COMPSCI 330: Design and Analysis of Algorithms Spring 2019

Lecture 15: Bipartite Matching

Lecturer: Rong Ge *Scribe: Shweta Patwa*

1 Motivation

A bipartite graph is used to model the relationship between two different objects.

Classroom assignment: Suppose there are n courses and m classes. Each course can only use a subset of classrooms, which is given to us.

Goal: Find a way to schedule all courses in the classrooms that are available (where one course only needs one classroom, and one classroom can only hold one course).

In the bipartite graph, we use vertices on the left-hand side to represent courses, and those on the right for classrooms. There is an edge between a vertex for a course and that for a classroom, if the classroom can be used to hold the course.

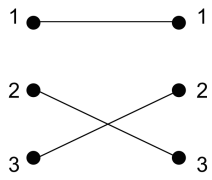
Observe that the solution here will be a set of edges that do not share any vertices. We leave it as a small exercise for students to see why this must be true.

2 Definitions

Bipartite graphs

1. Mostly used as abstractions for relations of two types of objects.
2. $G = (U, V, E)$, where U is the set of vertices of type 1, V is the set of vertices of type 2 and E is the set of edges between a vertex in U and a vertex in V. Thus, $E = \{(u,v) | u \in U, v \in V\}$. $|U| = n_1, |V| = n_2$ and $|E| = m$.

   The following graph has edges $(1,1), (2,3)$ and $(3,2)$.



3. Matching M of a bipartite graph is a subset of edges such that no edges in M share a vertex.

   Thus, the set of edges above is a valid matching. If we had as edge $(2,2)$ then M can contain either $(2,2)$, or $(2,3)$ and $(3,2)$, but not all three.
4. Maximum bipartite matching is a matching of maximum size.

5. Size of a matching M is the number of edges in M.
6. $\min\{n_1, n_2\}$ gives an upper bound on the size of a maximum matching.
7. For a bipartite graph G and matching M, a vertex is called matched if it is adjacent to an edge in the matching M.
   Call an edge e matched if $e \in M$, otherwise call it unmatched.

Thus, going back to the classroom assignment problem:

- Bipartite graph $\iff$ Courses and classrooms
- Edge (i, j) $\iff$ Course i can use classroom j
- Matching $\iff$ Assignment of courses to classrooms
- Maximum matching $\iff$ Schedule max #courses

3 First attempt: Greedy algorithm

Algorithm: For each course, if it has a classroom that is not taken by another course, then schedule it in that classroom.

But this does not work. In the example graph above, if we choose $(1,1)$ and $(2,2)$, then that is a smaller matching than $(1,1),(2,3)$ and $(3,2)$.

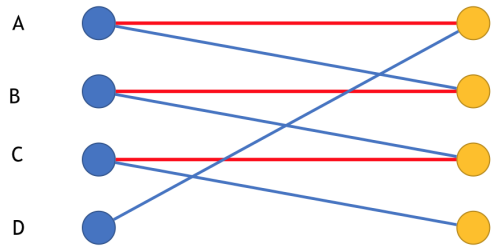
Thus, this very simple greedy approach does not work.

3.1 Trying to fix the problem

Idea: If a classroom is taken, ask its instructor to move to another room.

This means that if M contains $(2,2)$ but 3 on the left can only be matched to 2 on the right, then if we can switch $(2,2)$ with $(2,3)$, all vertices can have a matched edge incident to them. Thus, we swap $(2,2)$ out for $(2,3)$ and $(3,2)$.

This can involve more edges, or a longer path. Here is an example:



D: A, can you find another room? A: let me try..
A: B, can you find another room? B: let me try..
B: C, can you find another room? C: Yes sure
B$\rightarrow$A: I've found another room.

A→D: I've found another room.

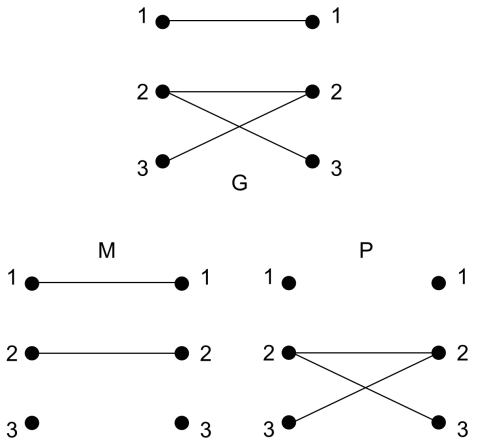
In the more complicated examples, instructors may have more rooms to choose from. We need more structure to be able to make this process efficient.

3.2 Formalizing the "room exchange" idea

Augmenting path: An augmenting path is a path from an unmatched course to an unmatched course, that alternates between unmatched and matched edges.

Fact: Length of an augmenting path is always odd. First and last edges of an augmenting path are unmatched.

Why do we care about augmenting paths?
We can improve the matching by finding an augmenting path and making the switch between matched and unmatched edges. Given a graph, a matching and an augmenting path, the XOR operation produces a new matching. Recall that $x \oplus y$ is 1 if and only if $x \neq y$.
All the unmatched edges in the augmenting path are now matched, and the matched edges in the augmenting path are now unmatched.



Here M is a matching and P is an augmenting path. If we do the XOR between these graphs, then we get a matching corresponding to the graph in the figure on page 1. This always results in a matching with one more edge than in the matching we started out with. We denote the resulting graph by $M' = M \oplus P$.

4 Augmenting path algorithm

Idea:

Initialize M to be empty

While there is an augmenting path P

$M = M \oplus P$

Questions:

1. How to find an augmenting path?
2. If we cannot find an augmenting path, does that mean we have found a maximum matching?

4.1 Finding an augmenting path

Start from an unmatched vertex, course in the case of classroom assignment. The first edge we follow needs to be an unmatched edge. If the next vertex is unmatched, then add this edge to the matching. However, if the vertex is matched, then we need to follow the incident matched edge, and so on. We use DFS to find such an augmenting path between two unmatched vertices.

Running time is same as that of DFS, thus it takes $O(m+n)$ time.

4.2 Proof of correctness

Theorem 1. *Given a bipartite graph G and an augmenting path M, if there is no augmenting path with respect to this matching M, then M is a maximum matching.*

Proof. Assume towards contradiction that M is not a maximum matching. Let M' be a maximum matching. We know that $|M'| > |M|$. Consider $M' \oplus M$.
Claim. $M' \oplus M$ is going to have only paths or cycles.
Proof sketch. In $M' \oplus M$, every vertex has degree at most 2. Any vertex can be adjacent to at most one edge in M' and at most one edge in M. Thus, the number of edges adjacent to any vertex in $M' \oplus M$ cannot exceed 2. This results in either paths or cycles in the graph.
For a cycle: it contains the same number of edges in M and M'.
For a path: we know one of the matching has one more edge. The path will alternate between the two matchings for paths (of odd length). We know that the number of edges in M' is larger than the number of edges in M. Therefore, there must be a path of odd length where first and last edge are in M', and this path is indeed an augmenting path for M. Edges in M' are in the graph but unmatched for M. Hence, we have somehow constructed an augmenting path for matching M, which contradicts with the assumption that M does not have any augmenting paths. □

(Algorithm is given on the next page.)

```
Find_Path(u)
    Mark u as visited
    FOR all edges (u,v)     //enumerate over classrooms that course u can use
        IF v is not visited THEN
            IF v is unmatched or Find_Path(match_room[v]) = true THEN
// (if either we found an empty classroom, or the current instructor of
// the classroom is able to switch to another room)
                match_course[u] = v
                match_course[v] = u
// I will take this room
                RETURN true        // I have found a room for course u.
     RETURN false
// I've tried all the possible rooms, they are not empty and their instructors
// cannot switch to another room, so I cannot find a room for course u.
Max_Matching
      Initially set all nodes to be unmatched.
      FOR u = 1 to n     //enumerate the courses
           Mark all vertices as unvisited    //initialize for the DFS
           Find_Path(u)                          //Try to schedule course u.
```