15-859E: Advanced Algorithms **CMU, Spring 2015**
Lecture #7: Graph matchings I: combinatorial algorithms January 28, 2015
Lecturer: Anupam Gupta Scribe: Evan Palmer

1 Contents

Matching on bipartite graphs (König)
Matching on non-bipartite graphs (Tutte Berge, Edmonds-Gallai)
Blossom algorithm

2 Notation and Definitions

During this lecture, we will consider only simple graphs.

We will denote an undirected unweighted graph G with the vertex set V of size n and the edge set E of size m as:

$$G = (V, E)$$

We will denote an undirected unweighted bipartite graph with left vertices L, right vertices R and edges E as:

$$\text{Bipartite} = (L, R, E)$$

Definition 7.1. Matching: A matching on a graph is some subset of the edges $M \subseteq E$ which have no endpoints in common.

Given a matching M on a graph, we will say that:
A vertex v is *open* or *exposed* if there is no edge in the matching adjacent to v
A vertex v is *closed* or *covered* if there exists an edge e in the matching such that v is an endpoint of e.

Definition 7.2. Perfect Matching: A perfect matching on a graph is a matching such that for every vertex in the graph, there is an edge in the matching which touches that vertex.

$$\forall v \in V, \exists e \mid e \text{ is an endpoint of } v$$

Definition 7.3. Maximum Matching: The maximum matching on a graph is the matching with the largest cardinality. We will denote the maximum matching as:

$$\text{Maximum Matching (G)} = MM(G)$$

Definition 7.4. Maximal Matching: A maximal matching on a graph is a matching where no additional edges can be added without violating the shared endpoint constraint.

Definition 7.5. Vertex Cover: A vertex cover of a graph is a set of vertices C such that every edge in the graph has at least one endpoint in C.

3 Bipartite Graphs

3.1 Ford-Fulkerson for Matchings

Consider a bipartite graph: Bipartite (L, R, E)

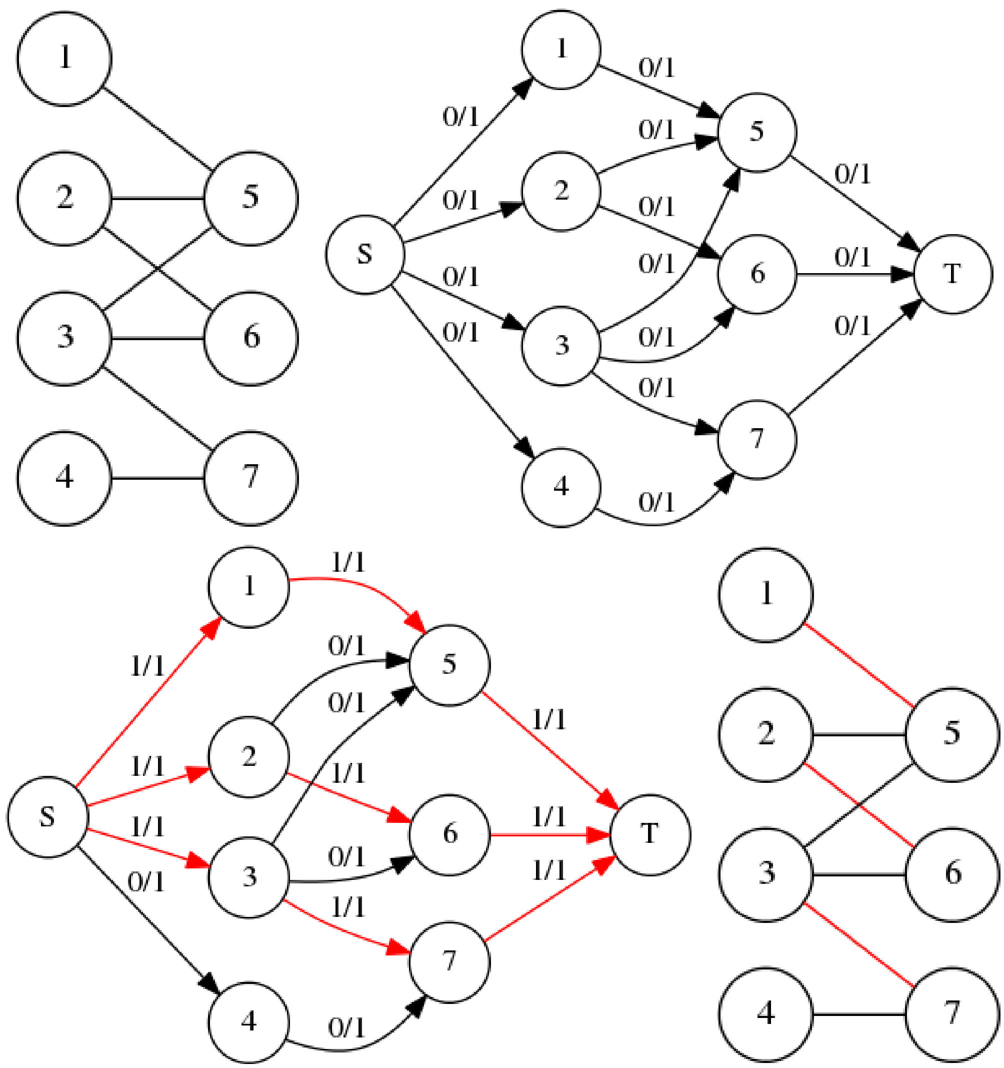


Figure 7.1: The use of Ford Fulkerson to determine a matching

We can do the following:

- Create a super source s and connect it to all of the vertices in L with a directed edge of capacity 1 from s to L.
- Create a super sink t and connect all of the vertices in R to it with a directed edge of capacity 1 from R to t.
- Change every edge in E into a directed edge which points from its endpoint in L to its endpoint in R, and give the edge capacity 1.

We can now run Ford-Fulkerson on this graph. Due to the integral flow theorem, we know that we can construct a max flow which gives a capacity of either 1 or 0 to each edge in our graph.

We will choose the edges which are fully saturated in the maxflow graph and present in our original graph. These edges will form a maximum matching.

3.2 Another algorithm for Matchings

Suppose we have some matching M on the graph G, and we would like to find an M-augmenting-path if such a path exists.

Any augmenting path must be of odd length, since it must start and end on opposite sides. Otherwise, when we toggle the edges on the path, it will not increase the size of the matching.

- Direct every edge in the matching left.
- Direct every edge not in the matching right.
- Try to find a path from an open vertex on the left to an open vertex on the right.

If we find an augmenting path of this form, we can toggle the edges in the path, and this will form a larger matching.

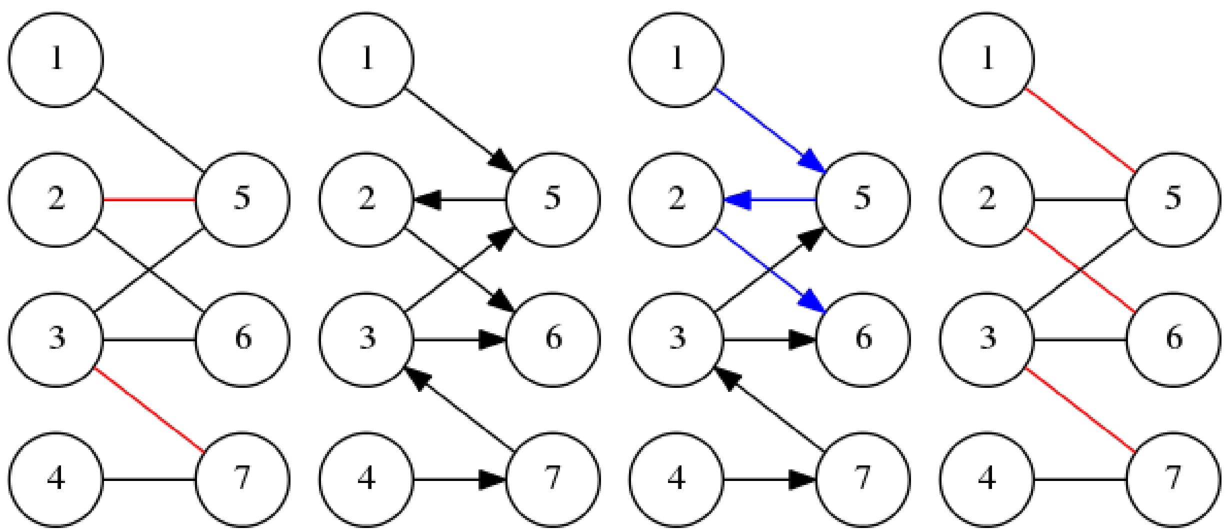


Figure 7.2: The transformation from a non maximum matching to a larger (coincidentally maximum) matching by finding an augmenting path

If there is any augmenting path, it must alternate between matched and unmatched edges, and therefore, we must be able to find it in this manner.

Each iteration takes $O(m)$ time, and we will run at most $O(n)$ iterations. This gives the algorithm an overall complexity of $O(mn)$. This is essentially Ford-Fulkerson. When we direct matching edges backwards we are doing the same thing as Ford-Fulkerson does with residual edges.

3.3 Other algorithms

Hopcroft-Karp: $O(m\sqrt{n})$ - This is the fastest current algorithm for matching on bipartite graphs. It finds many augmenting paths at once and then combines them in a clever way.

Evan, Tarjan: $O(\min(m\sqrt{m}, mn^{2/3})$ - An algorithm for computing maxflows on unit graphs.

3.4 Maximal matching and the vertex cover

Theorem 7.6 (König-1933)**.** *On a bipartite graph, the cardinality of the smallest vertex cover is the size of the largest possible matching.*

$$\min(|VC(G)|) = |MM(G)|$$

This is a special case of the max-flow min-cut theorem.

Proof. $(VC \geq MM)$

Since edges in the matching MM share no common endpoints, in order to have a vertex cover, we must have at least one vertex for each edge. □

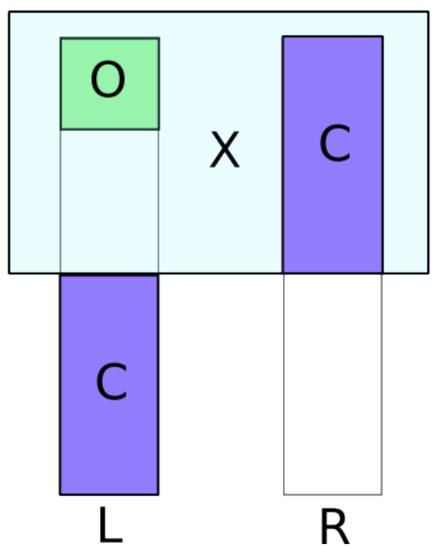
Proof. $\exists\, VC \mid (VC \leq MM)$

Consider the bipartite graph left over from running our second version of Ford-Fulkerson. There can be no augmenting paths since we have a maximum matching.

Let X be the set of vertices reachable from the open vertices O on the left side.

We will take our vertex cover C to be the vertices on the left side which are not in X (CL), and the vertices on the right side which are in X (CR).

$$C = (L \setminus X) \cup (R \cap X)$$



Claim 7.7. *C is a vertex cover.*

To show that this is a vertex cover, we must demonstrate that there can be no edges between $L \cap X$ and $R \setminus X$. Since this is a bipartite graph all other possible edges must have at least one endpoint in C

- There can be no unmatched edge from the open vertices $L \cap X$ to $R \setminus X$ since otherwise that vertex would be reachable from O and therefore in X. There cannot be a matched edge from an open vertex by definition.

- There can be no unmatched edge from a closed vertex in $L \cap X$ to $R \setminus X$ as this would make the endpoint in $R \setminus X$ reachable from O
- There can be no matched edge from $R \setminus X$ to $L \cap X$. Since everything in $L \cap X$ is reachable from O, it must have a matched edge from something in $R \cap X$. Accordingly it cannot have another matched edge due to matching constraints.

Claim 7.8. $|C| \leq |MM|$

- Every vertex in $R \setminus X$ must have an edge in the matching as otherwise it would be open, and there would be an augmenting path
- Every vertex in $L \cap X$ must have an edge in the matching since no vertices in $L \cap X$ are open.
- There can be no edges between $R \setminus X$ and $L \cap X$ as this would mean that things in $R \setminus X$ were reachable from X and therefore O

So, every vertex in $(L \cap X) \cup (R \setminus X)$ corresponds to a unique edge in the matching, and $|MM| \leq |C|$

□

4 Non-Bipartite Graphs

4.1 Tutte-Berge Theorem

Theorem 7.9 (Tutte-Berge)**.** - *Given a graph G, the size of the maximum matching is described by the following equation.*

$$MM(G) = \min_{U \subseteq V} \frac{n + |U| - odd(G \setminus U)}{2}$$

Here U is a set of vertices such that if U is removed from G, the remainder of G becomes disjoint with pieces $\{K_1, K_2, ..., K_t\}$. The quantity $odd(G \setminus U)$ is the number of such pieces with odd cardinality.

It is clear that the size of the matching is must be bounded by this quantity.

At most everything in U can be in the matching. Similarly, since the partitions K_i are disjoint, things not matched with U can only be matched within the partition. This gives us:

$$\begin{aligned} |M| &\leq |U| + \sum_{i=1}^{t} \lfloor \frac{k_i}{2} \rfloor \\ &\leq |U| + \frac{n - U}{2} - \frac{\text{odd}(G \setminus U)}{2} \\ &\leq \frac{|U| + n - \text{odd}(G \setminus U)}{2} \end{aligned}$$

To make some sense of this formula, we can consider the trivial case when $U = \emptyset$. Here the formula states that if the graph has an even size, then the maximum matching cannot be bigger than $n/2$, and if it has an odd size, then the maximum matching cannot be bigger than $(n - 1)/2$.

4.2 Gallai Edmonds Decomposition

We will break the vertices of the graph G up into three sets B, M, and T as follows

$$B(G) = \{v \mid \text{there exists a maximum size matching without v}\}$$
$$M(G) = \{v \mid v \text{ is a neighbor of some } u \in B(G), \text{but } v \notin B(G)\}$$
$$T(G) = V(G) \setminus (B(G) \cup M(G))$$

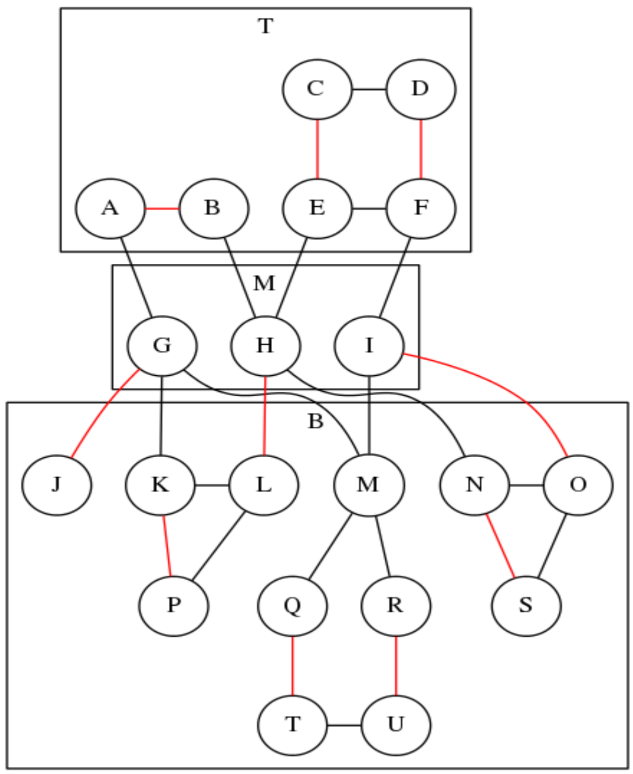


Figure 7.3: The Gallai Edmonds Decomposition

We then put the vertices in M in the middle, those in B to the bottom, and those in T to the top, we will have a graph which looks somewhat like Figure 7.3.

All of the components in T must be even since all of their vertices are in every maximum matching.

The set M will form our minimum U from the Tutte-Berge Theorem. Neat!

4.3 The Edmonds Blossom algorithm

We have an algorithm that finds either a m-augmenting-path or a m-blossom (which will reduce the size of the problem) if there exists an m-augmenting path.

Definition 7.10. A Blossom is an alternating path of even length which starts at an open vertex and ends at an odd length cycle.

The *root* of the blossom is the vertex where the alternating path intersects the odd cycle.

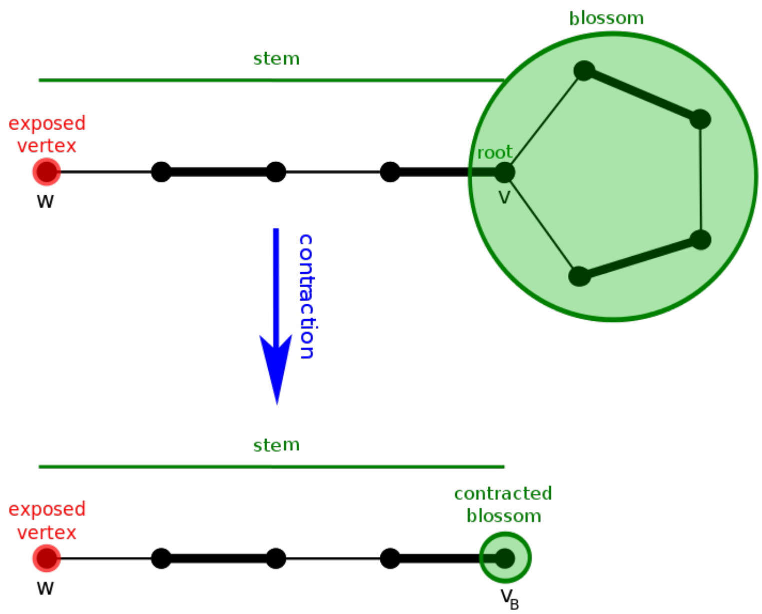


Figure 7.4: The shrinking of a blossom[1]

The *stem* of a blossom is the even length alternating path.

Given a graph G with a blossom, we can construct $G \setminus B$ by shrinking the blossom. This is done by replacing the odd length cycle with the root vertex and modifying all edges which had one endpoint in the odd length cycle to have an endpoint at the root vertex.

After shrinking a blossom in this manner, we can see that there will be an augmenting path in the shrunk graph $G \setminus B$ if and only if there is an augmenting path in G (shown in Figure 7.5).

This allows us to conclude that a matching M is maximum on G if and only if $M \setminus B$ is maximum on $G \setminus B$.

[1]Image found at `http://en.wikipedia.org/wiki/Blossom_algorithm`

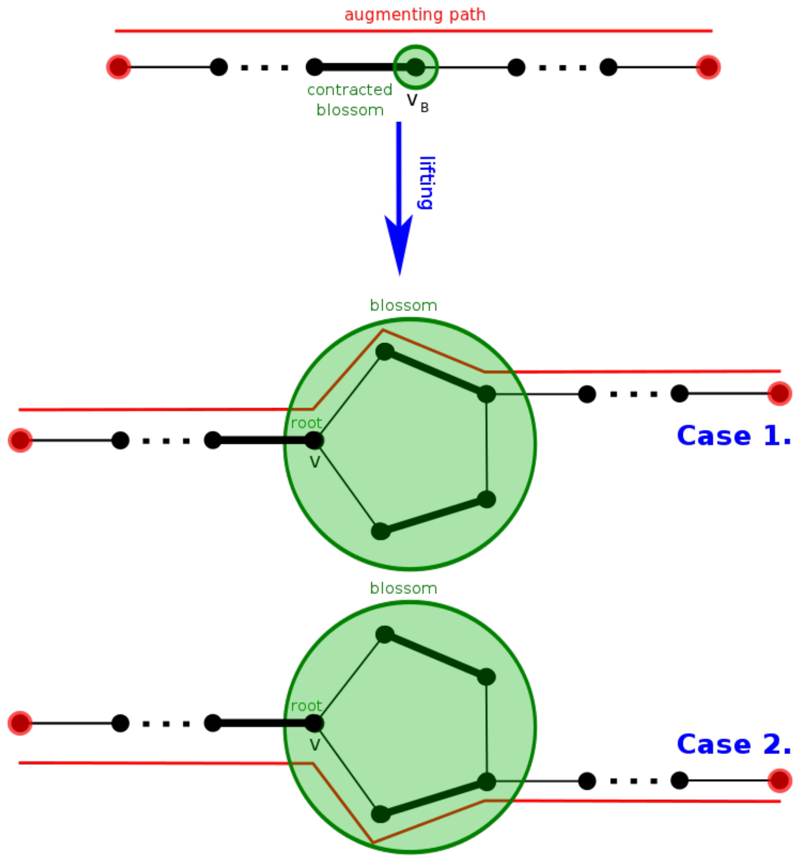


Figure 7.5: The translation of augmenting paths from $G \setminus B$ to G and back[2]

[2]Image found at http://en.wikipedia.org/wiki/Blossom_algorithm