S$.



4.1.8. Remark. *Disconnecting set vs. edge cut.* Every edge cut is a disconnecting set, since $G - [S, \bar{S}]$ has no path from S to \bar{S} . The converse is false, since a disconnecting set can have extra edges. Above we show a disconnecting set and an edge cut in bold; see also Exercise 13.

Nevertheless, *every minimal disconnecting set of edges is an edge cut* (when $n(G) > 1$). If $G - F$ has more than one component for some $F \subseteq E(G)$, then for some component H of $G - F$ we have deleted all edges with exactly one endpoint in H . Hence F contains the edge cut $[V(H), \bar{V(H)}]$, and F is not a minimal disconnecting set unless $F = [V(H), \bar{V(H)}]$. ■

The notation for edge-connectivity continues our convention of using a “prime” for an edge parameter analogous to a vertex parameter. Using the same base letter emphasizes the analogy and avoids the confusion of using many different letters – and running out of them.

Deleting one endpoint of each edge in an edge cut F deletes every edge of F . This suggests that $\kappa(G) \leq \kappa'(G)$. However, we must be careful not to delete the only vertex of a component of $G - F$ and thereby leave a connected subgraph.

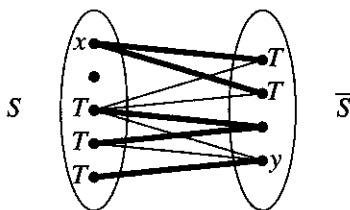
4.1.9. Theorem. (Whitney [1932a]) If G is a simple graph, then

$$\kappa(G) \leq \kappa'(G) \leq \delta(G).$$

Proof: The edges incident to a vertex v of minimum degree form an edge cut; hence $\kappa'(G) \leq \delta(G)$. It remains to show that $\kappa(G) \leq \kappa'(G)$.

We have observed that $\kappa(G) \leq n(G) - 1$ (see Example 4.1.2). Consider a smallest edge cut $[S, \bar{S}]$. If every vertex of S is adjacent to every vertex of \bar{S} , then $|[S, \bar{S}]| = |S| |\bar{S}| \geq n(G) - 1 \geq \kappa(G)$, and the desired inequality holds.

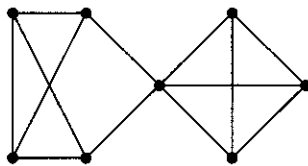
Otherwise, we choose $x \in S$ and $y \in \bar{S}$ with $x \not\sim y$. Let T consist of all neighbors of x in \bar{S} and all vertices of $S - \{x\}$ with neighbors in \bar{S} . Every x, y -path passes through T , so T is a separating set. Also, picking the edges from x to $T \cap \bar{S}$ and one edge from each vertex of $T \cap S$ to \bar{S} (shown bold below) yields $|T|$ distinct edges of $[S, \bar{S}]$. Thus $\kappa'(G) = |[S, \bar{S}]| \geq |T| \geq \kappa(G)$. ■



We have seen that $\kappa(G) = \delta(G)$ when G is a complete graph, a biclique, a hypercube, or a Harary graph. By Theorem 4.1.9, also $\kappa'(G) = \delta(G)$ for these graphs. Nevertheless, in many graphs the set of edges incident to a vertex of minimum degree is not a minimum edge cut. The situation $\kappa'(G) < \delta(G)$ is precisely the situation where no minimum edge cut isolates a vertex.

4.1.10. Example. Possibility of $\kappa < \kappa' < \delta$. For graph G below, $\kappa(G) = 1$, $\kappa'(G) = 2$, and $\delta(G) = 3$. Note that no minimum edge cut isolates a vertex.

Each inequality can be arbitrarily weak. When $G = K_m + K_m$, we have $\kappa(G) = \kappa'(G) = 0$ but $\delta(G) = m - 1$. When G consists of two m -cliques sharing a single vertex, we have $\kappa'(G) = \delta(G) = m - 1$ but $\kappa(G) = 1$. ■



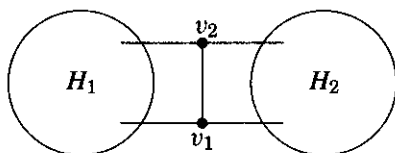
Various conditions force equalities among the parameters; for example, $\kappa'(G) = \delta(G)$ when G has diameter 2 (Exercise 25). For 3-regular graphs, connectivity and edge-connectivity are always equal.

4.1.11. Theorem. If G is a 3-regular graph, then $\kappa(G) = \kappa'(G)$.

Proof: Let S be a minimum vertex cut ($|S| = \kappa(G)$). Since $\kappa(G) \leq \kappa'(G)$ always, we need only provide an edge cut of size $|S|$. Let H_1, H_2 be two components of $G - S$. Since S is a minimum vertex cut, each $v \in S$ has a neighbor in H_1 and a neighbor in H_2 . Since G is 3-regular, v cannot have two neighbors in H_1 and

two in H_2 . For each $v \in S$, delete the edge from v to a member of $\{H_1, H_2\}$ where v has only one neighbor.

These $\kappa(G)$ edges break all paths from H_1 to H_2 except in the case below, where a path can enter S via v_1 and leave via v_2 . In this case we delete the edge to H_1 for both v_1 and v_2 to break all paths from H_1 to H_2 through $\{v_1, v_2\}$. ■



When $\kappa'(G) < \delta(G)$, a minimum edge cut cannot isolate a vertex. In fact, whenever $|[S, \bar{S}]| < \delta(G)$, the set S (and also \bar{S}) must be much larger than a single vertex. This follows from a simple relationship between the size of the edge cut $[S, \bar{S}]$ and the size of the subgraph induced by S .

4.1.12. Proposition. If S is a set of vertices in a graph G , then

$$|[S, \bar{S}]| = [\sum_{v \in S} d(v)] - 2e(G[S]).$$

Proof: An edge in $G[S]$ contributes twice to $\sum_{v \in S} d(v)$, while an edge in $[S, \bar{S}]$ contributes only once to the sum. Since this counts all contributions, we obtain $\sum_{v \in S} d(v) = |[S, \bar{S}]| + 2e(G[S])$. ■

4.1.13. Corollary. If G is a simple graph and $|[S, \bar{S}]| < \delta(G)$ for some nonempty proper subset S of $V(G)$, then $|S| > \delta(G)$.

Proof: By Proposition 4.1.12, we have $\delta(G) > \sum_{v \in S} d(v) - 2e(G[S])$. Using $d(v) \geq \delta(G)$ and $2e(G[S]) \leq |S|(|S| - 1)$ yields

$$\delta(G) > |S|\delta(G) - |S|(|S| - 1).$$

This inequality requires $|S| > 1$, so we can combine the terms involving $\delta(G)$ and cancel $|S| - 1$ to obtain $|S| > \delta(G)$. ■

As a set of edges, an edge cut may contain another edge cut. For example, $K_{1,2}$ has three edge cuts, but one of them contains the other two. The minimal non-empty edge cuts of a graph have useful structural properties.

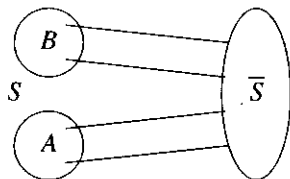
4.1.14. Definition. A **bond** is a minimal nonempty edge cut.

Here “minimal” means that no proper nonempty subset is also an edge cut. We characterize bonds in connected graphs.

4.1.15. Proposition. If G is a connected graph, then an edge cut F is a bond if and only if $G - F$ has exactly two components.

Proof: Let $F = [S, \bar{S}]$ be an edge cut. Suppose first that $G - F$ has exactly two components, and let F' be a proper subset of F . The graph $G - F'$ contains the two components of $G - F$ plus at least one edge between them, making it connected. Hence F is a minimal disconnecting set and is a bond.

For the converse, suppose that $G - F$ has more than two components. Since $G - F$ is the disjoint union of $G[S]$ and $G[\bar{S}]$, one of these has at least two components. Assume by symmetry that it is $G[S]$. We can thus write $S = A \cup B$, where no edges join A and B . Now the edge cuts $[A, \bar{A}]$ and $[B, \bar{B}]$ are proper subsets of F , so F is not a bond. ■

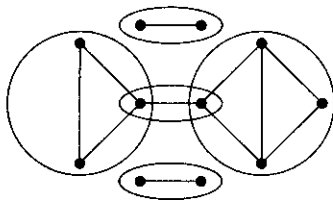


BLOCKS

A connected graph with no cut-vertex need not be 2-connected, since it can be K_1 or K_2 . Connected subgraphs without cut-vertices provide a useful decomposition of a graph.

4.1.16. Definition. A **block** of a graph G is a maximal connected subgraph of G that has no cut-vertex. If G itself is connected and has no cut-vertex, then G is a block.

4.1.17. Example. Blocks. If H is a block of G , then H as a graph has no cut-vertex, but H may contain vertices that are cut-vertices of G . For example, the graph drawn below has five blocks; three copies of K_2 , one of K_3 , and one subgraph that is neither a cycle nor a clique. ■



4.1.18. Remark. Properties of blocks. An edge of a cycle cannot itself be a block, since it is in a larger subgraph with no cut-vertex. Hence an edge is a block if and only if it is a cut-edge; the blocks of a tree are its edges. If a block has more than two vertices, then it is 2-connected. The blocks of a loopless graph are its isolated vertices, its cut-edges, and its maximal 2-connected subgraphs. ■