CMPSCI 611 Advanced Algorithms **Fall 2000**

Lecture 6: September 28

Lecturer: Micah Adler *Scribe: Matthew Hertz and Joel Sieh*

6.1 Bipartite Matching

Bipartite matching is a good place to begin looking at the intersection of matroids. In studying bipartite matchings, we will build the machinery that we will use throughout our examination of problems involving the intersection of matroids.

6.1.1 Description of Bipartite Matching

Input: A Bipartite Graph $G = (U, V, E)$, where U, V are two sets of vertices and E a set of edges. Since G is bipartite, $(\forall (u, v) \in E).(u \in U \text{ AND } v \in V)$.

Output: A matching of the bipartite graph G of maximum size.

Consider the bipartite graph shown in Figure 6.1.

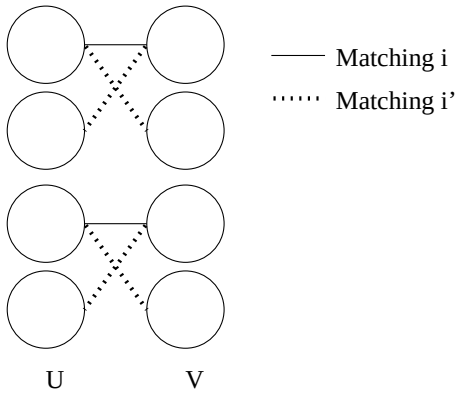

Figure 6.1: A bipartite graph G with two maximal matchings

Theorem 6.1 *Bipartite matching is not a matroid.*

Proof: In Figure 6.1, both i and i' are maximal matchings of G, but $|i| \neq |i'|$. This violates the Cardinality Theorem. Therefore, the bipartite matching cannot be a matroid. ■

However, we can express the problem as the intersection of the following two matroids:

(E, I) Consider U independently of V; E is E from the original problem and I contains the independent sets of edges such that at most one edge in an independent set is incident to any vertex in U. The following figure shows the maximum solution to the subset system (E, I)

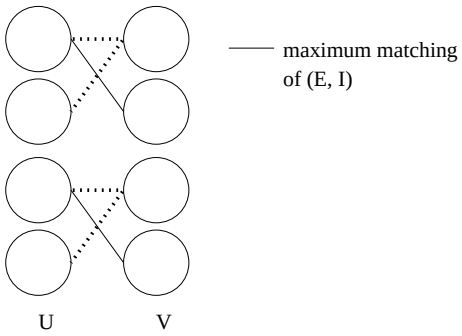

Figure 6.2: A bipartite graph G with the maximum solution to the first matroid subset

(E, J) Consider V independently of U; this problem is like the first one, but J contains all independent sets with at most one edge incident to any vertex in V. The following figure shows the maximum solution to the subset system (E, J)

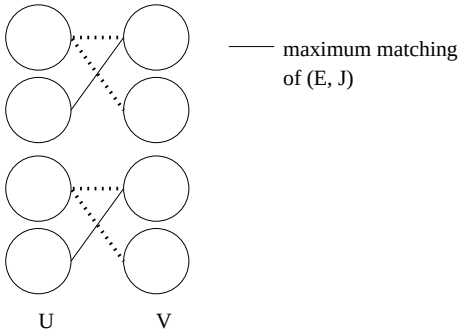

Figure 6.3: A bipartite graph G with the maximum solution to the second matroid subset

Sets in $(E, I) \cap (E, J)$ are the independent sets with at most one edge incident to any vertex in U and at most one edge incident to any vertex in V, or a matching of G. Problems that can be solved this way are known as **intersections of matroids**.

We can prove several useful properties of intersections of matroids.

6.1.2 Time bound on maximum cardinality set

Claim: Given two matroids (E, I) and (E, J) defined over the same set of elements E, the maximum cardinality set in $I \cap J$ can be found in time $O(|E|^3 * C(|E|))$. We define $C(|E|)$ to be the time required to test if an element i is a member of both I and I'.

We use time $O(C(|E|))$ because different matroids have varying complexities when testing for membership. For example, it is easy to test if a set is a bipartite matching, but difficult to test if a set is a linear independent set of columns.

Also note that this proof is to determine the **maximum cardinality set** and not the maximum weight set. The maximum weight set can be found, but the proof is more complex than what we will do in this class. For now we will prove this theorem only for bipartite matching.

6.1.2.1 Augmenting Paths

Augmenting paths is a way of building matchings one edge at a time (although we will later show how to use them in any intersection of matroids problem). Like the greedy method one edge will be added at a time. However, because this is not a matroid problem we does not have a guarantee that once an edge is accepted it will always be accepted. Therefore we must create some method of backtracking.

Consider the following example of a bipartite graph on which we have found a non-maximal matching:

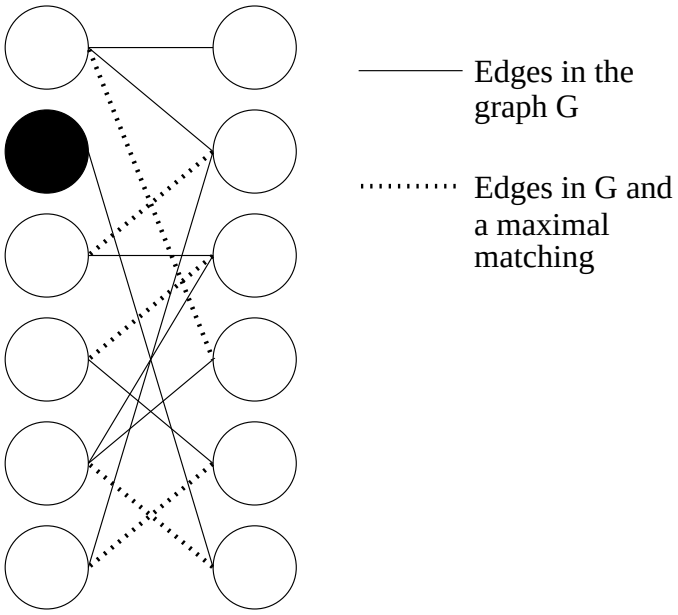

Figure 6.4: A bipartite graph G with a non-maximal matching

Definition 6.2 *Given a matching, a* **free vertex** *is a vertex that is not incident to any edge in the matching. In Figure 6.4 the black vertex is a free vertex.*

Definition 6.3 *Given a matching M, an* **augmenting path** *P for M is any path alternating matching and non-matching edges, beginning and ending on a free vertex. The following figure shows an augmenting path for the matching in Figure 6.4.*

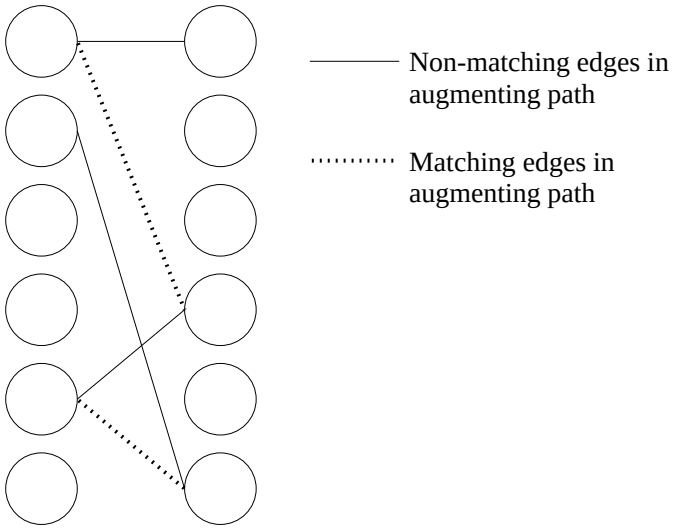

Figure 6.5: An augmenting path for the matching in Figure 6.4

Definition 6.4 *Given a matching M and an augmenting path P for M the* **symmetric difference** *between M and P is $M \cup P - M \cap P$ (i.e. all edges that exist in either M or P, but not both). If we call the symmetric difference M', we would write this as $M' = M \oplus P$.*

Figure 6.6 shows the symmetric difference between our the matching in Figure 6.4 and the augmenting path in Figure 6.5:

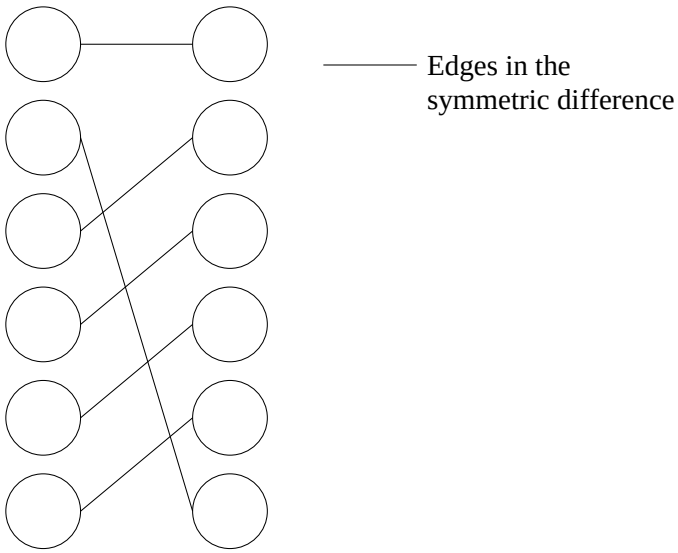

Figure 6.6: The symmetric difference between Figure 6.4 and Figure 6.5

Theorem 6.5 *Given any matching M and an augmenting path P, $M' = M \oplus P$ is also a matching.*

Proof: Consider edges $e_1, e_2 \in M'$, there are three possible cases for the membership of e_1 and e_2:

Case 1: $e_1, e_2 \in M - P$
Since M is a matching and $e_1, e_2 \in M$, they cannot share a common vertex.

Case 2: $e_1, e_2 \in P - M$
If a vertex is free, then there is at most one non-matching edge incident to it in the augmenting path. Every other vertex has two edges incident to it: one matching edge and one non-matching edge. Since e_1 and e_2 are in the augmenting path and both are non-matching (i.e. neither is in M), they cannot share a vertex.

Case 3: $e_1 \in M - P,\ e_2 \in P - M$
If e_2 is incident to a free vertex, by definition 6.2, e_2 cannot share the vertex with any edge $e_1 \in M$. If both e_2 and e_1 are incident to the same (non-free) vertex, then e_1 would have to be a member of both M and P, by definition 6.3. But $e_1 \in M - P$, so this is not possible. Therefore, e_1 and e_2 cannot share a non-free vertex.

Since no edge in M' shares a vertex with any other edge in M', M' is a matching. ■

Claim: $|M'| = |M| + 1$

Since the augmenting path P starts and ends on a free vertex and we alternate edges not in M with edges in M, every even edge in P will be removed. Therefore we must add 1 edge.

Note that we still have not determined how to find an augmenting path nor how to find if an augmenting path exists, but merely the properties we can use once we have an augmenting path. We will now try to show how we can find these augmenting paths.

Lemma 6.6 *If a matching M is not maximum, then there exists an augmenting path for M.*

Proof: Assume there exists a non-maximum matching M and a maximum matching M', $|M'| > |M|$.

Let $E' = M \oplus M', G' = (V, E')$. What do we know about G'? What does G' look like?

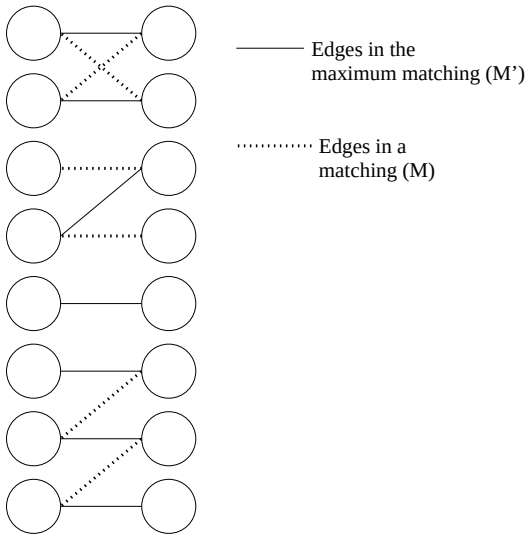

Figure 6.7: An example of a matching and the maximum matching

Consider the graph G' in Figure 6.7. All of the vertices have degree ≤ 2. If $v \in V$ has degree of 2, then there exists an edge incident to v in M and a different edge incident to v in M'. We also know the connected components in G' must either be cycles or augmenting paths. The cycles must be of even length and have equal numbers of edges from M and M'. However $|M'| > |M|$, so there must exist a path with more edges in M' than in M. Since that path cannot be a cycle, it must be an augmenting path. ■

6.1.3 Algorithm for Maximum Bipartite Matching

We can now being to define an algorithm for finding the maximum bipartite matching:

$M = \emptyset$
while there exists an augmenting path P
    $M \leftarrow M \oplus P$
endwhile
return(M)

All we have to do now is find the augmenting paths. If we didn't have the alternating matching and non-matching edge requirement, we could simply do a breadth-first-search (BFS) from a free vertex to find an augmenting path.

Note that if we do augmenting paths we will always take edges in the matching in one direction and edges not in the matching in the other direction. Figure 6.8 shows an example of this:

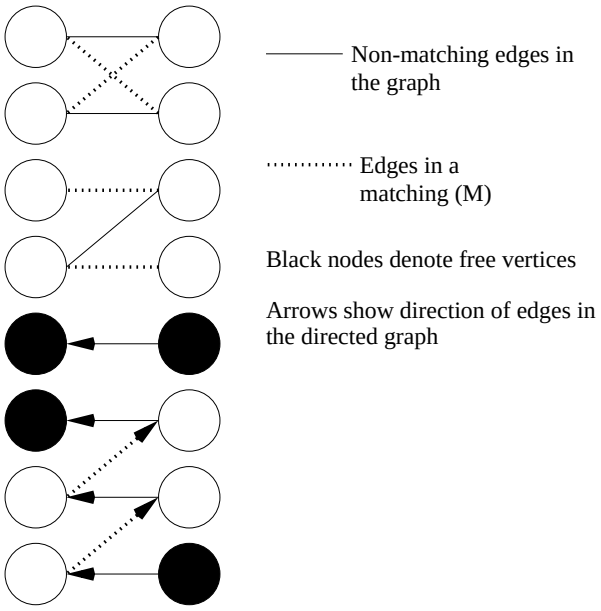

Figure 6.8: Augmenting paths from a free vertex

We can rewrite the graph so that all edges in the matching flow in one direction (e.g. $U \rightarrow V$) and all other edges flow in the other direction. For example, Figure 6.8 has all edges in the matching from the right set to the left set and vice versa. The augmenting path can now be found by doing a BFS from all the free vertices on the directed graph.

6.1.4 Running Time for Maximum Bipartite Matching

Now that we have a complete algorithm for maximum bipartite matching, what is the running time?

- Number of augmenting paths: We add one edge per augmenting path, therefore we will need as many augmenting paths as the size of the maximum bipartite matching. Since we cannot have more edges in the maximum matching than vertices in one set, the time is: $O(min(|U|, |V|))$
- Time to find each augmenting path: We need to start a BFS from each node. By saving our search results we only need to visit each vertex once. The time for each path is: $O(|E|)$

Total running time = $O(|E| * min(|U|, |V|))$.

6.2 Extension to Generic Intersection of Matroid Problems

How can we generalize these theorems to any intersection of matroid problem? We are going to essentially build the augmenting set, from which we will be able to take the symmetric difference. Consider $(E, I) \cap (E, J)$

Assume we start with an element m where m is a valid solution in $I \cap J$	
Find an e_1 such that $m + e_1 \in I$ and $m + e_1 \notin J$	`an edge from some free vertex`
Select an e_2 such that $m + e_1 - e_2 \in I \cap J$	`add a matching edge`
Select an e_3 such that $m + e_1 - e_2 + e_3 \in I$ and $m + e_1 - e_2 + e_3 \notin J$	`add the next non-matching edge`
...and so on until ...	`...and so on until ...`
$m + e_1 - e_2 + e_3 \ldots - e_{even} + e_{odd} \in I \cap J$	`add the last edge to a free vertex`

If this sequence exists, than for any intersection of matroids we can find it using BFS.

6.2.1 Application of Intersection of Matroids

6.2.1.1 Branching for a Directed Spanning Tree

Problem: Find a branching for a directed spanning tree (also known as an arborescence).

Input: A directed, acyclic graph $G = (V, E)$ with a root vertex r

Output: If it exists, a branching, which is a subset of edges such that:

1. the corresponding undirected graph is a tree
2. all edges in the branching point away from the root

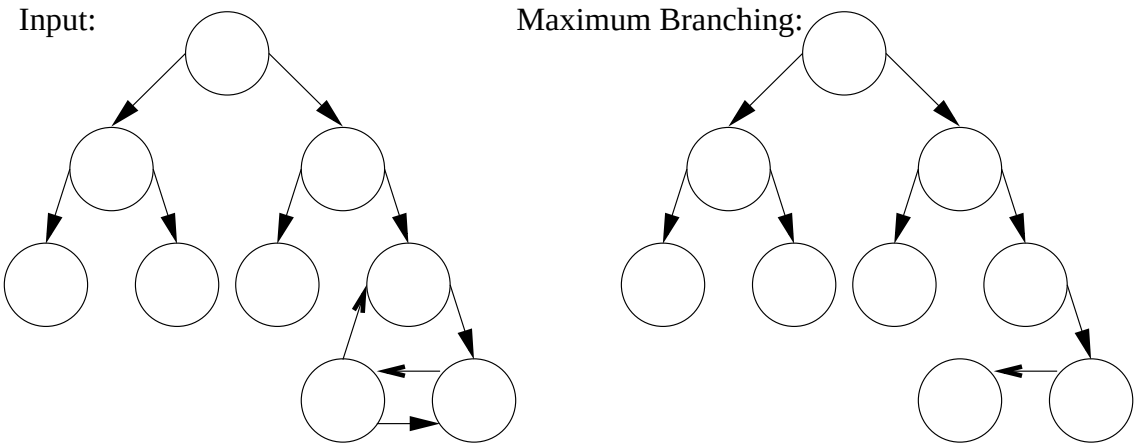

Figure 6.9: An example directed, acyclic graph and a branching of the graph

We can find the maximum cardinality of the branching by dividing the problem into two matroids:

(E, I): E = the edges of the graph and I = undirected edges which form a forest

(E, J): E is as above and J = all edges such that at most one edge is directed toward any vertex, except the root vertex which has no incoming edges

I corresponds to all possible solutions of the first requirement and J corresponds to all possible solutions of the second requirement. Since in a tree the root has no incoming edges and all other vertices have at most 1 incoming edge, $I \cap J$ would be the set of all, possibly disconnected, branchings.

To find if a graph has a branching, we can find the maximum cardinality of $I \cap J$. If the result is a connected graph (e.g. the maximum cardinality is $|V| - 1$), then the branching exists; if the graph is disconnected, the branching does not exist.