CS 435 : Linear Optimization Fall 2006

Lecture 30: Primal-Dual Algorithm for Matching in Bipartite Graphs (Conclusion)

Lecturer: *Sundar Vishwanathan* Scribe:*Ankit Aggarwal*
Computer Science & Engineering Indian Institute of Technology, Bombay

1 Primal-Dual Method

Primal	Dual
$\max \sum w_{uv} x_{uv}$	$\min \sum Y_u + \sum Z_v$
$\sum x_{uv} = 1$	$\forall \{u, v\},\ y_u + z_v \geq w_{uv}$
$x_{uv} = 0$	

2 Procedure

- If there doesn't exist any constraint such that the equality holds in it, we decrease all dual variables by some δ, such that atleast one of the constraints satisfy equality.
- Let $E^{'}$ be the edges such that $y_u + z_v = w_{uv}$
- Now solve,

$$\min \quad \sum y'_u + \sum z'_v \tag{1}$$

$$\forall u, v, \exists E, \quad y'_u + z'_v \geq 0 \tag{2}$$

- Find a maximum matching in this graph using the "original" algorithm.
- Now, suppose there are 2n vertices, then there are 2 cases:
  - If that matching has size n, the weight of the matching is

$$\sum_{(u,v) \text{in the matching}} w_{uv} = \sum y_u + \sum z_v = \text{Dual Cost} \tag{3}$$

  - If the matching is of size t, which is less than n, find a vertex cover S of size t.

- Vertex Cover : **(Algorithm)**
  - Decrease the vetex label in (V - S) by δ
  - Increase the vertex label in S by same δ
  - There is a net decrease in the cost.
  - Take those vertices which are at odd distances from unmatched vertices (see in Figure 1.

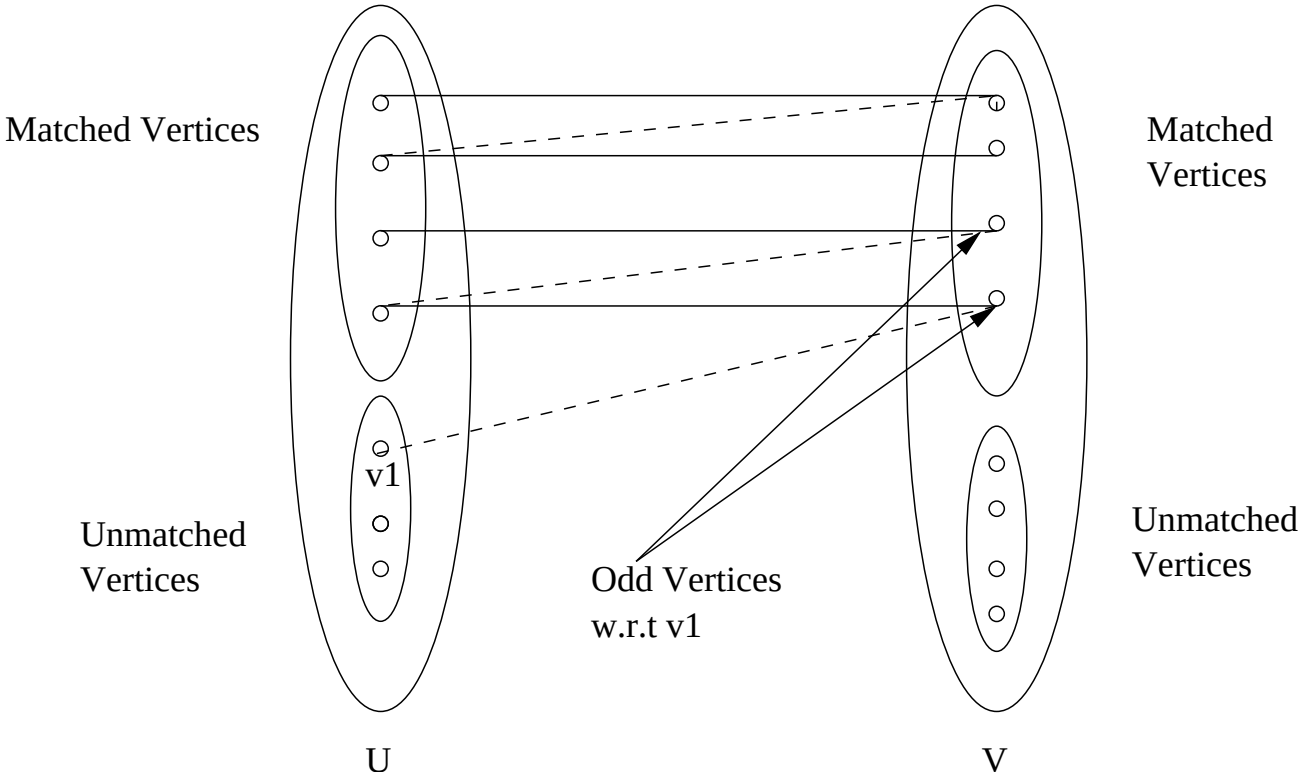

Figure 1: Showing the odd distant vertices w.r.t. unmatched vertex v1.

  - After this, there can be some matched edges for which neither vertices are taken.

- We also show that these decrease and increase in labels don't violate the constraints:

$$y_{\text{new}} = y_{\text{old}} + y' \tag{4}$$

$$z_{\text{new}} = z_{\text{old}} + z' \tag{5}$$

  - Because y_{old} and z_{old} are equal to corresponding w_{uv}, and y' and z' are positive, hence y_{new} and z_{new} are greater than equal to w_{uv}.

3 Time Complexity

Two Cases:

1. Either we increase the size of matching, or
2. we increase the size of the component.

So, by the end, atmost n^2 steps will be taken. Hence, the algo is $O(n^2)$, i.e., polynomial.

4 Few things to take over

1. At every step, we need not find maximal matching (because we could use the augmenting path from previous step).
2. This can be used for unweighted case also. See, how this looks overall.