Approximation Algorithms and Hardness of Approximation March 15, 2013

Lecture 8

Lecturer: Ola Svensson *Scribes: Slobodan Mitrović*

1 LP iterative rounding

In the previous lecture we were exploring properties of extreme point solutions, and showed how those properties can be used in one-iteration rounding. In this lecture we consider two problems, *Maximum Weighted Bipartite Matching* and *Generalized Assignment Problem*, and show how to apply *iterative rounding* over the corresponding LPs. This technique was introduced by Jain, in [1], to give a 2-approximation algorithm for the survivable network design problem. The main idea of the technique is to benefit from properties of extreme point solutions through an iterative algorithm. At each iteration of the algorithm, a subset of variables is rounded. Then, based on the rounded variables, the current instance of the problem is updated by reducing its size. Such an updated instance is passed to the next iteration for further processing. This general approach can be summarized as follows.

(1) Formulate IP/LP

(2) Characterize extreme point solutions

(3) Devise an iterative algorithm

(4) Analyze the running time and give a proof of correctness

Applications of iterative rounding led to new results on *network design* problems [1, 2, 3]. Some old problems solved by this technique have simpler and more elegant proofs than before. However, there is also a drawback of using iterative rounding – the LP has to be solved many times.

1.1 An LP iterative rounding prelude – Rank Lemma

Before we consider the two advertised problems, we state Lemma 1 which is the main ingredient in characterization of extreme point solutions of the two problems.

Lemma 1 *(Rank Lemma) Let $P = \{x \mid Ax \geq b, x \geq 0\}$, and let x be an extreme point solution of P such that $x_i > 0$ for each i. Then, any maximal number of linearly independent tight constraints of the form $A_i x = b_i$, where A_i is the i-th row of A, equals the number of variables.*

Intuitively, Rank Lemma says that an extreme point of a d-dimensional polytope P is defined as the intersection of d linearly independent constraints, as illustrated in Figure 1. However, more than d constraints can intersect at the same extreme point, but then some of the constraints are redundant in defining the point. For instance, c_1 and c_2 are sufficient to define A in Figure 1.

1.2 Maximum Weighted Bipartite Matching

In this section we consider Maximum Weighted Bipartite Matching, defined as follows

Maximum Weighted Bipartite Matching:
Given: a graph $G = (V_1 \cup V_2, E)$, $V_1 \cap V_2 = \emptyset$, with $w : E \to \mathbb{R}$.
Find: a matching of the maximum weight.

An example of the problem is provided in Figure 2. Maximum Weighted Bipartite Matching can be solved optimally in polynomial time using, for example, the well-known Hungarian algorithm. However, our goal of this analysis is not to provide a good approximation ratio for the problem, but rather to illustrate the structure of iterative rounding. We start by defining an LP relaxation for the problem.

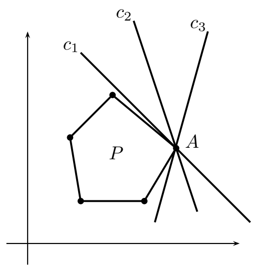

Figure 1: An illustration that two linearly independent constraints, e.g. c_1 and c_2, are sufficient to define an extreme point, i.e. to define A, of a 2-dimensional polytope.

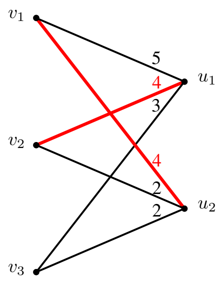

Figure 2: An example of a weighted matching in a bipartite graph. The maximal weight, which is 8, is obtained by choosing edges $\{v_1, u_2\}$ and $\{v_2, u_1\}$ as matching.

1.2.1 LP relaxation

In this subsection we define an LP relaxation for Maximum Weighted Bipartite Matching problem. Let $\delta(v)$ denote the set of all the edges incident to $v \in V_1 \cup V_2$. Then, the LP relaxation for the problem, denoted by $LP_{bm}(G)$, we define in the following way:

$$\begin{array}{lrl} \text{maximize} & \displaystyle\sum_{e \in E} w_e x_e & \\ \text{subject to} & \displaystyle\sum_{e \in \delta(v)} x_e \leq 1 & \forall v \in V_1 \cup V_2 \\ & x_e \geq 0 & \forall e \in E \end{array}$$

For each edge $e = (u, v) \in E$ there is a variable x_e denoting whether vertices u and v are matched or not.

1.2.2 Characterization of extreme point solutions

In this subsection we characterize extreme point solutions of $LP_{bm}(G)$. The first step towards the characterization is expressed via Lemma 2, which is a direct application of Rank Lemma. Before we state the lemma, for a set $F \subseteq E$ we define a characteristic vector $\chi(F) \in \mathbb{R}^{|E|}$ to be the vector with 1 at indices that correspond to the edges in F, and 0 at all the other indices.

Lemma 2 *Given any extreme point solution x to $LP_{bm}(G)$ such that $x_e > 0$, for each $e \in E$, there exists $W \subseteq V_1 \cup V_2$ such that*

(i) $x(\delta(v)) = 1$, for each $v \in W$.

(ii) The vectors in $\{\chi(\delta(v)) \mid v \in W\}$ *are linearly independent.*

(iii) $|W| = |E|$.

Finally, from Lemma 2 we deduce the following corollary, which is the skeleton of the iterative algorithm that we are going to provide in the sequel.

Corollary 3 *Given any extreme point solution* x *to* $LP_{bm}(G)$*, there must exists an edge* e *such that* $x_e = 0$ *or* $x_e = 1$.

Proof Suppose, towards a contradiction, that $0 < x_e < 1$ for each $e \in E$. Then, by Lemma 2 there exists $W \subseteq V_1 \cup V_2$ such that the constraints corresponding to W are linearly independent and tight, and $|W| = |E|$.

Next, our goal is to show $d(v) = 2$ for each $v \in W$, and $d(v) = 0$ for each $v \notin W$. This would imply that E is a cycle cover on the vertices in W, which would further imply that the constraints are not linearly independent.

For each $v \in W$, from $x(\delta(v)) = 1$ and $x_e < 1$ follows that $d(v) \geq 2$. That, along with $|W| = |E|$ and $x_e > 0$, implies the following chain of inequalities

$$2|W| = 2|E| = \sum_{v \in V} d(v) \geq \sum_{v \in W} d(v) \geq 2|W|. \tag{1}$$

Since the first and the last term in (1) are the same, all the inequalities in the chain can be replaced by equalities. Then, from the first inequality we have $d(v) = 0$ for each $v \notin W$, and from the second inequality we have $d(v) = 2$ for each $v \in W$. Therefore, E is a cycle cover on the vertices of W.
Let C by any such cycle with all its vertices in W. Since G is a bipartite graph, each edge of C is incident to a vertex in $V_1 \cap W$ and to a vertex in $V_2 \cap W$. Expressing it via characteristic vectors, we obtain the following

$$\sum_{v \in C \cap V_1} \chi(\delta(v)) = \sum_{v \in C \cap V_2} \chi(\delta(v)),$$

which contradicts the independence of the constraints corresponding to W. Therefore, any extreme point solution x of $LP_{bm}(G)$ has at least one edge e such that $x_e = 0$ or $x_e = 1$. This concludes the proof. ■

1.2.3 Iterative algorithm

In this subsection we provide an iterative rounding algorithm for Maximum Weighted Bipartite Matching problem. The algorithm explores properties of extreme point solutions provided by Corollary 3.

```
ITERATIVE-BM
1  $F \leftarrow \emptyset$
2  while $|E| \neq 0$
3      Find an optimal extreme point solution $x$ to $LP_{bm}$
4      if $\exists e$ such that $x_e = 0$
5          remove $e$ from $G$
6      else if $\exists e = (u, v)$ such that $x_e = 1$
7          $F \leftarrow F \cup \{e\}$
8          remove $u$ and $v$ from $G$
9  return $F$
```

1.2.4 Analysis of Iterative-BM

In this subsection we analyze algorithm Iterative-BM via the following theorem.

Theorem 4 *Algorithm* Iterative-BM *returns an integral solution with the optimal weight in time polynomial in the input size.*

Proof We split the proof into two parts: analysis of the running time; and showing that our algorithm returns an optimal solution.

The running time of the algorithm is polynomial. By Corollary 3, line 4 or line 6 of the algorithm is satisfied. Therefore, at each step of the loop we remove at least one edge of G, either by removing an edge explicitly, or by removing two adjacent vertices. It implies that the loop starting at line 2 executes $O(|E|)$ times. At each step of the loop one LP_{bm} is solved, and G is updated. Since both operations run in polynomial time, we conclude that the whole algorithm runs in polynomial time.

Our algorithm returns an optimal solution. We give a proof by induction on the number of iterations of the algorithm. The claim holds trivially if the algorithm does not make any step, since the edge set is empty, and thus the matching set is empty as well.

Let x be an extreme point solution at the current step. We distinguish two cases: there exists an edge e such that $x_e = 0$; and for each edge e we have $x_e \neq 0$.

Case $\exists e \in E, x_e = 0$. Let G' be a graph obtained by removing e from G. Then, a maximum matching, not necessarily integral, on G' is also a maximum matching on G. At that step, F remains unchanged. Then, by the inductive hypothesis we have that our algorithm will return $F = F'$ which is an integral maximum matching on G'. However, F' is an integral maximum matching on G as well.

Case $\forall e \in E, x_e \neq 0$. Then, by Corollary 3 there exists an edge e such that $x_e = 1$. Let G' be the graph obtained from G by removing e and its incident vertices. Let M' be a maximum matching, not necessarily integral, on G'. Since each vertex v in any matching is incident to edges having $x(\delta(v)) \leq 1$, we have that $\{e\} \cup M'$ is a maximum matching on G. By the inductive hypothesis, our algorithm will return an integral maximum matching on G', denoted by F'. Since F' and M' have the same cost, $\{e\} \cup F'$ is an integral maximum matching on G, which is exactly what is returned by the algorithm.

This completes the analysis. ■

1.2.5 And yet another property of extreme point solutions

In the previous subsections we have seen how to construct a maximal weighted matching for a given bipartite graph. We developed algorithm Iterative-BM whose main steps consist of traversing extreme points, and "putting" part of their integral parts into the output set. In this subsection we show that every extreme point of LP_{bm} is integral, and thus we can find an integral solution in one step only, solving only one LP. That property is expressed via the following lemma.

Lemma 5 *Each extreme point of LP_{bm} is integral.*

Proof Let x be an extreme point of LP_{bm}. Define w (weights of the edges in G) such that x is the unique maximum, as illustrated in Figure 3. Note that this is always possible since x is an extreme point.

By Theorem 4, Iterative-BM is going to return an optimal solution which is integral. Since x is the unique maximum, the algorithm is going to return x. This completes the proof. ■

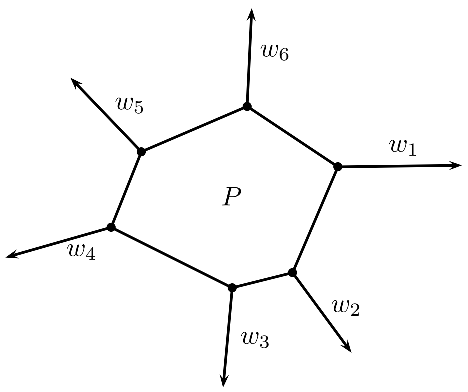

Figure 3: An illustration of weight functions for LP_{bm} for which there is a unique solution. Note that the angle between w_i, whenever $1 \leq i \leq 6$, and its neighboring sides is more than 90 degrees each.

1.3 Generalized Assignment Problem

In this section we consider Generalized Assignment Problem, defined as follows.

Generalized Assignment Problem:
Given:
a set of jobs $J = \{1, \ldots, n\}$,
a set of machines $M = \{1, \ldots, m\}$,
a processing time p_{ij} that job j requires if executed on machine i,
a cost c_{ij} of executing job j on machine i,
a deadline T_i to each $i \in M$.
Find: a min-cost schedule that respects the deadlines.

An example of the problem is provided in Figure 4. We present an approximation algorithm for Gen-

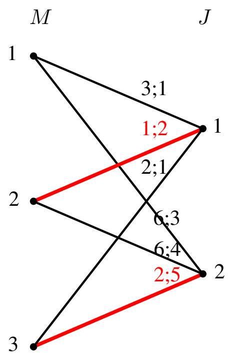

Figure 4: An example of a general assignment problem. We assume that the deadline for each machine is 10. Each edge $e = (i, j)$ is equipped with a pair $a; b$ representing the cost and the processing time, respectively, of job j on machine i. The min-cost schedule is achieved by assigning job 1 to machine 2, and job 2 to machine 3, having cost 3 and makespan 5. Note that the min makespan is achieved by assigning job 2 to machine 1. However, in that case the cost is at least 6, but we prioritize the cost of a schedule.

eralized Assignment Problem (the result was first obtained by Shmoys and Tardos [4]; the presented algorithm is in the thesis by Singh), which uses the rounding technique to obtain the approximation guarantee. The algorithm returns an assignment of the optimal cost, and with a $2T_i$ deadline guarantee for each machine i.

1.3.1 LP relaxation

In this subsection we provide an LP relaxation of Generalized Assignment Problem. First, we describe how to model the problem via a bipartite graph. We start with a complete weighted bipartite graph $G = (M \cup J, E = M \times J, c)$, where to each edge $e = (i, j) \in E$ is assigned a cost c_{ij}. The problem can be reduced to finding a subgraph $F = (M \cup J, E', c)$ of G, where an edge in $e = (i, j) \in E'$ denotes that job j is assigned to machine i. To model the constraint that each job has to be executed on exactly one machine, we require $d_F(j) = 1$ for each $j \in J$. To model the time constraints on the machines, we require $\sum_{e \in \delta(i) \cap F} p_{ij} \leq T_i$, for each machine i. In addition, we adjust this model by removing all edges $e = (i, j) \in E$ from G if $p_{ij} > T_i$, since no optimal solution will assign job j to machine i.

Next, we formulate an LP relaxation for the matching model described above, and denote it by LP_{ga}. Note that we impose the time constraints only on a subset of machines $M' \subseteq M$, which is initially set to M. In the LP formulation for each edge $e = (i, j) \in E$ we have a variable x_e denoting whether job j is assigned to machine i.

$$\begin{aligned}
\text{minimize} \quad & \sum_{e \in E} c_e x_e & \\
\text{subject to} \quad & \sum_{e \in \delta(j)} x_e = 1 & \forall j \in J \\
& \sum_{e \in \delta(i)} p_e x_e \leq T_i & \forall i \in M' \\
& x_e \geq 0 & \forall e \in E
\end{aligned}$$

1.3.2 Characterization of extreme point solutions

In this subsection we provide an insight into extreme point solutions. First, we apply Rank Lemma to LP_{ga} obtaining the following lemma as a result.

Lemma 6 *Let x be an extreme point solution to LP_{ga} such that $x_e > 0$, for each $e \in E$. Then, there exist $J' \subseteq J$ and $M'' \subseteq M'$ such that*

(i) $\sum_{e \in \delta(j)} x_e = 1$, for each $j \in J'$, and $\sum_{e \in \delta(i)} p_e x_e = T_i$, for each $i \in M''$.

(ii) The constraints corresponding to J' and M'' are linearly independent.

(iii) $|J'| + |M''| = |E|$.

As a consequence of Lemma 6 we have the following corollary.

Corollary 7 *Let x be an extreme point solution to LP_{ga}. Then, one of the following must hold.*

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

(b) There exists an edge $e \in E$ such that $x_e = 1$.

(c) There exists a machine $i \in M'$ such that $d(i) \leq 1$.

(d) There exists a machine $i \in M'$ such that $d(i) = 2$ and $\sum_{j \in J} x_{ij} \geq 1$.

Proof The main idea of the proof is to show that

$$\neg((a) \vee (b) \vee (c)) \Rightarrow (d).$$

If (a), (b), or (c) hold, then the proof is done. Therefore, assume that none of (a), (b), or (c) hold, and show it implies (d).

From $\neg((a) \vee (b))$ we conclude $0 < x_e < 1$, for each $e \in E$. It further implies, along with $\sum_{e \in \delta(j)} x_e = 1$, that $d(j) \geq 2$ for each $j \in J$. Assumption $\neg(c)$ implies $d(i) \geq 2$ for each $i \in M'$. From Lemma 6 we have $|J'| + |M''| = |E|$, which can be rewritten as follows

$$|J'|+|M''| = |E| = \frac{\sum_{j \in J} d(j) + \sum_{i \in M} d(i)}{2} \geq \frac{\sum_{j \in J} d(j) + \sum_{i \in M'} d(i)}{2} \geq |J|+|M'| \geq |J'|+|M''|. \quad (2)$$

The second inequality in (2) follows from $d(i) \geq 2$, for each $i \in M'$, and $d(j) \geq 2$, for each $j \in J$. Since the first and the last term in (2) are the same, all the inequalities in (2) can be replaced by equalities. Hence, we derive: $J' = J$; $M'' = M'$; $d(j) = 2$ for each $j \in J$; $d(i) = 2$ for each $i \in M'$; and $d(i) = 0$ for each $i \in M \setminus M'$.

The last sequence of observations implies that E is a cycle cover of the vertices in $J' \cup M''$. Consider any cycle C in the cover. The total number of jobs in C is equal to the total number of machines in C. Since $\sum_{i \in M} x_{ij} = \sum_{i \in M''} x_{ij} = 1$ for each $j \in J'$, there must exist a machine $i \in M''$ with $\sum_{j \in J'} x_{ij} \geq 1$. Then, $J' = J$, $M'' = M'$, and $d(i) = 2$ for each $i \in M''$ imply (d). This completes the analysis. ■

1.3.3 Iterative algorithm

In this subsection we present an algorithm that returns an assignment with the optimal cost, by possibly violating deadline constraints of machines, by a factor of at most 2.

```
ITERATIVE-GA
 1  $E(F) \leftarrow \emptyset$, $M' \leftarrow M$
 2  while $J \neq \emptyset$
 3        Solve $LP_{ga}$ to obtain $x$
          ▷ Case (a) of Corollary 7.
 4        if $\exists e$ such that $x_e = 0$
 5              remove $e$ from $G$
                ▷ Case (b) of Corollary 7.
 6        else if $\exists e = (i, j)$ such that $x_e = 1$
 7              $F \leftarrow F \cup \{e\}$
 8              $J \leftarrow J \setminus \{j\}$
 9              $T_i \leftarrow T_i - p_{ij}$
                ▷ Case (c) + (d) of Corollary 7.
10        else if $\exists i \in M'$ such that $d(i) \leq 1$, or $d(i) = 2$ and $\sum_{j \in J} x_{ij} \geq 1$
11              $M' \leftarrow M' \setminus \{i\}$
12  return $F$
```

The condition at line 10 and the update at line 11 should be understood as a relaxation, since by those two lines we relax the deadline constraint of a machine. However, in the sequel we show that the load of any machine i stays within $2T_i$.

1.3.4 Analysis of Iterative-GA

Analysis on Generalized Assignment Problem from the previous subsections is summarized in the following theorem.

Theorem 8 *Algorithm* ITERATIVE-GA *runs in polynomial time, and returns a scheduling with cost at most C such that load of each machine $i \in M$ is at most $2T_i$. C is the optimal cost of a scheduling in which the load of each machine $i \in M$ is at most T_i.*

Proof We split the proof into four parts: analysis of the running time; arguing that each job is assigned; showing optimality of the algorithm w.r.t. to the cost; proving that the deadline constraints are off by a factor of at most 2.

Running time. Since at each step of the loop either E, or J, or M' is reduced, it follows that after at most $|E| + |M| + |J|$ steps J will become empty. Each step of the loop is executed in polynomial time, and thus the whole algorithm runs in time polynomial in the input size.

Each job gets assigned. It is easy to see that this holds. Indeed, when a job gets removed from J at line 8, it is also assigned to a machine via line 7. Also, the algorithm runs until J gets empty, implying that all the jobs get assigned.

The returned scheduling has the optimal cost. We proceed by showing that at each iteration of the algorithm the cost given by F plus the cost of the current linear programming solution to LP_{ga} is at most the cost of the initial linear program. This can be shown by an induction on the number of iterations.

The claim trivially holds before the first iteration. Next, if at some step line 4 is executed, there is no change in costs. On the other hand, if at some step line 6 get executed, then the cost of F increases by c_{ij}, and the current linear programming solution decreases by $c_{ij}x_{ij} = c_{ij}$. Therefore, the sum of the costs remains the same. When line 10 is executed, the cost of F remains the same, while the cost of the current linear program can only decrease. Therefore, the claim holds in this case as well.

Load of each machine $i \in M$ is at most $2T_i$. Fix any machine $i \in M$. We first argue that if $i \in M'$, then at any iteration we must have $T'_i + T_i(F) \le T_i$, where T'_i is the time left on this machine, i.e. the time left after updates in that step, and $T_i(F)$ is the time used by jobs assigned to machine i in F. The claim can be proved by a simple induction as the claim above for the costs.

Now, consider when machine i is removed from M'. We distinguish the two possible cases.

Case $d(i) \le 1$. The total processing time on machine i, until the end of the algorithm, is at most $T_i + p_{ij} \le 2T_i$, where $j \in J$ is the only job i is connected to.

Case $d(i) = 2$ and $\sum_{j \in J} x_{ij} \ge 1$. Let x denote the solution to LP_{ga} when i got deleted. Let j_1 and j_2 be the two jobs adjacent to machine i. Then, we have the following chain of inequalities

$$\begin{aligned}
load(i) &\le T_i(F) + p_{ij_1} + p_{ij_2} \\
&\le (T_i - x_{ij_1}p_{ij_1} - x_{ij_2}p_{ij_2}) + p_{ij_1} + p_{ij_2} \\
&\le T_i + (1 - x_{ij_1})p_{ij_1} + (1 - x_{ij_2})p_{ij_2} \\
&\le T_i + (1 - x_{ij_1})T_i + (1 - x_{ij_2})T_i \\
&\le T_i + (2 - (x_{ij_1} + x_{ij_2}))T_i \\
&\le 2T_i.
\end{aligned}$$

Note that $p_{ij_1} \le T_i$ and $p_{ij_2} \le T_i$ hold because of the adjustment we have applied over the model in Section 1.3.1.

This completes the analysis. ■

References

[1] K. Jain, *A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem*, Combinatorica 21, 39–60, 2001.

[2] L.C. Lau, S. Naor, M. Salavatipour and M. Singh, *Survivable network design with degree or order constraints*, Proceedings of the 40th ACM Symposium on Theory of Computing, 651–660, 2007.

[3] N. Bansal, R. Khandekar and V.Nagarajan, *Additive guarantees for degree bounded directed network design*, in Proceedings of the Fourtieth Annual ACM Symposium on Theory of Computing (STOC), 2008.

[4] D. Shmoys, É. Tardos, *An approximation algorithm for the generalized assignment problem*, Mathematical Programming, 62(3), 461–474, 1993.