Lecture 3

1 Maximum Weighted Matchings

Given a weighted bipartite graph $G = (U, V, E)$ with weights $w : E \to \mathbb{R}$ the problem is to find the maximum weight matching in G. A matching is assigns every vertex in U to at most one neighbor in V, equivalently it is a subgraph of G with induced degree at most 1. By adding edges with weight 0 we can assume wlog that G is a complete bipartite graph. Finding maximum cardinality matchings is a standard problem in undergraduate algorithms.

1.1 Maximum Matching

An augmenting path is a path whose endpoints are unmatched vertices that alternates between unmatched and matched edges. A matching has maximum cardinality if and only if there are no augmenting paths.

Augmenting paths can be found by a breadth first search starting at unmatched vertices in U alternating between unmatched and matched edges. If the breadth first search reaches an unmatched vertex in V we have found an augmenting path and can extend the matching.

1.2 Maximum Weighted Matchings

The maximum weight matching problem is solved using the primal dual framework. It is useful to think in terms of upper bounds on the weight of a matching. The sum of the weights of the maximum weight edges incident on U is clearly an upper bound on the weight of a matching. The bound is not tight, for the path of length 3 where the middle edge has weight 2 and the others have weight 1 the upper bound is 3 while the maximum weight of a matching is 2.

The upper bound is generalized by assigning prices $p(u)$ to the nodes, such that the sum of the prices of the endpoints exceeds the weight of an edge, that is for all edges (u, v) we have $p(u) + p(v) \geq w(u, v)$. The sum of the prices over all the vertices is an upper bound on the weight of a matching,

$$\sum_{(u,v)\in M} w(u, v) \leq \sum_{(u,v)\in M} p(u) + p(v) = \sum_{w\in G} p(w) \tag{1}$$

This form of upper bound can be made tight for the length 3 path, by assigning 1 to the endpoints of the path.

The best upper bound on the weight of a matching using this approach corresponds to the weight cover for G, that is the smallest sum over prices such that all edges are covered; $p(u) + p(v) \geq w(u, v)$. The weight of a any cover is greater than the weight of a matching, this fact will be crucial for designing the algorithm.

Figure 1: Example of non-optimal and optimal price functions. Black edges have weight 1, and the blue edge has weight 2.

The primal dual algorithm maintains a cover and iteratively reduces the weight of the cover by finding maximum matchings on tight edges,

1. The initial cover is chosen to be $p(u) = \max_{u \in e} w(e), p(v) = 0$, this is a cover as $p(u) + p(v) \geq w(u, v)$ is true for all edges.

2. The edges (u, v) for which $p(u) + p(v) = w(u, v)$ are called the tight edges. Compute a maximum matching M on the tight edges, if the matching M has size $|U|$, the problem is solved as we found a matching and cover having the same weight.

3. Perform a breadth first search starting from unmatched vertices in U alternating between unmatched and matched edges. Let U' and V' be the vertices in U and V that are reachable in this manner.

   The augmenting path characterization of the maximum matching shows that there is no tight edge between U' and $V \setminus V'$. Let δ be the minimum excess over edges between U' and $V \setminus V'$.

4. Update the prices by setting $p(u) \rightarrow p(u) - \delta$ for $u \in U'$ and $p(v) \rightarrow p(v) + \delta$ for $v \in V'$. The new prices form a cover due to the choice of δ and the edges in the maximum matching continue to be tight.

   The total price has decreased due to this step as $|U'| > |V'|$, this follows as V' consists of matched vertices and the neighbors of these vertices belong to U'. At least one edge between U' and $V \setminus V'$ becomes tight for every update. Repeat step 2 with the new set of tight edges.

Analysis: Progress is made as the size of the maximum matching over tight edges is monotonically increasing. With some data structures, we can ensure that the size of the matching increases in $O(m \log n)$ time. Thus, in time $O(nm \log n)$ we have a perfect matching and a corresponding feasible price function of the same value. That is, we have a maximum weight matching.

1.3 Maximum Matching Game.

This solution was rather clever, standard tools can be applied by framing the maximum weighted matching problem as a two player zero sum game. This formulation can be found in the homework.