Chapter 1

Zachary Campbell

February 14, 2018

1 Bipartite graphs and matchings

Throughout this thesis we will be interested in a specific subclass of graphs known as bipartite graphs. Unless otherwise noted, our algorithms will assume a bipartite structure.

Definition (Bipartite graph). A *bipartite graph* is a graph whose vertices can be partitioned into two sets L and R such that all edges connect a vertex $l \in L$ to a vertex $r \in R$. We will denote this graph $G = (L, R, E)$, where E is the edge set.

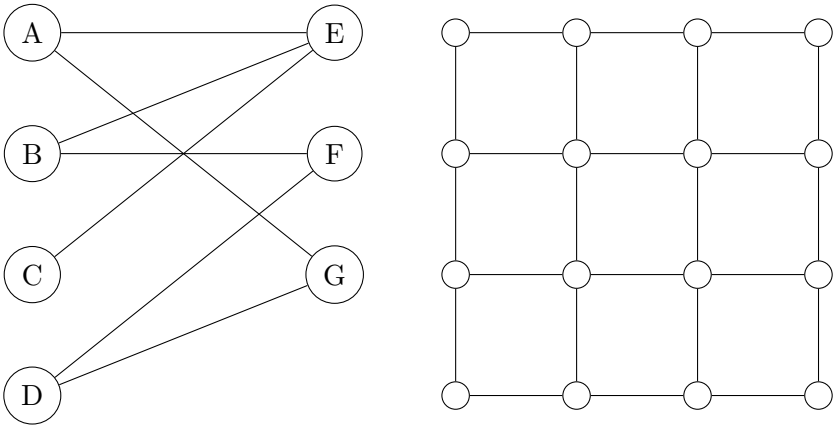


Figure 1: Examples of bipartite graphs

In Figure 1 we have a bipartite graph with vertex partition given by $L = \{A, B, C, D\}$ and $R = \{E, F, G\}$. All edges in this graph are between a node $l \in L$ and a node $r \in R$. The graph on the right is also bipartite. It may take a little more time to convince yourself that you can partition the vertices into disjoint L and R in a way that maintains the bipartite property. Try it!

Now that we know what we are working with, let's introduce a problem that we'd like to solve on these graphs. We now describe the notion of a matching on a graph.

Definition (Matching). Let $G = (V, E)$ be a graph. A subset $M \subset E$ is a *matching* if no two edges in M are incident to the same vertex.

We say that a vertex $v \in V$ is matched with respect to M if it is an endpoint of some edge in M. The following figure shows some examples of different matchings on one of the graphs from Figure 1.

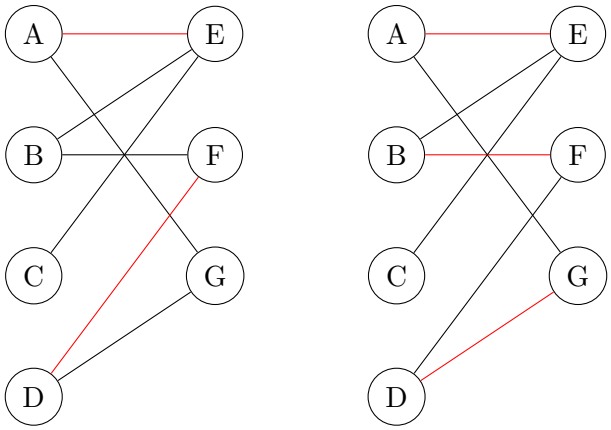


Figure 2: Examples of matchings on a bipartite graph

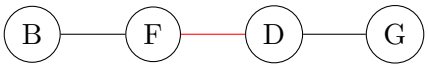
There many different valid matchings on the graph in Figure 2. Oftentimes, we want to find the largest matching on a graph. (ADD SENTENCE OR TWO ON MOTIVATION FOR MAXIMAL MATCHING.) This leads to the following definition.

Definition (Maximal matching). A *maximal matching* on G is a matching M such that if any other edge not in M is added to M, it is no longer a valid matching. Alternatively put, M is maximal if there is no matching $M^{'}$ such that $M \subset M^{'}$.

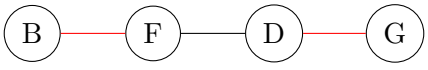
Both matchings in Figure 2 are maximal matchings; in each case there are no edges that we can add to M and have that M is still a matching. However, notice that the size of the matchings is different, even though both are maximal on G. This leads to the following definition.

Definition (Maximum matching). A matching M on a graph G is said to be a maximum matching if for all other matchings $M^{'}$ on G, $|M^{'}| \leq |M|$.

In our example, the matching on the right given by $M = \{(A, E), (B, F), (D, G)\}$ is a maximum matching (convince yourself). In general there may be many unique maximum matchings on a graph.
In this section we are interested in general methods for finding maximum matchings on bipartite graphs. One of the fundamental approaches is to look at certain subgraphs called alternating paths. Before we define what these are, let's look at a motivating example. Suppose we have the matching on the left in Figure 2. So $M = \{(A, E), (D, F)\}$. Consider the following sequence of vertices in the graph:



Call this sequence p. Let's perform an operation that we will denote $M \oplus p$, which operates like XOR: add to M each edge in p that isn't in M, and remove from M each edge in p that is in M. This gives us the following segment, where (B, F) and (D, G) are now in M, but (D, F) is not:



First, we must check that the new M is still a valid matching; we do this by noticing that B and G were originally unmatched, so it's okay for one of their incident edges to be added. Also, notice

that the size of our matching has grown by 1! In fact, this new matching is exactly the matching given by the graph on the right in Figure 2. This is a general technique in finding maximum matchings. We want to look for these paths that start and end at unmatched vertices, and whose edges are alternately matched and unmatched. If we can find one of these paths, we will be able to increase the size of matching. We define this formally now.

Definition (Alternating path). Let G be a graph and M some matching on G. An *alternating path* is a sequence of vertices and edges that begins with an unmatched vertex, and whose edges alternate between being in M and not in M.

Definition (Augmenting path). An *augmenting path* is an alternating path that starts and ends on unmatched vertices. When we augment M by an augmenting path p, we use the notation $M \oplus p$.

This motivates a general method for finding a maximum matching on a bipartite graph: just keep looking for augmenting paths, and augment the current matching by that augmenting path. Of course, we need to prove that this in fact gives us a maximum matching. The following theorem says exactly that.

Theorem (Berge, 1957). A matching M on G is a maximum matching if and only if G contains no augmenting paths with respect to M.

This gives us the following framework for finding maximum matchings in bipartite graphs.

```
ALG 1(G)
1  M := ∅
2  while there exists an augmenting path p
3        M := M ⊕ p
4  return M
```

Note that we have yet to describe the details of this algorithm. Before we do so, we are going to take a step back a bit and look at the maximum matching problem from a slightly different perspective. In doing so, we will develop a language for talking about this problem that will serve us throughout the rest of this thesis. At first, the approach will appear purely pedagogic, but hopefully the reader will understand the significance of it by the end of the thesis.

2 The vertex cover problem

We begin this section by defininig a new problem. Again, this is a problem on graphs in general, but we will be restricting our attention to bipartite graphs. This is a good idea for many reasons, but the most pertinent reason is that this problem is NP-complete in the general case.

Definition (Vertex cover). Let $G = (L, R, E)$ be a bipartite graph. A subset $C \subset L \cup R$ is said to be a *vertex cover* if for each $(i, j) \in E$ is such that at least one of $i, j \in C$. C is a *minimum* vertex cover if for any other cover C', $|C| \leq |C'|$.

Using what we've already learned, we can specify at least one relation between matchings and vertex covers: namely, the vertices of all edges in any maximal matching on a graph forms a vertex cover. Here are some examples of vertex covers on the graph we've been looking at.

You can convince yourself that the cover on the right is a minimum cover. This brings us to an important theorem. We first prove a lemma.

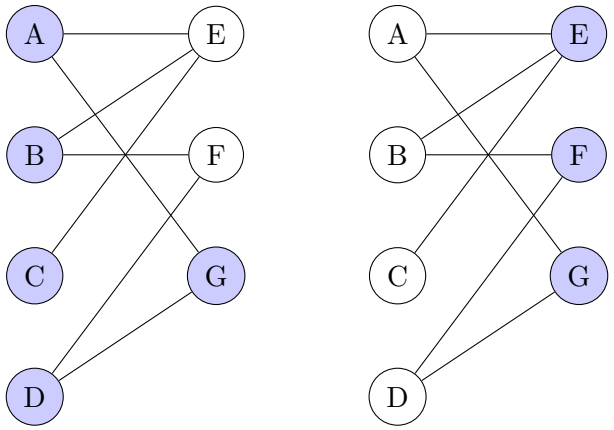


Figure 3: Examples of vertex covers

Lemma 1. Let $G = (L, R, E)$ be a bipartite graph. Let M be a matching on G and C a cover on G such that $|M| = |C|$. Then M is a maximum matching and C is a minimum covering.

Proof. Let $M^{'}$ be a maximum matching on G and $C^{'}$ a minimum covering on G. For each $(i, j) \in M^{'}$, $C^{'}$ must include either i or j, which tells us that $|M^{'}| \leq |C^{'}|$. Then we have

$$|M| \leq |M^{'}| \leq |C^{'}| \leq |C|.$$

Thus, if $|M| = |C|$ we have equalities above, which means that the size of a maximum matching is equal to the size of a minimum covering. □

Before we prove this, we define a couple of terms. Given a matching M, we say a vertex is *saturated* if it is incident to some edge in M. Also, an M-alternating path is an alternating path with respect to M.

Theorem (Kőnig-Egervary). For any bipartite graph G, if M is a maximum matching on G and C is a minimum vertex cover on G, then $|M| = |C|$.

Proof. Let $G = (L, R, E)$ be a bipartite graph, and let M be a maximum matching on G. Furthermore, define

$$A := \{s \in S \mid s \text{ unsaturated}\}$$

and

$$B := \{\text{all vertices connected to nodes in } A \text{ by } M\text{-alternating paths}\}.$$

Let $U = B \cap L$ and $V = B \cap R$. Then we have the following:

1. Every node in V is saturated.
2. $N(U) = V$,

where $N(U)$ denotes the set of all vertices connected to elements of U (the "neighbors" of U). The first claim comes from the fact that, if M is a maximum matching, then our alternating paths starting at nodes in A must have length ≥ 2, and must have even length (otherwise we would have an augmenting path, which contradicts our assumption that M is a maximum matching). The second comes from that fact that every node in $N(U)$ is connected to vertices in A by an

alternating path.
Now, define $K := (L \setminus U) \cup V$. Every edge in G must have one of its endpoints in K. If not, there there would be an edge with one end in U and one end in $R \setminus V$, which contradicts $N(U) = V$. So K is a covering of G. Moreover, $|K| = |M|$, since for each edge in M we've included one of its endpoints in K (the vertices we've chosen are those in $N(U)$ and those in $L \setminus U$). Thus, by the previous lemma, K is a minimum covering. □

All of this tells us that there is a deep relationship between maximum matchings and minimum vertex covers on bipartite graphs. Given a solution to one, we can turn it into a solution to the other. This is what we seek to accomplish next.

3 Linear programming

Add sentence relating to last section (we are formulating this things as optimization problems). For a more detailed treatment on linear programming, we will refer you to (CITE LP SOURCES). For the purposes of this thesis, we will treat linear programming more casually, only requiring few key results. Moreover, we will not be discussing methods of actually solving linear programs.
In the general linear-programming problem, our goal is to optimize some linear function that is constrained by a set of linear inequalities. These problems are ubiquitous in applied math and computer science, as they model a system in which something needs to be optimized according to competing resources. We can express a general *maximization* linear program as

$$\text{maximize} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_j, \quad i = 1, \ldots, m \tag{2}$$

$$x_j \geq 0, \quad j = 1, \ldots, n. \tag{3}$$

We call the function in (1) our *objective function*, and the linear inequalities (2) and (3) our constraints. Similarly, a *minimization* linear program takes the form

$$\text{minimize} \quad \sum_{j=1}^{n} c_j x_j \tag{4}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_j, \quad i = 1, \ldots, m \tag{5}$$

$$x_j \geq 0, \quad j = 1, \ldots, n. \tag{6}$$

Many problems which on the face may not appear to be optimization problems turn out to be easily rephrased as linear programs. Our goal in this section will be to describe our two problems, maximum matchings and minimum vertex covers, as linear programs. Before doing so, we describe duality theory, which allows us to draw relationships between certain linear programs.

Duality theory gives us a way to prove bounds on optimal solutions to linear programs. We first define the dual of a linear program.

Definition (Dual). Let

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned}$$

be our linear program, which we will call the *primal* linear program. Then we define the *dual* of this linear program to be the linear program

$$\begin{aligned} \text{minimize} \quad & \mathbf{b}^T\mathbf{y} \\ \text{subject to} \quad & A^T\mathbf{y} \leq \mathbf{c} \\ & \mathbf{y} \geq 0 \end{aligned}$$

The first thing to note is that the dual of the dual is the primal. Let us introduce some notation. First, let us denote the primal maximization problem by the letter Γ, and the dual minimization problem by the letter Ω. For a given linear program, we denote an optimal solution by **OPT**.

Theorem (Weak duality). If the primal linear program (in maximization form) and the dual (in minimization form) are both feasible, then

$$\mathbf{OPT}(\Gamma) \leq \mathbf{OPT}(\Omega).$$

What's surprising is the following theorem.

Theorem (Strong duality). Given two linear programs Γ and Ω that are duals of each other, if one is feasible and bounded, then so is the other. Additionally,

$$\mathbf{OPT}(\Gamma) = \mathbf{OPT}(\Omega).$$

Our goal now is to use this theory to relate the maximum matching problem to the minimum vertex cover problem.

Let's first turn maximum matching into a linear program. Our goal is to maximize the number of edges in our matching. Our constraint is that no edge is incident to more than one edge in the matching. So for each edge (i, j), we will need a corresponding x_{ij}. Our objective function is then pretty simple: maximize the number of x_{ij}. Now we need to figure out our constraint equations. For a fixed node i, the number of edges in the matching incident to i is given by $\sum_j x_{ij}$. So we want that this is ≤ 1. Similarly, for any node j, we want $\sum_i x_{ij} \leq 1$. This gives us the following linear program.

$$\begin{aligned} \text{maximize} \quad & \sum_{i,j} x_{ij} && && (7) \\ \text{subject to} \quad & \sum_j x_{ij} \leq 1, && \forall i \in S, && (8) \\ & \sum_i x_{ij} \leq 1, && \forall j \in T, && (9) \\ & x_{ij} \in \{0, 1\}. && && (10) \end{aligned}$$

Now let's try and construct the dual of this linear program. We will need a variable u_i for each vertex i in (8). Similarly, we need a variable v_j for each vertex j in (9). Our objective will be to minimize over the sum of these u_i, v_j. Since our constraint in the primal is the constant vector 1, our only constraint will be that $u_i + v_j \geq 1$. This gives us the dual linear program

$$\text{minimize} \quad \sum_{i,j}(u_i + v_j) \tag{11}$$

$$\text{subject to} \quad u_i + v_j \geq 1 \qquad \forall i, j, \tag{12}$$

$$u_i, v_j \in \{0, 1\}. \tag{13}$$

This dual problem tells us that each edge must be "covered" by at least one of its incident vertices. This is exactly the vertex cover problem! So for bipartite graphs, the linear programs for maximum matchings and minimum vertex covers are duals of each other. We will use this insight to construct our algorithms for solving the maximum matching problem.

4 Primal-dual matching algorithms

The reader may be familiar with matching algorithms via Ford-Fulkerson. They likely took a flow approach to the problem. Here, we take an approach motivated by the integer linear programs we described in the previous section.

Definition (Alternating tree). Let $G = (L, R, E)$ be a bipartite graph, and M a matching on G. An *alternating tree* with respect to M is a tree which satisfies two conditions:

- the tree contains exactly one node $i \in L$. We call i the *root* of the tree.
- all paths between the root and any other node in the tree are alternating paths.

The Kőnig Egervary theorem provides much of the motivation for this algorithm. Given a maximum matching, it tells us how to construct a minimum vertex cover. As this is a primal dual algorithm, we will be simultaneously constructing a maximum matching and a minimum vertex cover. Additionally, we want to maintain primal-dual feasibility at all steps in the computation. We initialize with the feasible matching $M = \emptyset$. Let's figure out what to initialize our vertex cover as. The most straightforward thing to do would be to initialize our vertex cover as $L \cup R$. This is certainly a feasible vertex cover, and a strict upper bound on the size of all other vertex covers. However, it turns out we can do a little better than this. Using the Kőnig-Egervary theorem and the weak duality theorem, let's construct an initial vertex cover based on the intial matching $M = \emptyset$. Using the notation from the Kőnig-Egervary theorem, our set $A = L$, since all vertices in L are unsaturated. Then our set $B = L \cup R$. Thus, our set K is equal to R. So we can set our initial vertex cover $C = R$. Note that this is a tighter upper bound on the size of vertex covers. The algorithm looks at exposed nodes in L, and seeks augmenting paths from that node. In this way, each exposed $i \in L$ becomes the root of an alternating tree. Additionally, each time we augment, we update the information regarding our cover. This means we update the sets A and B, (and thus U, V, and C). In order to explicitly state how we augment, we will associate with each vertex k a label π_k, which is equal to $\emptyset$ for exposed vertices in L; for vertices j in T that are exposed, their label will be equal to some vertex $i \in L$ where $(i, j) \in E$ and $(i, j) \notin M$. We use these labels to trace the alternating path all the way to the source, which will have label $\emptyset$.

ALG 2($G = (L, R, E)$)

```
1  M := ∅
2  A := L, B := L ∪ R, C := R
3  while |M| < |C|
4        for i ∈ L with label ∅
5              for (i, j) ∉ M
6                    label π_j := i unless already labeled
7        for j ∈ R
8              if j is exposed
9                    augment
10                   update A, B, C
11                   remove all labels on vertices
12                   label each exposed i ∈ L with ∅
13             else
14                   find the unique (i, j) ∈ M, label π_i := j.
15 return M and C
```

We know what an augmenting path means in the primal: our sum $\sum_{i,j} x_{ij}$ increases by one. But how can we interpret augmenting paths from the dual perspective? We know that it decreases the size of the set A by one.