Lecture 6 - Nonbipartite Matching and the Assignment Problem

Lecturer: Prof. Richard Karp
Scribe: Mark Whitney

February 6, 2006

1 Nonbipartite Matching

Problem: Given a graph G and a matching M, find an augmenting path or determine that there is none, in which case M is a maximum matching.

For the nonbipartite case, we require the concept of a *blossom*, which is constructed from the original graph by successively merging vertices that make up odd cycles.

A blossom B is a subgraph of G containing an odd number of vertices, one of which is called its *base*, such that

- M includes a perfect matching of all the vertices of B except the base
- Every vertex in B is reachable from the base by an even-length alternating path

An example of a blossom is given in Figure 1. This particular blossom is composed of a number of odd-length cycles.

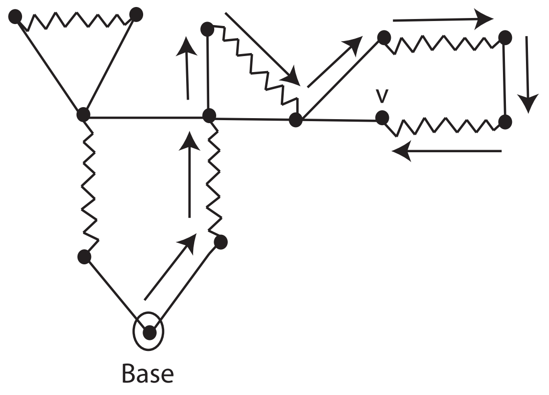

Figure 1: An example of a blossom. The arrows denote an alternating, even-length path from the base to a vertex v in the blossom.

At any point in time of finding a matching for a nonbipartite graph, we have the following:

- graph G
- matching M
- a set of disjoint blossoms
- A search tree T whose node set contains all the blossoms and some additional vertices of G, and whose edges are edges of G. Nodes at an even distance from the root are called *outer* and nodes at an odd distance are called *inner*.

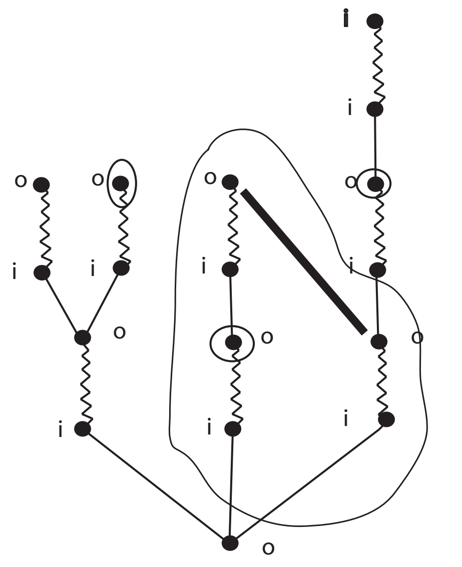

Figure 2: A new edge going from an outer node to outer node is added to the tree in bold, creating a new blossom. All the blossoms are circled. Inner and outer nodes are labelled with "i"s and "o"s

1.1 Search tree construction

The root of a search tree T is either a free vertex (w.r.t. M) or a blossom, and every blossom is an outer node. Along any path from the root, the edges are alternating solid and wiggly, and the path starts with a solid edge. The wiggly edge into a blossom is incident with its base, and the solid edges out of the blossom may be incident with any of its vertices. The leaves of the tree are outer. An example of a search tree is given in Figure 2.

The following procedure is followed to update a search tree T:

Algorithm 1 UpdateSearchTree(T)

```
if there exists a solid edge $(u, v)$ not in the search tree already joining 2 outer nodes (which may be blossoms) then
    merge the two outer nodes into a blossom where the base is the least common ancestor of the 2 outer nodes
else
    if there is a solid edge $(u, v)$ joining an outer node $u$ to a vertex $v$ not in search tree $T$ then
        if $v$ is a free vertex then
            construct an augmenting path.
        else
            if $v$ is matched to some node $w$ then
                insert edges $(u, v)$ and $(v, w)$, $v$ becomes inner and $w$ becomes outer.
            end if
        end if
    end if
end if
```

Construction of the augmenting path The path runs from the root (or its base if it is a blossom) to the free vertex v. If the root-leaf path of T passes through blossoms, the augmenting path will include a unique even-length alternating path from the base of each blossom to the exit point of the blossom. To get the full augmenting path, we insert the even length alternating path from the base node to the exit node.

Once an augmenting path is added, the search tree can be thrown out and a new one can be built from scratch, giving an $O(n^4)$ algorithm. Alternatively, the old search tree can be used to construct the next search tree and runtimes of $O(n^3)$ and $O(m\sqrt{n})$ (m=number of edges) are possible.

The construction of the tree T terminates when no outer nodes can be merged into a blossom and the tree cannot be extended. In this case, all the vertices in the tree can be set aside, as they will never participate in an augmenting path. The process ends when all free vertices have been set aside.

Proof that a maximum matching is obtained In the bipartite case, we used the Konig-Egervary theorem by showing a vertex cover could be constructed from the final matching. This does not work in the non-bipartite case.

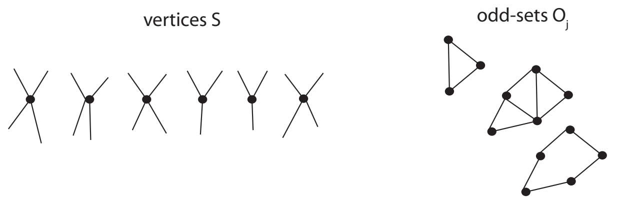

Figure 3: An example of an odd-set cover: a bunch of vertices S and disjoint odd-cardinality sets O_j

In a graph G, an *odd-set cover* consists of a collection of vertices S and odd-cardinality sets of vertices $O_1, O_2, ..., O_t$, disjoint from one another and from S as shown in Figure 3, such that every edge of G is either incident with a vertex in S or lies within an odd set O_j. The *capacity* of the odd-set cover is defined as:

$$|S| + \sum_{j=1}^{t} \frac{|O_j| - 1}{2} \tag{1}$$

Lemma If M is a matching and C is an odd-set cover then

$$|M| \leq \mathit{capacity}(C) \tag{2}$$

Proof M contains at most one edge incident with each vertex of S and at most $\frac{|O_j|-1}{2}$ edges contained in O_j.

This is due to the fact that each vertex in S can have at most one edge associated with the matching, and out of all the edges in odd set O_j, only one can be associated with each vertex pair, thereby leaving one out since the cardinality is odd.

We would now like to show that the above is an equality for the case of a maximum matching $|M| = \mathit{cap}(C)$.

Consider the information available upon termination of the blossom algorithm. We have a collection of trees from terminated searches, and the vertices not in those trees are perfectly matched.

We get an odd-set cover in which the odd sets are the blossoms, and the vertices are the inner vertices plus one end point of each edge of the matching in the perfectly matched part.

Exercise Verify that this is an odd-set cover

Exercise The capacity of this odd-set cover is equal to the cardinality of the matching M produced by the algorithm. Hence M is a maximum matching.

2 The Assignment Problem

Input: A complete bipartite graph with n boys, n girls, and a weight on each edge.

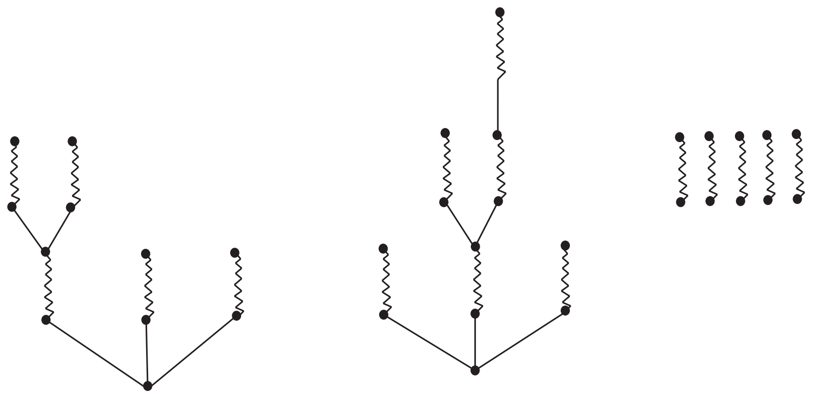

Figure 4: A bunch of search trees and perfectly matched vertices.

Goal: Construct a perfect matching of minimum total weight.

$$x_{ij} = \begin{cases} 1 & \text{if boy } i \text{ is matched with girl } j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

And let c_{ij} = weight of edge between boy i and girl j
The problem is:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

subject to:

$$\begin{aligned} \sum_{i=1}^{n} x_{ij} &= 1, \forall j && (4) \\ \sum_{j=1}^{n} x_{ij} &= 1, \forall i && (5) \\ x_{ij} &\in 0, 1\ \forall i, j && (6) \end{aligned}$$

If we relax the problem by letting the x_{ij} be fractional, we get a linear program:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{7}$$

subject to

$$\begin{aligned} \sum_{i=1}^{n} x_{ij} &= 1 && (8) \\ \sum_{j=1}^{n} x_{ij} &= 1 && (9) \\ x_{ij} &\geq 0 && (10) \end{aligned}$$

We will show that the basic feasible solutions of this linear program are integral. Therefore, there is an optimal integral solution, which gives an optimal perfect matching. Since we know there is always an optimal solution at a extreme point, we just have to show that all extreme points are zero-one vectors.

We will show that any solution in which some of the x_{ij} are fractional is the average of two other feasible solutions, and therefore not an extreme point, because an extreme point can never be an average of other solutions.

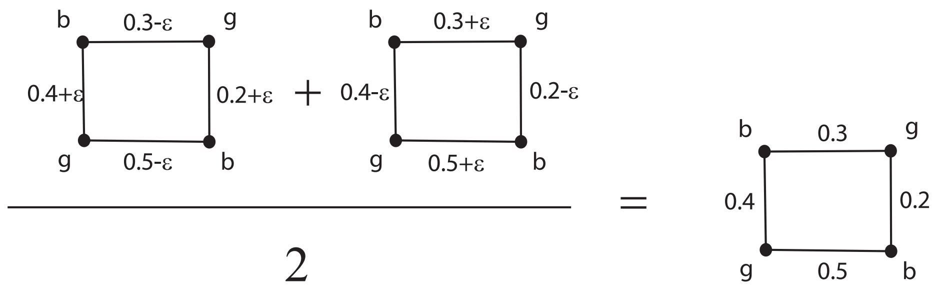

Figure 5: A fractional set of edge assignments is always an average of two other perturbed fractional assignments and therefore is not an extreme point.

Let (x_{ij}) be a feasible solution in which some variables are fractional. No row or column can contain exactly one fractional variable, since every row or column sum is 1. Consider the matrix for the bipartite graph with a boy vertex for each row and a girl vertex for each column and an edge from row i to column j if x_{ij} is fractional. This graph is nonempty and contains no vertex of degree 1 (because there cannot be exactly one fractional value of a row or column of (x_{ij})); therefore it contains a cycle, and the cycle must be even since the graph is bipartite. Now we can write (x_{ij}) as the average of two feasible solutions; one is obtained by alternately increasing and decreasing the x_{ij} around the cycle by a sufficiently small ϵ, and the other is obtained by reversing the signs of these changes. Figure 5 shows just such a perturbation situation where we have a 4-cycle of alternating boys and girls. Since the cycle is even, we perturb two edges by $+\epsilon$ and two edges by $-\epsilon$.

Thus we can find a perfect matching of minimum weight by computing a basic feasible solution of the linear program. Our algorithm will be based on the complementary slackness principle of linear programming, which we now derive. Consider the following dual pair of linear programs

Primal linear programming problem:

$$\begin{array}{rcl} \min & c'x & \\ \text{subject to } Ax & = & b \\ \text{and } x & \geq & 0 \end{array}$$

Dual linear programming problem:

$$\begin{array}{rcl} \max & y'b & \\ \text{subject to } y'A & \geq & c \end{array}$$

By the duality theorem, optimal solutions must satisfy $y^{*\prime}b = c'x^*$
Since $b = Ax^*$, this gives:

$$y^* Ax^* = c'x^* \tag{11}$$

$$(c\prime - y^{*\prime}A)x^* = 0 \tag{12}$$

Since $c' - y^{*\prime}A \geq 0$ by dual feasibility and $x^* \geq 0$, this means that, for all j:

$$(c' - y^{*\prime}A)_j = 0 \tag{13}$$

or

$$x_j^* = 0 \tag{14}$$

i.e. the primal variable x_j^* is positive only if the corresponding dual constraint is tight. This is called the complimentary slackness condition, and a pair of feasible solutions to the primal and dual problems are both optimal if and only if complementary slackness holds between them.

In our case, there is a dual variable u_i for each row and v_j for each column, and the following relations must hold:

$$\begin{aligned}
\sum_i x_{ij} &= 1 \text{ (primal feasibility)} \\
\sum_j x_{ij} &= 1 \\
x_{ij} &\geq 0 \\
c_{ij} - u_i - v_j &\geq 0 \text{ (dual feasibility)} \\
x_{ij} &> 0 \text{ only if } c_{ij} - u_i - v_j = 0 \text{ (complementary slackness)}
\end{aligned}$$