Matching: Basic Definitions

- Given an undirected graph $G = (V, E)$ with weights w, a *matching* is a subset $M \subseteq E$ such that no two edges in M share a vertex. The *maximum-weight matching problem* is to find a matching M such that the sum of the weights of the edges in M is maximum over all possible matchings.

Matching: Basic Definitions

- Given an undirected graph $G = (V, E)$ with weights w, a *matching* is a subset $M \subseteq E$ such that no two edges in M share a vertex. The *maximum-weight matching problem* is to find a matching M such that the sum of the weights of the edges in M is maximum over all possible matchings.
- If all weights are 1, we get the *unweighted matching problem*, which asks for a matching of maximum cardinality.

Matching: Basic Definitions

- Given an undirected graph $G = (V, E)$ with weights w, a *matching* is a subset $M \subseteq E$ such that no two edges in M share a vertex. The *maximum-weight matching problem* is to find a matching M such that the sum of the weights of the edges in M is maximum over all possible matchings.
- If all weights are 1, we get the *unweighted matching problem*, which asks for a matching of maximum cardinality.
- Given a matching M in $G = (V, E)$, an edge $e \in E$ is *matched* if $e \in M$, and *free* if $e \in E - M$. A vertex v is *matched* if v has an incident matched edge, *free* otherwise.

Matching: Basic Definitions

- Given an undirected graph $G = (V, E)$ with weights w, a *matching* is a subset $M \subseteq E$ such that no two edges in M share a vertex. The *maximum-weight matching problem* is to find a matching M such that the sum of the weights of the edges in M is maximum over all possible matchings.
- If all weights are 1, we get the *unweighted matching problem*, which asks for a matching of maximum cardinality.
- Given a matching M in $G = (V, E)$, an edge $e \in E$ is *matched* if $e \in M$, and *free* if $e \in E - M$. A vertex v is *matched* if v has an incident matched edge, *free* otherwise.
- A *perfect matching* is a matching in which evrey vertex is matched.

Matching: Basic Definitions

- Given an undirected graph $G = (V, E)$ with weights w, a *matching* is a subset $M \subseteq E$ such that no two edges in M share a vertex. The *maximum-weight matching problem* is to find a matching M such that the sum of the weights of the edges in M is maximum over all possible matchings.
- If all weights are 1, we get the *unweighted matching problem*, which asks for a matching of maximum cardinality.
- Given a matching M in $G = (V, E)$, an edge $e \in E$ is *matched* if $e \in M$, and *free* if $e \in E - M$. A vertex v is *matched* if v has an incident matched edge, *free* otherwise.
- A *perfect matching* is a matching in which evrey vertex is matched.
- Given a matching M in $G = (V, E)$, a path in G is *alternating* (w.r.t. M) if it is simple (with no repeated vertices) and consists of alternating matched and free edges. An alternating path is an *augmenting path* if its endpoints are free.
- Similarly for *alternating cycles*.

Using augmenting paths

- Notation: For two sets A and B, let $A \oplus B$ denote their symmetric difference: $(A - B) \cup (B - A)$.

Using augmenting paths

- Notation: For two sets A and B, let $A \oplus B$ denote their symmetric difference: $(A - B) \cup (B - A)$.

Lemma

If M is a matching and P is an augmenting path, then $M \oplus P$ is another matching of cardinality $|M| + 1$.

Using augmenting paths

- Notation: For two sets A and B, let $A \oplus B$ denote their symmetric difference: $(A - B) \cup (B - A)$.

Lemma

If M is a matching and P is an augmenting path, then $M \oplus P$ is another matching of cardinality $|M| + 1$.

Lemma

M is a matching of maximum cardinality if and only if it has no augmenting path.

Finding augmenting paths in bipartite graphs

- Given an undirected bipartite graph $G = (U, V, E)$ and a matching M, let $D(G, M)$ be the directed graph formed from G by orienting an edges from U to V if it is in M, and from V to U otherwise.

Finding augmenting paths in bipartite graphs

- Given an undirected bipartite graph $G = (U, V, E)$ and a matching M, let $D(G, M)$ be the directed graph formed from G by orienting an edges from U to V if it is in M, and from V to U otherwise.
- Let F be the set of free vertices.

Finding augmenting paths in bipartite graphs

- Given an undirected bipartite graph $G = (U, V, E)$ and a matching M, let $D(G, M)$ be the directed graph formed from G by orienting an edges from U to V if it is in M, and from V to U otherwise.
- Let F be the set of free vertices.

Lemma

The set of augmenting paths w.r.t. M are in one-to-one correspondence with paths in $D(G, M)$ going from $F \cap V$ to $F \cap U$.

Finding augmenting paths in bipartite graphs

- Given an undirected bipartite graph $G = (U, V, E)$ and a matching M, let $D(G, M)$ be the directed graph formed from G by orienting an edges from U to V if it is in M, and from V to U otherwise.
- Let F be the set of free vertices.

Lemma

The set of augmenting paths w.r.t. M are in one-to-one correspondence with paths in $D(G, M)$ going from $F \cap V$ to $F \cap U$.

Corollary

An augmenting path w.r.t. a matching in a bipartite graph, if it exists, can be found in time $O(m + n)$.

Notation: We always use m to denote $|E|$ and n to denote the number of vertices in a graph.

A naïve algorithm for unweighted bipartite matching

- The Algorithm: Start with an empty set M. Keep finding an augmenting path P w.r.t. M and updating M to $M \oplus P$, until no augmenting path can be found.

A naïve algorithm for unweighted bipartite matching

- The Algorithm: Start with an empty set M. Keep finding an augmenting path P w.r.t. M and updating M to $M \oplus P$, until no augmenting path can be found.
- Runtime: $O(mn)$ (assuming $m \geq \frac{n}{2}$).

A faster algorithm: Hopcroft and Karp

Lemma

Given a matching M in a bipartite graph, if the maximum matching is of size $|M| + k$, then there exist k vertex-disjoint augmenting paths w.r.t. M. Further, one of these paths is of length at most $\frac{n}{k}$, where n is the number of nodes.

A faster algorithm: Hopcroft and Karp

Lemma

Given a matching M in a bipartite graph, if the maximum matching is of size $|M| + k$, then there exist k vertex-disjoint augmenting paths w.r.t. M. Further, one of these paths is of length at most $\frac{n}{k}$, where n is the number of nodes.

Definition

Given a mathing M, a set $\mathcal{P}$ of augmenting paths is a *blocking set of augmenting paths* if

- all paths in $\mathcal{P}$ are of the same length, ℓ;
- all paths in $\mathcal{P}$ are vertex disjoint;
- there exists no augmenting path of length smaller than ℓ;
- any augmenting path of length ℓ must share a node with a path in $\mathcal{P}$.

A faster algorithm: Hopcroft and Karp

Lemma

Given a matching M in a bipartite graph, a blocking set of augmenting paths can be found in linear time.

A faster algorithm: Hopcroft and Karp

Lemma

Given a matching M in a bipartite graph, a blocking set of augmenting paths can be found in linear time.

Theorem (Hopcroft and Karp)

There is an $O(m\sqrt{n})$ algorithm for unweighted maximum bipartite matching.

A faster algorithm: Hopcroft and Karp

Lemma

Given a matching M in a bipartite graph, a blocking set of augmenting paths can be found in linear time.

Theorem (Hopcroft and Karp)

There is an $O(m\sqrt{n})$ algorithm for unweighted maximum bipartite matching.

The algorithm: Start with $M = \emptyset$. Find a block set of augmenting paths of minimum lengths. Augment M with the paths. Repeat, until no augmenting paths can be found.

Lemma

After each round of the algorithm, the minimum length of an augmenting paths increases by at least 2*.*

Analysis of Hopcroft and Karp's Algorithm

Lemma

After each round of the algorithm, the minimum length of an augmenting paths increases by at least 2.

Corollary

The algorithm will end after at most $2\sqrt{n}$ *rounds of finding blocking sets of augmenting paths.*

Analysis of Hopcroft and Karp's Algorithm

Lemma

After each round of the algorithm, the minimum length of an augmenting paths increases by at least 2.

Corollary

The algorithm will end after at most $2\sqrt{n}$ rounds of finding blocking sets of augmenting paths.

Lemma

Given a matching M, let p be an augmenting path of minimum length w.r.t. M; let M' be the matching $M \oplus p$. Let q be any augmenting path of M'. Then $|q| \geq |p| + 2|p \cap q|$.