Optimization Methods		Date:	*27 Feb 2018*
Instructor: *Sujit Prakash Gujar*		Scribes:	Aniruddha Patil Murtuza Bohra

Lecture 14: Computing Integral Optimal Solution

1 Recap

In last lecture we had seen, from LP relaxation, how to reach integer optimal solution of maximum weighted matching problem in bipartite graph. For bipartite graph, if $\exists$ a non-integral optimal solution such that for some decision variable $0 < x_e^* < 1$, then there will be a cycle in which all the decision variable corresponding to edges of the cycle are strictly between 0 and 1. So we substract small value ϵ from odd number edges in the cycle and add the same ϵ in even number edges, this lead us to an optimal integral solution in polynomial time.

To solve a general integer optimization problem, we solve it's LP relaxed problem and then compute nearest integral solution. We briefed branch and bound technique in last lecture, which can be used to compute an integral optimal solution from it's LP relaxed optimal solution. In this lecture we will see detailed branch and bound algorithm and also Balas algorithm for computing integral optimal solution.

2 Branch and Bound

2.1 Introduction

It is an "Divide and Conquer" approach to explore the set of feasible integer solutions. The "bound" part is done by estimating how good a solution we can get for each smaller problems. The optimal value from a subproblem will tell us whether there is a need to further divide or not-

Branch step: The bound on the optimal cost of a subproblem is obtained by solving the LP relaxation of the sub problem.

Divide Step: If the optimal solution x to the LP relaxation is not integer, we choose a component x_i for which x_i^* is not integer and create two subproblems, by adding either of the constraints -

$$\begin{aligned} x_i &\leq \lfloor x_i^* \rfloor, or \\ x_i &\geq \lceil x_i^* \rceil \end{aligned}$$

2.2 Algorithm:

let x^- is vector which contains the best solution of original optimization problem till i^{th} iteration, similarly l^- is the best objective value achieved till i^{th} iteration. Branch and bound algorithm ends when x^- is integral solution and l^- is the best optimal value of objective function among all integral solution of our original optimization problem.

```
Step 1: $x^- = \emptyset$ ($x^-$ is initialized to empty set)
        $l^- = -\infty$
Step 2: Problem $P$ = maximize $c^Tx$ such that $Ax \leq b,\ x \geq 0$
        PUSH ($P$, LIST)
Step 3: while (LIST $\neq \emptyset$)
         a: Problem = POP(LIST)
         b: Solve Problem
         c: If $x^* \in \mathbb{Z}^n$ and $C^Tx^* > l^-$
            $x^- = x^*$, $l^- = C^Tx^*$
         d: Else If $\lfloor C^Tx^* \rfloor > l^-$
            $\exists x_j \in \mathbb{Z}$
            Problem $P_1$: Problem $P \cup x_j \leq \lfloor x_j^* \rfloor$
            Problem $P_2$: Problem $P \cup x_j \geq \lceil x_j^* \rceil$
            PUSH($P_1$, LIST)
            PUSH($P_2$, LIST)
```

2.3 Example

Given the following problem, obtain an integer solution.

$$\begin{gathered} maximize\ x_1 + x_2 \\ \text{Such that} \\ x_2 \leq x_1 \\ x_2 \leq 35.5 - x_1 \\ x_1, x_2 \ \epsilon\ \mathrm{Z} \\ x_1 - x_2 \geq 3 \end{gathered}$$

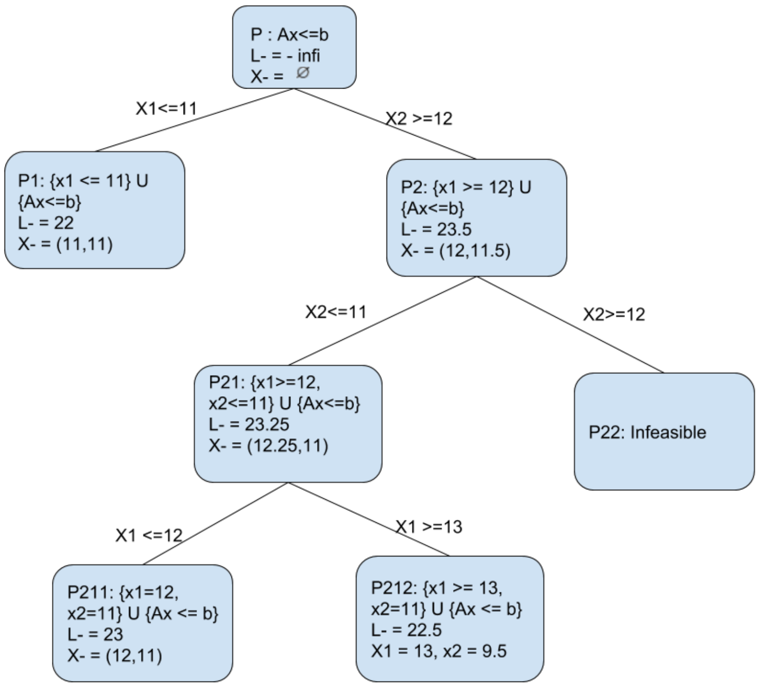

Figure 1: Tree that illustrates how Branch and Bound algorithm comes up with a optimal integral solution

3 Balas Algorithm

Balas Additive algorithm is a special branch and bound algorithm given by Egon Balas in 1965 for solving binary integer programs. It requires that the problem be put into some standard form which, for a binary integer problem is:

1. Objective function is of the form $min\ c^T x$
2. All the constraints have inequalities of $Ax \geq b$
3. All of the x_i where $i = 1, 2, ..., n$ are binary variables ($x_i \ \epsilon$ {0,1} $\forall\ i = 1, 2, ..., n$).
4. All objective function coefficients are non-negative
5. The variables are ordered according to their objective function coefficients so that $0 \leq c_1 \leq c_2 \leq ... \leq c_n$ (This can be done by introducing y_i variables such that $x_i = 1 - y_i$ as we shall show in a following example)

3.1 Intuition

- The objective function is to be minimized and all the coefficients are non-negative, so we would tend to set all the variables to zero to give the smallest value of the objective function.
- If we cannot set all of the variables to zero without violating one or more constraints, then we prefer to set the variable that has the smallest index to 1. This is because the variables are ordered so that those earlier in the list increase the objective function by the smallest amount.

3.2 Example

3.2.1 Converting to the required form

If a problem has objective $max\ c^T x$, $max\ c^T x$, it is equivalent to $min\ -c^T x$, however, the condition of $max\ 8x_1 + 11x_2 + 6x_3 + 4x_4$ may not be satisfied. The following example shows how any BIP can be converted to this standard form.

$$\begin{gathered} max\ 8x_1 + 11x_2 + 6x_3 + 4x_4 \\ \text{Such that} \\ 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ x_i\ \epsilon\ \{0,1\}\ \forall\ i = 1, 2, ..., 4 \end{gathered}$$

Since $max\ 8x_1 + 11x_2 + 6x_3 + 4x_4 \equiv min\ -8x_1 - 11x_2 - 6x_3 - 4x_4$, but there are negative coefficients, we do $x_i \rightarrow 1 - y_i\ \forall\ x_i$ with negative coefficients to give

$$\begin{gathered} min\ 8y_1 + 11y_2 + 6y_3 + 4y_4 \\ \text{Such that} \\ 5y_1 + 7y_2 + 4y_3 + 3y_4 \geq 5 \\ y_i\ \epsilon\ \{0,1\}\ \forall\ i = 1, 2, ..., 4 \end{gathered}$$

Now we rearrange the coefficients to satisfy the condition of $0 \leq c_1 \leq c_2 \leq ... \leq c_n$ to give

$$\begin{gathered} z_1 = y_4, z_2 = y_3, z_3 = y_1, z_4 = y_2 \\ min\ 4z_1 + 6z_2 + 8z_3 + 11z_4 \\ \text{Such that} \\ 5z_3 + 7z_4 + 4z_2 + 3z_1 \geq 5 \\ z_i\ \epsilon\ \{0,1\}\ \forall\ i = 1, 2, ..., 4 \end{gathered}$$

This formulation satisfies all conditions for Balas algorithm.

3.2.2 Using the algorithm

Consider the following example:

$$\begin{gathered} min\ 3x_1 + 5x_2 + 6x_3 + 9x_4 + 10x_5 + 10x_6 \\ \text{Such that} \\ -2x_1 + 6x_2 - 3x_3 + 4x_4 + x_5 - 2x_6 \geq 2 \\ -5x_1 - 3x_2 + x_3 + 3x_4 - 2x_5 + x_6 \geq -2 \\ 5x_1 - x_2 + 4x_3 - 2x_4 + 2x_5 - x_6 \geq 3 \end{gathered}$$

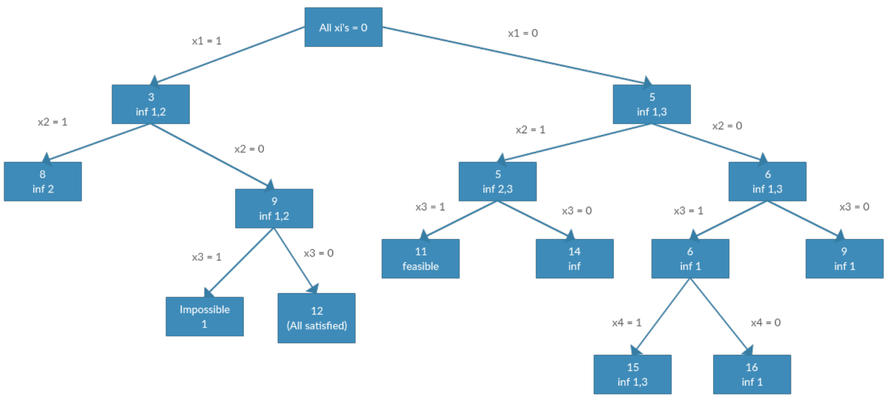

Figure 2: Tree that illustrates how Balas algorithm comes up with a feasible solution (inf. means infeasible)

Following is the explaination for computation of each node in above example in Bala's algorithm, and different condition to check before exploring a node further-

1. **Infeasibility at a node:** At a particular node, when certain variable are assigned value either 0 or 1 and rest of the unexplored variables are 0, then at that node if we see some constraints are violated, we list that set of constraints and call this node infeasible because of this set of constraints. e.g. In above example, at first left node from root, $X = (1, 0, 0, 0, 0, 0)$ is a infeasible solution because constraints 1 and 2 are violated.

2. **Impossible to branch further:** At a particular node, when certain variable are assigned value either 0 or 1 and rest of the unexplored variables are 0, but we obeserved that no matter whatever value these unexplored variables take while futher exploring into that branch, few constraints will always be violated. In that case we do not explore futher into that branch and call that particular node as **Impossible**. e.g. In the left subtree from the root, the leaf node labeled as impossible because of constraint 1, this is because at that node $X = (1, 0, 1, 0, 0, 0)$ is infeasible solution because of constraint 1, not only that but if we explore further for x_4, x_5 and x_6, no matter what values these variables take constraint 1 will always be violated.

3. **Stop Branching when:** we get a feasible solution in that branch because exploring futher will only increase the objective value, or we stop when a node is impossible to explore futher as mentioned above. e.g.In the above example, at $X = (1, 0, 0, 0, 0, 0)$ in the left sub-tree, we found a feasible solution, so we stop there because if we explore further than x_4, x_5 and x_6 will take values 1, which in turn increase objective value.