We continue the discussion for bipartite graphs.

Theorem: For any bipartite graph $G = [S, T; E]$ the following statements are equivalent.

(i) $\exists$ no P.M in G

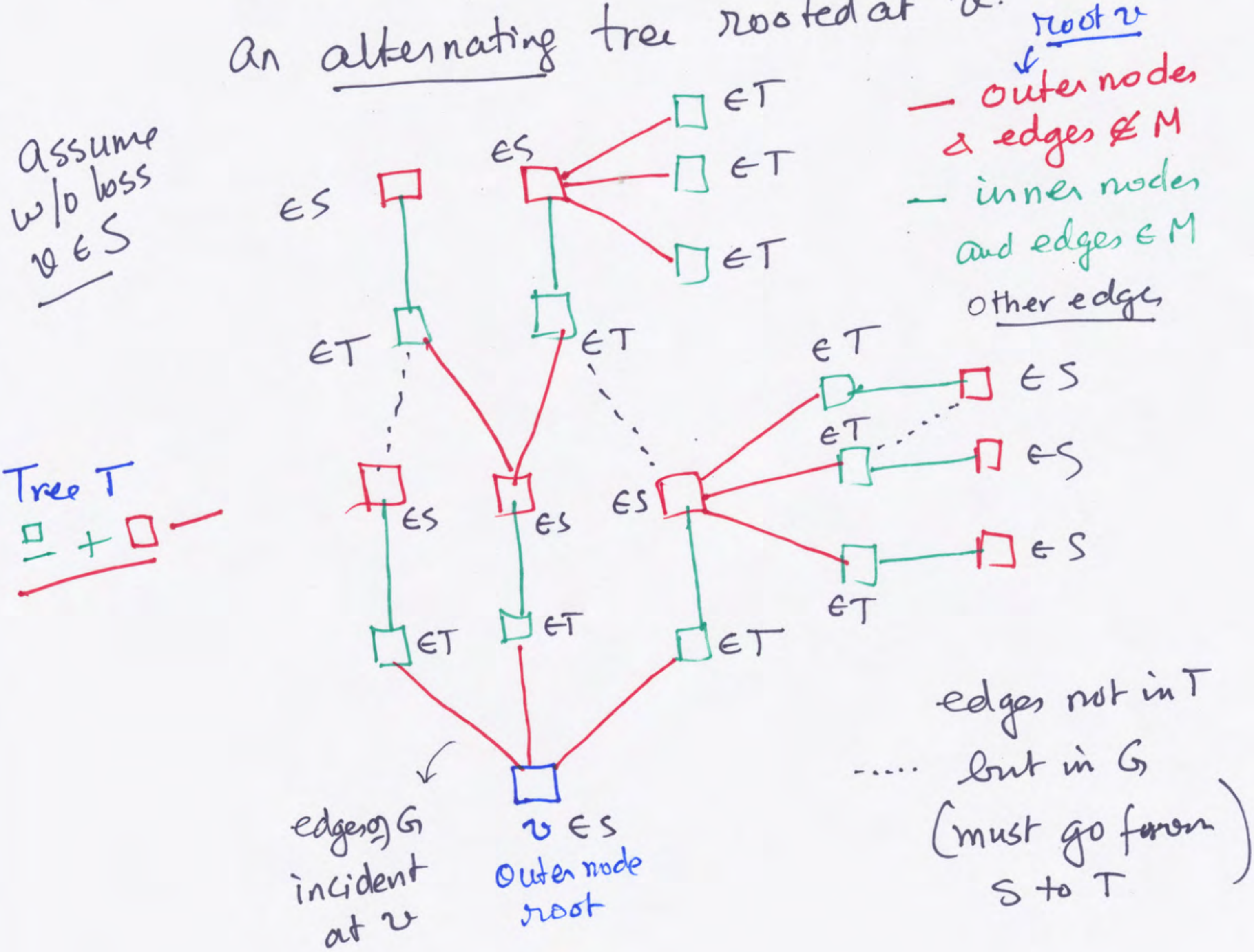
(ii) $\exists$ a Hungarian set H wholly contained in S or in T.

Pf: "Clearly" we have seen that ~~(i) ⇒ (ii)~~
(ii) $\Rightarrow$ (i)

Now we show (i) $\Rightarrow$ (ii) by showing an algorithm R1 which has two possible terminations: (a) it finds a PM in G
(b) " " a Hungarian set H That is wholly contained in S or in T.

This algorithm can also be thought of as one method of solving P1.

R1: Let us suppose in addition to input $G = [S, T; E]$ we have some (possibly empty) matching $M \subseteq E$. For the moment assume that M is not maximum Cardinality. We attempt to increase the size of the matching by trying to find an augmenting path relative to M. For this purpose, we select a "free" node v relative to M and "grow" an alternating tree rooted at v.



If any inner node of tree T is free or exposed, we get an augmenting path relative to M ~~since~~ and this path connects this node to the root of the tree and size of the Matching is increased. We repeat the process with the new M if it is not a PM.

Else, all leaf nodes of this tree are outer nodes and set of outer nodes forms a Hungarian set in the graph G.

Hence, the theorem follows.

Now we use this as a subroutine to solve P3 using ideas from Primal-Dual algorithms. We describe this on the next page.

Bipartite Version of P3:

replace by $x_{ij} \geq 0$ → $x_{ij} : \begin{cases} 1 & \text{if edge}(i,j) \text{ is in } M \\ 0 & \text{else} \end{cases}$

$$u_i : \sum_{j \in T} x_{i,j} = 1 \quad \forall\, i \in S$$

Unrestd

$$v_j : \sum_{i \in S} x_{ij} = 1 \quad \forall\, j \in T$$

(P) for bipartite graph

$$\text{Max } \cancel{\text{Max}} \sum_{j \in T} \sum_{i \in S} c_{ij} x_{ij}$$

(Graph: $G = [S, T; E]$; edgeweights c_{ij})

$$u_i + v_j \geq c_{ij} \quad \forall\, i \in S,\, j \in T$$

$$\text{Min}\left(\sum_{i \in S} u_i + \sum_{j \in T} v_j \right)$$

(P) : $x_e : \begin{cases} 1 & \text{if } e \in M \\ 0 & \text{else} \end{cases}$

for general graph.

$$y_i : \sum_{\substack{e:\ \text{incident} \\ \text{at node } i}} x_e = 1 \quad \forall\, i \in V$$

$$\text{Max} \sum_{e \in E.} c_e x_e$$

Algorithm (Primal-dual) for bipartite Case for P3.

Step 1: Start with (an easily found) feasible Solution for D.

Example [not restricted to this]

$$\text{Let } y_i = \max_{\substack{e:\,\text{incident}\\ \text{at } i}} C_e \quad \forall\, i \in V$$

(Can do much better; one such is given in my printed notes)

Step 2: Define $G^{eq} = [S, T; E^{eq}]$ where

$$E^{eq} = \{ e = (i,j) : \underbrace{y_i + y_j}_{f_e(y)} = c_e \}$$

Step 3: Find (what is readily available) ~~an~~ a ~~a~~ matching M in G^{eq}.

Step 4: If M is perfect, Stop; M is optimal to P and $\{y\}$ is opt. to Dual. Else goto 5

Step 5 Find a free node v in G^{eq} with respect to M and attempt to grow an alternating tree T rooted at v in G^{eq}.

Step 6: If an inner node of T is free, augment M and goto step 4.

Step 7: Else, T has all leaf nodes as outer nodes and none of the outer nodes are connected to any node not in T in G^{eq}.

Step 8: At this point, outer nodes of T is a Hungarian set in G^{eq}. If this is also a Hungarian set in G, G has no perfect matching → STOP.
P is infeasible. Else go to step 9.

Step 9: Change dual variables as follows:

$$y_i^{new} = \begin{cases} y_i^{old} + \epsilon & i \text{ inner node of } T \\ y_i^{old} - \epsilon & i \text{ outer node of } T \\ y_i & \text{else.} \end{cases}$$

where $\epsilon = \underset{\substack{e:\text{ with one end an outer node of T}\\ \text{and the other not in T}}}{\text{Min}} \left[f_e(y) - c_e \right] > 0$

and go to step 2.

Note : y^{new} is feasible to dual and all edges of M are still in new G^{eq} as are edges of the old Tree T. At least one edges of extra edge is now in new G^{eq} so that the tree can be "grown" further. The only edges that might be dropped from G^{eq} are ~~those~~ among those connecting inner nodes of the tree T and a node $\notin T$ that are not in either M or the tree T.

The new dual solution is an improved solution compared to the old.

This algorithm proves the following theorem.

Thm : For an input that has a perfect matching, algorithm produces such a matching and an optimal solution to both problems. P and D. Moreover if c_e are integral, optimal solution to D can also be chosen to be integral. If G has no PM, ~~opt~~. dual is unbounded.

P3 for General graph:

If we now try to solve the integer program as a linear program, we may not get an integer optimal solution. (We get multiples of 1/2). The trick is to add additional inequalities that are satisfied by all integer solutions but may not be satisfied by fractional solutions. Such inequalities are called Valid cuts.

The algorithms, prior to J. Edmonds' work added a valid cut After solving the LP One at a time. J. Edmonds was the first to think of the idea of adding all of these at once in the beginning.

This produces the Convex-hull of all PM incidence vectors right in the beginning So that LP Solution now gives an integer Solution. We describe this LP which now becomes our (P)

$$x_e = \begin{cases} 1 & \text{if } e \in M \\ 0 & \text{else} \end{cases} \rightarrow x_e \geq 0 \quad \forall e \in E$$

$$y_i : \quad \sum_{e:\ \text{incident at } i} x_e = 1 \quad \forall i \in V$$

unrestd

(P)

$$y(S) \geq 0 : \quad \sum_{e:\ \text{both ends in } S \subseteq V} x_e \leq \left\lfloor \frac{|S|}{2} \right\rfloor = q_S \quad \forall S \subseteq V.$$

Valid cuts

$$\text{Max} \sum_{e \in E} c_e x_e$$

$$y(S) \geq 0 \quad \forall S \subseteq V$$

(D)

$$f_e(y) = y_i + y_j + \sum_{\substack{S:\ i \in S \\ j \in S}} y(S) \geq c_e \qquad e = (i,j) \in E$$

$$\text{Min} \left[\sum_{i \in V} y_i + \sum_{S \subseteq V} q_S \, y(S) \right]$$

Now we use primal-dual algorithm to solve (P) and (D) and show that if G has a PM, we get optimal perfect matching for (P) and optimal y for (D).

This is done by generalizing our algorithm to yield a strongly polynomial algorithm to solve P3 for general graph.

Warning. This is a complicated algorithm.

Primal - Dual Algorithm for P3 for general graphs

Step 1: Find a feasible solution to the dual. For example: let $y(S) = 0 \ \forall S \subseteq N$; let y_i be chosen as in the bipartite case.

Step 2: Define $G^{eq} = [V; E^{eq}]$ where

$$E^{eq} : \{ e : e = (i,j);\ f_e(y) = y_i + y_j + \sum_{\substack{S:\\ i \in S\\ j \in S}} y(S) = c_e \}$$

Step 3: Using algorithm R1, try to find a perfect matching in G^{eq}. If one is found, then it is optimal to P3 and y is optimal to the dual.

Before going further, we describe R1

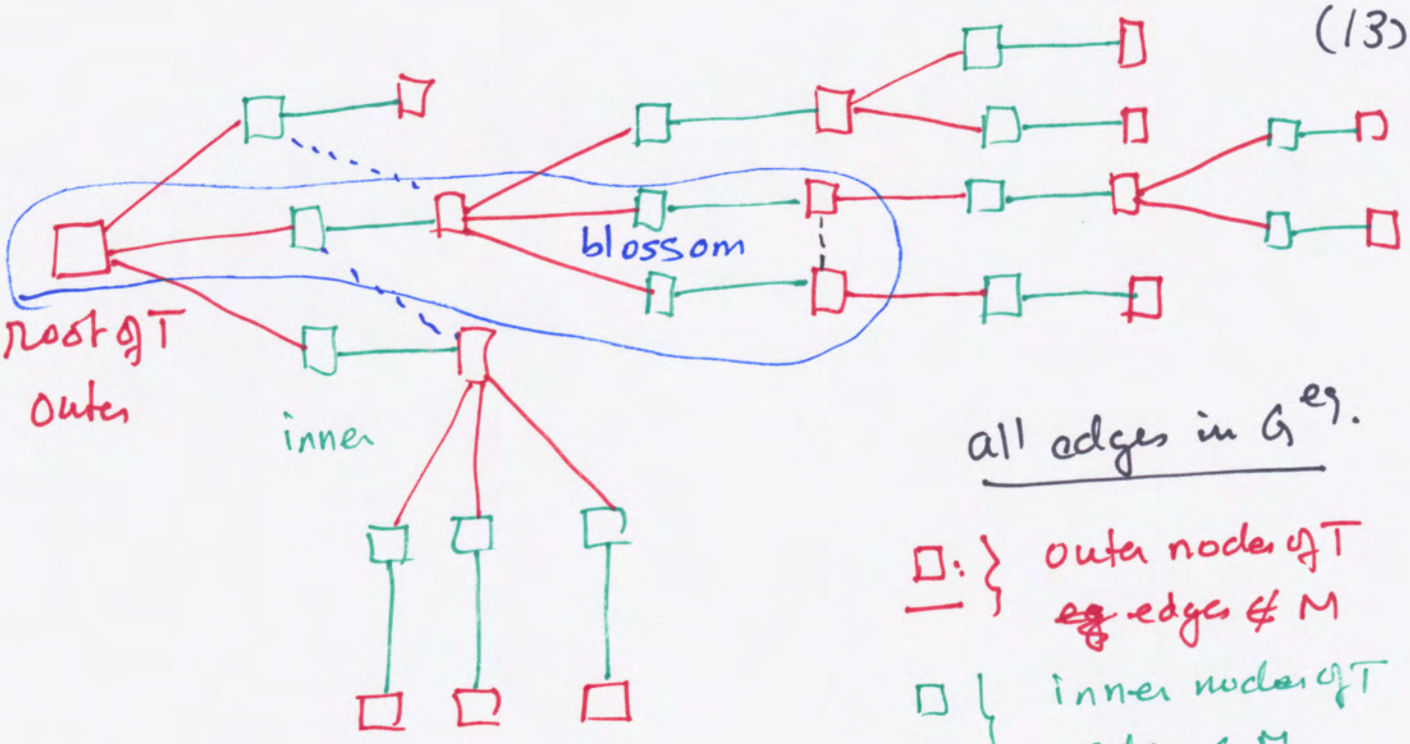
R2: is dual ~~variable~~ solution change and it needs the New R1 description.

R1: We start with an easily found (not necessarily perfect) matching M in G^{eq}, as before. If M is not perfect, we start "growing" an alternating tree T as before from a free node v as its root (outer node) with nodes alternating between outer and inner nodes. Also, edges alternatively are in M and not in M in every path from the root in the tree T as before. To expand the tree, we look for edges connecting outer nodes to nodes not in T ~~but~~ via edges in G^{eq}. If at any time, we have an inner node of T that is free, we get an augmenting path and we increase the size of the matching in G^{eq}. [This is our first preference, second being growing the tree T in size]. Unfortunately, for general graphs, this is not enough.

Whenever, the matching is augmented, the tree is dismantled and we start growing another tree from a new free node unless the new matching is perfect.

There is one more case which was not there in bipartite case. This occurs because, two Outer nodes of tree T <u>may be</u> connected by an edge in G^{eq}. (If this happens, the set of outer nodes <u>will not</u> form a Hungarian set in G^{eq}). This happens when G^{eq} has an <u>odd</u> cycle which is not possible if G is bipartite ($\Rightarrow G^{eq}$ is bipartite).

This situation is shown in the picture on the next page.



all edges in G^{eg}.

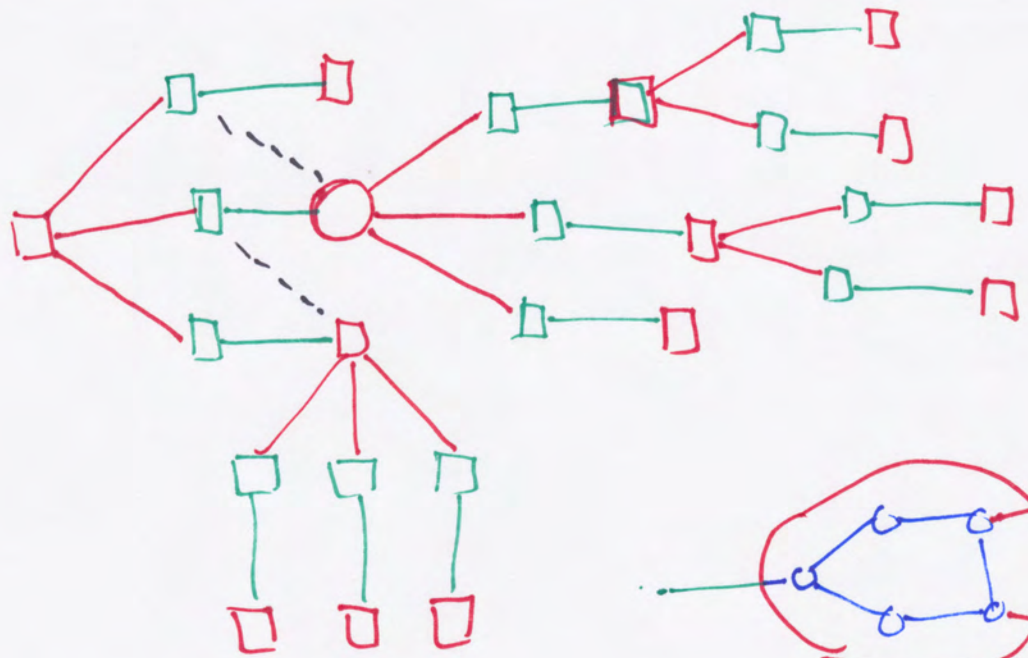
□: } outer nodes of T
— } ~~eg~~ edges $\notin M$

□ } inner nodes of T
— } edges $\in M$.

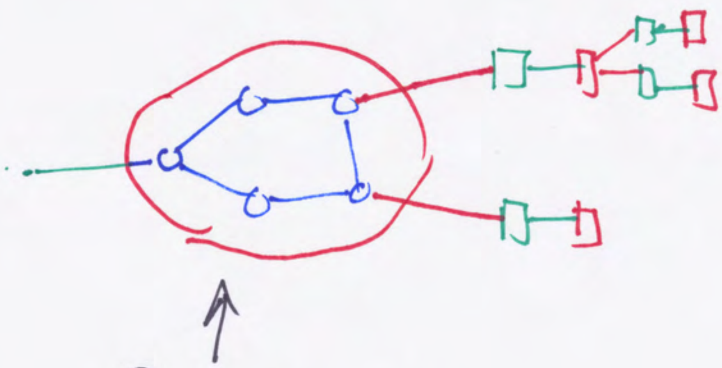
Other edges in G^{eg}.

Flower

Before "shrinking"



After shrinking



Pseudo-node

(14)

Shrinking process of an odd cycle, creates pseudo nodes. This is done on all four entities: G, G^{eq}, M, T to get shrunken version: G_S, G^{eq}_S, M_S, T_S. And we have to work with all of these. When we find an augmenting path and T is ~~dis~~ dismantled, we have inner & outer nodes in both real and pseudo nodes.

This is where the algorithm gets complicated. This is done in the next lecture.