# 4 Bipartite Matchings

*4.1 Introduction • 4.2 Maximum Weight Matchings • 4.3 Minimum Weight Perfect Matchings*

## 4.1 Introduction

The graphs $G$ in which we look for matchings will be undirected, sometimes weighted, sometimes not.
Recall that a *matching* in $G$ is an $M \subset E(G)$ such that $m_1 \cap m_2 = \emptyset$ for all $m_1, m_2 \in M$. In other words, a set of edges that don't touch each other at any endpoint. A matching is *perfect* if $\cup_{m \in M} m = V(G)$, i.e. if every vertex is an endpoint of some edge of the matching (it is 'matched' or 'covered').

- Maximum Cardinality Matching Problem: In an unweighted graph, find a maximum with the maximum number of edges.
- Maximum/Minimum Weight Matching Problem: In a weighted graph, find a matching with maximum/minimum weight.
- Maximum/Minimum Weight Perfect Matching Problem: In a weighted graph, find a perfect matching, if it exists, such that its weight is maximum/minimum among all perfect matchings.

- Note that looking for a minimum cardinality matching would be silly. The maximum cardinality matching problem is a special case of the maximum weight matching problem, where all weights are 1.
- For the weighted problems, the maximum and minimum versions are equivalent: one can be turned into the other by multiplying all weights by $-1$.
- When looking for a maximum weight matching, one can ignore edges with negative weight, because they would never be included. For a minimum weight watching, one ignored positive-weight edges.
-The max/min weight matching and max/min weight perfect matching problems are equivalent; you will be asked to prove this in the problem set.

Let's right away consider the corresponding integer program and the dual of its relaxation:

**IP for maximum weight matchings**

$$\text{maximize} \sum_{e \in E(G)} w_e x_e \quad \text{with } x \geq 0, \ \boxed{x \in \mathbb{Z}^{|E|}},$$
$$\sum_{e \ni v} x_e \leq 1 \quad \text{for } v \in V.$$

**Dual of relaxation**

$$\text{minimize} \sum_{v \in V} y_v \quad \text{with } y \geq 0,$$
$$y_u + y_v \geq w_e \quad \text{for } uv \in E.$$

**Theorem 4.1.** *If the graph is bipartite, then optimal solutions of the integer program above are also optimal for the relaxation.*

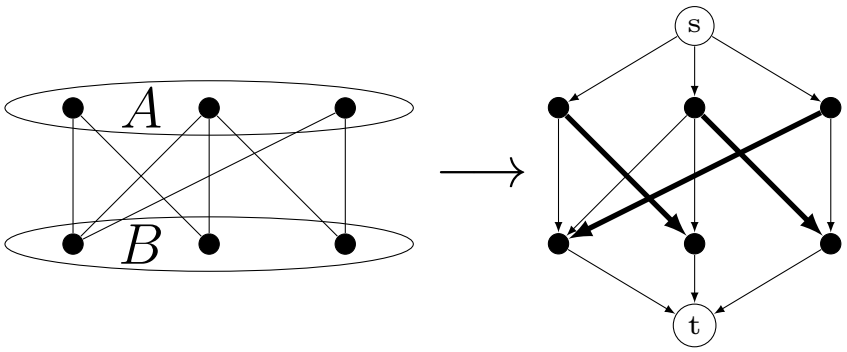
We will see a proof of this in a later lecture. We get the maximum cardinality matching problem if we choose all $w_e = 1$. An integral optimal dual solution, which must have all values 0 or 1, is then called a *vertex cover*: a set of vertices such that every edge has at least one of them as endpoint. The following standard theorem from graph theory now follows directly from the duality theorem:

**Theorem 4.2** (König)**.** *In a bipartite graph the maximum cardinality of a matching equals the minimum cardinality of a vertex cover.*

## 4.2 Bipartite Maximum Cardinality Matchings

We will first consider bipartite graphs, for which it is considerably easier to find matchings. So assume in this section that $G$ is bipartite, with partite sets $A$ and $B$. We will focus on the maximum weight matching problem.

One can reduce the maximum cardinality matching problem to a maximum flow problem, as follows. Direct all edges of $G$ from $A$ to $B$, and add vertices $s$ and $t$, with an edge from $s$ to all vertices of $A$, and an edge from each vertex of $B$ to $t$. Give all edges capacity 1.
Here is an example of what it looks like, with the capacities left out, and a matching with thick edges.



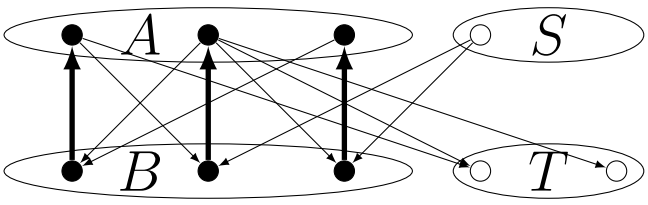

Now let $f$ be an integral $st$-flow in this graph, and let $M = \{e \in E(A,B) : f(e) = 1\}$. Then $M$ is a matching: if $f(ab) = 1$ and $f(ab') = 1$, then $f$ would fail the flow constraint at $a$:

$$\sum_{e \in \delta^{\text{out}}(a)} f(e) \geq f(ab) + f(ab') > f(sa) = \sum_{e \in \delta^{\text{in}}(a)} f(e).$$

Conversely, given any matching $M$, we get a flow defined by $f(e) = 1$ for $e \in M$, $f(e) = 0$ for $e \notin M$; $f(sa) = 1$ if there is an $ab \in M$, $f(sa) = 0$ otherwise; and $f(bt) = 1$ if there is an $ab \in M$, $f(bt) = 0$ otherwise.
So integral flows in this graph correspond exactly to matchings, and the value of a flow equals the cardinality of the corresponding matching,

$$\sum_{e \in \delta^{\text{out}}(a)} f(e) = |\{a \in A : \exists ab \in M\}| = |M|.$$

Hence the augmenting path algorithm for maximum flows gives a polynomial algorithm for maximum cardinality matchings (we do not have to worry about irrational capacities, since all capacities are 0 or 1). Actually, we could even use the edge-disjoint path algorithm that we saw.
Here is an example of what the auxiliary graph $G_M$, corresponding to a matching $M$ during the algorithm, will look like, ignoring $s$ and $t$:




The algorithm will look for a path from $S$ to $T$, because that is the only way from $s$ to $t$, since the edges from $s$ to matched vertices in $A$ are at capacity, as are the edges from matched

vertices in $B$ to $t$. That path will be *alternating*, in the sense that it alternates matching-edges and non-matching-edges, with one more non-matching than matching. Given such an alternating $ST$-path, the algorithm will augment on it, which means it will swap matching and non-matching edges on the path, increasing their number.
For instance, there is a path from $S$ to the second vertex of $B$, to the second vertex of $A$, to $T$. Augmenting along it will replace its one matching edge with the two other edges.
Let's state the resulting algorithm.

**Augmenting path algorithm for bipartite maximum cardinality matchings**

1. Set $M = \emptyset$ and $V(G_M) = V(G)$;
2. Set $E(G_M) = \{ba : \{a, b\} \in M\} \cup \{ab : \{a, b\} \notin M\}$;
   Let $S = A \backslash (\cup M)$, $T = B \backslash (\cup M)$;
3. Find any $ST$-path $Q$ in $G_M$; if none exists, go to 5;
4. Augment along $Q$: $M := M \Delta E(Q)$; go back to 2;
5. Return $M$.

Just for practice, and because we'll need the ideas below, let's prove correctness:

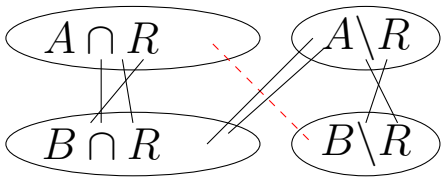
**Theorem 4.3.** *The augmenting path algorithm returns a maximum cardinality matching $M$ in the bipartite graph $G$.*

*Proof.* We will show that there is a vertex cover $C$ with $|C| = |M|$, which proves by duality that $M$ is maximum.
Given $S$ and $T$, the sets of unmatched vertices, define

$$R = \{v \in V(G) : \exists \text{ a path in } G_M \text{ from } S \text{ to } v\},$$

and consider

$$C = (A \backslash R) \cup (B \cap R).$$




We claim that $C$ is a vertex cover in $G$. If not, then there is an edge not touching $C$, which means it is an edge $ab$ between $A \cap R$ and $B \backslash R$. It cannot be in the matching, for then it would be oriented $ba$ in $G_M$ and the only edge of $G_M$ that enters $a$, so $a$ would not be in $R$ since $b$ is not. But since it is not in the matching, it is oriented $ab$, and it would provide an augmenting path by extending the path from $S$ to $a$.
Now for $|C| = |M|$. Duality (or König) right away gives $|M| \leq |C|$. To prove that $|C| \leq M$, we show that every $v \in C$ is matched by a different $m \in M$. We have $S \subseteq A \cap R$ and $T \subseteq B \backslash R$, so every $v \in C$ is matched by some $m \in M$. And no $a, b \in C$, so $a \in A \backslash R$ and $b \in B \cap R$, can be matched by the same $m \in M$, since that would imply $a \in R$, a contradiction. □

## 4.3 Bipartite Maximum Weight Perfect Matching

In the weighted cases, it turns out to be a bit more convenient to formulate the algorithm for perfect matchings. So we need the corresponding linear programs, which only differ in that there is equality in the primal constraints (and hence $y$ need not be nonnegative).

**LP for maximum weight perfect matchings**

$$\text{maximize} \sum_{e \in E(G)} w_e x_e \quad \text{with } x \geq 0,$$
$$\sum_{e \ni v} x_e = 1 \quad \text{for } v \in V.$$

**Dual**

$$\text{minimize} \sum_{v \in V} y_v \quad \text{with } y \in \mathbb{R}^{|V|},$$
$$y_u + y_v \geq w_{uv} \quad \text{for } uv \in E.$$

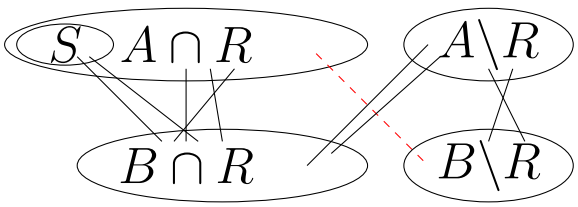
We will make use of the Complementary Slackness Theorem, which we state here specifically for matchings (it's not hard to prove directly, and you will be asked to do so in the problem set). Note that if we didn't have equality in the primal constraint, then there would be more conditions (that if $y_v = 0$ then the corresponding primal constraint is tight); this is why perfect matchings are more convenient here.

**Lemma 4.4.** *Let $x$ be a feasible primal solution and $y$ a feasible dual solution to the linear programs for maximum weight perfect matchings.*
*If $x_{ab} = 0$ whenever $y_a + y_b > w_{ab}$, then both are optimal solutions.*

We first describe the algorithm informally.
Given a feasible dual solution $y$, we will look at the graph $G_y$ of tight edges, defined by $V(G_y) = V(G)$ and

$$E(G_y) = \{ab \in E(G) : y_a + y_b = w_{ab}\}.$$

We find a maximum cardinality matching $M_y$ using the algorithm above. If it is perfect in $G_y$, then it will also be a perfect matching in $G$. Since it is complementary to $y$, both must be optimal by the lemma, so $M_y$ is a maximum weight perfect matching.
If $M_y$ is not perfect, we will try to improve $y$ by lowering its value, in such a way that at the same time we create new tight edges. I.e. we make at least one new edge satisfy $y_a + y_b = w_{ab}$, which will hopefully make it possible to find a larger matching among the tight edges.
To make this work we use ideas from the augmenting path algorithm and its correctness proof above. We will use the directed graph $G_y$, with matching edges directed from $B$ to $A$, and non-matching edges directed from $A$ to $B$. And we again consider the set $R$ of vertices reachable in $G_y$ from its uncovered vertices in $A$. This gives a configuration like this in $G_y$:




As we saw before, $C_R = (A \backslash R) \cup (B \cap R)$ is then a vertex cover in $G_y$, so there are no tight edges between $A \cap R$ and $B \backslash R$. That allows us to lower all the $y_a$'s in $A \cap R$, while simultaneously increasing the $y_b$'s in $B \cap R$, in such a way that at least one of the previously-slack edges between $A \cap R$ and $B \backslash R$ will become tight.
We will state the algorithm, and then prove in more detail that it works. For convenience,

we assume that $G$ has a perfect matching, which also implies that $|A| = |B|$. We could easily check this before starting the algorithm.

**Primal-dual algorithm for bipartite maximum weight perfect matchings**

1. Set $M_y = \emptyset$ and $V(G_y) = V(G)$;

   Define $y$ by $y_a = 0$ for $a \in A$, $y_b = \max_{e \in \delta(b)} w(e)$ for $b \in B$;

2. Set $E(G_y) = \{ab \in E(G) : y_a + y_b = w_{ab}\}$;

   Find a maximum cardinality matching $N$ in $G_y$;

   if $|N| > |M_y|$, set $M_y := N$ (else leave $M_y$ unchanged);

   if $M_y$ is perfect in $G$, go to 5; if not, set $S = A \backslash (\cup M)$;

3. Orient each $ab \in E(G_y)$ from $B$ to $A$ if it is in $M_y$, from $A$ to $B$ otherwise;

   Set $R = \{v \in V(G) : \exists$ a dipath in $G_y$ from $S$ to $v\}$;

   Compute $\delta = \min_{ab \in E_G(A \cap R, B \backslash R)} (y_a + y_b - w_{ab})$;

4. Augment by $\delta$: $\begin{cases} y_a := y_a - \delta \text{ for } a \in A \cap R, \\ y_b := y_b + \delta \text{ for } b \in B \cap R; \end{cases}$

   go back to 2;

5. Return $y$, $M_y$.

Here is a simpler version of the algorithm that we could have tried: If there is no perfect matching in $G_y$, then find a set $X \subset A$ such that $N(X) < X$ (like our $A \cap R$); decrease all $y_a$ in $X$ by an appropriate $\delta > 0$ and increase all $y_b$ in $N(X)$ by $\delta$; then $\sum y_v$ will strictly decrease by a multiple of $\delta$ as above. If the weights were integral, then we would have $\delta \geq 1$ throughout, so the algorithm would terminate. But it would not be polynomial, at least not if the weights can be arbitrarily large.
This is why we need to use the non-perfect matching like we do above. To deal with non-integral weights and to prove polynomiality, we need the more detailed observation about $|M_y|$ and $|B \cap R|$ in the lemma below.
Also note that it is not true that the number of tight edges always increases (which would have made things simpler); as we'll see in the proof below, edges between $B \cap R$ and $A \backslash R$ may go from tight to slack.

**Lemma 4.5.** *The algorithm is well-defined. After an augmentation (step 4), $y$ is still feasible, its value $\sum_{v \in V} y_v$ has decreased, and either $|M_y|$ or $|B \cap R|$ has increased.*

*Proof.* Note that $S$ is nonempty, because we assumed $|A| = |B|$, so when a matching is not perfect, it must leave vertices unmatched on both sides. Also $A \cap R \neq \emptyset$, since it contains $S$, and $B \backslash R \neq \emptyset$, since it contains $T$ (the unmatched vertices in $B$, which satisfy $|T| = |S|$). The minimum exists because $E_G(A \cap R, B \backslash R) \neq \emptyset$: otherwise $A \cap R$ could never be perfectly matched, because $N(A \cap R) = B \cap R$, and below we will show $|B \cap R| < |A \cap R|$. And finally $\delta > 0$, since the edges between $A \cap R$ and $B \backslash R$ cannot be in $G_y$, otherwise they would create

an augmenting path. So the steps are indeed well-defined.
Let $y'$ be the new dual solution. For an edge $ab$ between $A \cap R$ and $B \cap R$, the constraint is unchanged:

$$y'_a + y'_b - w_{ab} = (y_a - \delta) + (y_b + \delta) = y_a + y_b \geq w_{ab}.$$

Edges between $A \backslash R$ and $B \backslash R$ are clearly unaffected. For edges $ab$ between $B \cap R$ and $A \backslash R$, only $\delta$ is added to $y_b$, which cannot break the constraint. It could lose tightness, but that will not be a problem. Finally, edges $ab$ between $A \cap R$ and $B \backslash R$ will not get their constraints broken because of the way we choose $\delta$, and at least one of these must become tight. This proves that the new $y'$ is feasible.
To show that $\sum y_v$ strictly decreases, we show that $|A \cap R| > |B \cap R|$. This follows from

$$|A \cap R| = |A| - |A \backslash R| = \frac{|V|}{2} - |A \backslash R| > |C_R| - |A \backslash R| = |B \cap R|,$$

where we used that $|C_R| = |M_y| < \frac{|V|}{2}$, which follows from the fact that $M_y$ is not perfect.
Suppose $|M_y|$ does not increase. Then by design of step 2, $M_y$ remains the same. Since we have created at least one new tight edge in $ab \in E(A \cap R, B \backslash R)$, $b$ will be in $B \cap R$ in the next step. No vertex can leave $B \cap R$, since the only edges that can lose tightness are in $E(B \cap R, A \backslash R)$, and removing these from $G_y$ cannot break any path from $S$, so this does not affect $R$. Hence $|B \cap R|$ will have increased by at least 1. □

**Theorem 4.6.** *The primal-dual algorithm terminates, is polynomial, and returns a maximum weight perfect matching.*

*Proof.* The returned matching is clearly perfect, and it is maximum since it is a matching in the last $G_y$, which implies that it satisfies complementary slackness with the feasible dual solution $y$.
The algorithm terminates in polynomial time by the last statement of the lemma: There can be at most $\frac{|V|}{2}$ steps in which $|M_y|$ increases, and in between those steps there can be at most $|B| = \frac{|V|}{2}$ steps in which $M_y$ remains the same but $|B \cap R|$ increases. So there are fewer than $|V|^2$ iterations, and each iteration is also polynomial: the only expensive things it does is the maximum cardinality matching algorithm to find $N$, and it does a breadth-first search to determine $R$. □

A few words on primal-dual algorithms in general. The algorithm above is a special case of a linear programming algorithm called *primal-dual*, which similary works with a feasible dual, tries to find a feasible primal that satisfies complementary slackness, and if it fails it tries to improve the feasible dual. For linear programming in general it is not polynomial, but it turns out to work very well for linear programs that correspond to combinatorial optimization problems.
Many of the algorithms that we have seen (including Dijkstra's algorithm and the maximum flow algorithm) can be derived from this general method, without needing much insight. But this derivation is somewhat tedious, which is why I have only shown the result for the example of weighted perfect matchings. In case you go on to study more advanced combinatorial optimization, you will definitely see more of the primal-dual method.