| | | | |
|---|---|---|---|
| Optimization Methods | | Date: | *26/February/2018* |
| Instructor: *Sujit Prakash Gujar* | | Scribes: | ANKIT SARIN<br>K L BHANU<br>VASANT CHAITANYA |

# Lecture 13: Linear Programming - Maximum Weighted Perfect Matching and Branch & Bound method

## 1 Recap

1. Integer Programming : Typically, feasible region of integer program is not convex.
2. Application of integer programs are in entire class of problems referred to as sequencing, scheduling, and routing. Most of the graph theory problem are instances of Integer programming.
3. Totally unimodular Matrices : a matrix $A \in \mathbf{Z}^{mxn}$ is totally unimodular, if the determinant of each square sub-matrix of $A \in \{-1, 0, 1\}$
4. Max matching & minimum vertex cover on bipartite graph.
5. LP relaxation of min vertex cover and max matching are dual of each other.
6. Konig's theorem : size of max matching is same as size of minimum vertex cover.

## 2 Maximum Weighted Perfect Matching in bipartite Graphs

Let us consider the Maximum Weighted Matching problem in bipartite graphs, i.e Given a graph $G(V, E)$, $w_j$ being the weights of the $j^{th}$ edge, we choose a combination of non-adjacent vertices such that the sum of weights is maximized. Perfect matching, is when a matching matches all the vertices of the graph. So for our problem, we have $|X| = |Y|$ because, we need to cover all vertices, and an edge can only select one vertex from each set, therefore, to select all vertices, we need the number of vertices in each set to be equal. We assume that perfect matching exists and we are finding a perfect matching with maximum weight. The proof can be modified for general matching with some effort but we will skip that.
This is an Integer Linear Program, as we can't select partial edges. We can use the simplex algorithm to arrive at an integral solution, as the constraint matrix is unimodular in case of a bipartite graph (as seen in previous lecture), but the runtime of simplex algorithm may be exponential (not guaranteed to be polynomial), and we are looking for a heuristic for a faster integral solution. The ellipsoid method does give us a solution in polynomial time, but an integral solution is not guaranteed. So we are trying to come up with a method that will give us integral solution in polynomial time. The integer programming problem is as follows :

$$\text{maximize} \quad \sum_{j=1}^{m} w_j x_j$$

$$\text{subject to} \quad \sum_{\substack{\forall v, e\, is \\ incident\, on\, v}} x_e = 1, \quad x_e \in \{0, 1\}$$

We will consider its LP Relaxation which relaxes our constraint for $x_e$ to $x_e \in [0, 1]$. All other constraints remain the same. Now naturally a question arises, why do we use strict equality in the constraint $\sum x_e = 1$ instead of $\sum x_e \leq 1$. That is because we want a perfect matching. Now our LP relaxation becomes :

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{\substack{\forall v, e\, is \\ incident\, on\, v}} x_e = 1, \quad x_e \in [0,1] \end{aligned}$$

Let $(x_e^*)$ be the LP optimal solution. Now if $\forall e$ if $x_e^*$ is integer, we have an integral solution. If not, then let $k(x_e^*)$ be the number of non-integral assignments. Let $e_1 = (a_1, b_1), a_1 \in X, b_1 \in Y$ be an edge such that $x_{e1}$ is non-integer. Without loss of generality, let us look at a vertex, say $b_1$, it has to satisfy the constraint, so that means it has atleast one more edge $e_2 = (b_1, a_2), a_2 \in X$ such that $x_{e2}$ is non integer. Now, the same constraint applies to $a_2$ also, infering that $\exists e_3 = (a_2, b_2), b_2 \in Y$ and so on. But these series of edges has to terminate (of course we dont have infinite edges), which can possibly be only $a_1$ forming a cycle, ie $\exists e_t = (b_t, a_1), b_t \in Y$. Note that this cycle has to have even number of vertices, as this graph is a bipartite one and would take even number of traversals to start from $a_1$ and end at $a_1$. ie $t$ is even. Check the example given below to verify these statements.

Let $\tilde{x}$ be another feasible solution such that

$$\tilde{x} = \begin{cases} x_e^*, & \text{if } e \notin e_1, e_2, ...e_t \\ x_e^* - \epsilon, & \text{if } e \in e_1, e_3, ..., e_{t-1} \\ x_e^* + \epsilon, & \text{if } e \in e_2, e_4, ..., e_t \end{cases}$$

$$0 \leq \tilde{x} \leq 1 \tag{1}$$

$$\forall v \text{ e is incident on v} \tag{2}$$

Where $\epsilon$ is an arbitrary value. Our objective value becomes :

$$Z(\tilde{x}) = \sum_{e \notin e_1, e_2, .., e_t} w_e x_e^* + \sum_{e \in e_1, e_2, .., e_t} w_e (x_e^* \pm \epsilon) \tag{3}$$

That becomes

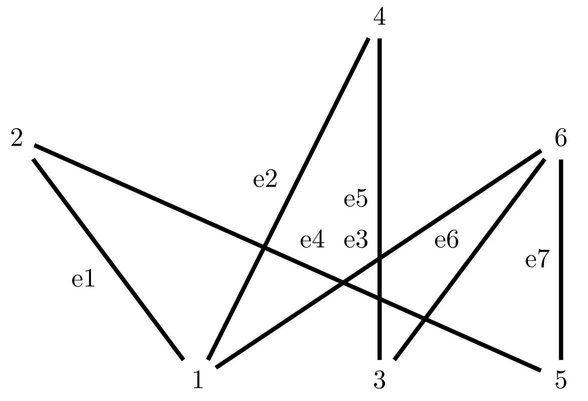
$$Z(\tilde{x}) = \sum_{e} w_e x_e^* + \sum_{i=1}^{t} (-1)^i \epsilon w_{e_i} \tag{4}$$

That is

$$Z(\tilde{x}) = \sum_{e} w_e x_e^* + \epsilon\delta \text{ where } \delta = \sum_{i=1}^{t} (-1)^i w_{e_i} \tag{5}$$

now as $\sum_e w_e x_e^*$ is the optimum objective value, and $\epsilon$ is an arbitrarily chose number, $\delta$ has to be equal to be zero, as $Z(\tilde{x})$ cannot exceed the LP optimum value. This allows us to change $\epsilon$ such that one of the edges of the cycle (with non-integral $x_e$) can be made integral. ie $k(\tilde{x}) < k(s_e^*)$. The process is repeated again, until all the edges have an integral $x_e$, ie $k(\tilde{x})$ becomes zero. The maximum number of times we might need to iterate this could be as many times the edges are present ie $O(|E|)$. Therefore, given a non-integral solution, we can always get an integral solution for this problem in polynomial time.

Let us consider an example, given a bipartite graph, and $x_e^*$, we will run this procedure until we get integral assignments of $x_e$ for all edges.



**A bipartite graph,** $X = 1, 3, 5$ **,** $Y = 2, 4, 6$

| Iter | Cycle | $\epsilon$ | $x_{e_1}$ | $x_{e_2}$ | $x_{e_3}$ | $x_{e_4}$ | $x_{e_5}$ | $x_{e_6}$ | $x_{e_7}$ |
|---|---|---|---|---|---|---|---|---|---|
| Initial | NA | NA | 0.1 | 0.4 | 0.5 | 0.9 | 0.6 | 0.4 | 0.1 |
| 1 | $e_1e_4e_7e_3$ | 0.1 | 0 | 0.4 | 0.6 | 1 | 0.6 | 0.4 | 0 |
| 2 | $e_2e_5e_6e_3$ | 0.4 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

Table 1: Instances of the algorithm to find integral solution to perfect bipartite matching

In the above example, we choose $\epsilon$ as the smallest $x_e$ in the cycle. Decrement $\epsilon$ for odd edges and increment for even edges in the cycle, this can be interchanged without loss of generality. Also, the same example can be used to verify that given an edge has non-integral $x_e$, it will always be a part of even sized cycle, with all it's edges having non-integral $x_e$.

## 3 A Heuristic approach to Integer program: Branch & Bound

Integer program optimization problems typically exponential in terms of time complexity and may require exploring all possible permutations in worst case.

A solution is to use **Brute Force** for solving such problems.
If number of variables $= n$
we will have $2^n$ assignments, check each to see if they satisfy the constraint, save maximum solution that satisfies constraint. This solution can be expressed as tree.
As of today a average computer can compute $2^{60}$ operations in an year.

*Can we come up with some heuristic which will remove some brute force?*

One solution is to use **Backtracking** to optimize the Brute Force solution. In the tree representation, we can do DFS(depth first search) of tree. If we reach a point where a solution no longer is feasible, there is no need to continue exploring.

We can do better (than backtracking), if we know a bound on best possible solution subtree rooted with every node. If the best in subtree is worse than current best, we can simply ignore this node and its subtrees. So we compute bound (best solution) for every node and compare the bound with current best solution before exploring the node.

**Branch & Bound** : Idea of Branch and bound is similar to LP relaxation.

1. Solve LP Relaxation
   if $x^* \in \mathbf{Z}^n$
   return $x^*$

2. If $\exists x_j^* \neq \mathbf{Z}^n$ (Integer solution)
   Add new constraints and split problem into two problem
   P1 : $x_j \leq \lfloor x_j^* \rfloor$ (floor of $x^*$)
   P2 : $x_j \geq \lceil x_j^* \rceil$ (Ceil of $x^*$)
   solve P1 and P2, reiterate on each branch until we get an optimal solution. STOP, if in any branch you get infeasible or impossible solution.

*Example :*

$$\begin{array}{lll} \text{maximize} & x_1 + x_2 & \\ \text{subject to} & & x_2 - x_1 \leq 0 \\ & & x_1 + x_2 \leq 3.5 \\ & & x_1, x_2 \geq 0 \end{array}$$

*Solution :*

With reference to *figure 1*, which shows feasible solution area defined by constraints and intuitively there are 5 integer solutions, IP solution space (P0) $\in \{(1,1)(1,2)(1,0)(2,0)(3,0)\}$

Problem (P0) : Optimal solution $\rightarrow (1.75, 1.75) \not\in \mathbb{Z}$
Maximum objective value = 3.5.
As this is not an integer solution, we will divide problem into two parts, selecting $x_1$ randomly because both the variables are non-integers.

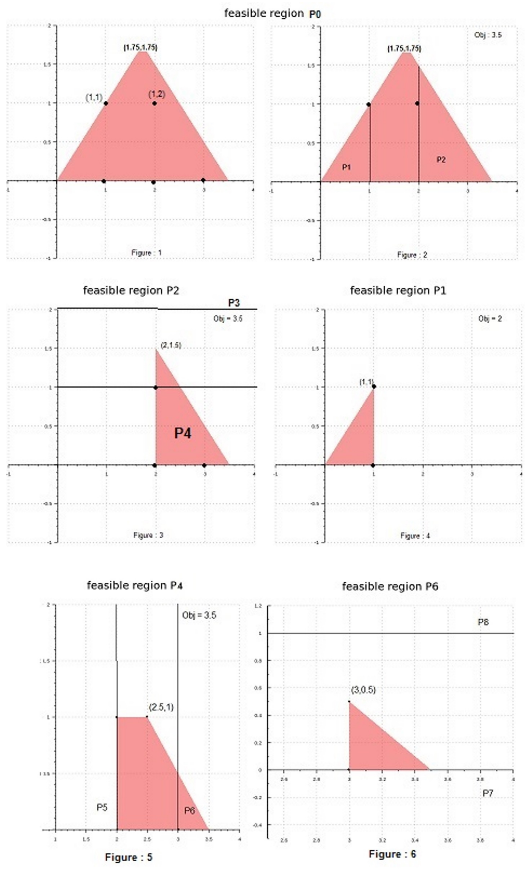


Figure 1: Graphical solution of example

Problem 1 (P1) : $x_1 \leq \lfloor 1.75 \rfloor \Rightarrow x_1 \leq 1$
Graphically optimal solution $\rightarrow (1, 1) \in \mathbf{Z}$
Maximum objective value $= 2$

Problem 2 (P2) :(refer 3rd graph in *figure 1*)
$x_1 \geq \lceil 1.75 \rceil \Rightarrow x_1 \geq 2$
Graphically optimal solution $\rightarrow (2, 1.5)$
Maximum objective value $= 3.5$. We can see that Obj(P2) $\geqslant$ Obj(P1) and optimal solution of P2 is non Integer, hence we need to select P2 and divide it into two problems P3 and P4.

Select $x_2$ because $x_2 \notin \mathbf{Z}$
Problem 3 (P3) : $x_2 \geq \lceil 1.5 \rceil \Rightarrow x_2 \geq 2$
Graphically optimal solution $\rightarrow$ Infeasible because of new constraint $x_2 \leq 1$
Maximum objective value $=$ Not Applicable
We don't have to continue further in this branch.

Problem 4 (P4) :(refer 5th graph in *figure 1*)
$x_2 \leq \lfloor 1.5 \rfloor \Rightarrow x_2 \leq 1$
Graphically optimal solution $\rightarrow (2.5, 1)$
Maximum objective value $= 3.5$. We can see that Obj(P4) $=$ Obj(P0) and optimal solution of P2 is non Integer, hence we need to select P4 and divide it into two problems P5 and P6.

Now, select $x_1$ because $x_1 \notin \mathbf{Z}$

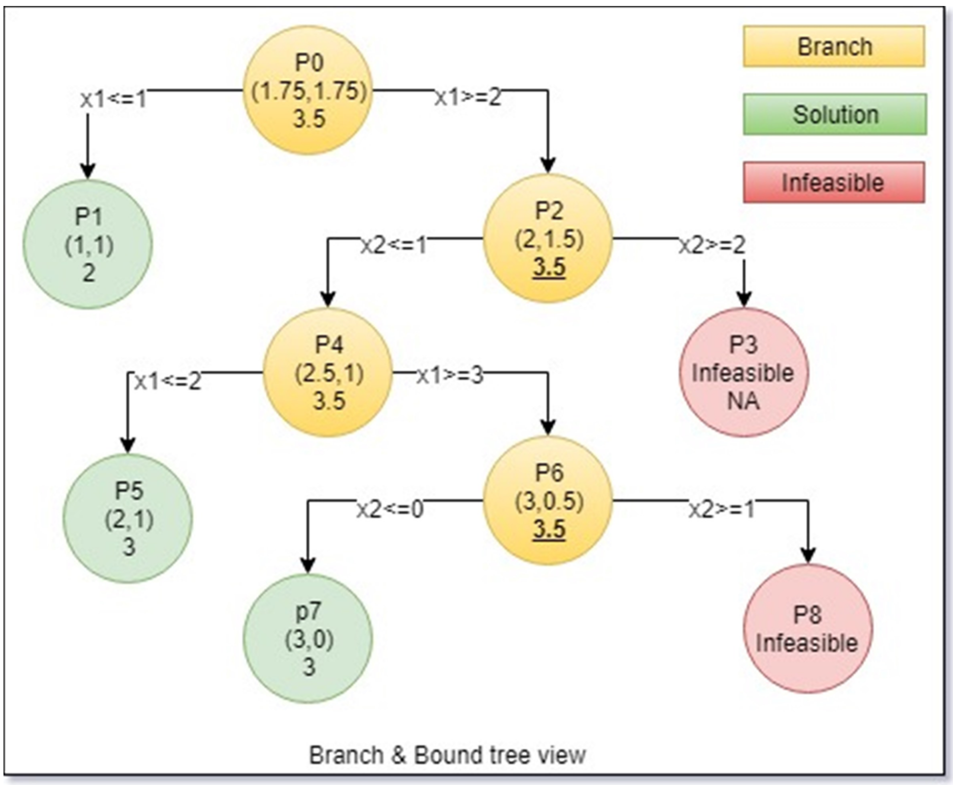


Figure 2: Tree view

Problem 5 (P5) : $x_1 \leq \lfloor 2.5 \rfloor \Rightarrow x_1 \leq 2$
P2 has constraint $x_1 \geq 2$ hence above constraint become $x_1 = 2$
Graphically optimal solution $\rightarrow$ (2,1) $\in \mathbf{Z}$
Maximum objective value = 3

Problem 6 (P6) : $x_1 \geq \lceil 2.5 \rceil \Rightarrow x_1 \geq 3$
Graphically optimal solution $\rightarrow (3, 0.5)$, (refer 6th graph in *figure 1*)
Maximum objective value = 3.5. We can see that Obj(P5) $\leqslant$ Obj(P6) and optimal solution of P6 is non Integer, hence we need to select P6 and divide it into two problems P7 and P8.

Select $x_2$ because $x_2 \notin \mathbf{Z}$
Problem 7 (P7) : $x_2 \leq \lfloor 0.5 \rfloor \Rightarrow x_2 \leq 0$
P0 has constraint $x_2 \geq 0$ hence above constraint become $x_2 = 0$
Graphically optimal solution $\rightarrow$ (3,0) $\in \mathbf{Z}$ & Maximum objective value = 3

Problem 8 (P8) : $x_2 \geq \lceil 0.5 \rceil \Rightarrow x_2 \geq 1$
Graphically optimal solution $\rightarrow$ Infeasible because of new constraint $x_2 \geq 1$
Maximum objective value = Not Applicable. We don't have to continue further in this branch.

We found that P5 & P7 gave us max. optimal value 3 hence integer optimal solution can be (2,1) or (3,0).

We divided problem into tree of subproblems (*figure 2*) P1 to P8 and we stopped if we get an integer solution or infeasible sol.

**References**

1. `https://www.draw.io/` for tree drawing

2. `https://quickmath.com/webMathematica3/quickmath/graphs/inequalities` for graph plotting

The End