Maximum Matchings

1 The Concepts

Formally, a **graph** is a pair (V, E) where V is some set and E is a subset of $V \times V$. When dealing with a simple graph, E is usually further restricted to $\binom{V}{2}$ and is required to be symmetrically closed, i.e. for $a, b \in V$, $(a, b) \in E \Leftrightarrow (b, a) \in E$.

Informally, and for the purposes of what is described here, a graph is a drawing of dots (called vertices) and lines (called edges). A path is a traversal of a subset of the vertices via edges and a cycle is a path beginning and ending at the same vertex. The length of a path is the number of edges contained in it.

Definition 1. For a graph G and vertices u and v in G, u and v are said to be **adjacent**, denotes $u \sim v$, if there exists an edge between u and v.

Definition 2. For a graph G and a subset S of its vertices, the **neighbor set** $N_G(S)$ of S is the set of vertices in G with edges to at least one vertex in S.

Definition 3. A **bipartite graph** is a graph with no odd-length cycles.

It can be proven that the vertices of any bipartite graph can be partitioned into two branches A and B such that no edges are between two vertices of A or two vertices of B, as in figure 1, though we will take this for granted.

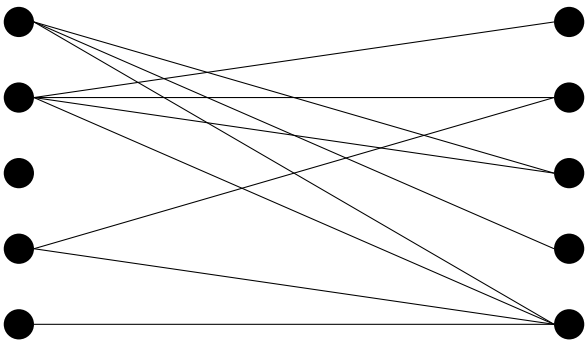

Figure 1: A bipartite graph

Informally, a **matching** in a graph is a collection of edges in the graph such that none of the chosen edges are both incident to a single vertex. An example of a matching of size 4 for the above graph is shown in figure 2.

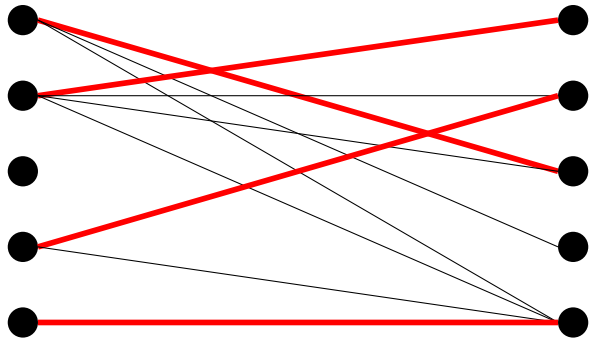

Figure 2: A bipartite graph with an indicated matching

For a bipartite graph, this is a matter of pairing up a subset of vertices in the left branch each with a distinct vertex from the right branch with which it has an edge. We call a matching a **perfect matching** if each vertex of the graph is incident to some edge in the matching. Thus, the graph above does not contain a perfect matching, while the graph in figure 3 does, as indicated.

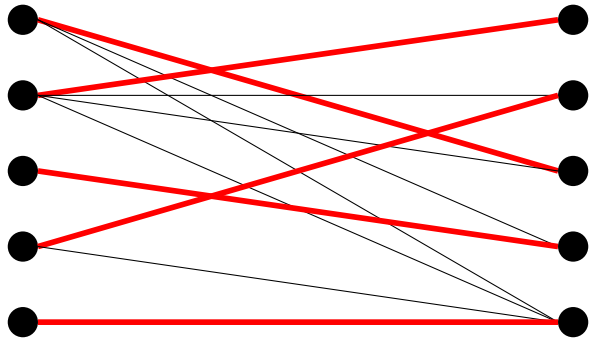

Figure 3: A bipartite graph with an indicated perfect matching

Definition 4. A **maximum matching** is a matching of largest size for a given bipartite graph.

For example, the matching given in figure 3 is a maximum matching (since it uses all vertices).

This project is concerned with an algorithm to find maximum matchings for bipartite graphs, to be discussed below; however, first we provide a related result.

Theorem 1 (Hall's Marriage Theorem)**.** *A bipartite graph G with branches A and B of equal size ($|A| = |B|$) has a perfect matching if and only if for every subset S of A, $|N_G(S)| \geq |S|$.*

In other words, for one of the directions, each subset of the left branch must have edges to more vertices in the right branch than its size. This condition is known as the "marriage condition".

Remark. If $|N_G(S)| \geq |S|$ for all $S \subseteq A$, then $|N_G(T)| \geq |T|$ for all $T \subseteq B$, i.e. having the marriage condition for the left branch implies having the marriage condition for the right branch, when $|A| = |B|$. Indeed, consider such $T \subseteq B$, so that $N_G(T) \subseteq A$. Then $N_G(A \setminus N_G(T)) \subseteq B \setminus T$, so $|N_G(A \setminus N_G(T))| \leq |B| - |T|$. Furthermore, from the marriage condition on the left branch, $|A \setminus N_G(T)| \leq |N_G(A \setminus N_G(T))|$. Then:

$$\begin{aligned} |A \setminus N_G(T)| &\leq |B| - |T| \\ \implies |A| - |N_G(T)| &\leq |B| - |T| \\ \implies |A| - |N_G(T)| &\leq |A| - |T| \\ \implies |T| &\leq |N_G(T)| \end{aligned}$$

We now prove Hall's Marriage Theorem.

Proof. The forward direction is simpler to prove and we show the contrapositive. If there exists $S \subset A$ such that $|N_G(S)| < |S|$, then it is clear that not all vertices of S can be matched, so that a perfect matching cannot exist.

For the backward direction, we proceed by induction on n, the common size of the branches. When $n = 1$, the branches are $A = \{a_1\}$ and $B = \{b_1\}$ and $|N_G(\{a_1\})| \geq |\{a_1\}| = 1$, so that a_1 is connected to b_1 and so the edge between them gives a perfect matching.

Now consider when $|A| = |B| = n \geq 2$. Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ and assume WLOG $a_1 \sim b_1$. If we delete a_1 and b_1 from the graph and maintain the marriage condition for the remaining graph, then we apply the induction hypothesis to get a perfect matching. Otherwise, upon deleting a_1 and b_1 to get a new graph G', there is a subset S of $A \setminus \{a_1\}$ such that the neighbor set of S in $B \setminus \{b_1\}$ has size less than $|S|$. Since the marriage condition was satisfied for G, this implies that $|N_G(S)| = |S|$ with one of the vertices in $N_G(S)$ being b_1, so that $|N_{G'}(S)| = |S| - 1$ and $|N_G(S)| = |S|$.

Now consider instead splitting the graph into $G^{(1)}$ composed of S and its neighbors in B (and all incident edges) and $G^{(2)}$ composed of the rest of G. (Note that $G^{(2)}$ is non-empty; it at least contains a_1.) Since all neighbors of vertices in S are within $G^{(1)}$, the original presence of the marriage condition for G implies the marriage condition for $G^{(1)}$.

As for $G^{(2)}$, recall from the remark above that since the left branch satisfies the marriage condition, the right branch also satisfies it. Since all vertices of $G^{(2)}$ in B only have edges to vertices of $G^{(2)}$ in A (when viewed in G), the marriage condition also holds for $G^{(2)}$ and the induction hypothesis applies here as well. □

Corollary 1. For a bipartite graph G with branches A and B (not necessarily with $|A| = |B|$), there exists a matching containing all of A (i.e. saturating A) if and only if for every subset S of A, $|N_G(S)| \geq |S|$.

Proof. As in the above proof, the forward direction is straightforward. For the backwards direction, note that for the condition to be met, $|A| \leq |B|$. If $|A| = |B|$, then apply the theorem above. Otherwise, add $|B| - |A|$ vertices to the A branch and connect them to all vertices of the B branch. The marriage condition is still met and we can now apply the above theorem to get a perfect matching. We can then remove the added vertices to get a matching covering A completely. □

Before turning to the algorithm, we require two additional definitions.

Definition 5. An **augmenting path** of a graph G with a matching M is a path P starting and ending at vertices not matched in M and such that every second edge of P is contained in M.

Notice that these restrictions on augmenting paths require that they are of odd length. Furthermore, the edges of the path P must alternate between not being present in M and being present in M. An example of an augmenting path is shown in figure 4 below, with the edges in alternating blue and green.

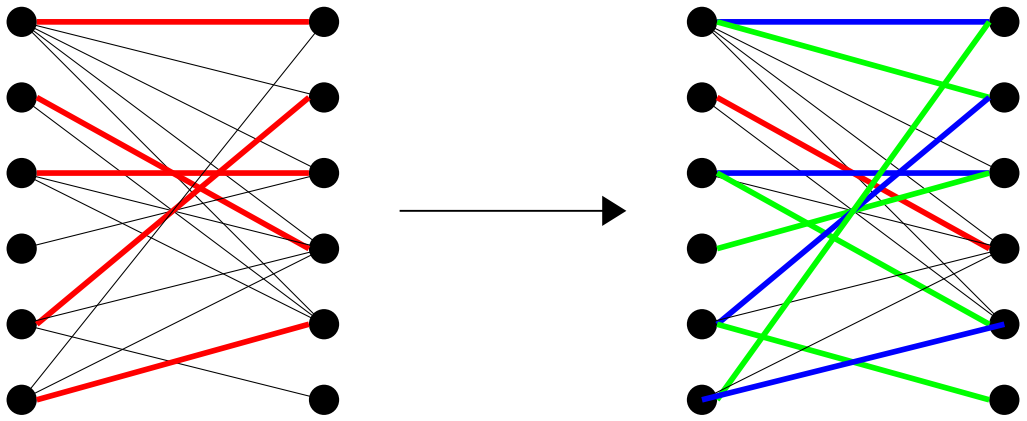

Figure 4: A bipartite graph with an augmenting path

By removing the edges from the matching that are on the augmenting path and adding the edges that were not in the matching to the augmenting path, we obtain a matching which has one extra edge, i.e. a larger matching, as continued in figure 5.

Definition 6. An **almost augmenting path** of a graph G with a matching M is a path P starting and ending at vertices in the left branch and such that every second edge is in M.

2 The Algorithm

For the purposes of determining a maximum matching in a given bipartite graph G, the basic structure of the algorithm is as follows:

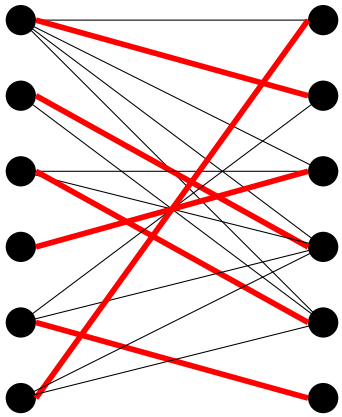

Figure 5: The above bipartite graph after utilizing an augmenting path.

1. Find an initial **maximal** matching.
2. Let U denote the set of unmatched vertices in the left branch and initialize the set S to U. S will be the set of vertices in A reachable from U along almost augmenting paths (we assume that the vertices of U themselves are almost augmenting paths). We repeatedly consider vertices u in B adjacent to a vertex in S. If such a vertex u is matched to a vertex v in A and $v \notin S$, add v to S. If u is unmatched, then proceed to step 3. If we run out of ways of expanding S and there is no unmatched vertex of B adjacent to a vertex of S, terminate the algorithm.
3. If u is not matched at all, then there exists an augmenting path from U to u. We can then toggle the edges of this augmenting path so that every other edge is part of the matching, beginning with the first. This increases the size of the matching by 1. We then return to step 2.

In step 1, by "maximal" we mean that the matching cannot be extended by the addition of any single edge. This can be achieved using a "greedy algorithm" where we start at the first vertex in the left branch and continue down the branch; at each vertex we look for the first edge emanating from it which is not incident to an edge of the matching so far. If no such edge emanates from the vertex, then we do not add any edge for it.

Remark. Such a maximal matching has the property that for any edge in the graph, at least one of the edge's incident vertices is contained in the matching. Otherwise, the edge itself could be added and the matching would not be maximal.

3 More Concepts

It is first worth noting that if a bipartite graph G has a perfect matching, then the algorithm will output a perfect matching as well.

Theorem 2. *The algorithm of section 2 gives rise to a perfect matching in a bipartite graph G if a perfect matching does in fact exist.*

Proof. The algorithm terminates when the set S cannot be expanded. S is composed of U and the vertices that were counted as being reachable from U by almost augmenting paths, i.e. the other $|S|-|U|$ vertices are matched with a set T of vertices of B (so $|S|-|U|=|T|$). Since S cannot be expanded and there is no vertex that is adjacent to a vertex of S and unmatched, all neighbors of vertices of S must be in T. Thus $|N_G(S)| \leq |T| = |S| - |U|$.

Under the assumption that a perfect matching has not been found, $|U| > 0$, so that $|N_G(S)| < |S|$, so that no perfect matching could exist anyway. □

Furthermore, the algorithm gives a maximum matching.

Theorem 3. *The algorithm of section 2 gives rise to a maximum matching.*

Proof. Here is a sketch. Assume for contradiction that the resulting matching M were not of maximum size. Then there exists a matching of larger size. Let M' be a maximum matching of G with a minimal number of edges which do not appear in M. Then there exists some vertex u_1 such that u_1 is matched in M' but not in M. Let its match in M' be v_1. If v_1 were unmatched in M, then the edge (u_1, v_1) would have been added to M, thus v_1 must be matched in M to some other vertex u_2. If u_2 is unmatched in M', then we can toggle the edges in the path from u_1 to u_2 in M' to get a maximum matching M'' with a smaller number of edges which do not appear in M. Thus, M' must match u_2 to some vertex v_2.

We continue in this way with adding additional vertices. Since the branches are of finite size, we must eventually find an unmatched vertex in B, so that M can be expanded, a contradiction. □