Lecture 4: Linear Programming Relaxations and Deterministic Rounding, Facility Location, GAP

30th January, 2015

1 Integer Programming Formulations

Most optimization problems can be easily cast as **integer** linear programming (ILP) problems. These are problems where one has to maximize or minimize a linear function of variables which have to satisfy certain linear inequalities *and* (some of them) have to be *integral*. Let us illustrate with an example. Take the set cover problem done in class. Here is an integer program for it.

$$\begin{aligned} \min \quad & \sum_{j=1}^{m} c(S_j)x_j && && \text{(ILP-Set cover)} \\ \text{subject to} \quad & \sum_{j:i\in S_j} x_j \geq 1 && \forall i \in U \\ & x_j \in \{0,1\} && \forall j = 1...m \end{aligned}$$

x_j indicates whether we pick set S_j in the cover or not, depending on whether it is set to 1 or 0. The constraint captures the fact that each element needs to be present in at least one of the sets that is picked.

Now, ILPs are also NP-hard to solve (in fact this should be clear from the discussion above). So it seems we haven't gained anything. But here is an extremely powerful and non-trivial fact (which some of you may hear in your optimization class): **linear programs can be solved in polynomial time**. That is, if one removes the integrality constraints (or replace it with 'boundary conditions'), then the residual program is indeed polytime solvable. This residual program is often called the **LP relaxation** of the problem. So the LP relaxation for the set cover problem is the following.

$$\begin{aligned} \min \quad & \sum_{j=1}^{m} c(S_j)x_j && && \text{(LP-Set cover)} \\ \text{subject to} \quad & \sum_{j:i\in S_j} x_j \geq 1 && \forall i \in U && (1) \\ & 1 \geq x_j \geq 0 && \forall j = 1...m && (2) \end{aligned}$$

We now see that the LP relaxation serves two purposes: (LP-Set cover) can be found in polynomial time **and** (LP-Set cover) is a **lower bound** on the optimum set cover. Why?

Therefore, we have obtained a 'nice' lower bound in a very systematic way. Recall the nice lower bound L is something which we have a handle on (can be computed in polytime, for instance), and $L \leq OPT$; one now wishes to design algorithms and bound their costs w.r.t. L since we understand in better, rather than OPT. This has tremendous success in the design of approximation algorithms.

An f-factor approximation for set cover. Straight off the bat, let us see how it helps for set cover by showing a different factor. Given a set system, let f_j be the frequency of an element j, that is, the number of different sets j appears in. Let $f = \max f_j$ over all elements. We now show an f-factor approximation which follows almost trivially. By the way, for the vertex cover problem, $f = 2$.

> Solve (LP-Set cover). Let x^* be the optimal solution. Pick S_i in the set cover iff $x_i^* \geq 1/f$.

It is clear that the cost of the solution picked is at most f times the LP value. We need to show that what the algorithm picks is indeed a set cover. To see this, consider an element j and focus on (1). Note that one of the S_i's must have $x_i \geq 1/f$ since there are at most f terms in the sum. Done.

How good if the LP lower bound? Integrality gap Given a particular LP relaxation for an optimization problem, there is a limit on the best approximation factor obtainable by the process described above. Take an instance I of the problem (set cover, say), and let $\texttt{lp}(I)$ be the value of the LP relaxation, and $\texttt{opt}(I)$ be the value of the optimum. Since our solution can't cost less than $\texttt{opt}(I)$, the best approximation factor we can hope for in this instance is $\frac{\texttt{opt}(I)}{\texttt{lp}(I)}$. Thus, the best approximation factor one can get for the minimization problem via this LP relaxation is at most

$$\sup_{\text{instances } I} \frac{\texttt{opt}(I)}{\texttt{lp}(I)}$$

This quantity is called the integrality gap of the LP relaxation and is very useful in judging the efficacy of a linear program. What is the integrality gap of the LP relaxation of (LP-Set cover) in terms of f?

2 Facility Location

We introduce a new problem now which will sound like a generalization of set cover. Indeed it is, but we will see how to make it 'easier'. The input is two sets: F, a set of facilities, and C a set of clients. The goal is to *open* a bunch of facilities F' and connect all clients to one open facility. The cost of opening facility i is f_i and the cost of connecting client j to a facility i is $c_{i,j}$. We need to find a scheme of minimum cost.

A moment's reflection will show you that this problem is at least as hard as set cover. In fact, can you design an $O(\log n)$-approximation algorithm for this problem? What would a greedy algorithm be for the problem?

Today, we are going to see an $O(1)$-approximation; however, we assume that the connection costs form a metric. In particular, given any two facilities a and b and any two clients p and q, we have $c(a,p) \leq c(b,p) + c(b,q) + c(a,q)$. We start with an LP relaxation.

LP Relaxation

$$\min \quad \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} c(i,j) x_{ij} \qquad (x_{ij}, y_i \geq 0) \tag{3}$$

$$\text{subject to} \quad \sum_{i \in F} x_{ij} \geq 1 \qquad \forall j \in C \tag{4}$$

$$y_i \geq x_{ij} \qquad \forall i \in F,\ \forall j \in C \tag{5}$$

Let (x, y) be a fractional solution to the above LP. We define some notation. Let $F_{LP} := \sum_{i \in F} f_i y_i$. Let $C_j := \sum_{i \in F} c(i,j) x_{ij}$. Let $C_{LP} = \sum_{j \in C} C_j$. We now show an algorithm which returns a solution of cost at most $2F_{LP} + 6C_{LP} \leq 6\texttt{opt}$. The algorithm proceeds in two stages.

Filtering. Given a client j, order the facilities in increasing order of $c(i,j)$. That is, $c(1,j) \leq c(2,j) \leq \cdots \leq c(n,j)$. The fractional cost of connecting j to the facilities is C_j; our goal is to pay not much more than this cost. So, we look at the set of facilities $N_j := \{i : c(i,j) \leq 2C_j\}$. Now it is possible that $x_{ij} > 0$ for some $i \notin N_j$. However, we can change the solution so that j is fractionally connected only to facilities in N_j. This is called filtering the fractional solution.

Define $\overline{x}_{ij}$ as follows. $\overline{x}_{ij} = 2x_{ij}$ for $i \in N_j$, $\overline{x}_{ij} = 0$ for $i \notin N_j$.

Claim 1. *$\overline{x}$ satisfies (4), that is, $\sum_i \overline{x}_{ij} \geq 1$.*

Proof. Let us consider the sum $C_j = \sum_i c(i,j) x_{ij} = \sum_{i \in N_j} c(i,j) x_{ij} + \sum_{i \notin N_j} c(i,j) x_{ij}$. Since $c(i,j) > 2C_j$ for all $i \notin N_j$, we get $\sum_{i \notin N_j} x_{ij} < \frac{1}{2}$, for otherwise the second term in the RHS above exceeds C_j. This implies $\sum_{i \in N_j} x_{ij} \geq 1/2$ implying $\sum_{i \in F} \overline{x}_{ij} \geq 1$. □

We say that a facility i is a good facility for client j if $\overline{x}_{ij} > 0$; note that if all clients were eventually connected to some good facility, the total connection cost would be $\leq 2C_{LP}$. We will **not** be able to do this for all clients. But for clients j we are not able to do so, using the metric property we will show that the connection cost is at most $6C_j$. However, we have to argue about the facility opening costs as well. This is done using a clustering idea.

Clustering. Suppose we could find sets of disjoint facilities $F_1, F_2, \ldots$ such that in each F_k, we have $\sum_{i \in F_k} y_i \geq 1/\alpha$ for some $\alpha \geq 1$, and we only open the minimum cost facility i_k^* in each F_k. Then, our total facility opening cost would be $\sum_k f_{i_k^*} \leq \alpha \sum_k \sum_{i \in F_k} f_i y_i \leq \alpha F_{LP}$. We now show how to obtain such a clustering.

The clustering algorithm proceeds as follows.

- Let X be the set of unconnected clients, initially $X = C$.
- We maintain a set $H \subseteq C$ of *hubs*, initially empty. For each hub $j \in H$, we will maintain a subset $F_j \subseteq F$ of facilities, and a subset $S_j \subseteq C$ of clients. We will maintain: F_j and S_j's will be disjoint, and every client will either be a hub or will be in some S_j.
- Repeat until $X = \emptyset$:
  - Pick client $j \in X$ with minimum C_j. Add j to H.

- Define $F_j := \{i : \overline{x}_{ij} > 0\}$ be the set of facilities which $\overline{x}$ says j can go to.
- Let $S_j = \{j \in X : \overline{x}_{ij} > 0 \text{ for some } i \in F_j\}$. That is, S_j is the set of unconnected clients which have a good facility in F_j.
- Remove $j \cup S_j$ from X.

- For each hub $j \in H$, open the *cheapest* facility in F_j. Connect **all** clients in $j \cup S_j$ to F_j.

It is clear that the above solution is feasible since every client is either a hub j or in the spoke set S_j for some hub j. The following claim shows that each cluster of facilities has large LP mass.

Claim 2. *For any two hubs j and j', $F_j \cap F_{j'} = \emptyset$.*

Proof. Say j was added to H before j'. Note that if there is a facility $i \in F_j \cap F_{j'}$, then since $\overline{x}_{ij'} > 0$ we would add j' to S_j and j' can't be a hub. Therefore, $F_j \cap F_{j'} = \emptyset$. □

Claim 3. *For every hub $j \in H$, $\sum_{i \in F_j} y_i \geq 1/2$.*

Proof. For every $i \in F_j$ where j is the hub, we have from the LP $y_i \geq x_{i,j} = \frac{1}{2}\overline{x}_{ij}$. Therefore, $\sum_{i \in F_j} y_i \geq \frac{1}{2} \sum_{i \in F_j} \overline{x}_{ij} \geq 1/2$ from Claim 1. □

Claim 4. *The total cost of facilities opened by the algorithm is at most $\leq 2F_{LP}$.*

Proof. Since $\sum_{i \in F_j} y_i \geq 1/2$ for all $j \in H$, and we only open the minimum cost facility i_j^* in each F_j, our total facility opening cost would be $\sum_{j \in H} f_{i_j^*} \leq 2 \sum_{j \in H} \sum_{i \in F_j} f_i y_i \leq 2F_{LP}$. □

Now let us argue about connection costs. Consider a client j. If it is a hub, it is connected to a good facility, and so the connection cost $\leq 2C_j$. Suppose the client $j \in S_h$ for some hub h. Since it is in S_h, there is a facility $f \in F_h$ with $\overline{x}_{fj} > 0$. This facility however may not have been opened. Say $i \in S_h$ was opened and j was connected to i. We need to upper bound $c(i,j)$ with respect to C_j. Now, by the metric property, $c(i,j) \leq c(f,j) + c(f,h) + c(i,h)$ where h, recall, is the hub. Now $c(f,j) \leq 2C_j$ since $\overline{x}_{fj} > 0$. Similarly, $c(f,h) \leq 2C_h$ and $c(i,h) \leq 2C_h$. So we get $c(i,j) \leq 2C_j + 4C_h$. But since h was the hub and not j, we know that $C_h \leq C_j$. QED.

3 Generalized Assignment Problem

In GAP, we are given m items J, and n bins I. Each bin i can take a maximum load of B_i. Each item j weighs w_{ij} in bin i, and gives profit p_{ij} when put in it. The goal is to find a max-profit feasible allocation.

LP Relaxation

$$\begin{aligned}
\max \quad & \sum_{i \in I, j \in J} p_{ij} x_{ij} && (x_{ij} \geq 0) && (6)\\
\text{subject to} \quad & \sum_{j \in J} w_{ij} x_{ij} \;\leq\; B_i && \forall i \in I && (7)\\
& \sum_{i \in I} x_{ij} \;\leq\; 1 && \forall j \in J && (8)
\end{aligned}$$

Solve the LP to get a fractional solution x. Suppose all the weights w_{ij} were B_i, say. Then the first constraint is equivalent to $\sum_{j \in J} x_{ij} \leq 1$. Consider the bipartite graph (I, J, E) where we have an edge (i, j) if $x_{ij} > 0$. Then, the solution x above is a maximum weight *fractional matching* in this bipartite graph. This implies that there is an *integral matching* of the same weight, and thus there is an allocation equalling the LP value (and hence the optimum value). This is a non-trivial combinatorial optimization fact and is key to the approximation algorithm below. Let's come to this fact later.

Of course w_{ij} is not equal to B_i, although we can assume $w_{ij} \leq B_i$ since if not we know x_{ij} has to be 0 for such a pair. (Note this must be put into the above LP explicitly, that is, we must set $x_{ij} = 0$ for such pairs as constraints.) Thus, in general, $\sum_{j \in J} x_{ij} \geq 1$. Let $n_i := \lceil \sum_{j \in J} x_{ij} \rceil$. The algorithm proceeds as follows: it uses x to define a fractional matching on a different bipartite graph, use it to get an integral matching of equal weight in it, and then do a final step of pruning.

New Bipartite Graph. One side of the bipartition is J. The other side is I' which consists of n_i copies of each bin i in I. For each bin $i \in I$, consider the items in J in *increasing* weight order. That is, suppose, $w_{i1} \leq w_{i2} \leq \cdots \leq w_{im}$. Consider the fractions $x_{i1}, x_{i2}, \ldots, x_{im}$ in this order. Let $j_1, j_2, \ldots, j_{n_i - 1}$ be the "boundary" items defined as follows. $x_{i1} + \cdots + x_{j_1} \geq 1$, and $x_{i1} + \cdots + x_{j_1 - 1} < 1$; $x_{i1} + \cdots + x_{j_2} \geq 2$, and $x_{i1} + \cdots + x_{j_1 - 2} < 2$; and so on. Formally, for each $1 \leq \ell \leq n_i - 1$,

$$\sum_{j=1}^{j_\ell} x_{ij} \geq \ell; \quad \sum_{j=1}^{j_\ell - 1} x_{ij} < \ell$$

Recall there are n_i copies of the bin i in I'. The ℓth copy, call it i_ℓ, has an edge to items $j_{\ell-1}$ to j_ℓ (j_0 is the item 1). The n_ith copy has an edge to items $j_{n_i - 1}$ to m.

New Fractional Matching. x' is so defined such that for every copy i_ℓ, the total fractional weight incident on it is at most 1 (in fact, it'll be exactly 1 for all copies but the n_ith copy). The total fractional weight incident on item j is the same as that induced by x. It's clear how to do it given the way edges are defined above; formally it's the mess given below.

$$\begin{aligned}
x'_{i_\ell, j} &= x_{ij}, && \text{for } j_{\ell-1} < j < j_\ell \\
x'_{i_\ell, j_{\ell-1}} &= x_{i, j_{\ell-1}} - x'_{i_{\ell-1}, j_{\ell-1}} && \text{(if } \ell = 1\text{, then the second term is 0).} \\
x'_{i_\ell, j_\ell} &= 1 - \sum_{j = j_{\ell-1}}^{j_\ell - 1} x'_{i_\ell, j}
\end{aligned}$$

Integral Matching and Pruning. Note that x' defines a fractional matching in (I', J, E'). Thus, there is an allocation of items to I' whose profit is at least the LP profit (and thus at least `opt`). The assignment to I' implies an assignment to I in the natural way: bin i gets all items allocated to the n_i copies in I'. Is this feasible? Not necessarily. But, the total weight allocated to bin i is not too much larger than B_i.

Claim 5. *The total weight of items allocated by the integral matching above is at most $B_i + \Delta_i$, where $\Delta_i := \max_{j \in J} w_{ij}$.*

Proof. We use the fact that the items (for bin i) were ordered in increasing order of weights. Let J_ℓ be the set of items from $j_{\ell-1}$ to j_ℓ, and let J_{n_i} be the items from j_ℓ to m. Since x is a feasible solution to the LP, we get

$$B_i \geq \sum_{j \in J} w_{ij} x_{ij} = \sum_{\ell=1}^{n_i} \sum_{j \in J_\ell} w_{ij} x'_{i_\ell, j} \geq \sum_{\ell=1}^{n_i} \left(w_{i,j_{\ell-1}} \cdot \sum_{j \in J_\ell} x'_{i_\ell, j} \right) \geq w_{i,j_1} + \cdots + w_{i,j_{n_i-1}}$$

The second inequality uses the increasing order of weights, the last uses the fact that x' forms a fractional matching.

Now, bin i gets at most one item from each J_ℓ (since each copy i_ℓ gets at most item from it's neighbors in J_ℓ.) Note that the heaviest item in J_ℓ weighs w_{i,j_ℓ}, and $\Delta_i = w_{i,n_i}$. Thus, the load on bin i is at most $w_{i,j_1} + \cdots + w_{i,j_{n_i-1}} + w_{i,j_{n_i}}$ which is at most $B_i + \Delta_i$. □

To summarize, we have found an allocation whose profit is at least `opt` and each bin i has load at most $B_i + \Delta_i$. For each bin i now consider the heaviest item j allocated to it. We know that $w_{ij} \leq B_i$. We also know weight of all the *other* items allocated to it is $\leq B_i$. To make the allocation feasible, keep either j or the rest, whichever gives more profit. In this pruned allocation, each bin thus gets profit at least half of what it got in the earlier infeasible allocation. Thus, the profit of this new feasible allocation is at least `opt`$/2$. Thus, the above algorithm is a $1/2$-approximation.

3.1 Fractional Matching to Integral Matching

Given a fractional matching x, construct the fractional bipartite graph $G = (I, J, E_f)$ where there is an edge between $i \in I$ and $j \in J$ with $0 < x_{ij} < 1$. Since x is a fractional matching, we get $\sum_{i \in I} x_{ij} \leq 1$ for all $j \in J$ and $\sum_{j \in J} x_{ij} \leq 1$ for all $i \in I$. We now describe a procedure which takes x and converts it to x' such that two things occur: a) the number of edges in the corresponding E_f is strictly less, and b) $\sum_{i,j} w_{ij} x'_{ij} \geq \sum_{i,j} w_{ij} x_{ij}$. If we show this, then repeating this $|E_f|$ times will give an integral matching of weight no smaller than the fractional weight.

Procedure ROTATE.

1. Pick a path or cycle in $G = (I, J, E_f)$. Call the edges picked F. Decompose F into two matchings M_1 and M_2 in the natural way.
2. Let
$$\varepsilon_1 := \min \left(\min_{(i,j) \in M_1} x_{ij}, \min_{(i,j) \in M_2} (1 - x_{ij}) \right)$$
$$\varepsilon_2 := \min \left(\min_{(i,j) \in M_2} x_{ij}, \min_{(i,j) \in M_1} (1 - x_{ij}) \right)$$
3. Define $x^{(1)}$ as follows. For each $(i,j) \in M_1$, $x^{(1)}_{ij} = x_{ij} - \varepsilon_1$, for each $(i,j) \in M_2$, $x^{(1)}_{ij} = x_{ij} + \varepsilon_1$, for all other edges $x^{(1)}_{ij} = x_{ij}$. Define $x^{(2)}$ similarly.
4. Let x' be the $x^{(1)}$ or $x^{(2)}$ with larger $\sum_{i,j} w_{ij} x'_{ij}$.

It is clear that the number of edges in E_f decreases in this procedure. Also note that the sum $\sum_{i,j} w_{ij}x'_{ij}$ is at least $\sum_{i,j} w_{ij}x_{ij}$. This is because the "increase" in weight in $x^{(1)}$ over x is precisely $\varepsilon_1 \left(\sum_{(i,j)\in M_2} w_{ij} - \sum_{(i,j)\in M_1} w_{ij}\right)$, and that in $x^{(2)}$ is $\varepsilon_2 \left(\sum_{(i,j)\in M_1} w_{ij} - \sum_{(i,j)\in M_2} w_{ij}\right)$. One of them is at least 0. The following claims ends the analysis of the procedure.

Claim 6. *x' is a feasible fractional matching.*

Proof. If F forms a cycle, then it's clear that the fractional "load" on any vertex remains unchanged. If F forms a path, then we need to only concern about end vertices. Let i be such a vertex. Note that there are no edges (i, j') with $x_{ij'} = 1$ incident on it, and exactly one edge $(i, j) \in E_f$ incident on it. In the end, $x'_{ij} \leq 1$. □