Maximum flow problem

Network flows

Network

- Directed graph G = (V, E)
- **Source node** $s \in V$, sink node $t \in V$
- **Edge** capacities: cap : $E \rightarrow \mathbb{R}_{>0}$
- Flow: $f : E \to \mathbb{R}_{>0}$ satisfying
 - 1. Flow conservation constraints

$$\sum_{e: \text{target}(e)=v} f(e) = \sum_{e: \text{source}(e)=v} f(e), \text{ for all } v \in V \setminus \{s, t\}$$

2. Capacity constraints

$$0 \leq f(e) \leq cap(e)$$
, for all $e \in E$

Maximum flow problem

• Excess:

$$excess(v) = \sum_{e:target(e)=v} f(e) - \sum_{e:source(e)=v} f(e)$$

• If f is a flow, then excess(v) = 0, for all $v \in V \setminus \{s, t\}$

- Value of a flow: val(f) = excess(t)
- Maximum flow problem:

```
\max\{\operatorname{val}(f) \mid f \text{ is a flow in } G\}
```

Can be seen as a linear programming problem.

Lemma.

If f is a flow in G, then excess(t) = -excess(s).

Maximum flow problem (2)

Proof. We have

$$excess(s) + excess(t) = \sum_{v \in V} excess(v) = 0.$$

• First "=": excess(
$$v$$
) = 0, for $v \in V \setminus \{s, t\}$

Second "=": For any edge e = (v, w), the flow through e appears twice in the sum, positively in excess(w) and negatively in excess(v).

Cuts

- A cut is a partition (S, T) of V, i.e., $T = V \setminus S$.
- (S, T) is an (s, t)-cut if $s \in S$ and $t \in T$.
- Capacity of (S, T)

$$\operatorname{cap}(S, T) = \sum_{E \cap (S \times T)} \operatorname{cap}(e)$$

A cut is saturated by *f* if *f*(*e*) = cap(*e*), for all *e* ∈ *E* ∩ (*S* × *T*), and *f*(*e*) = 0, for all *e* ∈ *E* ∩ (*T* × *S*).

Lemma.

If f is a flow and (S, T) an (s, t)-cut, then

$$\operatorname{val}(f) = \sum_{e \in E \cap (S \times T)} f(e) - \sum_{e \in E \cap (T \times S)} f(e) \leq \operatorname{cap}(S, T).$$

If S is saturated by f, then val(f) = cap(S, T).

Maximum flow problem (2)

Proof. We have

$$val(f) = -excess(s) = -\sum_{u \in S} excess(u)$$
$$= \sum_{e \in E \cap (S \times T)} f(e) - \sum_{e \in E \cap (T \times S)} f(e)$$
$$\leq \sum_{e \in E \cap (S \times T)} cap(e)$$
$$= cap(S)$$

For a saturated cut, the inequality is an equality.



Remarks.

- A saturated cut proves the optimality of a flow.
- To show: for every maximal flow there is a saturated cut proving its optimality.

Residual network

The residual network G_f for a flow f in G = (V, E) indicates the capacity unused by f. It is defined as follows:

- G_f has the same node set as G.
- For every edge e = (v, w) in G, there are up to two edges e' and e'' in G_f :
 - 1. if f(e) < cap(e), there is an edge e' = (v, w) in G_f with residual capacity r(e') = cap(e) f(e).
 - 2. if f(e) > 0, there is an edge e'' = (w, v) in G_f with residual capacity r(e'') = f(e).



Maximum flows and the residual graph

Theorem.

Let *f* be an (s, t)-flow, let G_f be the residual graph w.r.t. *f*, and let S be the set of all nodes reachable from *s* in G_f .

- If $t \in S$, then f is not maximum.
- If $t \notin S$, then S is a saturated cut and f is maximum.

Maximum flows and the residual graph (2)

Proof (part 1).

If *t* is reachable from *s* in G_f , then *f* is not maximal.

- Let p be a simple path from s to t in G_f .
- Let δ be the minimum residual capacity of an edge in p.
 By definition, r(e) > 0, for all edges e in G_f. Therefore, δ > 0.
- Construct a flow f' of value val $(f) + \delta$:

$$f'(e) = \begin{cases} f(e) + \delta, & \text{if } e' \in p \\ f(e) - \delta, & \text{if } e'' \in p \\ f(e), & \text{if neither } e' \text{ nor } e'' \text{ belongs to } p. \end{cases}$$

• f' is a flow and $val(f') = val(f) + \delta$.

Maximum flows and the residual graph (3)

Example.



Maximum flows and the residual graph (4)

Proof (part 2).

If t is not reachable from s in G_f , then f is maximal.

- Let S be the set of nodes reachable from s in G_f , and let $T = V \setminus S$.
- There is no edge (v, w) in G_f with $v \in S$ and $w \in T$.
- Hence
 - \triangleright f(e) = cap(e), for any $e \in E \cap (S \times T)$, and
 - ▷ f(e) = 0, for any $e \in E \cap (T \times S)$.
- Thus S is saturated and, by the Lemma, f is maximal.

Max-Flow-Min-Cut Theorem

Theorem.

The maximum value of a flow is equal to the minimum capacity of an (s, t)-cut:

```
\max\{\operatorname{val}(f) \mid f \text{ is a flow}\} = \min\{\operatorname{cap}(S, T) \mid (S, T) \text{ is an } (s, t) \text{-cut}\}
```

Ford-Fulkerson Algorithm

- **1**. Start with the zero flow, i.e., f(e) = 0, for all $e \in E$.
- 2. Construct the residual network G_f .
- 3. Check whether *t* is reachable from *s*.
 - if not, stop.
 - if yes, increase flow along an augmenting path, and iterate.



- Let |V| = n and |E| = m.
- Each iteration takes time O(n + m).
- If capacities are arbitrary reals, the algorithm may run forever.

Integer capacities

- Suppose capacities are integers, bounded by C.
- v^* := value of maximum flow can be up to (n-1)C.
- All flows constructed are integral (proof by induction).
- Every augmentation increases flow value by at least 1.
- Running time is $O((n + m)v^*) \rightarrow pseudo-polynomial.$

Edmonds-Karp Algorithm

- Compute shortest augmenting path, i.e., a shortest path from s to t in the residual network G_f, where each edge has distance 1.
- Apply, e.g., breadth-first search
- Resulting maximum flow algorithm can be implemented in $O(nm^2)$.

Bipartite matching

- G = (V, E) undirected graph
- *Matching*: Subset of edges $M \subseteq E$, no two of which share an endpoint.
- *Maximum matching*: Matching of maximum cardinality.
- *Perfect matching*: Every vertex in V is matched.
- *G* bipartite: $V = A \cup B$, $A \cap B = \emptyset$, and each edge in *E* has one end in *A* and one end in *B*.





Reduction to a network flow problem

- Add a source s and edges (s, a) for $a \in A$, with capacity 1.
- Add a sink t and edges (b, t) for $b \in B$, with capacity 1.
- Direct edges in *G* from *A* to *B*, with capacity 1.
- Integral flows f correspond to matchings M, with val(f) = |M|.
- Ford-Fulkerson takes time O((m + n)n), since $v^* \le n$.
- This can be improved to $O(\sqrt{n} m)$.

Illustration



Marriage theorem

Theorem (Hall).

A bipartite graph $G = (A \cup B, E)$, with |A| = |B| = n, has a perfect matching if and only if for all $B' \subseteq B$, $|B'| \le |N(B')|$, where N(B') is the set of all neighbors of nodes in B'.



Proof

• Let (S, T) be an (s, t)-cut in the corresponding network.

• Let
$$A_S = A \cap S$$
, $A_T = A \cap T$, $B_S = B \cap S$, $B_T = B \cap T$.

$$\begin{aligned} \operatorname{cap}(S,T) &= \sum_{e \in E \cap S \times T} \operatorname{cap}(e) \\ &= |A_T| + |B_S| + |N(B_T) \cap A_S| \\ &\geq |N(B_T) \cap A_T| + |N(B_T) \cap A_S| + |B_S| \\ &= |N(B_T)| + |B_S| \\ &\geq |B_T| + |B_S| = |B| = n \end{aligned}$$

By the max-flow min-cut theorem, the maximum flow is at least *n*.

Illustration





- G = (V, E) undirected graph
- $C \subseteq V$ is a vertex cover if every edge of G has at least one end in C.
- Lemma: For any matching M and any vertex cover C, we have $|M| \leq |C|$.
- **Theorem (König).** For a bipartite graph *G*,

 $\max\{|M| : M \text{ a matching }\} = \min\{|C| : C \text{ a vertex cover }\}.$

Network connectivity

- G = (V, E) directed graph, $s, t \in V, s \neq t$.
- Theorem (Menger). The maximum number of arc-disjoint paths from s to t equals the minimum number of arcs whose removal disconnects all paths from node s to node t.
- Theorem (Menger). The maximum number of node-disjoint paths from s to t equals the minimum number of nodes whose removal disconnects all paths from node s to node t.

Duality in linear programming

Primal problem

$$z_P = \max\{ c^T x \mid Ax \le b, x \in \mathbb{R}^n \}$$
 (P)

Dual problem

$$w_D = \min\{\mathbf{b}^T u \mid \mathbf{A}^T u = \mathbf{c}, u \ge 0\}$$
(D)

General form

	(P)			(D)	
min	c ^T x		max	и ^т b	
w.r.t.	$A_{j*}x \geq b_j,$	$i \in M_1$	w.r.t	$u_i \ge 0$,	$i \in M_1$
	$A_{j*}x \leq b_j,$	$i \in M_2$		$u_i \leq 0$,	$i \in M_2$
	$A_{j*}x = b_j,$	$i \in M_3$		u _i free,	$i \in M_3$
	$x_j \ge 0$,	$j \in N_1$		$(A_{*j})^T u \leq c_j,$	$j \in N_1$
	$x_j \leq 0$,	$j \in N_2$		$(A_{*j})^T u \geq c_j,$	$j \in N_2$
	<i>x_j</i> free,	$j \in N_3$		$(A_{*j})^T u = c_j,$	$j \in N_3$

Duality theorems

• Weak duality If x^* is primal and u^* is dual feasible, then

$$c^T x^* \leq z_P \leq w_D \leq b^T u^*.$$

- Strong duality If both (P) and (D) have a finite optimum, then $z_P = w_D$.
- Only four possibilities
 - 1. z_P and w_D are both finite and equal.
 - **2.** $z_P = +\infty$ and (D) is infeasible.
 - 3. $w_D = -\infty$ and (P) is infeasible.
 - 4. (P) and (D) are both infeasible.

Maximum flow and duality

Primal problem

$$\begin{array}{ll} \max & \sum\limits_{e: \text{source}(e) = s} x_e - \sum\limits_{e: \text{target}(e) = s} x_e \\ \text{s.t.} & \sum\limits_{e: \text{target}(e) = v} x_e - \sum\limits_{e: \text{source}(e) = v} x_e = 0, \quad \forall v \in V \setminus \{s, t\} \\ & 0 \leq x_e \leq c_e, \qquad \forall e \in E \end{array}$$

Dual problem

$$\begin{array}{ll} \min & \sum\limits_{e \in E} c_e y_e \\ \text{s.t.} & z_w - z_v + y_e \geq 0, \quad \forall e = (v, w) \in E \\ & z_s = 1, z_t = 0 \\ & y_e \geq 0, \qquad \quad \forall e \in E \end{array}$$

Maximum flow and duality (2)

- Let (y^*, z^*) be an optimal solution of the dual.
- Define $S = \{v \in V \mid z_v^* > 0\}$ and $T = V \setminus S$.
- (S, T) is a minimum cut.
- Max-flow min-cut theorem is a special case of linear programming duality.

Total unimodularity

- A matrix A is totally unimodular if each subdeterminant of A is 0, +1 or -1.
- Theorem (Hoffman and Kruskal). A ∈ Z^{m×n} is totally unimodular iff the polyhedron P = {x ∈ ℝⁿ | Ax ≤ b, x ≥ 0} is integral, i.e., P = conv(P ∩ Zⁿ), for any b ∈ Z^m.
- Corollary. $A \in \mathbb{Z}^{m \times n}$ is totally unimodular iff for any $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$ both optima in the LP duality equation

$$\max\{c^T x \mid Ax \leq b, x \geq 0\} = \{\min b^T u \mid A^T u \geq c, u \geq 0\}$$

are attained by integral vectors (if they are finite).

 Proposition. The constraint matrix A arising in a maximum flow problem is totally unimodular.

References

- K. Mehlhorn: Data Structures and Efficient Algorithms, Vol. 2: Graph Algorithms and NP-Completeness, Springer, 1986, http://www.mpi-sb.mpg.de/ ~mehlhorn/DatAlgbooks.html
- R. K. Ahuja, T. L. Magnanti and J. L. Orlin: Network flows. Prentice Hall, 1993
- S. Krumke and H. Noltemeier: Graphentheoretische Konzepte und Algorithmen. Teubner, 2005