A&W

Flow Applications

Image Segmentation · Edge-Disjoint Paths · Bipartite Matching

Detailed Walkthrough

Goal of these notes. These notes work through three classical applications of max-flow / min-cut theory. Each one starts with a combinatorial problem that doesn't *look* like a flow problem, constructs a network whose flows or cuts encode the combinatorial objects of interest, and lets the max-flow / min-cut machinery do the work. The running pattern is *problem* → *network* → *max flow* → *combinatorial answer* — but the difficulty of the last arrow varies, and it's worth seeing all three side by side.

The three applications.

1. **Image segmentation.** Partition an image's pixels into foreground and background. The trick is a *quality-to-cost reformulation*: maximising a mixed-sign "quality" becomes minimising a non-negative cost, which is then exactly the capacity of an s–t cut. Min cut, max flow, segmentation done.

2. **Edge-disjoint paths.** Count the maximum number of pairwise edge-disjoint u–v paths. The reduction needs three subtleties — integrality of max flow, antiparallel cancellation, flow decomposition — and yields *Menger's theorem* on the side.

3. **Maximum bipartite matching.** Find a cardinality-maximum matching in a bipartite graph. The cleanest of the three: only integrality is needed, the matching falls out of conservation equations directly, and we get *König's theorem* as a one-paragraph corollary.

What you should take from this. Each application is self-contained, but read together they expose a recurring construction toolkit (unit capacities, source/sink gadgets, antiparallel arc pairs) and three classical duality theorems (max-flow/min-cut, Menger, König) that all live inside the same machinery. The closing summary chain at the end of each application traces the same four-step pipeline through its specific objects.

Contents

1 Image Segmentation via Min-Cut / Max-Flow

Given an image, partition its pixels into a foreground set A and a background set B. We phrase this as an optimization (maximize a quality q), reformulate it as a minimization (minimize q'), realize that minimization as an s–t min-cut on a constructed network N_G, and solve it with max-flow. Each step is justified.

1.1 Setup and notation

The image is modelled as an undirected graph

$$G = (P, E)$$

where:

- P is the set of *pixels* (vertices). Real images push $|P|$ into the millions.
- $E \subseteq \binom{P}{2}$ encodes adjacency. Typically each pixel is connected to its 4 or 8 grid neighbours.
- $\chi : P \to$ colors assigns each pixel a color (e.g. an RGB triple). This is the raw image data.

1.1.1 Per-pixel and per-edge estimates

In addition to the raw colors we are given three non-negative functions, computed in advance from χ:

$$\alpha : P \to \mathbb{R}_0^+, \qquad \beta : P \to \mathbb{R}_0^+, \qquad \gamma : E \to \mathbb{R}_0^+.$$

Their interpretations:

- α_p large $\implies$ pixel p *looks like foreground*. Concretely, α_p might be the (negative log) likelihood that $\chi(p)$ was sampled from a foreground color model.
- β_p large $\implies$ pixel p *looks like background*.
- γ_e large for $e = \{p, p'\}$ $\implies$ pixels p, p' *look similar* and should belong to the same region. A typical choice is $\gamma_e = \exp(-\|\chi(p) - \chi(p')\|^2/\sigma^2)$.

The first two terms encode **data fidelity** (each pixel pulled toward the label that fits its color); the third is the **smoothness** term (neighbouring similar pixels prefer to share a label).

1.2 The quality function

We want to choose a partition $P = A \sqcup B$ where A is foreground and B is background. Define

$$q(A, B) := \sum_{p \in A} \alpha_p + \sum_{p \in B} \beta_p - \sum_{\substack{e \in E, \\ |e \cap A| = 1}} \gamma_e \qquad \text{maximize.}$$

Term by term:

- $\sum_{p \in A} \alpha_p$ rewards putting foreground-looking pixels into A.
- $\sum_{p \in B} \beta_p$ rewards putting background-looking pixels into B.
- $\sum_{e:\, |e \cap A| = 1} \gamma_e$ sums γ_e over edges *cut* by the partition (one endpoint in A, one in B). Such edges are penalised: the more similar their endpoints look, the higher the cost of separating them. This is the smoothness term.

Remark 1.1: Why a quality and not a cost?

Maximising q is intuitive: high quality means confident labels and few cuts through similar pixels. But maximisations with mixed-sign terms are hard to massage into a flow network. The fix is a tiny algebraic trick: pass to a closely related *cost* q'.

1.3 From maximizing q to minimizing q'

Define the alternative

$$q'(A,B) := \sum_{p\in A} \beta_p + \sum_{p\in B} \alpha_p + \sum_{\substack{e\in E,\\ |e\cap A|=1}} \gamma_e \qquad \text{minimize.}$$

Note the swaps: α and β are exchanged in the per-pixel sums, and the smoothness term changes sign. The claim is that maximising q and minimising q' pick out the *same* optimal partition.

Proposition 1.1 (q and q' are complementary)**.** *For every partition $A \sqcup B = P$,*

$$q(A,B) + q'(A,B) \;=\; \sum_{p\in P} \big(\alpha_p + \beta_p\big) \;=:\; C,$$

which depends only on the input data, not on (A,B). Hence

$$\arg\max_{(A,B)} q(A,B) \;=\; \arg\min_{(A,B)} q'(A,B).$$

Proof. Add the two expressions and group by term type. For the α-terms,

$$\underbrace{\sum_{p\in A} \alpha_p}_{\text{from } q} + \underbrace{\sum_{p\in B} \alpha_p}_{\text{from } q'} = \sum_{p\in P} \alpha_p.$$

The same identity holds for β-terms (with A and B roles swapped). The γ-terms appear with opposite signs and cancel:

$$-\sum_{e:\,|e\cap A|=1} \gamma_e + \sum_{e:\,|e\cap A|=1} \gamma_e \;=\; 0.$$

So $q(A,B) + q'(A,B) = \sum_{p\in P} \alpha_p + \sum_{p\in P} \beta_p = C$. □

Key Idea 1.1: Reduction to a minimisation

Maximising the (mixed-sign) quality q is the same as minimising the (all non-negative) cost q':

$$\max_{(A,B)} q(A,B) \;=\; C \;-\; \min_{(A,B)} q'(A,B).$$

All three terms of q' are non-negative, so we can hope to express $q'(A,B)$ as the *capacity of an s–t cut* in some network. That is exactly what comes next.

1.4 Building the flow network N_G

We construct a directed network

$$N_G \;=\; \big(P \cup \{s,t\},\; \overrightarrow{E},\; c,\; s,\; t\big)$$

with a fresh source s and sink t ($s \neq t$). The directed edge set $\overrightarrow{E}$ and capacity $c : \overrightarrow{E} \to \mathbb{R}_0^+$ are given by three rules.

(S) **Source edges.** For every $p \in P$, add $s \to p$ with capacity

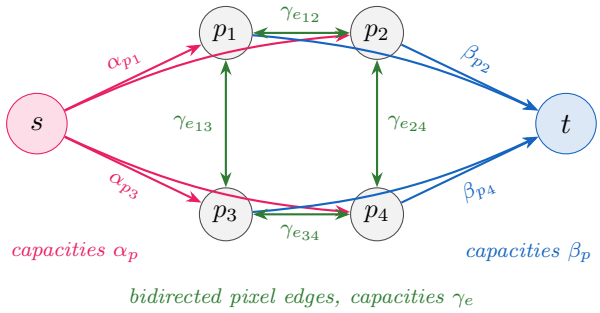
$$c(s,p) \;=\; \alpha_p.$$

(T) **Sink edges.** For every $p \in P$, add $p \to t$ with capacity

$$c(p,t) \;=\; \beta_p.$$

(P) **Pixel-pixel edges.** For every undirected edge $e = \{p,p'\} \in E$, add *two* directed edges $p \to p'$ and $p' \to p$, both with capacity

$$c(p,p') \;=\; c(p',p) \;=\; \gamma_e.$$



Remark 1.2: Why a pair of directed edges for each pixel-pixel edge?

G is undirected, but flow networks are directed. We want the smoothness penalty γ_e to be paid *whenever the cut separates p from p', regardless of which side each ended up on.* Adding both $p \to p'$ and $p' \to p$ ensures exactly one of the two crosses any s–t cut, so the contribution is γ_e either way.

1.5 Cuts in N_G are partitions of P

Recall: an *s–t cut* in N_G is a partition $V(N_G) = S \sqcup T$ with $s \in S$ and $t \in T$. Its capacity is

$$\mathrm{cap}(S,T) \;=\; \sum_{\substack{u \in S,\, v \in T \\ (u,v) \in \overrightarrow{E}}} c(u,v),$$

the total capacity of edges going *from S to T* (edges from T to S do **not** count).

There is an obvious bijection

$$\big\{\text{partitions } P = A \sqcup B\big\} \;\longleftrightarrow\; \big\{s\text{-}t \text{ cuts of } N_G\big\}, \qquad (A,B) \;\longmapsto\; \big(\{s\} \cup A,\; \{t\} \cup B\big).$$

The next theorem says this bijection is capacity-preserving in the sense that $q'(A,B)$ equals the cut capacity.

Theorem 1.2 (Cut capacity equals q')**.** *Let $A \sqcup B = P$ and let $S = \{s\} \cup A$, $T = \{t\} \cup B$. Then*

$$\mathrm{cap}(S,T) \;=\; q'(A,B).$$

Proof. Sum capacities of the three types of edges going from S to T.

(S) Source edges. An edge $s \to p$ goes from S to T iff $p \in T$, i.e. $p \in B$. Contribution:

$$\sum_{p \in B} \alpha_p.$$

(T) Sink edges. An edge $p \to t$ goes from S to T iff $p \in S$, i.e. $p \in A$. Contribution:

$$\sum_{p \in A} \beta_p.$$

(P) Pixel-pixel edges. For an undirected edge $e = \{p, p'\} \in E$ we added two directed edges $p \to p'$ and $p' \to p$, each with capacity γ_e. Three cases for how e is split by the cut:

- Both endpoints in A ($\subseteq S$): neither directed copy crosses $S \to T$. Contribution 0.
- Both endpoints in B ($\subseteq T$): neither directed copy crosses $S \to T$. Contribution 0.
- Endpoints split, say $p \in A$ and $p' \in B$: then $p \in S$, $p' \in T$, so $p \to p'$ *does* cross from S to T, contributing γ_e. The reverse edge $p' \to p$ goes from T to S and does *not* count.

Pixel-pixel edges contribute exactly γ_e for each cut edge of G:

$$\sum_{\substack{e \in E \\ |e \cap A| = 1}} \gamma_e.$$

Adding the three pieces:

$$\mathrm{cap}(S,T) \;=\; \sum_{p \in B} \alpha_p \;+\; \sum_{p \in A} \beta_p \;+\; \sum_{e:\, |e \cap A| = 1} \gamma_e \;=\; q'(A,B). \qquad \square$$

Key Idea 1.2: No fudge factor

Theorem 1.2 says q' *is exactly* the capacity function of N_G, restricted to cuts induced by partitions of P. The construction was reverse-engineered for this to hold on the nose: capacities $\alpha_p, \beta_p, \gamma_e$ were placed precisely where they would appear in the right cases.

1.6 Min-cut = max-flow: solving the segmentation

We can now stitch everything together.

Theorem 1.3 (Segmentation via max-flow)**.** *Let $f^\star$ be a maximum s–t flow in N_G. Then*

$$\min_{(A,B)} q'(A,B) \;=\; \mathrm{val}(f^\star),$$

and an optimal partition $(A^\star, B^\star)$ can be read off from any min-cut associated with $f^\star$.

Proof. By Theorem 1.2, minimising $q'(A,B)$ over partitions of P equals minimising the capacity of s–t cuts of N_G — every cut comes from some partition (it must split P into the source side and sink side) and vice versa. Therefore

$$\min_{(A,B)} q'(A,B) \;=\; \min_{(S,T)\ \text{s-t cut}} \mathrm{cap}(S,T) \overset{\text{MFMC}}{=} \max_{f} \mathrm{val}(f) \;=\; \mathrm{val}(f^\star),$$

using the Max-Flow Min-Cut Theorem in the middle. Given $f^\star$, the standard construction (the set of vertices reachable from s in the residual graph) yields a min-cut $(S^\star, T^\star)$. Setting $A^\star = S^\star \setminus \{s\}$ and $B^\star = T^\star \setminus \{t\}$ gives the optimal segmentation. □

Remark 1.3: Reading the slide carefully

The slide writes

$$q'(A,B) \;=\; \max_{f \text{ flow in } N_G} \mathrm{val}(f),$$

which is a slight abuse: q' depends on (A,B) while the right-hand side is a number. The precise statement is that the *minimum* of q' over partitions equals the max flow value, which is what Theorem 1.3 says.

Key Idea 1.3: Recipe

Given an image with α_p, β_p, γ_e:

1. Build N_G as in Section 4.
2. Run any max-flow algorithm (Edmonds-Karp, Dinic, push-relabel) to obtain $f^\star$ and a residual graph.
3. Let $A^\star$ be the pixels reachable from s in the residual graph; let $B^\star = P \setminus A^\star$.
4. Output the segmentation $(A^\star, B^\star)$. The cost is $\mathrm{val}(f^\star) = q'(A^\star, B^\star)$, and the corresponding quality is $C - \mathrm{val}(f^\star)$.

1.7 Why this is good news

Combinatorial difficulty disappears. A priori, segmentation is an optimisation over $2^{|P|}$ partitions, with a non-linear smoothness term. The reduction shows it is solvable in *polynomial time* — in fact, with specialised network structure, in essentially $O(|P| \cdot |E|)$ or better with modern push-relabel variants on grid graphs.

Globally optimal, not heuristic. The result is a *provably* optimal partition for the given α, β, γ. The user's modelling choices (how to set the per-pixel and per-edge weights) determine quality; the optimisation itself is exact.

Submodularity in disguise. The reason this trick works is that the energy q' is a *submodular* pseudo-boolean function on the indicator variables $x_p = \mathbf{1}[p \in A]$. Submodular minimisation with pairwise interactions can always be encoded as a min-cut (Kolmogorov & Zabih, 2004); image segmentation is the textbook example.

Watch out 1.1: When the reduction breaks

The reduction relies on $\gamma_e \geq 0$ (non-negative capacities). Smoothness terms that *reward* disagreement, or higher-order terms involving triples of pixels, generally cannot be handled by a single min-cut and need iterative methods (alpha-expansion, QPBO, etc.).

$$q \xrightarrow{\textit{complement}} q' \xrightarrow{\textit{cut bijection}} \mathrm{cap}(S,T) \xrightarrow{\textit{MFMC}} \mathrm{val}(f^\star).$$

2 Edge-Disjoint Paths via Max-Flow

Given an undirected graph G and two distinguished vertices u, v, count the maximum number of pairwise *edge-disjoint* u–v paths. We turn G into a directed flow network N_G with unit capacities, prove that $\max \mathrm{val}(f) = \#$edge-disjoint u-v paths, and explain the two non-obvious steps in the equivalence: antiparallel cancellation and flow decomposition. The result is Menger's theorem in disguise.

2.1 The problem

Definition 2.1 (Edge-disjoint paths)**.** Let $G = (V, E)$ be an undirected graph and $u, v \in V$ with $u \neq v$. A collection of u–v paths $P_1, \ldots, P_k$ is *edge-disjoint* if no edge $e \in E$ is used by two distinct paths, i.e. $E(P_i) \cap E(P_j) = \emptyset$ for all $i \neq j$. (Vertices may be reused.)

Given: a graph $G = (V, E)$ and $u, v \in V$ with $u \neq v$.
To find: the maximum number of edge-disjoint u–v paths in G. Call this number $\nu(G; u, v)$.

Remark 2.1: Why care?

Edge-disjoint paths model bandwidth, fault tolerance, and routing multiplicity. If a network can route k messages from u to v on disjoint links, then up to $k-1$ link failures still leave u and v connected. Computing $\nu(G; u, v)$ also unlocks *Menger's theorem*, the foundational connectivity duality result (Section 2.6).

2.2 The network construction

We turn G into a directed network

$$N_G \;=\; (V,\; A,\; c,\; u,\; v)$$

with the following recipe:

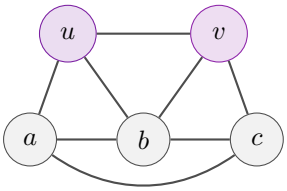
(A) **Bidirect every edge.** Set

$$A \;:=\; \big\{(x, y),\, (y, x) \;\big|\; \{x, y\} \in E\big\}.$$

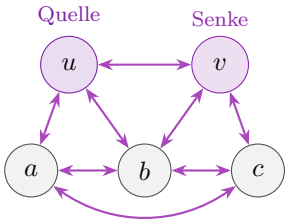
Each undirected edge of G becomes a pair of antiparallel directed arcs.

(B) **Unit capacities.** Set $c \equiv 1$, i.e. $c(a) = 1$ for every arc $a \in A$.

(C) **Source and sink.** Designate u as the source (*Quelle*) and v as the sink (*Senke*).



Graph G



Network N_G, $c \equiv 1$

2.2.1 Why bidirect every edge?

A naive idea would be to pick an orientation for each undirected edge. But in G a path from u to v may traverse an edge $\{x, y\}$ in either direction; pre-committing to one orientation would forbid paths that needed the other. Bidirecting keeps both options open.

2.2.2 Why unit capacities?

We want each edge of G to be *usable at most once* across all paths combined. Capacity 1 on each arc encodes "at most one unit of flow crosses this edge in this direction." The fact that we will obtain an *integral* max flow (Section 2.4.1) then means each arc carries flow 0 or 1.

Watch out 2.1: One subtle issue

With both (x, y) and (y, x) at capacity 1, an integral flow could in principle put 1 unit on both — using the underlying edge $\{x, y\}$ *twice*, once in each direction. We will deal with this by an antiparallel-cancellation argument (Section 2.4.2). At an optimum we can always assume at most one of $(x, y), (y, x)$ carries flow.

2.3 Easy direction: paths ⇒ flow

Proposition 2.2 (Lower bound)**.** *If G admits k edge-disjoint u–v paths, then N_G admits an integral flow f of value* $\mathrm{val}(f) = k$.

Proof. Let $P_1, \dots, P_k$ be k edge-disjoint u–v paths in G. Walking each P_i from u to v orients its edges; each undirected edge $\{x, y\} \in E(P_i)$ is traversed in some direction, say $x \to y$, giving an arc $(x, y) \in A$. Define

$$f(a) \;=\; \begin{cases} 1, & a \text{ is used by some } P_i, \\ 0, & \text{otherwise.} \end{cases}$$

Capacity: $f(a) \in \{0, 1\} \le 1 = c(a)$. ✓
Conservation. At any internal vertex $w \notin \{u, v\}$ of some P_i, the path enters w on one arc and leaves on another, both contributing 1 to the in/out-flow. Edge-disjointness across $P_1, \dots, P_k$ ensures these contributions never collide, so $\delta^- f(w) = \delta^+ f(w)$. At vertices used by no path the flow values are all 0. ✓
Value. Each path contributes one arc leaving u and one arc entering v. Summed: $\mathrm{val}(f) = \delta^+ f(u) - \delta^- f(u) = k$. ✓ □

2.4 Hard direction: flow ⇒ paths

This is the substantive direction. We need three ingredients.

2.4.1 Integrality of max flow

Lemma 2.3 (Integrality)**.** *Since N_G has integer capacities, a maximum integer-valued flow exists, and* $\max \mathrm{val}(f) \in \mathbb{Z}_{\geq 0}$.

Sketch. Ford-Fulkerson augments along an augmenting path and increases flow by the bottleneck residual capacity, which is a positive integer when capacities and the current flow are integral. Starting from $f \equiv 0$ (integral) and augmenting until no augmenting path remains preserves integrality. The final flow has integer value, and by max-flow/min-cut it is optimum. □

So we may assume $f^\star : A \to \{0, 1\}$.

2.4.2 Antiparallel cancellation

Lemma 2.4 (Cancellation)**.** *Let f be an integral max flow. There is an integral max flow $\tilde{f}$ with the same value such that for every $\{x, y\} \in E$, at most one of $\tilde{f}(x, y)$, $\tilde{f}(y, x)$ is nonzero.*

Proof. For each $\{x, y\} \in E$ with $f(x, y) = f(y, x) = 1$, set $\tilde{f}(x, y) = \tilde{f}(y, x) = 0$ and leave $\tilde{f}$ equal to f on every other arc. *Conservation.* At x, we removed one outgoing unit (along $x \to y$) *and* one incoming unit (from $y \to x$); these cancel, leaving the balance at x unchanged. Likewise at y. ✓ *Capacity & integrality.* Trivially $\tilde{f} \in \{0, 1\}^A$ and $\tilde{f} \leq c$. ✓ *Value.* Conservation at u is preserved, so $\mathrm{val}(\tilde{f}) = \mathrm{val}(f)$ unless $u \in \{x, y\}$. If $u = x$, then f was sending a unit out along $x \to y$ and pulling a unit back along $y \to x$, net contribution 0 to $\mathrm{val}(f)$ from this pair; removing both leaves the value unchanged. Symmetrically for $u = y$. □

Remark 2.2: Geometric reading

Antiparallel cancellation says: a max flow that simultaneously pushes a unit along $x \to y$ *and* along $y \to x$ is wasteful. Erasing both units doesn't change anything except the bookkeeping. After cancellation, the support of $\tilde{f}$ in A corresponds to an *orientation* of a subset of edges of G — each edge appears at most once.

2.4.3 Flow decomposition

Lemma 2.5 (Flow decomposition into u–v paths)**.** *Let $\tilde{f} : A \to \{0, 1\}$ be a 0/1-valued flow with $\mathrm{val}(\tilde{f}) = k$ that uses each undirected edge of G in at most one direction (as guaranteed by Lemma 2.4). Then there exist k directed u–v paths $\widetilde{P}_1, \ldots, \widetilde{P}_k$ in N_G whose arc sets are pairwise disjoint and contained in $\{a \in A : \tilde{f}(a) = 1\}$.*

Proof. We argue by induction on k. The base case $k = 0$ is vacuous.

For $k \geq 1$, $\mathrm{val}(\tilde{f}) = k > 0$ means $\delta^+\tilde{f}(u) > \delta^-\tilde{f}(u)$, so there is an arc (u, w_1) with $\tilde{f}(u, w_1) = 1$. By conservation at w_1 (or $w_1 = v$ already), $\delta^+\tilde{f}(w_1) \geq \delta^-\tilde{f}(w_1) \geq 1$, giving an arc (w_1, w_2) with $\tilde{f}(w_1, w_2) = 1$, unless $w_1 = v$.

Why the walk reaches v. Continue the walk $u, w_1, w_2, \ldots$ Conservation guarantees a nonzero out-arc at every intermediate vertex. The walk cannot continue forever in N_G on a finite vertex set without revisiting a vertex, but *revisiting is allowed in the construction*; we just need to argue it eventually hits v. If it ever revisits a vertex $w_i = w_j$ with $i < j$, the cycle $w_i \to w_{i+1} \to \cdots \to w_j$ uses unit flow on each arc, so we can reduce the flow on those arcs by 1 (preserving conservation and value) and remove the cycle from the walk. Repeating this finitely many times, we obtain a simple directed walk that hits v — call it $\widetilde{P}_1$.

Subtract and recurse. Set $\tilde{f}^{(1)} := \tilde{f} - \mathbf{1}[a \in \widetilde{P}_1]$. Each arc of $\widetilde{P}_1$ had flow 1, so $\tilde{f}^{(1)} \geq 0$; conservation is preserved at all internal vertices of $\widetilde{P}_1$ (one in, one out subtracted); and at u, v the value drops by exactly 1, so $\mathrm{val}(\tilde{f}^{(1)}) = k - 1$. By the inductive hypothesis, $\tilde{f}^{(1)}$ decomposes into $k-1$ arc-disjoint u–v paths $\widetilde{P}_2, \ldots, \widetilde{P}_k$. None of these uses an arc of $\widetilde{P}_1$ (we set those flows to 0), so all k paths are arc-disjoint. □

2.4.4 Putting it together

Proposition 2.6 (Upper bound)**.** *If N_G admits an integral flow of value k, then G admits k edge-disjoint u–v paths.*

Proof. Take a max integral flow f (Lemma 2.3), apply antiparallel cancellation to obtain $\tilde{f}$ (Lemma 2.4), then decompose into k arc-disjoint directed paths $\widetilde{P}_i$ in N_G (Lemma 2.5). Project each $\widetilde{P}_i$ back to G by forgetting arc directions; this yields a u–v walk in G, which contains a u–v path P_i.
Edge-disjointness in G. Suppose, for contradiction, that P_i and P_j share an undirected edge $\{x, y\}$ for some $i \neq j$. Then $\widetilde{P}_i$ used some arc on $\{x, y\}$ — say (x, y) — and $\widetilde{P}_j$ used an arc on $\{x, y\}$ as well. By Lemma 2.4, only one direction has flow, hence both used (x, y). But $\widetilde{P}_i$ and $\widetilde{P}_j$ are *arc-disjoint* by Lemma 2.5 — contradiction. So $P_1, \ldots, P_k$ are edge-disjoint in G. □

2.5 The main theorem

Theorem 2.7 (Edge-disjoint paths = max flow)**.** *For any undirected graph $G = (V, E)$ and distinct $u, v \in V$,*

$$\nu(G; u, v) \;=\; \max_{f \text{ flow in } N_G} \mathrm{val}(f).$$

Proof. $\nu \leq \max \mathrm{val}(f)$: by Proposition 2.2, any collection of ν edge-disjoint paths produces a flow of value ν.
$\max \mathrm{val}(f) \leq \nu$: by Proposition 2.6, any flow of value k produces k edge-disjoint paths. □

Key Idea 2.1: The reduction in one line

Replace each undirected edge by two unit-capacity antiparallel arcs and ask for the max u-to-v flow. By integrality, antiparallel cancellation, and flow decomposition, the max-flow value is exactly the maximum number of edge-disjoint u–v paths.

2.6 Connection to Menger's theorem

The duality side of Theorem 2.7 is one of the oldest results in combinatorics.

Theorem 2.8 (Menger's theorem, edge version)**.** *$\nu(G; u, v)$ equals the minimum number of edges whose removal disconnects u from v in G. Call that number $\lambda(G; u, v)$.*

Proof. By Theorem 2.7, $\nu(G; u, v) = \max \mathrm{val}(f)$ in N_G. By the max-flow/min-cut theorem, $\max \mathrm{val}(f) =$ min capacity of an s–t cut in N_G. An s–t cut (S, T) in N_G with $u \in S, v \in T$ has capacity equal to the number of arcs $(x, y) \in A$ with $x \in S, y \in T$; since arcs come in antiparallel pairs and only the $S \to T$ direction counts, this is exactly the number of undirected edges of G with one endpoint in S and one in T. Such an edge set disconnects u from v in G when removed, and conversely any disconnecting edge set defines such a cut. So $\max \mathrm{val}(f) = \lambda(G; u, v)$, completing the proof. □

Remark 2.3: Menger gives both inequalities for free

The harder inequality of Menger's theorem ($\nu \geq \lambda$) — every disconnecting set of size λ admits λ edge-disjoint paths around it — is precisely the augmenting-path/max-flow construction. The reverse direction ($\nu \leq \lambda$) is trivial: each path must use at least one edge of every disconnecting set, so k disjoint paths need at least k such edges.

2.7 Algorithm and complexity

Key Idea 2.2: Recipe

1. Build N_G: bidirect every edge, set all capacities to 1, designate u, v as source/sink.
2. Run a max-flow algorithm to obtain an integral max flow $f^\star$.
3. Cancel antiparallel pairs (Lemma 2.4) to get $\tilde{f}$.
4. Repeatedly find a u–v path in the support of $\tilde{f}$ and subtract it from the flow (Lemma 2.5). After $k = \mathrm{val}(f^\star)$ iterations, you have k edge-disjoint paths.

Complexity. On N_G with $|V| = n$ vertices and $|A| = 2|E|$ arcs, all of unit capacity, max flow can be computed in $O(|E|\sqrt{n})$ time using Dinic's algorithm — the classical *Hopcroft-Karp* regime. The cancellation and decomposition phases each run in $O(n + |E|)$. Total: $O(|E|\sqrt{n})$.
Why so fast. Unit-capacity networks have at most $\mathrm{val}(f^\star) \leq \min(\delta^+ u, \delta^- v) = O(n)$ augmenting phases in Ford-Fulkerson, but Dinic's blocking-flow analysis tightens this to $O(\sqrt{n})$ phases on simple graphs.

Example 2.1: The 5-vertex graph from the slide

Take $V = \{u, v, a, b, c\}$ with edges $\{u,v\}, \{u,a\}, \{u,b\}, \{v,b\}, \{v,c\}, \{a,b\}, \{b,c\}, \{a,c\}$. The maximum number of edge-disjoint u–v paths is **3**, realised by:

$$P_1 : u \to v, \qquad P_2 : u \to a \to c \to v \text{ (via } \{a,c\} \text{ and } \{c,v\}), \qquad P_3 : u \to b \to v.$$

Verifying edge-disjointness: P_1 uses $\{u,v\}$; P_2 uses $\{u,a\}, \{a,c\}, \{c,v\}$; P_3 uses $\{u,b\}, \{b,v\}$. No edge is used twice. The remaining edges $\{a,b\}, \{b,c\}$ are unused — there is no room for a fourth path because u has only 3 neighbours, capping $\nu(G; u, v) \leq 3$ at the source.

---

undirected $G \xrightarrow{\text{bidirect, cap 1}}$ *network* $N_G \xrightarrow{\text{max flow}}$ *integral* $f^\star \xrightarrow{\text{cancel \& decompose}}$ k *edge-disjoint paths.*

3 Maximum Bipartite Matching via Max-Flow

Given an undirected bipartite graph $G = (U \uplus W, E)$, find a *cardinality-maximum matching* — the largest set of edges sharing no endpoint. We turn G into a directed network N_G by adding a source s, a sink t, and three layers of unit-capacity arcs (source-side, middle, sink-side), then prove

$$\nu(G) \;=\; \max_{f \text{ flow in } N_G} \operatorname{val}(f).$$

The proof is much smoother than the edge-disjoint-paths case: no antiparallel arcs, no cancellation step, the matching is read off directly from any integral max flow. Along the way we obtain **König's theorem** as a one-paragraph consequence of max-flow/min-cut.

3.1 The problem

Definition 3.1 (Bipartite graph)**.** A graph $G = (V, E)$ is *bipartite* with parts U, W if $V = U \uplus W$ (disjoint union) and every edge $e \in E$ has one endpoint in U and one in W. Equivalently, $E \subseteq \big\{\{u, w\} : u \in U, w \in W\big\}$.

Definition 3.2 (Matching)**.** A *matching* in G is a set $M \subseteq E$ of pairwise vertex-disjoint edges, i.e. no vertex is incident to two edges of M. The *cardinality* of M is $|M|$. The *matching number* is

$$\nu(G) \;:=\; \max_{M \text{ matching in } G} |M|.$$

Given: a bipartite graph $G = (U \uplus W, E)$, unweighted and undirected.
To find: a matching $M^\star$ with $|M^\star| = \nu(G)$.

Remark 3.1: Why bipartite specifically?

The bipartite restriction is exactly what makes the max-flow reduction clean. For *general* (non-bipartite) graphs the matching problem is still polynomial-time solvable but requires substantially more machinery (Edmonds' blossom algorithm). The bipartite structure lets us orient edges $U \to W$ once and for all, so no antiparallel-arc gymnastics are needed.

3.2 The network construction

We turn G into a directed network

$$N_G \;=\; (U \uplus W \uplus \{s, t\},\; A,\; c,\; s,\; t)$$

with two fresh vertices: a source s and a sink t ($s \neq t$, $s, t \notin U \cup W$). The arc set splits into three pieces:

$$A \;=\; \big(\{s\} \times U\big) \;\cup\; \big\{(u, w) \in U \times W \;\big|\; \{u, w\} \in E\big\} \;\cup\; \big(W \times \{t\}\big)$$

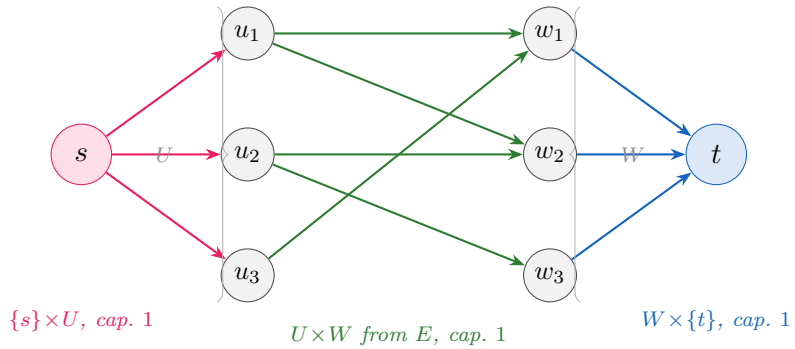
and all capacities are 1:

$$c \;\equiv\; 1.$$

Reading the three pieces:

- **Source-side arcs** (s, u) for every $u \in U$ — one arc from s into each U-vertex, capacity 1.
- **Middle arcs** (u, w) for every $\{u, w\} \in E$ — every original (undirected) edge becomes a single directed arc, oriented from U to W, capacity 1.
- **Sink-side arcs** (w, t) for every $w \in W$ — one arc from each W-vertex into t, capacity 1.

Picture



3.2.1 Why this orientation works without antiparallel arcs

In the edge-disjoint-paths construction, each edge of G had to become a *pair* of antiparallel arcs because a u–v path in G might traverse an edge in either direction. Here the situation is fundamentally different: every s–t path in N_G has the shape

$$s \to u \to w \to t, \quad u \in U,\ w \in W,$$

so middle arcs only ever need to be traversed from U to W. One orientation suffices.

3.2.2 Why unit capacities

Each $u \in U$ has exactly one incoming arc (s, u) of capacity 1. By flow conservation at u, the total flow leaving u on middle arcs is ≤ 1. Integrally this means *at most one* middle arc out of u carries flow — in other words, u gets matched to at most one W-vertex. The same argument at each $w \in W$ caps the matching degree on the other side. Capacities are doing the matching constraint for us.

3.3 Easy direction: matching ⇒ flow

Proposition 3.3 (Lower bound)**.** *If G admits a matching M of size k, then N_G admits an integral flow f of value* $\mathrm{val}(f) = k$.

Proof. For each matched edge $\{u, w\} \in M$, set

$$f(s,u) \;=\; f(u,w) \;=\; f(w,t) \;=\; 1,$$

and set $f(a) = 0$ on every other arc a.
Capacity: every $f(a) \in \{0, 1\} \leq 1$. ✓
Conservation. Pick any $u \in U$. If u is matched in M, then by the assignment above we have $f(s, u) = 1$ flowing in and exactly one $f(u, w) = 1$ flowing out (corresponding to u's unique matched edge); all other middle arcs out of u carry 0. So inflow = outflow = 1 at u. If u is unmatched, all flows touching u are 0. ✓Symmetric argument at every $w \in W$.
Value. $\mathrm{val}(f)$ counts the flow leaving s, which is $\sum_{u\in U} f(s, u)$ = (number of matched U-vertices) $= |M| = k$. ✓ □

3.4 Hard direction: flow ⇒ matching

3.4.1 Integrality of max flow

Lemma 3.4 (Integrality). *Since N_G has integer capacities, a maximum flow with integer values on every arc exists, and* $\max \mathrm{val}(f) \in \mathbb{Z}_{\geq 0}$.

Sketch. Ford-Fulkerson augments along an augmenting path and increases flow by the bottleneck residual capacity, which is a positive integer when capacities and the current flow are integral. Starting from $f \equiv 0$ and augmenting until no augmenting path exists preserves integrality, and the final flow is optimum by max-flow/min-cut. □

So we may assume $f^\star : A \to \{0, 1\}$.

3.4.2 Reading off the matching

Proposition 3.5 (Upper bound). *If N_G admits an integral flow f of value k, then G admits a matching of size k.*

Proof. Define

$$M \;:=\; \big\{\{u, w\} \in E \;:\; f(u, w) = 1\big\}.$$

We must show M is a matching and $|M| = k$.

Matching property at U. Fix $u \in U$. The only arc into u is (s, u) with capacity 1, so $f(s, u) \in \{0, 1\}$. By flow conservation,

$$\sum_{w:\,(u,w)\in A} f(u, w) \;=\; f(s, u) \;\in\; \{0, 1\}.$$

Each $f(u, w) \in \{0, 1\}$ and they are non-negative integers summing to at most 1, so *at most one* of them equals 1. Therefore u is incident to at most one edge of M. ✓

Matching property at W. Fix $w \in W$. The only arc out of w is (w, t) with capacity 1, so $f(w, t) \in \{0, 1\}$. By conservation,

$$\sum_{u:\,(u,w)\in A} f(u, w) \;=\; f(w, t) \;\in\; \{0, 1\},$$

so at most one $f(u, w) = 1$, i.e. w is incident to at most one edge of M. ✓

Cardinality.

$$|M| \;=\; \sum_{(u,w)\in A} f(u, w) \;=\; \sum_{u\in U}\sum_{w} f(u, w) \;=\; \sum_{u\in U} f(s, u) \;=\; \mathrm{val}(f) \;=\; k. \qquad \square$$

Key Idea 3.1: The matching falls out of the conservation equations

Conservation at each $u \in U$ together with ≤ 1 inflow forces "u sends flow to at most one neighbour." Conservation at each $w \in W$ with ≤ 1 outflow forces "w receives flow from at most one neighbour." Those two constraints are exactly the matching condition. The unit capacities on (s, u) and (w, t) are doing the work.

3.5 The main theorem

Theorem 3.6 (Bipartite matching = max flow)**.** *For any bipartite graph $G = (U \uplus W, E)$,*

$$\nu(G) \;=\; \max_{f \text{ flow in } N_G} \operatorname{val}(f).$$

Proof. $\nu(G) \le \max \operatorname{val}(f)$ by Proposition 3.3: every matching of size $\nu(G)$ produces a flow of value $\nu(G)$.
$\max \operatorname{val}(f) \le \nu(G)$ by Proposition 3.5: every integral flow of value k produces a matching of size k, and by Lemma 3.4 an integral max flow exists. □

Remark 3.2: Why this is gentler than edge-disjoint paths

For edge-disjoint paths we needed three steps to go from a flow back to a combinatorial object: integrality, antiparallel cancellation, and flow decomposition into paths. Here we only need integrality. The reason is structural: the network N_G is acyclic, has only one kind of s–t path (length 3, of the form $s \to u \to w \to t$), and conservation alone witnesses the matching property. No path-decomposition argument is needed because each unit of flow already *is* a length-3 s–t path corresponding to one matched edge.

3.6 König's theorem as a free corollary

The dual side of Theorem 3.6 is the celebrated *König-Egerváry theorem.*

Definition 3.7 (Vertex cover)**.** A *vertex cover* of G is a set $C \subseteq V$ such that every edge of G has at least one endpoint in C. Write $\tau(G)$ for the minimum cardinality of a vertex cover.

Theorem 3.8 (König's theorem)**.** *For any bipartite graph G, $\nu(G) = \tau(G)$.*

Proof. By Theorem 3.6 and max-flow/min-cut,

$$\nu(G) \;=\; \max \operatorname{val}(f) \;=\; \min_{(S,T)\ \text{s-t cut}} \operatorname{cap}(S,T).$$

We claim $\min \operatorname{cap}(S,T) = \tau(G)$.
From cut to cover. Let (S,T) be a min cut with $s \in S$, $t \in T$. Suppose some middle arc (u,w) crosses the cut, i.e. $u \in S$ and $w \in T$. Move w to S: this removes the contribution of (u,w) to the cut (capacity 1) and adds the contribution of (w,t) (capacity 1). The net change is 0. Repeating, we may assume the min cut contains *no middle arcs.* After this normalisation, the only arcs crossing the cut are source-side (s,u) arcs with $u \in T$, and sink-side (w,t) arcs with $w \in S$. Define

$$C \;:=\; \underbrace{(U \cap T)}_{U_T} \cup \underbrace{(W \cap S)}_{W_S}.$$

We verify C is a vertex cover. Take any edge $\{u,w\} \in E$. The corresponding middle arc (u,w) does *not* cross the cut, so we cannot have $u \in S$ and $w \in T$ simultaneously. Therefore $u \in T$ or $w \in S$, i.e. $u \in C$ or $w \in C$. ✓

$$|C| \;=\; |U_T| + |W_S| \;=\; \operatorname{cap}(S,T).$$

Hence $\tau(G) \le \min \operatorname{cap}(S,T)$.
From cover to cut. Conversely, given a vertex cover C, define

$$S \;:=\; \{s\} \cup (U \setminus C) \cup (W \cap C), \qquad T \;:=\; \{t\} \cup (U \cap C) \cup (W \setminus C).$$

Then (S,T) is an s–t cut, and we count its capacity by checking each arc type:

- Source arcs (s, u) cross $S \to T$ iff $u \in U \cap C$. Count: $|U \cap C|$.
- Middle arcs (u, w) cross iff $u \in U \setminus C$ and $w \in W \setminus C$. But C is a vertex cover, so every edge $\{u, w\}$ has $u \in C$ or $w \in C$ — no middle arc crosses. Count: 0.
- Sink arcs (w, t) cross iff $w \in W \cap C$. Count: $|W \cap C|$.

Total: $\text{cap}(S, T) = |U \cap C| + |W \cap C| = |C|$. So $\min \text{cap}(S, T) \leq \tau(G)$. □

Key Idea 3.2: The duality at a glance

Max matching = Max flow = Min cut = Min vertex cover. The three equalities are, respectively, our reduction, max-flow/min-cut, and the cut-cover correspondence above.

3.7 Algorithm and complexity

Key Idea 3.3: Recipe

1. Build N_G: source s, sink t, three layers of unit-capacity arcs.
2. Run a max-flow algorithm to obtain an integral max flow $f^\star$.
3. Read off $M^\star = \{\{u, w\} : f^\star(u, w) = 1\}$. By Proposition 3.5 this is a matching of size $\text{val}(f^\star)$.

Augmenting paths = alternating paths

In N_G, an *augmenting path* (a directed s–t path in the residual graph) of the form

$$s \to u_1 \to w_1 \to u_2 \to w_2 \to \cdots \to u_k \to w_k \to t$$

alternates between forward middle arcs (currently unsaturated, so $f(u_i, w_i) = 0$ in the original direction) and backward middle arcs (where the residual goes "against" a saturated arc, so $f(u_{i+1}, w_i) = 1$). Translated back to G, this is exactly an *M-alternating path* starting at an unmatched u_1 and ending at an unmatched w_k, alternating between non-matching and matching edges. Augmenting along it flips the alternation, increasing $|M|$ by 1.

Complexity. Run **Hopcroft-Karp**, which is exactly Dinic's algorithm specialised to unit-capacity bipartite graphs. It finds all shortest augmenting paths in each phase using BFS+DFS in $O(|V|+|E|)$, and the number of phases is $O(\sqrt{|V|})$. Total:

$$\text{Time} = O\big(|E|\sqrt{|V|}\big).$$

Watch out 3.1: Don't try this for general matching

Without the bipartite hypothesis, the construction breaks: orienting edges $U \to W$ requires the bipartition. General-graph matching needs Edmonds' blossom algorithm, where odd cycles (*blossoms*) get contracted to allow augmenting paths to traverse them. Running time is $O(|V|^3)$ classically, $O(|E|\sqrt{|V|})$ with Micali-Vazirani.

Example 3.1: The figure's graph

With $U = \{u_1, u_2, u_3\}$, $W = \{w_1, w_2, w_3\}$, and edges

$\{u_1, w_1\}, \{u_1, w_2\}, \{u_2, w_2\}, \{u_2, w_3\}, \{u_3, w_1\}$, a maximum matching has size **3**:

$$M^\star \;=\; \big\{\{u_1, w_2\}, \{u_2, w_3\}, \{u_3, w_1\}\big\}.$$

The corresponding flow saturates the three middle arcs above, plus all three source arcs and all three sink arcs. A min vertex cover of size 3 is e.g. $C = \{u_1, u_2, u_3\}$ (verifying König: $\nu = \tau = 3$).

---

bipartite $G \xrightarrow{\text{add } s,t,\ \text{orient}}$ *network* $N_G \xrightarrow{\text{max flow}}$ *integral* $f^\star \xrightarrow{\text{read off middle arcs}}$ *maximum matching* $M^\star$.