CS880: Approximation and Online Algorithms	**Scribe:** Jennifer Cao
Lecture 7: Iterative Rounding of LPs	**Date:** 09/30/2019

Last time we talked about applying LP relaxation and rounding methodology. By solving the linear programming relaxation, and round the possibly fractional solution of the LP to an integer solution, we can finally get a solution within an integer gap.

In this lecture, we will introduce iterative rounding by looking into three examples, *Bipartite Matching*, *Generalized Assignment Problem* and *Degree Bounded Spanning Tree*. In general, the iterative algorithm typically include one or more rounding and relaxation steps, and we can reduce the problem to a residual version at each iteration. By recomputing an optimal extreme point solution and iterating in this manner, we are able to set all variables when iteration ends.

7.1 Bipartite Matching

Consider the perfect matching problem with minimum cost in bipartite graphs in figure 7.1.1. Given a bipartite graph $(U \cup V, E)$, where E is edge set with a size of $|E| = M$. Each edge $e \in E$ has a cost c_e, and the goal is to find a matching with min-cost.

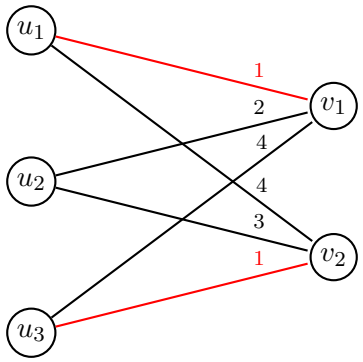

Figure 7.1.1: Bipartite Matching Problem. In this graph, edges (u_1, v_1) and (u_3, v_2) are chosen because it can obtain the minimum cost by 2.

7.1.1 Linear Programming Relaxation

An LP relaxation of this problem is as follows:

$$\begin{aligned} \min \quad & \sum_e c_e \cdot x_e \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in U \cup V \\ & x_e \geq 0, \forall e \in E \end{aligned}$$

where for edge $e = \{u, v\}$, x_e denotes whether vertices u and v are matched or not.

7.1.2 Characterization of Vertex Solutions

Before charactering vertex solutions for bipartite matching problem, we state Lemma 7.1.1 first.

Lemma 7.1.1 (Rank Lemma[1]) *Let $P = \{x : Ax = b, x \geq 0\}$ and let x be an extreme point solution of P such that $x_i > 0$ for each i. Then the number of variables is equal to the number of linearly independent tight constraints of A, i.e., the rank of A.*

Intuitively, Rank Lemma says that given any polytope in m dimensions, then any extreme point of this polytope is given by the intersection of m linearly independent tight constraints.

By Rank Lemma 7.1.1, bipartite matching polytope $P = \{x_e \geq 0 : x_e(\delta(v)) \leq 1, \forall v\}$ has following nice properties:

Lemma 7.1.2 *Given any vertex solution x to bipartite matching problem such that $x_e > 0$ for each $e \in E$, there exists $W \subseteq V$ such that*

- *$x(\delta(v)) = 1$ for each $v \in W$.*
- *the vectors in $\{\chi(\delta(v))|v \in W\}$ are linearly independent. $\chi(\delta(v)) \in E$ denotes 1 when e is an edge in $\delta(v)$, and 0 otherwise.*
- *$|W| = |E|$.*

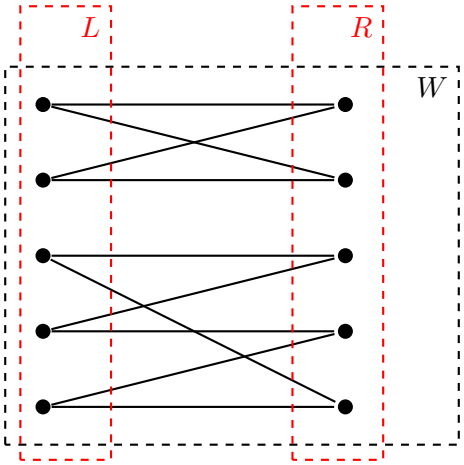

Figure 7.1.2: Intuition of bipartite matching polytope properties. $\sum_{v \in W \cup L} \sum_{e \in \delta(v)} x_e = \sum_{v \in W \cup R} \sum_{e \in \delta(v)} x_e$, these vertices are linearly independent.

Lemma 7.1.2 can deduce another nice property below, which is the skeleton of the iterative algorithm.

Lemma 7.1.3 *At any vertex of the bipartite matching polytope P, at least one x_e is 0 or 1.*

Proof: Let $W \subseteq V$ that are "tight", i.e., $\sum_{e \in \delta(v)}(x_e) = 1$, $\forall v \in W$. By Lemma 7.1.2, the constraints corresponding to W are linearly independent and tight, and $|W| = |E|$.

Suppose $x_e \neq 0$ and $x_e \neq 1$ for any edge e, then $0 < x_e < 1$. The constraints of the LP imply that each vertex in V must have at least two edges incident on it; so v's degree $deg(v) \geq 2$, $\forall v \in W$. This means

$$2|E| \geq \sum_{v \in V} deg(v) \geq \sum_{v \in W} deg(v) \geq 2|W| = 2|E|$$

This implies all inequalities hold as equalities, thus $deg(v) = 0$ for each $v \notin W$ as $\sum_{v \in W} deg(v) = 2|W|$, and $deg(v) = 2$ for each $v \in W$ as $\sum_{v \in V} deg(v) = \sum_{v \in W} deg(v)$. ■

If $x_e = 0$, we delete edge e from the graph. Otherwise we include it in our matching and delete the end point of e from the graph. By iteration we can end up with a minimum cost perfect matching solution.

7.2 Generalized Assignment Problem

Another application of iterative rounding is Generalized Assignment Problem in figure 7.2.3.

Given a set of jobs $J = \{1, 2, ..., n\}$, a set of machines $M = \{1, 2, ..., m\}$. A machine can run multiple jobs untile it hits the deadline T_i. If job j runs on machine i, it involves a running time p_{ij} and a cost c_{ij}, and we assume that $\forall (i, j) \in E$, $p_{ij} \leq T$. In generalized assignment problem, our goal is to find a min-cost schedule that respects deadlines.

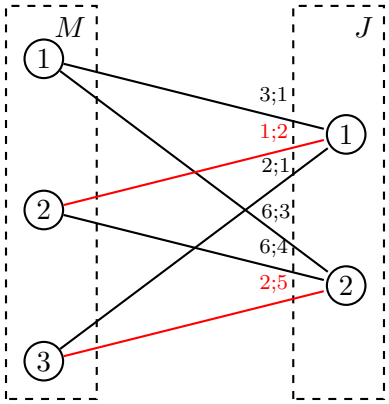

Figure 7.2.3: Generalized Assignment Problem. Each pair of $c_{ij}; p_{ij}$ represents cost and running time for each edge. In this example, we can achieve the goal by assign job 1 to machine 2, and job 2 to machine 3.

7.2.1 Linear Programming Relaxation

To obtain the linear programming relaxation form, we first model the problem as a bipartite matching problem. Recall the previous section, in generalized assignment problem, job set J and machine set M can be treated as two sides in bipartite matching graph, and each edge has a cost of c_{ij}. Moreover, $deg(j) = 1$ as each job has to be executed on exactly one machine. Hence we reduce the problem into finding a subgraph F of $G = M \cup J$ such that $\sum_{e \in \delta(i) \cap F} p_{ij} < T_i$, $\forall i$.

Thus we can give the following LP formulation:

$$\begin{aligned} \min \quad & \sum_{ij} c_{ij} \cdot x_{ij} \\ \text{subject to} \quad & \sum_{i:(i,j)\in E} x_{ij} = 1, \quad \forall j \\ & \sum_{j:(i,j)\in E} x_{ij} \cdot p_{ij} \le T_i, \quad \forall i \\ & x_{ij} \ge 0, \quad \forall (i,j) \in E \end{aligned}$$

7.2.2 Iterative Algorithm

In this section, we presents an algorithm that returns an assignment with optimal cost. Details is as below:

Iterative Generalized Assignment Algorithm

1. Initialize $F := \emptyset$, $\tilde{M} := M$, $T_i := T$ for all machines i.
2. Solve the LP to obtain a vertex solution x.
3. Consider 3 different cases.
   (a) If $x_{ij} = 0$ for some (i, j), remove (i, j) from E.
   (b) If $x_{ij} = 1$ for some (i, j), update $F \leftarrow F \cup \{(i, j)\}$, $J \leftarrow J \backslash \{j\}$, $T_i \leftarrow T_i - P_{ij}$.
   (c) **(Relaxation)** If there is a machine i such that $deg(i) \le 1$ or $deg(i) \le 2$ and $\sum_j x_{ij} \ge 1$, then drop i's constraint, remove i from $\tilde{M}$, i.e., $\tilde{M} \leftarrow \tilde{M} \backslash \{i\}$.
4. if $E \neq \emptyset$, return to 2.
5. Output F.

The following lemma shows we never increase LP's cost and the algorithm makes progress at each step of the algorithm. Hence this iterative algorithm can terminate in linear number of steps with an assignment.

Lemma 7.2.1 (Shmoys-Tardos[2]) *At any extreme point of Generalized Assignment Problem, at one of the following cases holds:*

(a) $x_{ij} = 0$ *for some* $(i, j) \in E$*;*

(b) $x_{ij} = 1$ *for some* $(i, j) \in E$*;*

(c) $degree_E(i) = 1$ *for some* i*;*

(d) $degree_E(2) = 1$ *and* $\sum_{j:(i,j)\in E} x_{ij} \ge 1$ *for some* i*.*

Additionally, in case (d) we can prove that additional load over each machine i is bounded by T_i

$$\text{increase in load on } i \leq \sum_{j:(i,j)\in E} (1 - x_{ij})p_{ij} \leq \sum_{j:(i,j)\in E} (1 - x_{ij})T_i \leq T_i$$

This lemma shows that in step 2 of the above algorithm, one of the four cases must happen and either $|E|$ or $|\tilde{M}|$ decreases. Hence it can terminate in linear number of steps.

Proof: By properties of extreme point solution, any extreme point is given by intersection of $|E|$ linearly independent constraints.
Let J denotes a set of jobs. Therefore, for $I \subseteq \tilde{M}$, $|J| + |I| = |E|$.
If there exists $x_{ij} \in 0, 1$, then the proof is complete. Suppose (a)(b)(c) do not hold, we have

$$2|E| \geq \sum_{i\in I} deg(i) + \sum_{j\in J} deg(j) \geq 2(|J| + |I|) = 2|E|$$

which indicates all $i \in I$ and $j \in J$ have degree of 2, while everything else has degree of 0.
Because

$$\sum_{i\in I}\sum_{j} x_{ij} = \sum_{j\in J}\sum_{i} x_{ij} = |J| = |I|$$

We can conclude that there must exists a machine $i \in I$ such that $\sum_j x_{ij} \geq 1$. Hence (d) holds as long as (a)(b)(c) don't, that complete the analysis. ■

7.3 Degree-Bounded Spanning Tree

Given a graph $G(V, E)$ with a cost c_e on edges $e \in E$ and degree bounds B_v for each $v \in W$, which is a subset $W \subseteq V$. The goal of a Degree-Bounded Spanning Tree is to find a minimum cost spanning tree T such that $deg_T(v) \leq B_v$, $\forall v \in W$.

7.3.1 Linear Programming Relaxation

First, we can define a variable for each edge that indicates whether or not this edge in part of the tree:

$$\begin{aligned} x_e &= 1 \quad \text{if edge } e \text{ is in tree} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Consider spanning tree as a minimal 1-edge-connected subgraph, then each cut must have at least one edge. Since our goal is to find a min-cost spanning tree, we can write the LP relaxation formulation as

$$\begin{aligned} \min \quad & \sum_{e\in E} c_e \cdot x_e \\ \text{subject to} \quad & \sum_{e\in\delta(v)} x_e \geq 1, \quad \forall v \in V \\ & \sum_{e\in\delta(v)} x_e \leq B_v, \quad \forall v \in W \\ & x_e \geq 0, \quad \forall e \in E \end{aligned}$$

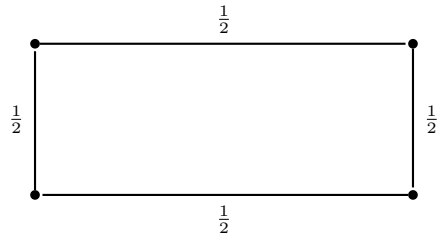

Figure 7.3.4: Undirected LP fails for Degree-Bounded Spanning Tree. Give a cycle with 4 vertices, any spanning tree needs 3 edges. However, if we set $x_e = \frac{1}{2}$, the formulation gives a cost of only 2.

However, this formulation fails in some occasions. Figure 7.3.4 provides a counter case.
Thus, we consider an alternative formulation as below

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e \cdot x_e \\
\text{subject to} \quad & \sum_{e \in E(S)} x_e \leq |S| - 1, \quad \forall S \subseteq V, S \neq \emptyset \\
& \sum_{e \in E(V)} x_e \leq |V| - 1 \\
& \sum_{e \in \delta(v)} x_e \leq B_v, \quad \forall v \in W \\
& x_e \geq 0, \quad \forall e \in E
\end{aligned}$$

Separations over these constraints can be carried out in polynomial time.

7.3.2 Iterative Algorithm

The following lemma can ensure that the algorithm terminates.

Lemma 7.3.1 *For any extreme point of degree-bounded spanning tree polytope, one of the following holds:*

(a) There exists an edge e such that $x_e = 0$.

(b) There exists an edge e with $x_e = 1$.

(c) There exists a vertex $v \in V$ with $deg_E(v) \leq 1$.

(d) There exists a vertex $v \in V$ with $deg_E(v) \leq 3$.

Also, characterization of extreme point solution in this polytype satisfies the following lemma.

Lemma 7.3.2 *Let x be an extreme point solution to degree-bounded spanning tree polytope. Then there exists a laminar family $L \subseteq 2^v$ and a subset of nodes $Z \subseteq W$ such that*

(a) The constraints corresponding to $s \in L$ and $v \in Z$ are tight.

(b) The constraints corresponding to L and Z are linearly independent.

(c) $|L| + |Z| = |E|$.

We are going to present the proof for both of the two lemmas in the next lecture.

Iterative Degree-Bounded Spanning Tree Algorithm

1. Initialize F as $\emptyset$.
2. While $|V(G)| \geq 2$, repeat
   (a) Find an optimal extreme point solution x.
   (b) if there exists $x_e = 0$, discard the edge from G.
   (c) if there exists $x_e = 1$ or a vertex with $deg_E(v) \leq 1$, include the edge into F. Update $F \leftarrow F \cup \{e\}$, $G \leftarrow G \backslash \{v\}$, $W \leftarrow W \backslash \{v\}$, and set $B_v \leftarrow B_v 1$.
   (d) **(Relaxation)** If there exists a vertex $v \in W$ with $deg_E(v) \leq 3$, then drop v from W since we should never increase cost. Update $W \leftarrow W \backslash \{v\}$.
3. return F.

This iterative polynomial-time algorithm returns a tree of optimal cost and only violates the degree bound within an additive error of two.

References

[1] L. C. Lau, R. Ravi, and M. Singh, *Iterative methods in combinatorial optimization.* Cambridge University Press, 2011, vol. 46.

[2] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical programming*, vol. 62, no. 1-3, pp. 461–474, 1993.