Flow notes

Diego Hurtado de Mendoza

Let $G(V, E)$ be a Graph

Flows

An $(s - t)$ flow is a function $f: E \to \mathbb{R}$ that satisfies the following conditions:

- **Conservation constraint**: $\forall\, u \in V - \{s, t\}$,
$$\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u)$$
- **Capacity constraint**: $\forall\, (u, v) \in E,\ 0 \leq f(u, v) \leq c(u, v)$

The value of the flow is $|f| = \sum_{u \in V} f(s, u) - \sum_{u \in V} f(u, s)$ and the problem is to find the flow with maximum value. It's not hard to see that also $|f| = \sum_{u \in V} f(u, t) - \sum_{u \in V} f(t, u)$.

To make the implementation easier we usually use an equivalent formulation:

- **Skew symmetry:** $\forall\, (u, v) \in E,\ f(v, u) = -f(u, v)$
- **Conservation constraint**: $\forall\, u \in V - \{s, t\}$,
$$\sum_{v \in V} f(u, v) = 0$$
- **Capacity constraint:** $\forall\, (u, v) \in E$,
$$f(u, v) \leq c(u, v), \qquad f(v, u) \leq c(u, v)$$

Properties:

- **Flow Decomposition Theorem:** Any $(s - t)$ flow can be written as a linear combination of directed $(s - t)$ simple paths and directed cycles.

- **Uniqueness condition:** A maximum $(s-t)$ flow is unique iff the residual graph is acyclic
- Ford-Fulkerson algorithm may not terminate with irrational capacities. Edmonds-Karp and Dinic's algorithm work fine with irrational capacities.

Cuts

A cut (S,T) is a partition of V into S and $T = V - S$ such that s (source) belongs to S and t (sink) belongs to T.

The capacity of the cut is

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$

Algorithm to find minimum cut

Find maximum flow and define S = {all vertices such that there exists a path from them to s in the final residual network} and $T = V - S$. Then (S,T) will be a minimum cut.

Properties:

- If (u,v) is part of any minimum edge cut, then (v,u) is part of no minimum edge cut.
- Let's (S,T), (S',T') be two minimum cuts. Then $(S \cap S', T \cup T')$ and $(S \cup S', T \cap T')$ are also minimum cuts
- Let f be a maximum flow. Let C_s be the vertices reachable from s in the residual network of f. Let C_t be the vertices that can reach t in the residual network of f. Then $(C_s, V - C_s)$ and $(V - C_t, C_t)$ are minimum cuts, and for any minimum cut $(C, V - C)$ it holds that $C_s \subseteq C \subseteq V - C_t$
- **Uniqueness condition:** A maximum $(s-t)$ flow is unique iff $C_s = V - C_t$

Unbalanced flow (a.k.a. Flow with supplies/demands)

We have a network G and we want to find a **feasible** flow that satisfies the **capacity constraint**, but now the conservation constraint will be transformed into a **balance constraint**. Each node $u \in V$ will have a balance function $b: V \to \mathbb{R}$.

- **Balance constraint**: $\forall\, u \in V$,
$$\sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) = b(u)$$
- **Capacity constraints**: $\forall\, (u,v) \in E,\ 0 \leq f(u,v) \leq c(u,v)$

Positive balances represent supplies and we can see these nodes as sources; similarly, negative balances represent demands and we can see these nodes as sinks. Nodes with zero balance are intermediate nodes.

A necessary condition for the existence of a feasible flow is that

$$\sum_{u \in V} b(u) = 0$$

To solve this problem, add additional nodes s, t and add the following extra edges:

- (s,u) for every u with $b(u) > 0$ with capacity $b(u)$
- (u,t) for every u with $b(u) < 0$ with capacity $-b(u)$

Then find a maximum flow from s to t. This flow will be valid if and only if is a saturating flow (i.e. its value equals the sum of supplies or the sum of demands).

$$|f| = \sum_{u \in V, b(u) > 0} b(u) = \sum_{u \in V, b(u) < 0} -b(u)$$

Maximum Unbalanced flow

We have a network G and two nodes s, t we want to find the **maximum** flow that satisfies the **capacity constraint** and **balance constraint**. Each node $u \in V - \{s, t\}$ will have a balance function $b: V \to \mathbb{R}$.

- **Balance constraint**: $\forall\, u \in V - \{s, t\}$,
$$\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u) = b(u)$$
- **Capacity constraints**: $\forall\, (u, v) \in E,\ 0 \leq f(u, v) \leq c(u, v)$

The value of a flow is defined as $|f| = \sum_{u \in V} f(s, u) - \sum_{u \in V} f(u, s)$.

Notice that now it can happen that

$$\sum_{u \in V} f(s, u) - \sum_{u \in V} f(u, s) \neq \sum_{u \in V} f(t, u) - \sum_{u \in V} f(u, t)$$

Because now we have

$$\sum_{u \in V} f(s, u) - \sum_{u \in V} f(u, s) = \sum_{u \in V} f(t, u) - \sum_{u \in V} f(u, t) - \sum_{u \in V - \{s,t\}} b(u)$$

To solve this problem we can set $b(s) = 0, b(t) = 0$ and add an artificial vertex z with $b(z) = -\sum_{u \in V - \{s,t\}} b(u)$ and the edges $(z, s), (t, z)$ with infinite capacity.

Intuitively this node will make the sum of balance of all nodes equal to zero and the flow that passes through (z, s) will be the value of the (s, t) flow. We can find any feasible flow f' in the new graph (with the algorithm of the previous section) and convert it to an (s, t) flow f in the original graph. Now every node in the graph have its correct balance function so we can know forget about it and we reduce it to the classic maximum flow algorithm where we just have to push flow along the augmenting paths from s to t. If we want **minimum flow** we just have to push augmenting paths from t to s.

Unbalanced flow with lower bounds

We will generalize the previous problem and now each edge will have a lower bound (in addition to the upper bound a.k.a. capacity). Again, we have a network G and we want to find a **feasible** flow that satisfies the **balance constraint** and the new **capacity constraint**.

- **Balance constraint**: $\forall\, u \in V$,
$$\sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) = b(u)$$
- **Capacity constraint**: $\forall\, (u,v) \in E,\ \ l(u,v) \leq f(u,v) \leq c(u,v)$

Notice that it may possible for $l(u,v)$ or $c(u,v)$ to be negative. Let's transform it another graph with non-negative lower/upper bounds. We have three case for each edge:

- $l(u,v) \geq 0$, then the edge it's not modified
- $c(u,v) < 0$, then we replace it with an edge (v,u) with $c(v,u) = -l(v,u)$ and $\ l(v,u) = -c(u,v)$
- $l(u,v) < 0 \leq c(u,v)$, then we replace it with two edges. First we add (v,u) with $l(v,u) = 0$ and $r(v,u) = -l(u,v)$ and then we set $l(u,v) = 0$.

To solve the problem, for each edge $e \in E$ we set the flow $f_L(u,v) = l(u,v)$. Now our flow f_L satisfies the capacity constraints but it does not necessarily satisfy the balance constraint. So, we are going to add to it another flow f' such that $f_L + f'$ satisfy both constraints.

- **New balance constraint**: $\forall\, u \in V$,
$$\sum_{v \in V} f'(u,v) - \sum_{v \in V} f'(v,u) = b(u) + \sum_{v \in V} l(v,u) - \sum_{v \in V} l(u,v)$$
- **New capacity constraint:** $\forall\, (u,v) \in E$,
$$0 \leq f(u,v) \leq c(u,v) - l(u,v)$$

Finally, we can find f' using the previous solution for flow with supplies/demands.

Maximum Unbalanced flow with lower bounds

We have a network G and two nodes s, t and we want to find the maximum (s, t) flow that satisfies the **balance constraint** and the **capacity constraint** with lower bounds.

- **Balance constraint**: $\forall\ u \in V - \{s, t\}$,
$$\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u) = b(u)$$
- **Capacity constraint**: $\forall\ (u, v) \in E,\ \ l(u, v) \leq f(u, v) \leq c(u, v)$

The value of a flow is defined as $|f| = \sum_{u \in V} f(s, u) - \sum_{u \in V} f(u, s)$.

To solve this problem, similarly to the maximum unbalanced flow, we can set $b(s) = 0, b(t) = 0$ and add an artificial vertex z with $b(z) = -\sum_{u \in V - \{s,t\}} b(u)$ and the edges $(z, s), (t, z)$ with infinite capacity.

We now used the algorithm of Unbalanced flow with lower bounds of the previous section and we will have a feasible flow $f^* = f_L + f'$. So now we can ignore the lower bounds and the balance functions.

Finally, we will improve f' pushing flows through the augmenting path as in the classic maximum flow problem. But remember that the edges in f' will have capacity equal to $c(u, v) - l(u, v)$.

Coverings, Matching, Independent Set

Source: https://www.epfl.ch/labs/dcg/wp-content/uploads/2018/10/GT-4-Covers.pdf

Preliminaries

Bipartiteness:

A graph is bipartite if its vertices can be divided into two disjoints sets such that there is no edge between vertices of the same set.

Necessary and sufficient condition:

A graph is bipartite iff it doesn't have and odd cycle.

Definitions

- **Matching :** Is a set $M \subset E$ such that the edges in M are pairwise disjoint
- **Vertex Cover:** Is a set $C \subset V$ such that every edge of G is incident to a vertex of C.
- **Edge Cover:** Is a set $C \subset E$ such that every vertex of G is incident to an edge in C (this concept is only defined in graph without isolated vertex)
- **Independent set:** Is a set $I \subset V$ such that no two vertices in I are adjacent.

Inequalities

For any arbitrary Graph:

$$|maximum\ matching| \leq |minimum\ vertex\ cover|$$

For any arbitrary Graph without isolated vertices:

$$|maximum\ independent\ set| \leq |minimum\ edge\ cover|$$

Gallai Theorem:

For any arbitrary Graph:

$$|maximum\ independent\ set| + |minimum\ vertex\ cover| = |V|$$

For any arbitrary Graph without isolated vertices:

$$|maximum\ matching| + |minimum\ edge\ cover| = |V|$$

Konig Theorem:

Source: https://www.epfl.ch/labs/dcg/wp-content/uploads/2018/10/GT-3-Matchings.pdf

If the graph is bipartite,

$$|maximum\ matching| = |minimum\ vertex\ cover|$$

If, additionally, doesn't have isolated vertices,

$$|maximum\ independent\ set| = |minimum\ edge\ cover|$$

Hall's Theorem:

- **Definition:** A matching M "covers" $A \subset V$ if every vertex in A is an endpoint of an edge of the matching.
- **Definition:** $N(S)$ is the set of neighbours of each node of S

Theorem: Let G be a bipartite graph with bipartition $V = A \cup B$. Then G has a matching that covers A if and only if for all $S \subset A$ we have $|N(S)| \geq |S|$.

Algorithm for finding each of them in Bipartite Graph:

Let say that our bipartite graph G has the partition $V = L \cup R$

- **Maximum matching**: Run the max flow algorithm on G. All the edges between $L\ and\ R$ that have flow are edges of a maximum matching

- **Minimum edge cover**: Let denote the maximum matching size by |M|. Take the |M| edges of the maximum matching. For the other |V| - 2 |M| unmatched vertices, take one of its edges (the other endpoint must be matched). This set of edges is a minimum edge covering.

- **Minimum vertex cover**: Find a minimum cut (S, T) . Take all the edges of the cut (those that goes from S to T). All the vertices that belong to those edges (except from the source and the sink) form a minimum vertex cover.

(Source: http://theory.stanford.edu/~trevisan/cs261/lecture14.pdf)

- **Maximum Independent set:** Take all the vertices that are not in the minimum vertex cover. These vertices form a maximum independent set.

Vertex/Edge Connectivity

Menger's Theorem:

- Maximum number of edge-disjoint paths from s to t equal the minimum $s-t$ edge cut (minimum number of edges whose removal disconnects s and t)
- Maximum number of vertex-disjoint paths from s to t equal the minimum $s-t$ vertex cut (minimum number of vertices whose removal disconnects s and t)

Both of these statements are also a consequence of the max flow min cut theorem.

Partially Ordered Sets

Definitions:

- **Partial Order:** A (strict) partial order over a set V is a binary relation, $\prec$, over V that is:
  1. irreflexive: for all $x, y \in V$ and $x \neq y$, $x \prec y$ implies $y \not\prec x$
  2. transitive: for all $x, y, z \in V$, $x \prec y$ and $y \prec z$ implies $x \prec z$.

  Also, if $x \prec y$ or $y \prec x$, then we say that these elements are comparable; otherwise they are incomparable.

  We can represent a poset (partially ordered set) as a DAG.

- **Chain:** Is a subset of V such that every pair of elements is comparable
- **Antichain:** Is a set of V such that every pair of elements is incomparable.
  Note: A one element is both a chain and an antichain
- **Chain partition:** Is a partition of V (group of pairwise disjoint non-empty subsets of V) such that each subset is a chain.
- **Antichain partition:** Is a partition of V such that each subset is an antichain.
- **Height:** The size of the maximum chain
- **Width:** The size of the maximum antichain

Inequations:

$$|any\ chain| \leq |any\ antichain\ partition|$$

$$|any\ anti\ chain| \leq |any\ chain\ partition|$$

Mirsky's Theorem:

Statement: In a poset, it holds that

$$|maximum\ chain| = |minimum\ antichain\ partition|$$

That means that a poset of **height** H can be partitioned in H chains

Construction of the minimum antichain partition: Recursively remove the minimal (maximal) elements of the poset. Note that all minimal (maximal) elements at each iteration, form an antichain.

Minimal (maximal) elements in a DAG are the ones with outdegree (indegree) equals 0.

Construction of maximum chain: We can start with the nodes with indegree 0 and trying to pick the best choice of the chain using dp (or topological sorting).

Dilworth Theorem:

Inductive proof: https://pwp.gatech.edu/math3012openresources/lecture-videos/lecture-14/

Constructive proof: https://web.stanford.edu/class/cs361b/files/cs261-Jan2014-notes.pdf

Statement: In a poset, it holds that

$$|maximum\ anti\ chain| = |minimum\ chain\ partition|$$

That means that a poset of **width** W can be partitioned in W chains.

Also

$$|maximum\ matching| + |minimum\ chain\ partition| = |V|$$

$$|maximum\ matching| + |maximum\ antichain| = |V|$$

Construction:

Let's denote the DAG of the poset as $G(V, E)$

Let's construct the bipartite graph $G'(V', E')$ where

$V' = \{a_i, b_i \mid x_i \in V\}$, that means we create 2 nodes in G' for each node in G.

$E' = \{(a_i, b_j) \mid x_i \prec x_j \; in \; G\}$ that means that we create and edge in the G' for each pair of vertex in G such that x_i is an ancestor of x_j.

Construction of minimum chain partition:

First build the maximum matching in G' with max flow algorithm. Then map each edge of this matching with an edge in G. If you consider only the mapped edges in G , each connected component form a chain, and the union of all of them is the minimum chain partition.

Construction of maximum antichain:

First build the minimum vertex cover in G' using the nodes of the min cut. Then map each node of this vertex cover with a node in G (some may be repeated) and call this set S. Then the antichain is form by the set of vertex that is not in S.