– Graph Theory and Its Applications –

Network Flows and Connectivity

Xinmin Hou

Department of Computational and Applied Mathematics
xmhou@ustc.edu.cn

Contents

1 Network Flows **4**

Exercises 13

2 Menger's Theorem **15**

Menger's Theorem (vertex form) 21

Exercises 28

3 Connectivity **29**

Exercises 46

4 Design of Transport Plans – Maximum-Flow Algorithm for Capacity Networks **48**

Exercises 59

5 Design of Optimal Transport Plans – Minimum-Cost Maximum-Flow Algorithm* 60

Exercises* 74

6 Chinese Postman Problem 76

Eulerian case 77

Non-Eulerian case 80

Exercises 91

7 Construction of Squared Rectangles 93

1 Network Flows

- A network is a weighted connected graph $(D, \mathbf{w})$ with two distinguished vertices x and y, denoted by $N = (D_{xy}, \mathbf{w})$; the vertices x and y are called the source and sink, respectively.
- If $\mathbf{w}$ is a non-negative **capacity function** $\mathbf{c}$, then the network $N = (D_{xy}, \mathbf{c})$ is called a capacity network; the value $\mathbf{c}(a)$ of $\mathbf{c}$ on an edge a is called the capacity of a.
- If $\mathbf{c}(a)$ is a non-negative integer for every $a \in E(D)$, then N is called an integer capacity network. Figure 1 (a) shows an integer capacity network, where the number on an edge a is $\mathbf{c}(a)$.

- Let $N = (D_{xy}, \mathbf{c})$ be a capacity network. If there exists $\mathbf{f} \in \mathcal{E}(D)$ such that

$$0 \leqslant \mathbf{f}(a) \leqslant \mathbf{c}(a), \quad \forall\ a \in E(D), \tag{1}$$

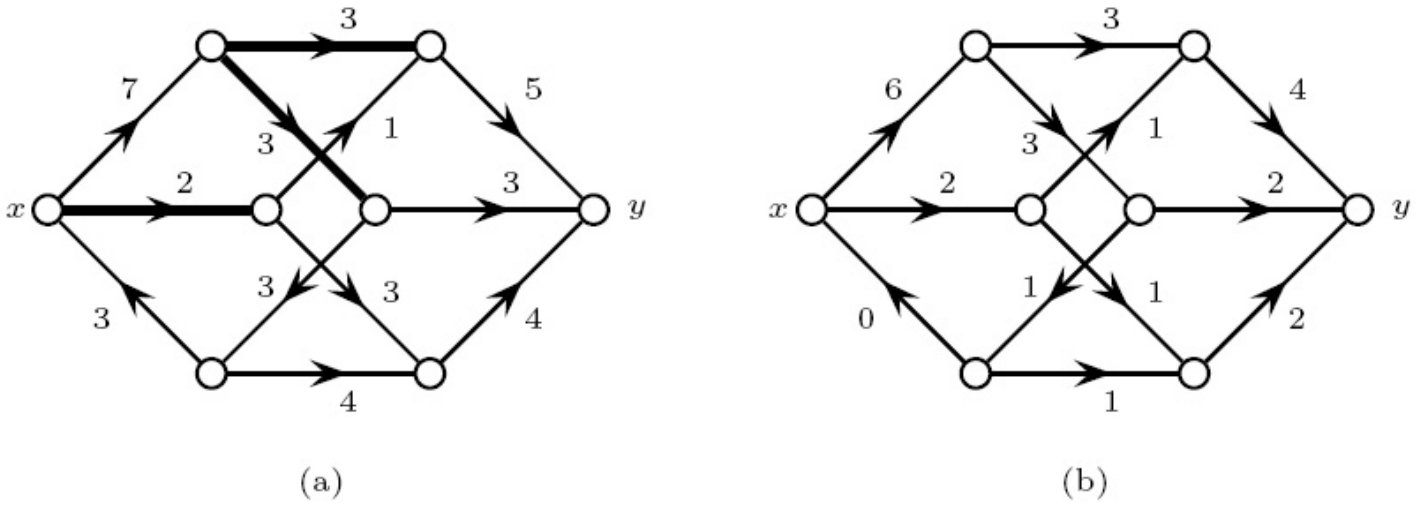

Figure 1: (a) the network $N = (D_{xy}, \mathbf{c})$ (b) an (x,y)-flow $\mathbf{f}$ in N

and

$$\mathbf{f}^+(u) = \mathbf{f}^-(u), \quad \forall\ u \in V(D) \setminus \{x,y\}, \tag{2}$$

then $\mathbf{f}$ is called an flow from x to y in N, abbreviated as an (x,y)-flow.

- Every network admits at least one flow, for example $\mathbf{f}(a) = 0$ for all

$a \in E(D)$, called the zero flow. Figure 1 (b) shows an (x,y)-flow $\mathbf{f}$ of the integer capacity network $N = (D_{xy}, \mathbf{c})$ in (a); the number on an edge a is $\mathbf{f}(a)$.

- From condition (2) one readily verifies that for every (x,y)-flow $\mathbf{f}$ in $N = (D_{xy}, \mathbf{c})$,

$$\mathbf{f}^+(x) - \mathbf{f}^-(x) = \mathbf{f}^-(y) - \mathbf{f}^+(y). \quad \text{(HOW TO PROVE IT?)} \qquad (3)$$

  The common value of the two sides is called the value of the flow, denoted by val $\mathbf{f}$.

- For example, the value of the (x,y)-flow $\mathbf{f}$ in Figure 1 (b) is val $\mathbf{f} = 8$.

- An (x,y)-flow of maximum value in N is called a maximum flow in N.

- A directed edge set B of D of the form $(S, \overline{S})$ is called an (x,y)-edge cut, where $x \in S$, $y \in \overline{S}$.

$$\text{cap}B = \mathbf{c}(B) = \sum_{a \in B} \mathbf{c}(a)$$

is called the **capacity** of B. An (x,y)-edge cut of minimum capacity is called a minimum cut.

- For example, in the capacity network $N = (D_{xy}, \mathbf{c})$ of Figure 1 (a), the bold edges form an (x,y)-edge cut B with capacity cap $B = 8$.

An efficient algorithm for finding a maximum flow and a minimum cut will be given in Section 4.4. First we prove the following celebrated result on the relation between the maximum flow value and the minimum cut capacity:

Theorem 4.1 (Max-Flow Min-Cut Theorem (L. R. Ford, Jr. & D. R. Fulkerson, 1956)) In every capacity network N, the maximum flow value equals the minimum cut capacity.

$$\max_{\mathbf{f}}\{val(\mathbf{f}) : \mathbf{f} \text{ is an } (x,y)\text{-flow}\}$$
$$= \min_{B}\{cap(B) : B \text{ is an } (x,y)\text{-edge cut}\}.$$

Proof

- Let $N = (D_{xy}, \mathbf{c})$ be a capacity network, let $\mathbf{f}$ be a maximum flow in N, and let $B = (S, \overline{S})$ be a minimum (x,y)-edge cut in N.
- From the definitions of a flow and its value, for every $u \in S$,

$$\mathbf{f}^+(u) - \mathbf{f}^-(u) = \begin{cases} \operatorname{val}\mathbf{f}, & \text{if } u = x; \\ 0, & \text{if } u \in S \setminus \{x\}. \end{cases}$$

  Hence (Exercise 4.1.2),

$$\begin{aligned} \operatorname{val}\mathbf{f} = \quad & \mathbf{f}^+(x) - \mathbf{f}^-(x) = \sum_{u \in S} (\mathbf{f}^+(u) - \mathbf{f}^-(u)) \\ = \quad & \mathbf{f}^+(S) - \mathbf{f}^-(S) \leq \mathbf{f}^+(S) \leqslant \mathbf{c}^+(S) = \operatorname{cap} B. \end{aligned}$$

- It remains to prove $\operatorname{val}\mathbf{f} \geq \operatorname{cap} B$.

- To this end, let D' be the graph obtained from D by adding, between the two end-vertices of each edge a, a new edge $\overleftarrow{a}$ with the opposite direction. Define $\widetilde{\mathbf{f}} \in \mathcal{E}(D')$ by

$$\widetilde{\mathbf{f}} = \begin{cases} \mathbf{c}(a) - \mathbf{f}(a), & \text{if } a \text{ is an old edge;} \\ \mathbf{f}(a), & \text{if } a \text{ is a new edge.} \end{cases}$$

- Note that $\tilde{\mathbf{f}}(a) + \tilde{\mathbf{f}}(\overleftarrow{a}) = \mathbf{c}(a)$.

- For example, the network in Figure 2 (a) is the graph D' obtained from the network in Figure 1 by the above procedure, where the curved directed edges denote the newly added edges $\overleftarrow{a}$. The number on an edge a is the value $\widetilde{\mathbf{f}}(a)$ of $\widetilde{\mathbf{f}}$ on a.

- Let $D^* = D'_{\widetilde{\mathbf{f}}}$, i.e. the support subgraph of $\widetilde{\mathbf{f}}$ in D' (its definition was given in Section 2.3). For example, the graph in Figure 2 (b) is the support graph D^* of $\widetilde{\mathbf{f}}$ in the graph D' of (a).

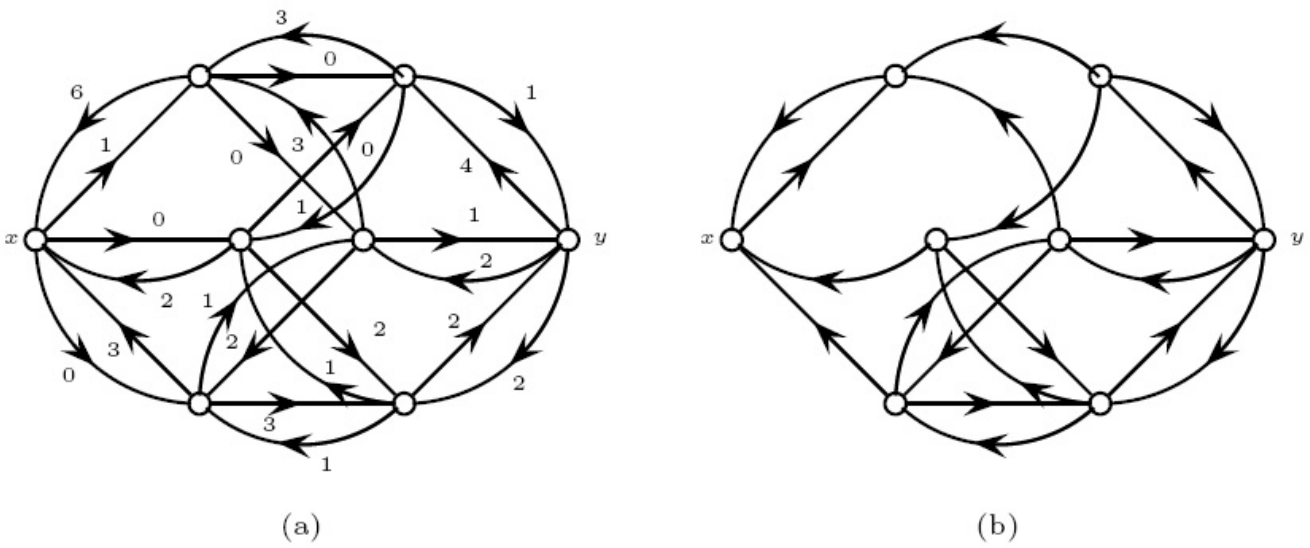

Figure 2: (a) D' and $\widetilde{\mathbf{f}}$; (b) $D^* = D'_{\widetilde{\mathbf{f}}}$

- Claim: D^* contains no (x,y)-path.
- Otherwise, let P be an (x,y)-path in D^*. Set

$$\sigma = \min\{\widetilde{\mathbf{f}}(a) : \ a \in E(P)\}.$$

Then $\sigma > 0$.

- Define $\mathbf{f}' \in \mathcal{E}(D)$ by

$$\mathbf{f}'(a) = \begin{cases} \mathbf{f}(a) + \sigma, & \text{if } a \in E(P); \\ \mathbf{f}(a) - \sigma, & \text{if } \overleftarrow{a} \in E(P); \\ \mathbf{f}(a), & \text{otherwise.} \end{cases}$$

- Then $\mathbf{f}'$ is an (x,y)-flow in N with val $\mathbf{f}' =$ val $\mathbf{f} + \sigma >$ val $\mathbf{f}$ (please prove it!).

- This contradicts the maximality of $\mathbf{f}$. Hence D^* contains no (x,y)-path.

- Set

$$S' = \{u \in V(D^*) : \ D^* \text{ contains an } (x,u)\text{-path}\}.$$

Then $x \in S'$, $y \notin S'$.

- Thus $B' = (S', \overline{S'})$ is an (x,y)-edge cut of D, and in D' we have

$$\widetilde{\mathbf{f}}(a) = \begin{cases} 0, & \text{if } a \in (S', \overline{S}'); \\ \mathbf{c}(a), & \text{if } a \in (\overline{S}', S'), \end{cases} \tag{4}$$

  so in D we have

$$\mathbf{f}(a) = \begin{cases} \mathbf{c}(a), & \text{if } a \in (S', \overline{S}'); \\ 0, & \text{if } a \in (\overline{S}', S'). \end{cases}$$

- Therefore, for the (x,y)-edge cut $B' = (S', \overline{S'})$ of D,

$$\operatorname{val}\mathbf{f} = \mathbf{f}^+(S') - \mathbf{f}^-(S') = \mathbf{f}^+(S') = \operatorname{cap}B' \geqslant \operatorname{cap}B.$$

  The theorem is proved. ∎

By Ford–Fulkerson's augmenting algorithm and MFMC Theorem, we have

Corollary 4.1 (Integer Max-Flow Min-Cut Theorem) Every integer capacity network admits an integer maximum flow whose value equals the minimum cut capacity. ∎

The Max-Flow Min-Cut Theorem is the foundation of the network-flow analytic method. It will be seen later that, by means of this theorem, one can not only find the maximum flow and minimum cut in a network, but also derive many well-known theorems in graph theory.

Exercises

4.1.3 Suppose that $N = (D_{xy}, \mathbf{c})$ is a capacity network and $B = (S, \overline{S})$ is an (x, y)-edge cut. Prove that for every (x, y)-flow $\mathbf{f}$ in N,

$$\mathbf{f}(B) - \mathbf{f}^{-}(S) = \mathbf{f}^{+}(S) - \mathbf{f}^{-}(S) = \begin{cases} \mathrm{val}\,\mathbf{f}, & \text{if } x \in S; \\ -\mathrm{val}\,\mathbf{f}, & \text{if } y \in S. \end{cases}$$

4.1.6 Let $N = (D_{xy}, \mathbf{c})$ be a capacity network, let $\mathbf{f}$ be an (x, y)-flow in N, and let $B = (S, \overline{S})$ be an (x, y)-edge cut. Prove that (a) if $\mathbf{f}(a) = \mathbf{c}(a)$ for every $a \in (S, \overline{S})$ and $\mathbf{f}(a) = 0$ for every $a \in (\overline{S}, S)$, then $\mathbf{f}$ is a maximum flow and B is a minimum cut; (b) if $\mathbf{f}$ is a maximum flow and B is a minimum cut, then $\mathbf{f}(a) = \mathbf{c}(a)$ for every $a \in (S, \overline{S})$ and $\mathbf{f}(a) = 0$ for every $a \in (\overline{S}, S)$.

2 Menger's Theorem

- Two paths P and Q in D are called internally vertex-disjoint if they have no common internal vertex; they are called edge-disjoint if they have no common edge.
- Let

$$\zeta_D(x,y) = \max\{\text{number of internally vertex-disjoint } (x,y)\text{-paths}\},$$

$$\eta_D(x,y) = \max\{\text{number of edge-disjoint } (x,y)\text{-paths}\}.$$

- For example, in the graph D of Figure 3 (b), $\zeta_D(x,y) = 2$, $\eta_D(x,y) = 3$.

- Let D be a digraph and let x, y be distinct vertices of D. An (x,y)-edge cut is a set $B \subseteq E(D)$ such that $D - B$ contains no directed (x,y)-path. Define

$$\lambda_D(x,y) = \min\{|B| : B \text{ is an } (x,y)\text{-edge cut of } D\}.$$

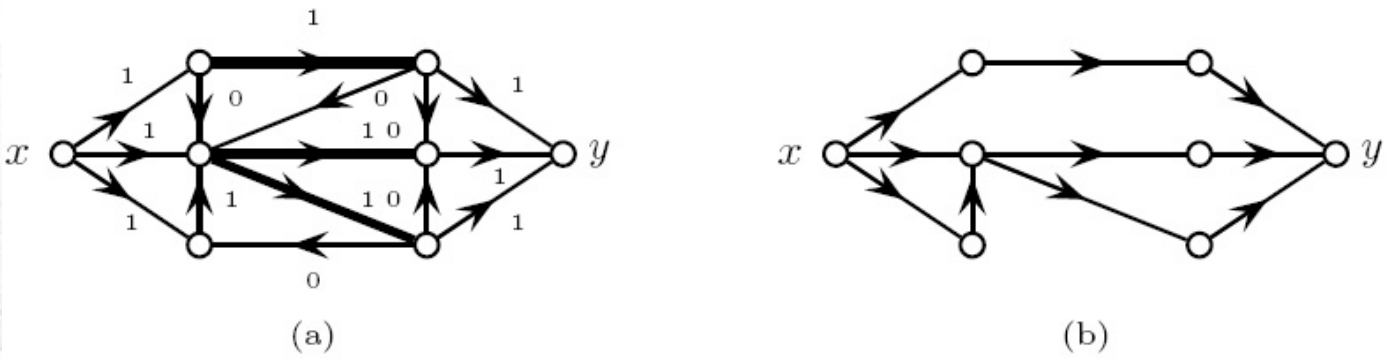

Figure 3: (a) a maximum flow $\mathbf{f}$ and a minimum cut B; (b) $H = D_{\mathbf{f}}$

The following theorem is one of the most famous theorems in graph theory, and it is the basic theorem of the network-flow theory. It was discovered by K. Menger (1927).

Theorem 4.2.1 (Menger's Theorem (edge form)) Let D be a digraph and x, y be two distinct vertices of D. Then

$$\eta_D(x,y) = \lambda_D(x,y).$$

That is, the maximum number of edge-disjoint (x,y)-paths equals the minimum number of edges in an (x,y)-edge cut.

Proof

- It is clear that
$$\eta_D(x,y) \leqslant \lambda_D(x,y).$$
For if B is an (x,y)-edge cut, then every (x,y)-path must contain an edge of B, so the number of edge-disjoint (x,y)-paths is at most $|B|$.

- To prove $\eta_D(x,y) \geqslant \lambda_D(x,y)$, let $l = \lambda_D(x,y)$. If $l = 0$, the conclusion is immediate.

- Regard D as a network with source x, sink y, and capacity one on every edge. The minimum cut capacity is l.

- By the max-flow min-cut theorem and the integrality theorem, there exists an integer flow $\mathbf{f}$ of value l. Since every edge has capacity one, $\mathbf{f}(e) \in \{0, 1\}$ for every $e \in E(D)$.

- For any flow $\mathbf{g}$, let $D_{\mathbf{g}}$ be its support digraph, with vertex set $V(D)$ and edge set

$$E(D_{\mathbf{g}}) = \{e \in E(D) : \mathbf{g}(e) > 0\}.$$

  We claim that whenever vol $\mathbf{g} > 0$, $D_{\mathbf{g}}$ contains a directed (x, y)-path.

- Indeed, let S be the set of vertices reachable from x in $D_{\mathbf{g}}$. If $y \notin S$, no edge leaving S carries positive flow. Thus the net flow out of S is nonpositive. On the other hand, by flow conservation, it equals the positive value of $\mathbf{g}$, a contradiction. Hence $y \in S$, proving the claim.

- Starting with $\mathbf{g} = \mathbf{f}$, choose a directed (x,y)-path in $D_{\mathbf{g}}$ and subtract one unit of flow along it. The resulting flow is again a nonnegative integer flow, and its value decreases by one. Moreover, every edge of the chosen path now has flow zero, so none of these edges can occur in any subsequently chosen path.

- Repeating this procedure yields l pairwise edge-disjoint directed (x,y)-paths. Therefore

$$\eta_D(x,y) \geqslant l = \lambda_D(x,y).$$

Together with the reverse inequality, this proves the theorem. ∎

Corollary 4.2.1.1 Let G be an undirected graph and x, y be two distinct vertices of G. Then

$$\eta_G(x,y) = \lambda_G(x,y).$$

That is, the maximum number of edge-disjoint xy-paths equals the minimum number of edges in an xy-edge cut. ∎

Corollary 4.2.1.2 (L. Lovász, 1973) A connected graph G is Eulerian if and only if for every partition

$$V(G) = V_1 \cup V_2, \quad V_1 \cap V_2 = \emptyset,$$

one has

$$\lambda_G(V_1, V_2) \equiv 0 \pmod 2 \text{ and } \lambda_G(V_1, V_2) \geq 2.$$

Proof

- ($\Rightarrow$) If G is Eulerian, then every vertex has even degree. For any partition $V = V_1 \cup V_2$, counting degrees inside V_1 gives

$$\lambda_G(V_1, V_2) = \sum_{v \in V_1} \deg_G(v) - 2e(G[V_1]) \equiv 0 \pmod 2,$$

and $\lambda_G(V_1, V_2) \geqslant 2$ because G is connected.

- ($\Leftarrow$) Suppose every cut $\lambda_G(V_1, V_2)$ is even. For each vertex v, take the partition $V_1 = \{v\}$; then

$$\deg_G(v) = \lambda_G(\{v\}, V \setminus \{v\})$$

is even. Thus every vertex of G has even degree, and since G is connected, G is Eulerian.

∎

Menger's Theorem (vertex form)

- Let x and y be two distinct vertices of D with no edge from x to y. A vertex set

$$S \subseteq V(D) \setminus \{x, y\}$$

is called an (x,y) separating set if $D - S$ contains no (x,y)-path.

- The parameter

$$\kappa_D(x,y) = \min\{\, |S| : S \text{ is an } (x,y) \text{ separating set}\}$$

is called the local vertex-connectivity from x to y.

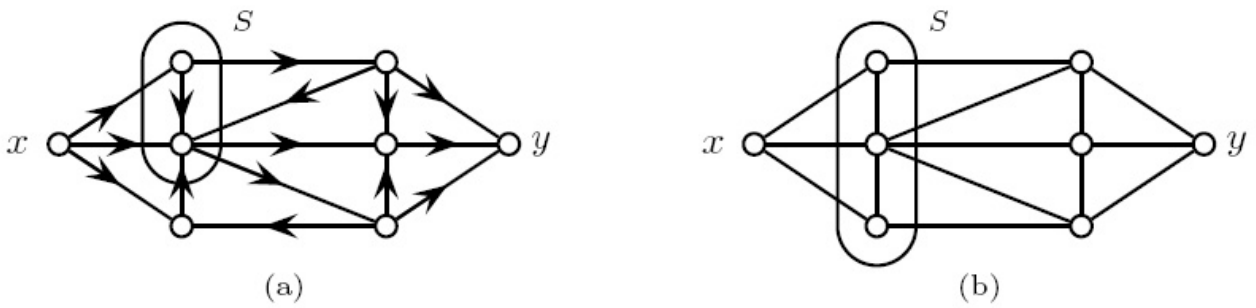

Figure 4: (a) a digraph D and an (x, y) separating set S; (b) an undirected graph G and and xy separating set S

- The following operation is useful. A vertex u of D is split into two vertices u' and u'' by replacing u with a new vertex u' and a new vertex u'', joining u' to u'' by a new edge, replacing every edge entering u with an edge entering u', and replacing every edge leaving u with an edge leaving u'' (see Figure 5).

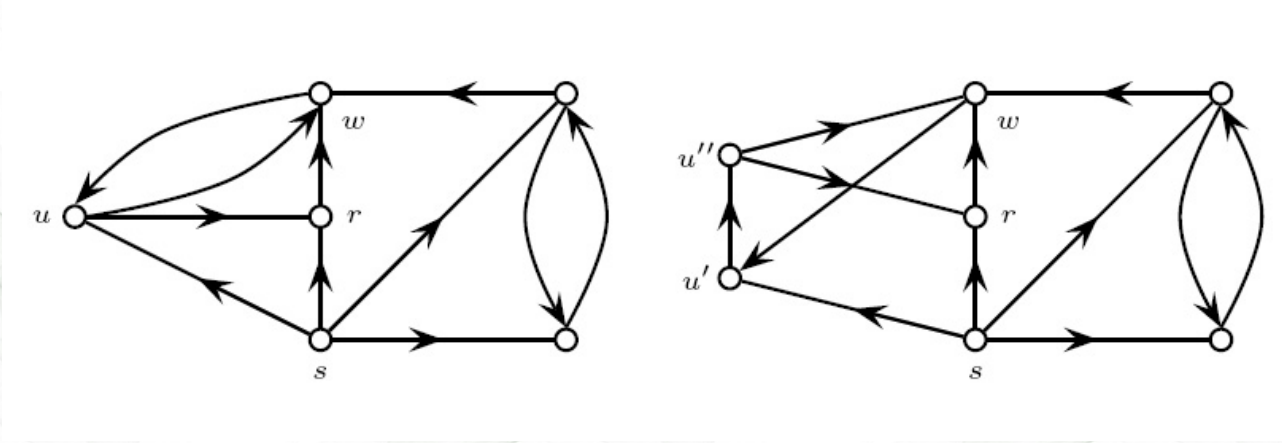

Figure 5: the split operation on a vertex u

- For example, Figure 6 shows the graph H obtained by splitting the vertices of the digraph D in (a).

Theorem 4.2.2 (Menger's Theorem (vertex form), K. Menger, 1927) Let D be a digraph and x, y be two distinct vertices of D with no edge

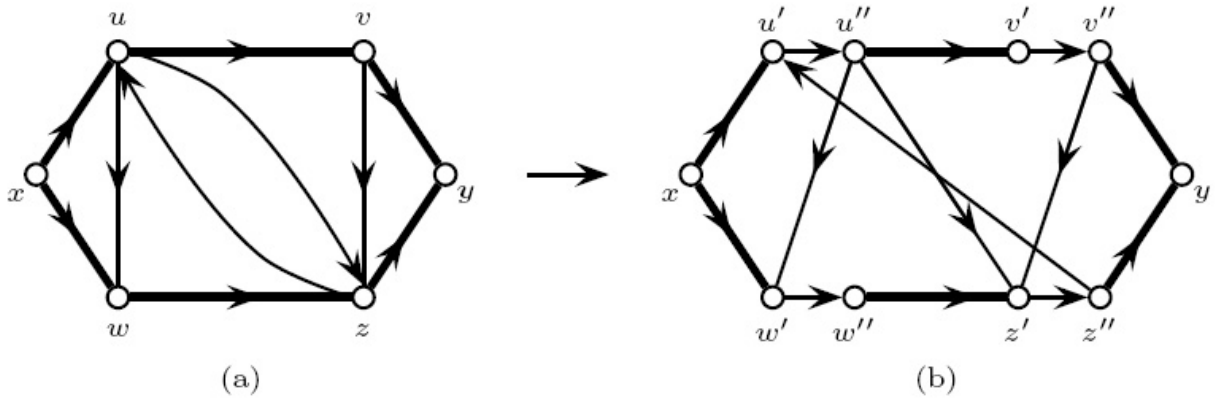

Figure 6: (a) a digraph D; (b) the split graph H

from x to y in D. Then

$$\zeta_D(x,y) = \kappa_D(x,y).$$

That is, the maximum number of internally vertex-disjoint (x,y)-paths equals the minimum number of vertices in an (x,y) separating set.

Proof

- It is clear that

$$\zeta_D(x,y) \leqslant \kappa_D(x,y).$$

  For if S is an (x,y) separating set, then every (x,y)-path contains a vertex of S, so the number of internally vertex-disjoint (x,y)-paths is at most $|S|$.

- To prove the reverse inequality, split every vertex u (other than x and y) into u' and u'' as above, and regard the resulting graph H as a capacity network in which the edge (u',u'') and every other edge has unit capacity.

- An (x,y)-path in the split graph H corresponds to an (x,y)-path in D, and two such paths in H are edge-disjoint if and only if the corresponding paths in D are internally vertex-disjoint: any path passing through u must use the unique capacity-1 edge (u',u''), so two paths share u exactly when they share that edge. Hence

$$\eta_H(x,y) = \zeta_D(x,y).$$

- By Theorem 4.2.1 (the edge form of Menger's theorem), the maximum number of edge-disjoint (x,y)-paths in the split graph H equals the minimum (x,y) cut capacity, that is,

$$\eta_H(x,y) = \lambda_H(x,y).$$

- It remains to show $\lambda_H(x,y) \geqslant \kappa_D(x,y)$. Let B be a minimum (x,y)-cut of H, so that $|B| = \lambda_H(x,y)$. Write B as the set of edges from V' to V'' for some partition $V(H) = V' \cup V''$ with $x \in V'$ and $y \in V''$ (this is without loss of generality: taking V' to be the set of vertices reachable from x in $H - B$ gives such a partition with $E_H(V', V'') = B$).

- Let S be the set of tails (initial vertices) of the edges of B. Then $|S| \leqslant |B|$, and $H - S$ contains no (x,y)-path: every (x,y)-path must use an edge of B, whose tail has been deleted. Let

$$S_D = \{u \in V(D) \setminus \{x,y\} : u' \in S \text{ or } u'' \in S\}.$$

Then $|S_D| \leqslant |S|$, and $D - S_D$ contains no (x,y)-path: otherwise the corresponding (x,y)-path in H would avoid every vertex of S, contradicting that $H - S$ has no (x,y)-path. Thus S_D is an (x,y) separating set of D, and

$$\kappa_D(x,y) \leqslant |S_D| \leqslant |S| \leqslant |B| = \lambda_H(x,y).$$

- Combining the steps above,

$$\zeta_D(x,y) = \eta_H(x,y) = \lambda_H(x,y) \geqslant \kappa_D(x,y),$$

and together with $\zeta_D(x,y) \leqslant \kappa_D(x,y)$ from the first step this gives $\zeta_D(x,y) = \kappa_D(x,y)$. The theorem is proved. ∎

Corollary 4.2.2 Let G be an undirected graph and x, y be two distinct vertices of G with $xy \notin E(G)$. Then

$$\zeta_G(x,y) = \kappa_G(x,y).$$

That is, the maximum number of internally vertex-disjoint xy-paths equals the minimum number of vertices in an xy-separating set. ∎

Exercises

4.2.1 Prove that for any two distinct nonadjacent vertices x and y of a connected graph G,

$$\kappa_G(x,y) \leqslant \lambda_G(x,y) \leqslant \min\{d_G(x), d_G(y)\}.$$

4.2.4 Prove that a connected graph G is a block if and only if any two vertices of G lie on a common cycle. Here a block of G is a maximal subgraph of G containing no cut-vertex.

3 Connectivity

- In the following, we define the total connectivity of a graph. First we consider the directed case.
- Let D be a nontrivial digraph. A vertex set $S \subseteq V(D)$ is called a separating set of D if $D - S$ is not strongly connected (or has only one vertex). The parameter

$$\kappa(D) = \min\{\, |S| : S \text{ is a separating set of } D\}$$

  is called the strong connectivity of D.
- If $\kappa(D) \geq k$, then D is called a k-strongly connected graph. For example, a 1-strongly connected graph is a strongly connected graph.
- An edge set $B \subseteq E(D)$ is called an edge-cut of D if $D - B$ is not strongly connected. The parameter

$$\lambda(D) = \min\{\, |B| : B \text{ is an edge-cut of } D\}$$

is called the strong edge-connectivity of D.

- If $\lambda(D) \geq k$, then D is called a k-strongly edge-connected graph.
- An edge-cut of size $\lambda(D)$ is called a λ-edge-cut.
- One readily sees that

$$\lambda(D) = \min\{\lambda_D(x,y) : \ \forall\ x,y \in V(D)\}.$$

- The connectivity and edge-connectivity of an undirected graph can be defined analogously.
- For the complete graph K_n on n vertices,

$$\kappa(K_n) = \lambda(K_n) = n-1;$$

- for the cycle C_n on n $(\geqslant 3)$ vertices,

$$\kappa(C_n) = \lambda(C_n) = 2;$$

- for any tree T on $n \geqslant 2$ vertices,

$$\kappa(T) = \lambda(T) = 1.$$

- Let G be an undirected graph and D be the symmetric digraph of G. It is easy to see that

$$S \text{ is a separating set of } G \Leftrightarrow S \text{ is a separating set of } D;$$

$$B = [S, \overline{S}] \text{ is an edge-cut of } G \Leftrightarrow (S, \overline{S}) \text{ is an edge-cut of } D.$$

  Therefore,

$$\kappa(G) = \kappa(D), \qquad \lambda(G) = \lambda(D).$$

- In this section we only discuss digraphs; the stated results concerning κ and λ naturally hold for undirected graphs, and will not be repeated.

Let $\delta(D) = \min\{\delta^+(D), \delta^-(D)\}$. The parameters $\kappa(D)$, $\lambda(D)$ and $\delta(D)$ satisfy the following relation. Its undirected form was first discovered by H. Whitney (1932), so in the literature it is usually called **Whitney's inequality**.

Theorem 4.3.1 [Whitney's inequality]

$$\kappa(D) \leqslant \lambda(D) \leqslant \delta(D).$$

Proof

- If $K_n^* \subseteq D$, the conclusion is immediate.
- We may assume that D is a nontrivial, strongly connected, loopless digraph, and choose $x \in V(D)$ with $d_D^+(x) = \delta(D)$.

- Since $E_D^+(x)$ is an edge-cut of D, we have

$$\lambda(D) \leqslant \delta(D).$$

- We now prove $\kappa(D) \leqslant \lambda(D)$.
- If $\lambda(D) \geq n(D) - 1$, the conclusion is immediate.
- Assume $\lambda(D) < n(D) - 1$.
- Choose an edge-cut $B = (S, \overline{S})$ of D with $|B| = \lambda(D)$.
- Let $C \subseteq S$ be the set of tails of the edges in B; then $|C| \leq |B|$.
- If there exists $v \in S$ with $E_D^+(v) \cap B = \emptyset$, then C separates v from $\overline{S}$. Hence

$$\kappa(D) \leq |C| \leq |B| = \lambda(D).$$

- Otherwise, take any $v \in C$ $(= S)$.
- Claim: $|N^+(v)| \leq |B|$.

- For each $w \in N^+(v)$: if $(v,w) \in B$, then w corresponds to an edge of B; if $(v,w) \notin B$, then $E^+(w) \cap B \neq \emptyset$, so again an edge of B corresponds to w. This correspondence is clearly injective. The claim follows.
- From $|N^+(v)| \leq |B| < n(D) - 1$ we have $V - (N^+(v) \cup \{v\}) \neq \emptyset$, i.e. $N^+(v)$ separates v from $V - (N^+(v) \cup \{v\})$.
- Hence

$$\kappa(D) \leq |N^+(v)| \leq \lambda(D).$$

∎

Corollary 4.3.1.1 For an undirected graph G,

$$\kappa(G) \leqslant \lambda(G) \leqslant \delta(G).$$

∎

Corollary 4.3.1.2 For every simple planar graph G,

$$\kappa(G) \leqslant \lambda(G) \leqslant 5.$$

∎

- The graph G shown in Figure 7 has $\kappa(G) = 2$, $\lambda(G) = 3$ and $\delta(G) = 4$.

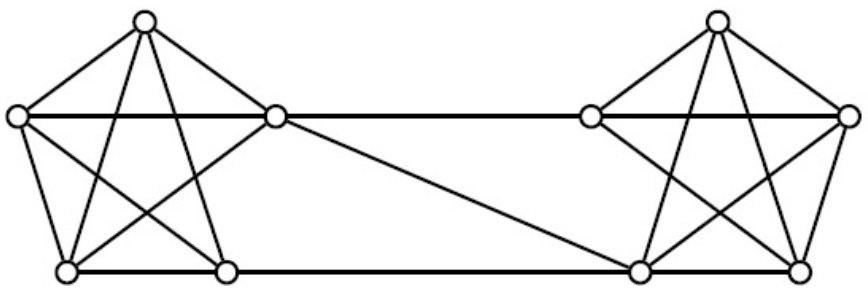

Figure 7: $\kappa(G) = 2, \lambda(G) = 3$ and $\delta(G) = 4$

- In fact, for any three positive integers κ, λ and δ ($\kappa \leqslant \lambda \leqslant \delta$), there exists an undirected graph G with $\kappa(G) = \kappa$, $\lambda(G) = \lambda$ and $\delta(G) = \delta$ (Exercise 4.3.17).

- Note that, by considering the symmetric digraph D of G, there also exists a digraph satisfying the same conditions.

- For the complete graph K_n, the cycle C_n ($n \geqslant 3$) and a tree T,

$$\begin{aligned}
&\kappa(K_n) = \lambda(K_n) = \delta(K_n) = n - 1,\\
&\kappa(C_n) = \lambda(C_n) = \delta(C_n) = 2,\\
&\kappa(T) = \lambda(T) = \delta(T) = 1.
\end{aligned}$$

For a general undirected or directed graph, it is of interest to find conditions under which $\kappa = \delta$ or $\lambda = \delta$. The following example gives a sufficient condition for $\lambda = \delta$ (J. L. Jolivet (1972); its undirected form was obtained by J. Plesnik (1975)).

Proposition 3.1 *Let D be a strongly connected simple graph with diameter $d(D) \leqslant 2$. Then $\lambda(D) = \delta(D)$.*

Proof

- If $d(D) = 0$, then D is the trivial graph, and $\lambda(D) = 0 = \delta(D)$.
- If $d(D) = 1$, then $D \cong K_n^*$, so $\lambda(D) = n - 1 = \delta(D)$.
- Assume $d(D) = 2$. By Whitney's inequality (Theorem 4.3.1), it suffices to prove $\lambda(D) \geqslant \delta(D)$.
- Let B be an edge-cut of D with $|B| = \lambda(D) = \lambda$. There is a nonempty proper subset S of $V(D)$ with $(S, \overline{S}) = B$ (Exercise 4.3.2 (a)).
- Let X be the set of tails of B in S, and Y be the set of heads of B in $\overline{S}$.
- Claim: $|S \setminus X| \cdot |\overline{S} \setminus Y| = 0$.

- Otherwise, choosing $x \in S \setminus X$ and $y \in \overline{S} \setminus Y$ would give $d_D(x,y) > 2$, contradicting $d(D) = 2$.
- Without loss of generality, $\overline{S} \setminus Y = \emptyset$, and let $Y = \{y_1, y_2, \cdots, y_b\}$.
- Set

$$s_i = |N_D^-(y_i) \cap X|, \quad i = 1, 2, \cdots, b.$$

Since D is simple,

$$\sum_{i=1}^{b} s_i = |B| = \lambda(D) \geqslant b \geqslant 1.$$

- Since $\delta = \delta(D) = \min\{\delta^+(D), \delta^-(D)\}$,

$$\delta \leqslant d_D^-(y_i) \leqslant s_i + (b-1).$$

Hence

$$b\delta \leqslant \sum_{i=1}^{b} s_i + b(b-1) = \lambda + b(b-1),$$

that is (noting $\delta \geqslant \lambda \geqslant b \geqslant 1$),

$$\lambda \geqslant b\delta - b(b-1) = \delta + (\delta - b)(b-1) \geqslant \delta.$$

The conclusion follows. ∎

The following example gives a sufficient condition for a graph G to be k-connected; it was obtained independently by J. A. Bondy (1969) and F. T. Boesch (1974).

Proposition 3.2 *Let G be a simple undirected graph with $V(G) = \{x_1, x_2, \cdots, x_n\}$, let $d_i = d_G(x_i)$ with $d_1 \leqslant d_2 \leqslant \cdots \leqslant d_n$, and let $n \geqslant 2$, $1 \leqslant k \leqslant n-1$. If there exists some $i \leq \lfloor \frac{1}{2}(n-k+1) \rfloor$ such that*

$$d_i \leqslant k+i-2 \implies d_{n-k+1} \geqslant n-i,$$

then $\kappa(G) \geqslant k$.

Proof (by contradiction)

- Suppose $\kappa(G) < k$.
- Since $1 \leqslant k \leqslant n-1$, G has a separating set S with

$$\kappa(G) \leqslant |S| = k-1.$$

- Let H be a connected component of $G-S$ of smallest order, and write $n(H) = i$. Then

$$i \leqslant \lfloor \frac{n-k+1}{2} \rfloor,$$

  i.e. $k \leqslant n-2i+1$.

- Hence for every $x \in V(H)$,

$$d_G(x) \leqslant n(H) - 1 + |S| = i+k-2 \ (\leqslant n-i-1).$$

- It follows that $d_i \leq i+k-2$.
- Since $i = n(H) \leq \lfloor \frac{1}{2}(n-k+1) \rfloor$ and $d_i \leqslant k+i-2$, the hypothesis gives $d_{n-k+1} \geqslant n-i$.
- For every $x_j \in V(G) \setminus (V(H) \cup S)$, one has $d_j \leqslant n-i-1$, so the vertices x_j with $d_j \geqslant n-i$ all lie in S.
- Moreover,
$$d_n \geqslant d_{n-1} \geqslant \cdots \geqslant d_{n-k+1} \geqslant n-i,$$
so
$$|S| \geqslant n-(n-k+1)+1 = k,$$
contradicting $|S| = k-1$.
- Therefore $\kappa(G) \geqslant k$. ∎

Note: Replacing G by D and setting $d_i = \min\{d_D^+(x_i), d_D^-(x_i)\}$, the condition in the example above becomes a sufficient condition for D to be k-strongly connected; its proof requires only minor modifications.

The following theorem gives a necessary and sufficient condition for a graph to be k-strongly connected or k-strongly edge-connected. Its undirected form was first given by H. Whitney (1932), so it is sometimes called Whitney's k-connectivity criterion.

Theorem 4.3.2 (H. Whitney, 1932) Let $k \geqslant 1$ and let D be a digraph on n $(\geqslant k+1)$ vertices. Then

(i) $\kappa(D) \geqslant k \Leftrightarrow \zeta_D(x,y) \geqslant k,\ \forall\ x,y \in V(D)$ (i.e. between any two vertices there are at least k internally vertex-disjoint directed paths);

(ii) $\lambda(D) \geqslant k \Leftrightarrow \eta_D(x,y) \geqslant k,\ \forall\ x,y \in V(D)$ (i.e. between any two vertices there are at least k edge-disjoint directed paths).

Proof (i) For $k = 1$, the conclusion follows immediately from the definitions of a strongly connected graph and ζ_D. Assume $k \geqslant 2$.

$(\Rightarrow)$

- Let x and y be two distinct vertices of the k-strongly connected graph D.
- If $E_D(x,y) = \emptyset$, then by Menger's theorem (4.2),

$$k \leqslant \kappa(D) \leqslant \kappa_D(x,y) = \zeta_D(x,y).$$

- Assume $E_D(x,y) \neq \emptyset$, and let $\mu = |E_D(x,y)|$. If $\mu \geqslant k$, then

$$\zeta_D(x,y) \geqslant |E_D(x,y)| = \mu \geqslant k.$$

- Assume $\mu < k$, and let $D' = D - E_D(x,y)$. We show that still $\zeta_D(x,y) \geqslant k$.

- (by contradiction) If $\zeta_D(x,y) < k$, then

$$1 \leqslant \zeta_{D'}(x,y) < k - \mu.$$

- Hence by Menger's theorem (4.2), D' contains an (x,y) separating set $S \subseteq V \setminus \{x,y\}$ with

$$|S| = \zeta_{D'}(x,y) \leqslant k - \mu - 1.$$

- Then

$$|V| - |S| \geqslant k + 1 - (k - \mu - 1) = \mu + 2,$$

so there exists $z \in (V \setminus \{x,y\}) \setminus S$.

- Claim: $\zeta_{D'-S}(x,z) \geqslant 1$; likewise $\zeta_{D'-S}(z,y) \geqslant 1$.
- If $E_{D'}(x,z) \neq \emptyset$, then $\zeta_{D'-S}(x,z) \geqslant 1$.
- If $E_{D'}(x,z) = \emptyset$, then by Menger's theorem (4.2),

$$\zeta_D(x,z) = \kappa_D(x,z) \geq k,$$

so $\zeta_{D'}(x,z) \geqslant k-\mu$.

- Since $|S| \leqslant k-\mu-1$, one still has $\zeta_{D'-S}(x,z) \geqslant 1$.
- Likewise $\zeta_{D'-S}(z,y) \geqslant 1$. Therefore $\zeta_{D'-S}(x,y) \geqslant 1$, contradicting the assumption that S is an (x,y) separating set in D'.
- Hence, when $E_D(x,y) \neq \emptyset$, one still has $\zeta_D(x,y) \geqslant k$.

($\Leftarrow$)

- Since $k \geqslant 1$, the assumption implies that D is strongly connected.
- If D contains the spanning subgraph K_n^*, then $\kappa(D) = n-1 \geqslant k$.
- If D does not contain K_n^*, let S be a κ-separating set of D.
- Then $D-S$ is not strongly connected, so there exist $x,y \in V(D)$ with no (x,y)-path in $D-S$, i.e. S is an (x,y) separating set in D. Hence $|S| \geqslant \kappa_D(x,y)$.

- By the assumption $\zeta_D(x,y) \geqslant k$, Menger's theorem (4.2) gives

$$\kappa(D) = |S| \geqslant \kappa_D(x,y) = \zeta_D(x,y) \geqslant k.$$

- The proof of (ii) is left to the reader (Exercise 4.3.1). ∎

Corollary 4.3.2 Let D be a strongly connected digraph. Then

(i) $\kappa(D) \geqslant k \Rightarrow \begin{cases} \kappa(D-x) \geqslant k-1, & \forall\ x \in V(D), \\ \kappa(D-a) \geqslant k-1, & \forall\ a \in E(D); \end{cases}$

(ii) $\lambda(D) \geqslant k \Rightarrow \lambda(D-a) \geqslant k-1,\ \forall\ a \in E(D)$. ∎

Exercises

4.3.5 (a) Prove that if G is a simple graph with $\delta \geqslant \lfloor \frac{|V(G)|}{2} \rfloor$, then $\lambda(G) = \delta(G)$.

(b) Give an example of a simple graph G with $\delta(G) = \lfloor \frac{|V(G)|}{2} \rfloor - 1$ and $\lambda(G) < \delta(G)$.

4.3.10 Let G be a 3-connected graph with $|V(G)| \geqslant 5$. Prove that there exists $e \in E(G)$ such that $\kappa(G \cdot e) \geqslant 3$. (W. T. Tutte, 1961)

4.3.11 Let $\kappa(D) \geqslant k$, and let $x_0, x_1, \cdots, x_k$ be any $k+1$ vertices of D. Prove that D contains k internally vertex-disjoint (x_0, x_i)-paths and k internally vertex-disjoint (x_i, x_0)-paths, $i = 1, 2, \cdots, k$.

4 Design of Transport Plans – Maximum-Flow Algorithm for Capacity Networks

- The routes along which goods are transported from the places of production to the markets form a transportation system. Suppose the transport capacity of each segment of the system is given. Design a transport plan that moves the goods from the places of production to the markets with the maximum throughput. If the throughput is to be increased further, on which segments should the transport capacity be increased?
- Regard this transportation system as a capacity network $N = (D_{x,y}, \mathbf{c})$, where D is the simple connected graph formed by the transportation system, the source x and sink y are the places of production and the markets, respectively, and the capacity function $\mathbf{c} \in \mathcal{E}(D)$ is the transport capacity. The problem then amounts to finding a maximum (x, y)-flow and a minimum-capacity (x, y)-edge cut in N. This section presents an efficient algorithm for this

problem.

- Let $\mathbf{f} \in \mathcal{E}(D)$ be an (x,y)-flow in $N = (D_{xy}, \mathbf{c})$, let u be a vertex of D with $u \neq x$, and let P be a path in D joining x and u (not necessarily directed).

- Give P the positive direction from x to u, and denote by P^+ and P^- the sets of edges in $E(P)$ that agree with and are opposite to the positive direction of P, respectively. Set

$$\sigma(a) = \begin{cases} \mathbf{c}(a) - \mathbf{f}(a), & \text{if } a \in P^+, \\ \mathbf{f}(a), & \text{if } a \in P^-. \end{cases}$$

and set

$$\sigma_P(u) = \min\{\sigma(a) : \ a \in E(P)\}.$$

- If $\sigma_P(u) = 0$, then P is called an **f**-saturated path.

- If $\sigma_P(u) > 0$, then P is called an **f**-unsaturated path.

- An (x,y)-path P that is $\mathbf{f}$-unsaturated is called an $\mathbf{f}$-incrementing path.

- It is called an incrementing path because the value of the flow $\mathbf{f}$ can be increased along this path. Indeed, the $\widetilde{\mathbf{f}} \in \mathcal{E}(D)$ defined by

$$\tilde{\mathbf{f}}(a) = \begin{cases} \mathbf{f}(a) + \sigma_P(y), & \text{if } a \in P^+, \\ \mathbf{f}(a) - \sigma_P(y), & \text{if } a \in P^-, \\ \mathbf{f}(a), & \text{otherwise} \end{cases} \tag{5}$$

  is an (x,y)-flow in N with $\operatorname{val} \widetilde{\mathbf{f}} = \operatorname{val} \mathbf{f} + \sigma_P(y)$ (Exercise 4.4.1). The flow $\widetilde{\mathbf{f}}$ is called the revised flow based on the $\mathbf{f}$-incrementing path P.

Example 4.4.1 Consider the network $N = (D_{xy}, \mathbf{c})$ shown in Figure 8 (a); the ordered pairs on the edges are the values of the flow $\mathbf{f}$ and the capacity $\mathbf{c}$, respectively, with $\operatorname{val} \mathbf{f} = 6$. The path $P = x(s,x)s(w,s)w(w,y)y$ is a path

in D joining x and y (shown in bold). Give P the positive direction from x to y. Then P is an $\mathbf{f}$-unsaturated xy-path, hence an $\mathbf{f}$-incrementing path. The revised flow $\widetilde{\mathbf{f}}$ defined by (5) is shown in Figure 8 (b), with val $\widetilde{\mathbf{f}} =$ val $\mathbf{f} + 2 = 8$.

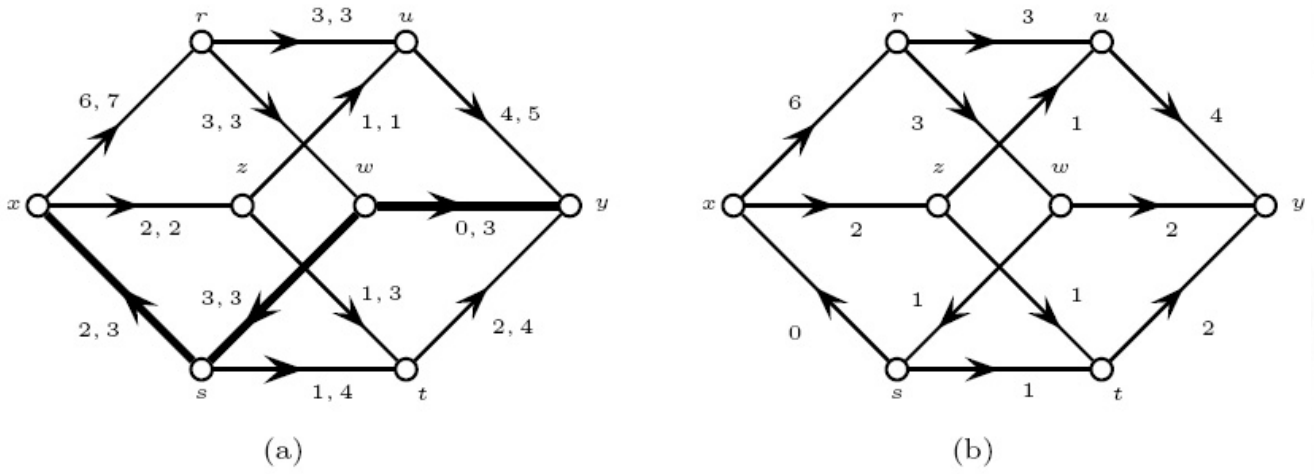

Figure 8: (a) $N = (D_{xy}, \mathbf{c})$ and the flow $\mathbf{f}$; (b) the revised flow $\widetilde{\mathbf{f}}$

Theorem 4.4.1 (L. R. Ford, Jr. & D. R. Fulkerson, 1957) An (x,y)-flow $\mathbf{f}$ in $N = (D_{xy}, \mathbf{c})$ is maximum if and only if N contains no $\mathbf{f}$-incrementing path.

Proof

($\Rightarrow$) This follows immediately from the definition and the properties of an incrementing path.

($\Leftarrow$)

- Suppose N contains no $\mathbf{f}$-incrementing path.
- Let

$$S = \{u \in V(D) : \ N \text{ contains an } \mathbf{f}\text{-unsaturated } xu\text{-path}\}.$$

  Then $x \in S$, $y \notin S$.

- Hence $B = (S, \overline{S})$ is an (x,y)-edge cut of D.

- Claim: for every $a \in (S, \overline{S})$, $\mathbf{f}(a) = \mathbf{c}(a)$; for every $a \in (\overline{S}, S)$, $\mathbf{f}(a) = 0$.
- Indeed, take any $a \in (S, \overline{S})$, say $a = (u, w)$. Then $u \in S$, $w \in \overline{S}$. Since $u \in S$, there is an $\mathbf{f}$-unsaturated xu-path Q.
- If $\mathbf{f}(a) < \mathbf{c}(a)$, then $Q + a$ is an $\mathbf{f}$-unsaturated xw-path in N, i.e. $w \in S$, contradicting $w \notin S$. Hence $\mathbf{f}(a) = \mathbf{c}(a)$.
- Likewise, if $a \in (\overline{S}, S)$, then $\mathbf{f}(a) = 0$. Consequently, val $\mathbf{f}$ = cap B.
- Let B^* be a minimum-capacity (x, y)-edge cut in N and $\mathbf{f}^*$ be a maximum (x, y)-flow in N. By the max-flow min-cut theorem (4.1),

$$\operatorname{val}\mathbf{f} \leqslant \operatorname{val}\mathbf{f}^* = \operatorname{cap}B^* \leqslant \operatorname{cap}B = \operatorname{val}\mathbf{f}.$$

- It follows that $\mathbf{f}$ is a maximum (x, y)-flow in N, and B is a minimum-capacity (x, y)-edge cut. ∎

Theorem 4.4.1 provides an efficient algorithm for finding a maximum flow and minimum cut in an integer capacity network. This algorithm is called the labelling method. It was first proposed by Ford and Fulkerson (1957) and slightly modified by J. Edmonds and R. M. Karp (1972).

The basic idea of the labelling method is to start from any known (x,y)-flow $\mathbf{f}$ in N (for example the zero flow), recursively construct a sequence of flows of increasing value, and terminate at a maximum flow. After each new flow $\mathbf{f}$ is obtained, if there is an $\mathbf{f}$-incrementing path P, form the revised flow $\widetilde{\mathbf{f}}$ based on P, then run the algorithm again with $\widetilde{\mathbf{f}}$ as the initial flow. If there is no $\mathbf{f}$-incrementing path, the algorithm stops; by Theorem 4.4.1 $\mathbf{f}$ is then a maximum flow.

Labelling method:

1. Choose any (x,y)-flow $\mathbf{f}$ in N (for example the zero flow), label x with $(-,\infty)$, and set $L=\{x\}$.

2. Remove the first element u from L (the "first-in-first-out" principle, which ensures that the (x,y)-incrementing path in step 4 is a shortest path in D).

   (1) If there is an unlabelled vertex z such that
   (i) $a=(u,z)\in E(D)$ and $\mathbf{f}(a)<\mathbf{c}(a)$, label z with $(u^+,\sigma(z))$;
   or
   (ii) $a=(z,u)\in E(D)$ and $\mathbf{f}(a)>0$, label z with $(u^-,\sigma(z))$,
   where $\sigma(z)=\sigma_P(z)$ for an $\mathbf{f}$-unsaturated xz-path P in D, and append z to the end of L.

   (2) If there is no such vertex z satisfying (i) or (ii) and $L=\emptyset$, stop; then $\mathbf{f}$ is a maximum flow.

3. If y is labelled, go to step 4; if y is not labelled, return to step 2.

4. The labelled vertices form an $\mathbf{f}$-incrementing path P in D. Construct the revised flow $\widetilde{\mathbf{f}}$ based on the incrementing path P of the flow $\mathbf{f}$. Remove all labels, replace $\mathbf{f}$ by $\widetilde{\mathbf{f}}$, and return to step 1.

Theorem 4.4.2 Let N be an integer capacity network, and let $\mathbf{f}$ and S be the (x,y)-flow and the set of labelled vertices obtained when the labelling method terminates. Then $\mathbf{f}$ is a maximum (x,y)-flow and $(S,\overline{S})$ is a minimum-capacity (x,y)-edge cut.

Proof

- By step 2 (2) of the algorithm, when it stops there is no $\mathbf{f}$-incrementing path in N. Hence, by Theorem 4.4.1, $\mathbf{f}$ is a maximum (x,y)-flow and $(S,\overline{S})$ is a minimum-capacity (x,y)-edge cut, where S is the set of labelled vertices.

- Since N is an integer capacity network, each execution of step 4 constructs a flow $\widetilde{\mathbf{f}}$ whose value is at least one larger than $\mathbf{f}$. For every (x,y)-flow $\widetilde{\mathbf{f}}$ in N,

$$\mathrm{val}\,\widetilde{\mathbf{f}} \leqslant \mathrm{cap}\,(E_D^+(x)),$$

where $E_D^+(x)$ is the set of edges leaving x in D; so step 4 is repeated at most cap $(E_D^+(x))$ times, and the algorithm terminates in finitely many steps. ∎

Some remarks:

- The "first-labelled-first-removed" principle in the labelling method is essential: it ensures that the incrementing path is a shortest xy-path in D, which guarantees the efficiency of the algorithm.

  Note: the complexity of the Ford-Fulkerson algorithm is $O((Val_{\max} f)|E|)$, and that of the Edmonds-Karp algorithm is $O(|V||E|^2)$.

- Consider the capacity network $N = (D_{xy}, \mathbf{c})$ shown in Figure 9. Clearly N has an (x,y)-flow $\mathbf{f}$ of value $2m$. Starting from the zero flow and alternately choosing $(x, x_1, x_2, x_5, x_6, y)$ and $(x, x_4, x_5, x_2, x_3, y)$ as incrementing paths, each revision increases the

flow by one unit, and $2m+1$ labelling procedures are required. Since m is arbitrary, the number of operations needed to run the labelling algorithm cannot be bounded above by a polynomial in $n(D)$ and $e(D)$. In other words, it is not an efficient algorithm.

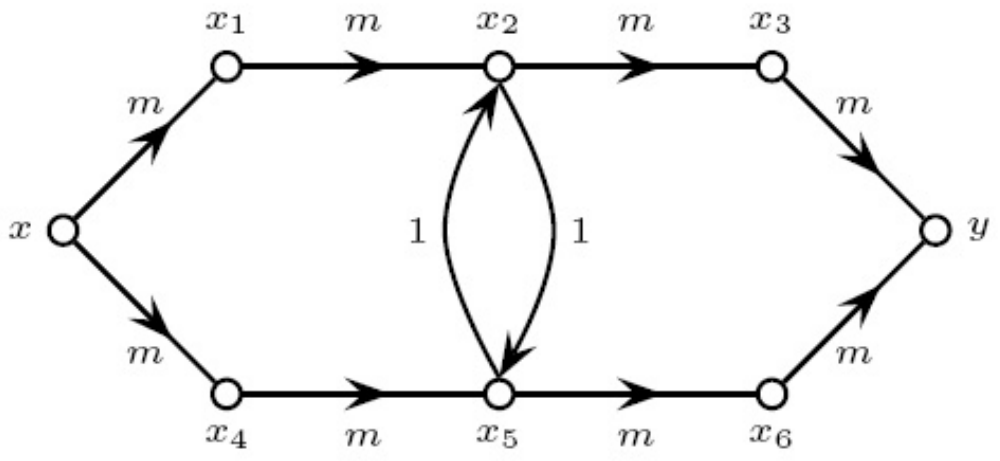

Figure 9: an application of the labelling method

- It is worth noting that for an irrational-capacity network, Ford and Fulkerson gave an example showing that the labelling method need not terminate in finitely many steps, and that the sequence of flows obtained during the computation does not converge to a maximum flow.

- When $\mathbf{c} \equiv 1$, the maximum flow $\mathbf{f}$ found by the labelling method satisfies $\lambda_D(x,y) = \text{val } \mathbf{f}$. In other words, the labelling method provides an efficient algorithm for finding the strong edge-connectivity of a graph.

Exercises

4.4.3 Use the labelling method to find a maximum (x,y)-flow in the capacity network shown in Figure 9.

5 Design of Optimal Transport Plans – Minimum-Cost Maximum-Flow Algorithm*

- In Section 4.4, when using the labelling method to design a transport plan, we considered only the flow value and took no account of the cost (or, equivalently, assumed that the cost per unit transported on every segment is the same).
- In practical problems the cost factor is important. Because the transport equipment differs from segment to segment, the cost per unit transported is not the same everywhere.
- We therefore seek a transport plan that maximizes the throughput while minimizing the total transport cost. Such a plan is called an optimal transport plan.
- We represent the transportation system by a so-called cost-capacity

network $N = (D_{xy}, \mathbf{b}, \mathbf{c})$, where $\mathbf{b} \in \mathcal{E}(D)$ is the **cost function** per unit of flow and $\mathbf{c} \in \mathcal{E}(D)$ is the **capacity function**.

- Figure 10 (a) shows a cost-capacity network $N = (D_{xy}, \mathbf{b}, \mathbf{c})$; the ordered pairs on an edge a are the values $\mathbf{b}(a)$ and $\mathbf{c}(a)$ of the cost function and the capacity function on a, respectively.

- Let $\mathbf{f}$ be an (x, y)-flow in N. Then

$$\mathbf{b}(\mathbf{f}) = \sum_{a \in E(D)} \mathbf{f}(a)\mathbf{b}(a)$$

  is defined as the cost (cost) of $\mathbf{f}$.

- If for every (x, y)-flow $\mathbf{f}'$ in N with value equal to val $\mathbf{f}$ one has

$$\mathbf{b}(\mathbf{f}) \leqslant \mathbf{b}(\mathbf{f}'),$$

  then $\mathbf{f}$ is called a minimum-cost flow.

- In the language of networks, the design of an optimal transport plan is to find, in the cost-capacity network $N = (D_{xy}, \mathbf{b}, \mathbf{c})$, a maximum (x, y)-flow $\mathbf{f}$ whose cost $\mathbf{b}(\mathbf{f})$ is minimum. Such a flow is called a minimum-cost maximum flow.

- Let $N = (D_{xy}, \mathbf{b}, \mathbf{c})$ be a cost-capacity network, let $\mathbf{f}$ be an (x, y)-flow in N, and let C be a cycle of D with a specified positive direction. Set

$$\sigma_{\mathbf{f}}(a) = \begin{cases} \mathbf{c}(a) - \mathbf{f}(a), & \text{if } a \in C^+; \\ \mathbf{f}(a), & \text{if } a \in C^-, \end{cases}$$
$$\sigma_{\mathbf{f}}(C) = \min\{\sigma_{\mathbf{f}}(a) : \ a \in E(C)\}.$$

- If the cycle C in D can be oriented so that $\sigma_{\mathbf{f}}(C) > 0$, then C is called an $\mathbf{f}$ increment cycle.

- For an $\mathbf{f}$ increment cycle C and any σ $(0 < \sigma \leqslant \sigma_{\mathbf{f}})$, we may define

$$\widetilde{\mathbf{f}}_{\sigma}(a) = \begin{cases} \mathbf{f}(a) + \sigma, & \text{if } a \in C^+; \\ \mathbf{f}(a) - \sigma, & \text{if } a \in C^-; \\ \mathbf{f}(a), & \text{otherwise.} \end{cases} \tag{6}$$

- It is easy to verify that $\widetilde{\mathbf{f}}_\sigma$ is an (x,y)-flow in N, and that val $\widetilde{\mathbf{f}} =$ val $\mathbf{f}$ (Exercise 4.5.1).
- The flow $\widetilde{\mathbf{f}}_\sigma$ is called the revised flow based on the $\mathbf{f}$ increment cycle C with respect to σ.

Example 4.5.1 Consider the network $N = (D_{xy}, \mathbf{b}, \mathbf{c})$ shown in Figure 10 (a). In Example 4.4.2 the maximum (x,y)-flow $\mathbf{f}$ was already found; it is shown in Figure 10 (b). The cycle $C = s(w,s)w(w,y)y(t,y)t(s,t)s$ is an $\mathbf{f}$ increment cycle, with positive direction clockwise. The revised flow $\widetilde{\mathbf{f}}$ defined by (6) is shown in Figure 10 (c), with val $\mathbf{f} = 8 =$ val $\widetilde{\mathbf{f}}$. A computation gives $\mathbf{b}(\mathbf{f}) = 62$, whereas $\mathbf{b}(\widetilde{\mathbf{f}}) = 56$. ∎

Let C be an $\mathbf{f}$ increment cycle. Define the cost of the increment cycle C by

$$\mathbf{b}(C,\mathbf{f}) = \sum_{a \in C^+} \mathbf{b}(a) - \sum_{a \in C^-} \mathbf{b}(a).$$

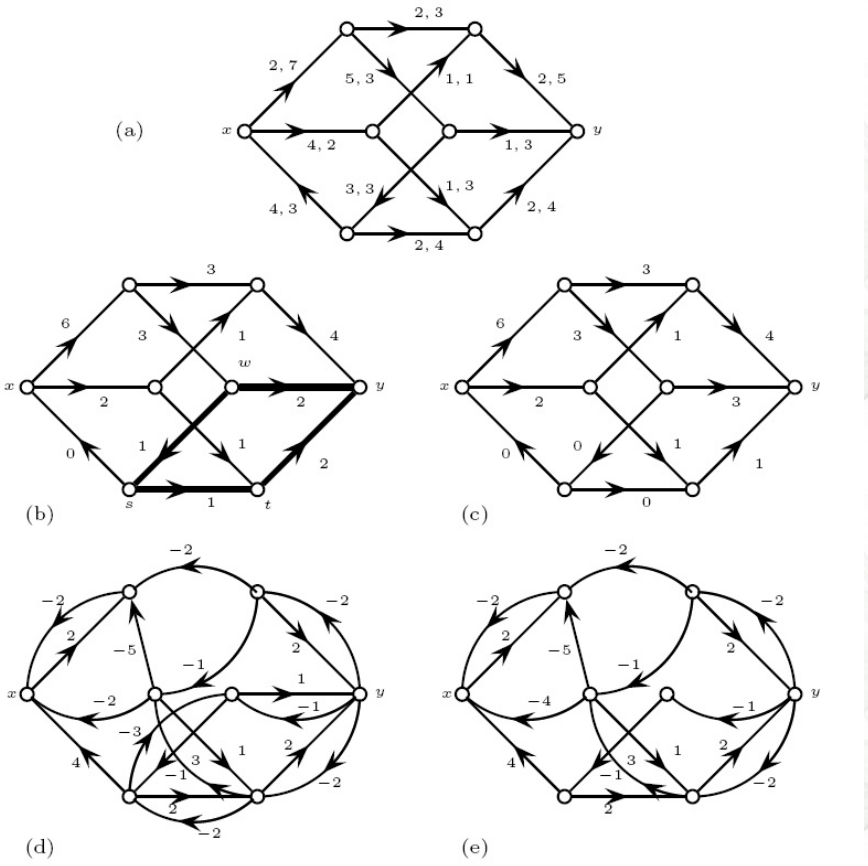

Figure 10: an application of Klein's algorithm

To find an $\mathbf{f}$ increment cycle of negative cost, we construct the auxiliary graph $D(\mathbf{f})$.

- Let $a \in E(D)$, and denote by $\overleftarrow{a}$ the new edge obtained by reversing the direction of a.
- Define the weighted graph $(D(\mathbf{f}), \mathbf{w})$ as follows: $V(D(\mathbf{f})) = V(D)$.

  For $a \in E(D)$:

  $\star$ if $\mathbf{f}(a) = 0$, then $a \in E(D(\mathbf{f}))$ and $\mathbf{w}(a) = \mathbf{b}(a)$;

  $\star$ if $\mathbf{f}(a) = \mathbf{c}(a)$, then $\overleftarrow{a} \in E(D(\mathbf{f}))$ and $\mathbf{w}(\overleftarrow{a}) = -\mathbf{b}(a)$;

  $\star$ if $0 < \mathbf{f}(a) < \mathbf{c}(a)$, then $a, \overleftarrow{a} \in E(D(\mathbf{f}))$, $\mathbf{w}(a) = \mathbf{b}(a)$ and $\mathbf{w}(\overleftarrow{a}) = -\mathbf{b}(a)$.
- For example, in Figure 10, the graph in (d) is the $D(\mathbf{f})$ of the (x, y)-flow $\mathbf{f}$ in (b); the curved directed edges represent $\overleftarrow{a}$.

- Let C be an $\mathbf{f}$ increment cycle in D. For every $a \in E(C^+)$ one has $\mathbf{f}(a) < c(a)$, hence $a \in E(D(\mathbf{f}))$; for every $a \in E(C^-)$ one has $0 < \mathbf{f}(a)$, hence $\overleftarrow{a} \in E(D(\mathbf{f}))$. Replacing a by $\overleftarrow{a}$ gives a directed cycle in $D(\mathbf{f})$ corresponding to C, denoted $\overleftarrow{C}$.

- Let $\overleftarrow{C}$ be the directed cycle in $D(\mathbf{f})$ corresponding to C, with $C^+ \subseteq E(\overleftarrow{C})$.

- Claim: C is an $\mathbf{f}$ increment cycle if and only if $\overleftarrow{C}$ is a directed cycle in $D(\mathbf{f})$, and

$$\mathbf{b}(C,\mathbf{f}) = \mathbf{w}(\overleftarrow{C}).$$

- Set $\overleftarrow{C}^+ = C^+$ and $\overleftarrow{C}^- = E(\overleftarrow{C}) \setminus \overleftarrow{C}^+$. Then the cost of C is

$$\begin{aligned}
\mathbf{b}(C,\mathbf{f}) &= \sum_{a\in C^+} \mathbf{b}(a) - \sum_{a\in C^-} \mathbf{b}(a) \\
&= \sum_{a\in \overleftarrow{C}^+} \mathbf{w}(a) - \sum_{\overleftarrow{a}\in \overleftarrow{C}^-} (-\mathbf{w}(\overleftarrow{a})) \\
&= \sum_{a\in E(\overleftarrow{C})} \mathbf{w}(a) = \mathbf{w}(\overleftarrow{C}).
\end{aligned}$$

- Conversely, let $\overleftarrow{C}$ be a directed cycle in $D(\mathbf{f})$, and set

$$\overleftarrow{C}^+ = \{a \in E(\overleftarrow{C}) :\ \mathbf{w}(a) > 0\}, \quad \overleftarrow{C}^- = E(\overleftarrow{C}) \setminus \overleftarrow{C}^+.$$

  Then $\overleftarrow{C}^+ \subset E(D)$ and the edges in $\overleftarrow{C}^-$ are all of the form $\overleftarrow{a}$.

- Set

$$C^+ = \overleftarrow{C}^+, \quad C^- = \{a \in E(D) :\ \overleftarrow{a} \in \overleftarrow{C}^-\}.$$

It is easy to verify that $C^+ \cup C^-$ forms a cycle in D, and

$$\begin{aligned}\mathbf{w}(\overleftarrow{C}) &= \sum_{a \in E(\overleftarrow{C})} \mathbf{w}(a) = \sum_{a \in \overleftarrow{C}^+} \mathbf{w}(a) + \sum_{\overleftarrow{a} \in \overleftarrow{C}^-} \mathbf{w}(\overleftarrow{a}) \\ &= \sum_{a \in C^+} \mathbf{b}(a) - \sum_{a \in C^-} \mathbf{b}(a) = \mathbf{b}(C, \mathbf{f}).\end{aligned}$$

- Thus finding an $\mathbf{f}$ increment cycle in D of cost < 0 is equivalent to finding a directed cycle of negative total weight in $D(\mathbf{f})$. Such a directed cycle is called a negative cycle.

Theorem 4.5.1 [Minimum-Cost Flow Criterion] An (x, y)-flow $\mathbf{f}$ in $N = (D_{xy}, \mathbf{b}, \mathbf{c})$ is a minimum-cost flow if and only if every $\mathbf{f}$ increment cycle C satisfies

$$\mathbf{b}(C, \mathbf{f}) \geqslant 0,$$

which is equivalent to $D(\mathbf{f})$ containing no negative cycle.

Proof

($\Rightarrow$) Let $\mathbf{f}$ be a minimum-cost flow, let C be an $\mathbf{f}$ increment cycle, and let $\widetilde{\mathbf{f}}$ be the revised flow based on C. Since val $\widetilde{\mathbf{f}}$ = val $\mathbf{f}$,

$$\begin{aligned} 0 \leqslant \mathbf{b}(\widetilde{\mathbf{f}}) - \mathbf{b}(\mathbf{f}) = & \quad \sigma(C) \left(\sum_{a \in C^+} \mathbf{b}(a) - \sum_{a \in C^-} \mathbf{b}(a) \right) \\ = & \quad \sigma(C) \mathbf{b}(C, \mathbf{f}). \end{aligned}$$

Since $\sigma(C) > 0$, it follows that $\mathbf{b}(C, \mathbf{f}) \geqslant 0$.

($\Leftarrow$)

- Let $\mathbf{f}$ be an (x, y)-flow in N such that every $\mathbf{f}$ increment cycle C has $\mathbf{b}(C, \mathbf{f}) \geqslant 0$.
- To prove that $\mathbf{f}$ has minimum cost, it suffices to show that every minimum-cost (x, y)-flow $\mathbf{f}^*$ in N with the same value as $\mathbf{f}$ satisfies $\mathbf{b}(\mathbf{f}^*) \geqslant \mathbf{b}(\mathbf{f})$.

- Define $\mathbf{h} \in \mathcal{E}(D(\mathbf{f}))$ by

$$\begin{array}{ll} \mathbf{h}(a) = \mathbf{f}^*(a) - \mathbf{f}(a) & \text{if } \mathbf{f}^*(a) > \mathbf{f}(a), \\ \mathbf{h}(\overleftarrow{a}) = \mathbf{f}(a) - \mathbf{f}^*(a) & \text{if } \mathbf{f}^*(a) < \mathbf{f}(a), \\ \mathbf{h}(a) = \mathbf{h}(\overleftarrow{a}) = 0 & \text{otherwise.} \end{array}$$

- It is easy to verify that $\mathbf{h}$ is a non-negative cycle vector of $D(\mathbf{f})$, i.e. $\mathbf{h} \in \mathcal{C}(D(\mathbf{f}))$.

- One can show that if $\mathbf{f}$ is a non-negative cycle vector of a digraph D, then

$$\mathbf{f} = \sum_{C \in \mathcal{C}} \lambda(C) \chi^C,$$

where $\lambda(C) \geq 0$, $\mathcal{C}(D)$ is the set of directed cycles of D, and χ^C is the incidence vector of the directed cycle C, i.e.

$$\chi^C(a) = \begin{cases} 1 & \text{if } a \in C, \\ 0 & \text{otherwise.} \end{cases}$$

- Hence there exists a function $\lambda : \mathcal{C}(D(\mathbf{f})) \to \mathbb{R}^+$ such that

$$\mathbf{h} = \sum_{\overleftarrow{C} \in \mathcal{C}(D(\mathbf{f}))} \lambda(\overleftarrow{C}) \chi^{\overleftarrow{C}}.$$

- That is,

$$\mathbf{f}^* - \mathbf{f} = \sum_{C \in \mathcal{C}(D)} \lambda(C) \chi^C,$$

where C is the increment cycle in D corresponding to $\overleftarrow{C}$, $\lambda(C) = \lambda(\overleftarrow{C}) \geq 0$, and

$$\chi^C(a) = \begin{cases} 1 & \text{if } a \in C^+, \\ -1 & \text{if } a \in C^-, \\ 0 & \text{otherwise.} \end{cases}$$

- Therefore, from $\mathbf{b}(C) = \mathbf{b}(C, \mathbf{f}) = \mathbf{b}(C, \mathbf{f}^*)$,

$$\mathbf{b}(\mathbf{f}^*) - \mathbf{b}(\mathbf{f}) = \sum_{C \in \mathcal{C}(D)} \lambda(C) \mathbf{b}(C) \geq 0,$$

where $\mathbf{b}(\chi^C) = \sum_{a \in E(D)} \mathbf{b}(a)\chi^C(a) = \sum_{a \in C} \mathbf{b}(a) = \mathbf{b}(C)$. The sufficiency is proved.

- The theorem is proved. ∎

Theorem 4.5.1 provides an algorithm for the minimum-cost maximum-flow problem. It was proposed by Klein (1967). The basic idea is: start from any (x, y) maximum flow $\mathbf{f}$ in N and examine each $\mathbf{f}$ increment cycle. If every $\mathbf{f}$ increment cycle has non-negative cost, then $\mathbf{f}$ is the desired minimum-cost maximum flow. If there is an $\mathbf{f}$ increment cycle C with $\mathbf{b}(C, \mathbf{f}) < 0$, replace $\mathbf{f}$ by the revised flow $\widetilde{\mathbf{f}}$ and repeat the procedure.

The key step in passing from one maximum flow $\mathbf{f}$ to another $\widetilde{\mathbf{f}}$ is to find an $\mathbf{f}$ increment cycle of negative cost, which can be done by finding a negative cycle in the auxiliary graph $D(\mathbf{f})$.

T. Yamada and H. Kinoshita (2002) proposed an efficient algorithm for

finding all negative cycles in a weighted digraph. We now outline Klein's algorithm.

Klein's algorithm:

1. Find a maximum (x, y)-flow $\mathbf{f}$ in N;
2. Construct $D(\mathbf{f})$;
3. Find a negative cycle in $D(\mathbf{f})$.
    - If there is no negative cycle, stop; then $\mathbf{f}$ is a minimum-cost maximum flow.
    - If $D(\mathbf{f})$ contains a negative cycle $\overleftarrow{C}$, then $C^+ \cup C^-$ is an $\mathbf{f}$ increment cycle (its positive direction agrees with that of $\overleftarrow{C}$); form the revised flow $\widetilde{\mathbf{f}}$, replace $\mathbf{f}$ by $\widetilde{\mathbf{f}}$, and return to step 1.

Example 4.5.2 Consider the network shown in Figure 10 (a); the ordered pairs on an edge a are $\mathbf{b}(a)$ and $\mathbf{c}(a)$, respectively. Example 4.4.2 found the maximum flow $\mathbf{f}$ shown in Figure 10 (b). The $D(\mathbf{f})$ constructed by Klein's algorithm is shown in Figure 10 (d), where the bold arc indicates a negative cycle; the corresponding $\mathbf{f}$ increment cycle is shown in bold in Figure 10 (b). The revised flow $\widetilde{\mathbf{f}}$ based on this increment cycle is shown in Figure 10 (c). Running Klein's algorithm again constructs $D(\widetilde{\mathbf{f}})$, shown in Figure 10 (e), which contains no negative cycle; hence $\widetilde{\mathbf{f}}$ is a minimum-cost maximum flow. Indeed, $\mathbf{b}(\mathbf{f}) = 62 > 56 = \mathbf{b}(\widetilde{\mathbf{f}})$. ∎

Exercises*

Supplementary 1 Prove that if $\mathbf{f}$ is a non-negative cycle vector of a digraph D, then

$$\mathbf{f} = \sum_{C \in \mathcal{C}} \lambda(C)\chi^C,$$

where $\lambda(C) \geq 0$, $\mathcal{C}(D)$ is the set of directed cycles of D, and χ^C is the incidence vector of the directed cycle C, i.e.

$$\chi^C(a) = \begin{cases} 1 & \text{if } a \in C, \\ 0 & \text{otherwise.} \end{cases}$$

4.5.3 Let $\mathbf{f}$ be an (x,y)-flow, and let P be an $\mathbf{f}$ incrementing path of minimum cost $\mathbf{b}(P,\mathbf{f})$ among all $\mathbf{f}$ incrementing paths; let $\widetilde{\mathbf{f}}$ be the revised flow based on P. (R. G. Busacker & P. J. Gowen, 1961)

(a) Prove that if $\mathbf{f}$ is a minimum-cost flow, then $\widetilde{\mathbf{f}}$ is also a minimum-cost flow.

(b) Prove that finding a minimum-cost $\mathbf{f}$ incrementing path in D is equivalent to finding a shortest xy-path in $D(\mathbf{f})$.

(c) Use (a) and (b) to design an algorithm for the minimum-cost maximum-flow problem.

6 Chinese Postman Problem

- A postman must traverse every street in his delivery area on each round, and return to the post office after finishing. How should he choose his route so that the total distance he walks is minimum?
- The problem was first posed and studied by Professor Guan Meigu of China (1960). Internationally it is therefore known as the Chinese Postman Problem.
- We regard the postman's delivery area as a connected weighted digraph $(D, \mathbf{w})$, where the vertices of D are the street intersections, the (one-way) streets are the edges, and the weights (necessarily positive) are the street lengths. A directed closed walk that traverses every edge of $(D, \mathbf{w})$ at least once is called a post tour. A post tour of minimum weight is called an optimal post tour.
- Solving the Chinese Postman Problem means finding an optimal post tour in a connected, positively weighted graph $(D, \mathbf{w})$.

- In real life many problems, such as the optimal routes of municipal water-sprinklers, snowploughs, refuse-collection vehicles, or tours of exhibition halls, can all be reduced to the Chinese Postman Problem.
- Because of its wide applicability, the Chinese Postman Problem has attracted great interest and many good solution methods have been proposed. This section presents an efficient algorithm for the problem, first put forward by Edmonds and Johnson (1973).

Eulerian case

First consider the simplest case, in which D is Eulerian. If D is Eulerian, then any Eulerian directed circuit is a post tour that traverses every edge of D exactly once, and hence is an optimal post tour.

Let D be Eulerian.

Edmonds-Johnson algorithm (an algorithm for finding an Eulerian directed circuit):

1. Choose any $x_0 \in V(D)$, find an out-arborescence T rooted at x_0, and set $P_0 = x_0$.

2. Suppose the directed trail $P_i = x_i a_i x_{i-1} \cdots a_2 x_1 a_1 x_0$ has been determined. Choose $a_{i+1} \in E(D) \setminus \{a_1, a_2, \cdots, a_i\}$ such that

   (i) $a_{i+1} = (x_{i+1}, x_i)$;

   (ii) $a_{i+1} \notin E(T)$ unless no other edge is available.

3. When step 2 can no longer be carried out, stop.

Theorem 4.6.1 The directed trail constructed when the above algorithm terminates is an Eulerian directed circuit in D. **Proof**

- Let

$$P_n = x_n a_n x_{n-1} \cdots a_2 x_1 a_1 x_0$$

be the directed trail obtained when the algorithm terminates. Since D is Eulerian, it is balanced.

- We first show that $x_n = x_0$. Otherwise,

$$d^+_{P_n}(x_n) = d^-_{P_n}(x_n) + 1.$$

Since D is balanced, there must be an unused edge entering x_n, contradicting the termination of the algorithm. Thus P_n is a closed directed trail.

- Let $R = D - E(P_n)$. Since both D and P_n are balanced, R is also balanced.

- For each $v \neq x_0$, let t_v be the unique edge of T entering v. By step 2 (ii), when t_v is used, every other edge entering v has already been used. Consequently, if any edge entering v remains unused, then t_v also remains unused.

- Suppose that $E(R) \neq \emptyset$, and choose an edge $(u,v) \in E(R)$. Then $d^+_R(u) > 0$. If $u \neq x_0$, the balance of R gives $d^-_R(u) > 0$, so $t_u \in E(R)$.

- Write the directed path in T from x_0 to u as

$$x_0 = u_0, u_1, \ldots, u_r = u.$$

  Applying the preceding argument successively to $u_r, u_{r-1}, \ldots, u_1$, we find that every edge (u_{j-1}, u_j) of this path belongs to $E(R)$. Hence $d_R^+(x_0) > 0$. This conclusion also holds immediately if $u = x_0$.

- Since R is balanced, $d_R^-(x_0) > 0$. Thus an unused edge enters $x_0 = x_n$, contradicting the termination of the algorithm.

- Therefore $E(R) = \emptyset$, and P_n is an Eulerian directed circuit in D. ∎

Non-Eulerian case

We now discuss the case in which D has no Eulerian directed circuit. First we settle the existence of a post tour in this case. (Recall: a post tour is a directed closed walk traversing all edges of D.) The graph D in Figure 11

has no post tour, because there is no directed path from $\{y_1, y_2, y_3\}$ to $\{x_1, x_2, x_3\}$, i.e. D is not strongly connected.

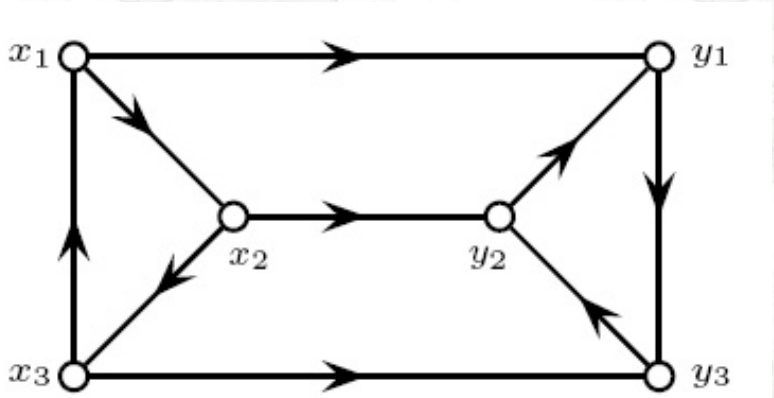

Figure 11: a digraph with no post tour

Theorem 4.6.2 A graph D has a post tour if and only if D is strongly connected.

Proof

($\Rightarrow$) From the definition of a post tour, D is immediately seen to be strongly connected.

($\Leftarrow$)

- Let D be strongly connected. Then D contains a directed cycle.
- Choose a directed closed walk C in D that contains as many edges of D as possible.
- If C is not a post tour, then there is an edge $a \in E(D)$ with $a \notin E(C)$.
- Let $a = (x, y)$ and choose $u \in V(C)$.
- Since D is strongly connected, there is a (u, x)-path P and a (y, u)-path Q in D.

- Then $C' = C \oplus (P + a) \oplus Q$ is a directed closed walk in D, and it contains at least one more edge of D than C (for example a).
- This contradicts the choice of C. Hence C is a post tour in D. ∎

- Let $(D, \mathbf{w})$ be a non-balanced, strongly connected weighted graph, and let P be a post tour in D. Then some edges occur repeatedly in P.
- Let $p(a)$ denote the number of times the edge $a \in E(D)$ occurs in P.
- For each $a \in E(D)$, let D^* be the graph obtained by replacing the edge a in D with $p(a)$ edges having the same direction as a. Then D^* is a supergraph of D and is balanced.
- Hence a post tour in D corresponds to an Eulerian directed circuit in the supergraph D^*.

- The Chinese Postman Problem can therefore be restated as follows: Given a strongly connected weighted graph $(D, \mathbf{w})$ with $\mathbf{w} > 0$,

  (i) construct a balanced supergraph D^* of D such that the added edge set E^* has minimum weight;

  (ii) find an Eulerian directed circuit in D^*.

The Edmonds-Johnson algorithm already provides an efficient method for (ii). We now discuss the method for (i), also proposed by Edmonds and Johnson (1973).

- For each $x \in V(D)$, set

$$\rho(x) = d_D^-(x) - d_D^+(x),$$

  and let

$$X = \{x \in V(D) : \ \rho(x) > 0\}, \quad Y = \{y \in V(D) : \ \rho(y) < 0\}.$$

- Since D is non-balanced, Theorem 1.6 gives $X \neq \emptyset$, $Y \neq \emptyset$, and

$$\sum_{x \in X} \rho(x) = -\sum_{y \in Y} \rho(y).$$

  Denote the common value of the two sides by $\rho(D)$.

- For example, for the weighted digraph $(D, \mathbf{w})$ in Figure 12 (a),

$$\rho(x_1) = -1,\ \rho(x_2) = 0,\ \rho(x_3) = 2,\ \rho(x_4) = 1,\ \rho(x_5) = -2.$$

  Thus $X = \{x_3, x_4\}$, $Y = \{x_1, x_5\}$, and $\rho(D) = 3$.

- Suppose the supergraph D^* and edge set E^* required in (i) have been chosen.

- For example, in Figure 12, the graph in (b) is the supergraph D^* of the graph D in (a); the edges in E^* are the curved directed edges.

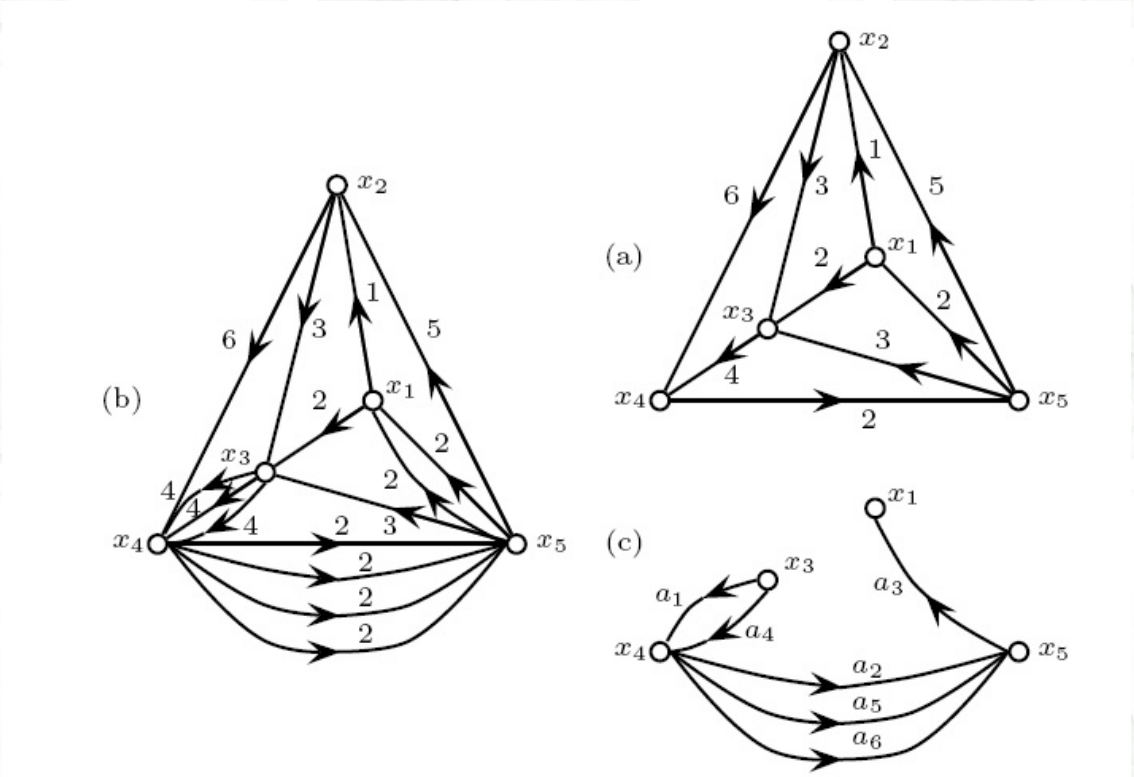

Figure 12: (a) the weighted graph $(D, \mathbf{w})$; (b) $(D^*, \mathbf{w}^*)$; (c) $H = G^*[E^*]$

- Let $H = D^*[E^*]$. Then H is the union of $\rho(D)$ edge-disjoint directed paths whose tails lie in X and whose heads lie in Y (how to prove this?).

- For example, in Figure 12 (b), the subgraph H induced by the curved edge set E^* (shown in (c)) contains $\rho(D) = 3$ edge-disjoint directed paths:

$$P_1 = x_3 a_1 x_4 a_2 x_5 a_3 x_1, \quad P_2 = x_3 a_4 x_4 a_5 x_5, \quad P_3 = x_4 a_6 x_5.$$

- Conversely, choose $\rho(D)$ directed paths in D from X to Y such that each $x \in X$ is the initial vertex of exactly $\rho(x)$ of these paths and each $y \in Y$ is the terminal vertex of exactly $-\rho(y)$ of these paths. If their total weight is minimum among all such choices, then adding a separate copy of every edge of each path gives a solution of (i).

- Thus the solution of (i) amounts to choosing $\rho = \rho(D)$ directed paths $P_1, P_2, \ldots, P_\rho$ in D from X to Y such that each $x \in X$ is the initial vertex of exactly $\rho(x)$ of these paths and each $y \in Y$ is the terminal vertex of exactly $-\rho(y)$ of these paths, while minimizing the total weight

$$\mathbf{w}(P_1) + \mathbf{w}(P_2) + \cdots + \mathbf{w}(P_\rho).$$

- To this end, construct the cost-capacity network $N = (D'_{x_0y_0}, \mathbf{b}, \mathbf{c})$, where D' is obtained from D by adding two new vertices x_0 and y_0, joining x_0 to every vertex $x \in X$ by an edge of capacity $\rho(x)$ and cost 0, and joining every vertex $y \in Y$ to y_0 by an edge of capacity $-\rho(y)$ and cost 0.

- For $a \in E(D)$, set $\mathbf{b}(a) = \mathbf{w}(a)$ and $\mathbf{c}(a) = \infty$. Figure 13 (a) shows D' and $N = (D'_{x_0y_0}, \mathbf{b}, \mathbf{c})$, where D is the graph in Figure 12 (a).

- Each unit (x_0, y_0)-flow $\mathbf{f}_0$ in N can be regarded as an (x, y)-path P_0 in D, with $x \in X$, $y \in Y$; its cost $\mathbf{b}(\mathbf{f}_0) = \mathbf{w}(P_0)$.

- Since both $E^+_{D'}(x_0)$ and $E^-_{D'}(y_0)$ are (x_0, y_0) cut edge sets of capacity $\rho(D)$, and every other (x_0, y_0) cut edge set has capacity ∞, both $E^+_{D'}(x_0)$ and $E^-_{D'}(y_0)$ are minimum-capacity (x_0, y_0) cut edge sets in N.

- By the max-flow min-cut theorem (4.1), the value of a maximum (x_0, y_0)-flow $\mathbf{f}$ in N is val $\mathbf{f} = \rho(D)$. Hence solving (i) reduces to the minimum-cost maximum-flow problem in N, for which Klein's algorithm already exists (Section 4.5).

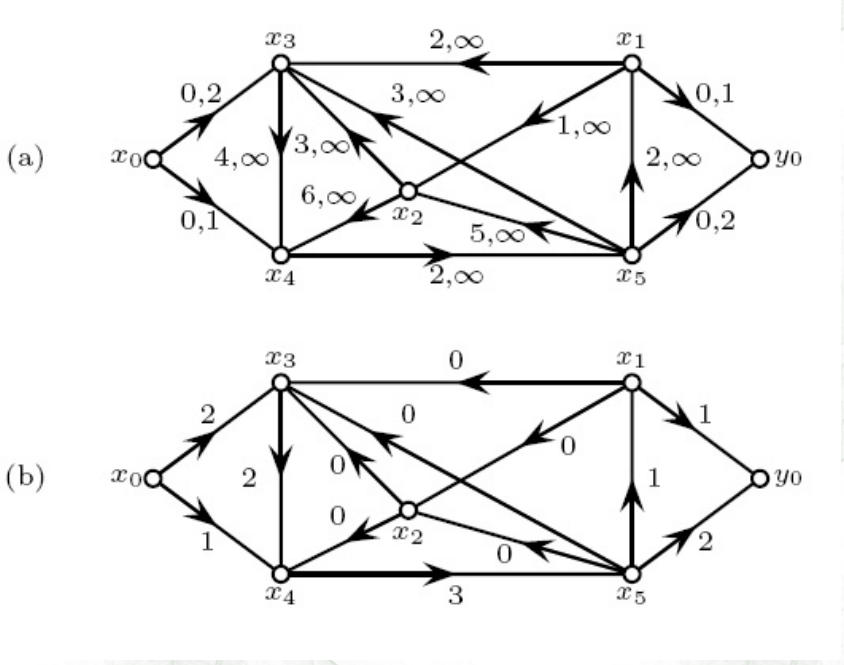

Figure 13: (a) the cost-capacity network $N = (N'_{x_0y_0}, \mathbf{b}, \mathbf{c})$; (b) a minimum-cost maximum flow in N

Summarizing the above procedure, we state:

Edmonds-Johnson algorithm:

1. Construct D' and $N = (D'_{x_0y_0}, \mathbf{b}, \mathbf{c})$;
2. Find a minimum-cost maximum flow in N;
3. Construct D^*;
4. Find an Eulerian directed circuit in D^*, which is an optimal post tour in $(D, \mathbf{w})$.

Example 4.6.2 Consider the strongly connected weighted graph $(D, \mathbf{w})$ in Figure 12 (a). By the Edmonds-Johnson algorithm, first construct D'

and $N = (D'_{x_0y_0}, \mathbf{b}, \mathbf{c})$, as shown in Figure 13 (a); the numbers on an edge a are the cost $\mathbf{b}(a)$ and the capacity $\mathbf{c}(a)$. Using Klein's algorithm (here, of course, by inspection) find the minimum-cost maximum flow $\mathbf{f}$, shown in Figure 13 (b), where $\mathbf{f}(a)$ is the number of times the edge a occurs in E^*. The graph D^* (shown in Figure 12 (b)) and the optimal post tour

$$P = (x_1, x_2, x_3, x_4, x_5, x_2, x_4, x_5, x_3, x_4, x_5, x_1, x_3, x_4, x_5, x_1)$$

are shown in Figure 12 (b), and moreover $\mathbf{w}(P) = 44$. ∎

Exercises

4.6.3 Let G be Eulerian. The following is the efficient **Fleury algorithm** for finding an Eulerian circuit in G.

1. Set $P_0 = x_0$;
2. Suppose $P_i = x_0e_1x_1 \cdots x_{i-1}e_ix_i$ has been determined. Choose $e_{i+1} \in E(G) \setminus \{e_1, e_2, \cdots, e_i\}$ such that

(i) $e_{i+1} = x_i x_{i+1}$;

(ii) unless no other edge is available, e_{i+1} is not a bridge of $G_i = G - \{e_1, e_2, \cdots, e_i\}$.

3. When step 2 can no longer be carried out, stop.

(a) Prove that if G is Eulerian, the trail P constructed by Fleury's algorithm is an Eulerian circuit in G.

7 Construction of Squared Rectangles

- A **squared rectangle** is a rectangle that can be dissected into finitely many (at least two) squares.
- If no two squares in the dissection have the same size, the squared rectangle is called **perfect**.
- The number of squares in the dissection is called the **order** of the squared rectangle.
- If the squares in the dissection contain a smaller squared square, the squared rectangle is called **compound**; otherwise it is called **simple**.
- Clearly every squared rectangle is composed of several simple squared rectangles.

Order	9	10	11	12	13	14	15	16	17	18
Number	2	6	22	67	213	744	2609	9016	31427	110384

- The squared-rectangle problem is a typical combinatorial problem. It has attracted interest since the 1930s. In the 1940s, R. L. Brooks, W. T. Tutte and others (1940), using graph theory, developed the construction of perfect rectangles into a systematic method, first gave a detailed list of perfect rectangles of orders 9–11, and proved that the minimum order of a perfect rectangle is 9. Later, C. J. Bouwkamp and others (1964), using an electronic computer, listed all perfect rectangles of orders 9–18. The numbers of perfect rectangles of each order are shown in the table above.

- For a long time no perfect squared square was known. R. Sprague (1939), by fitting together some of the perfect rectangles already found, discovered the first perfect squared square: a square of side 4205 and order 55. R. L. Brooks and others (1940), using the same method, obtained a square of side 608 and order 26. In 1967, J. C. Wilson, using an electronic computer, found a square of side 112 and order 25. In 1978, A. J. W. Duijvestijn, using an electronic

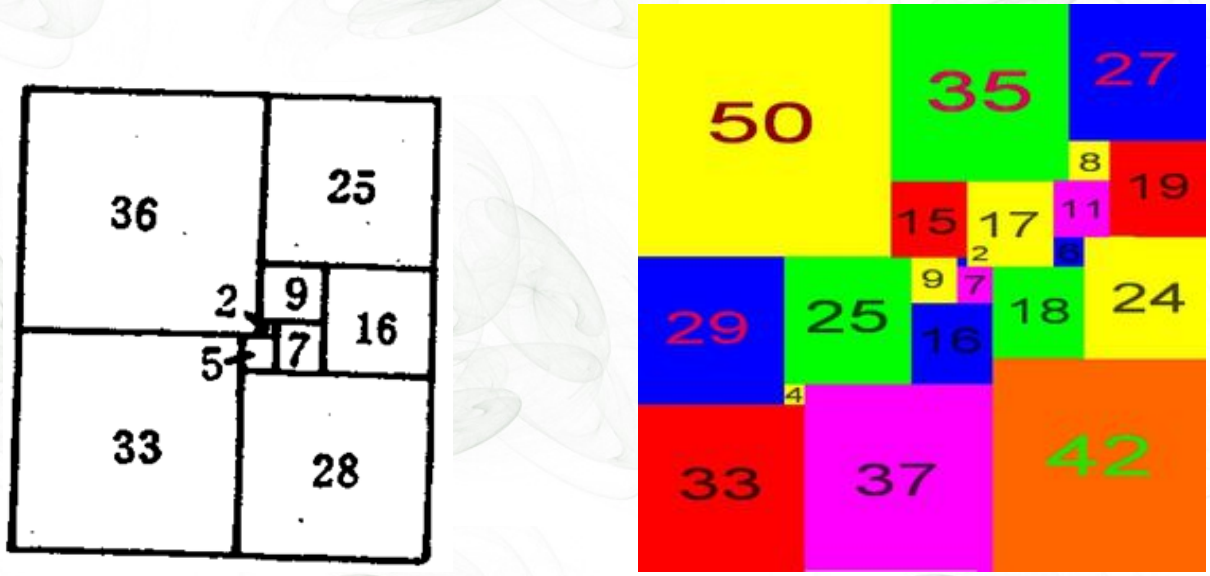

Figure 14: perfect squared squares

computer, found a square of side 112 and order 21 (see Figure 14); it is the unique perfect squared square of order 21 and is the perfect squared square of minimum order (as was proved by Duijvestijn, no perfect squared square of order less than 21 exists). Since then, the perfect squared squares have been completely enumerated up to order 37: the orders 22–26 were completed (and published) by

Bouwkamp and Duijvestijn in the 1980s–90s, and the orders 27–37 were enumerated in 2012–2018 (S. E. Anderson and collaborators; the orders 36 and 37 were completed by J. Williams). The numbers of perfect squared squares of the lower orders are shown in the table below.

Order	21	22	23	24	25	26	27	28	29	30
Number	1	8	12	30	172	541	1372	3949	10209	26234

The table shows how rapidly the number of perfect squared squares grows with the order. The counts for the remaining orders 31–37 are 71892, 196357, 528866, 1420439, 3784262, 10012056 and 26048712.

- The squared-rectangle problem is a typical combinatorial problem; like the Ramsey problem, it is interesting but very difficult. It is left for interested students to study on their own.

Thank you!