,7

CMPUT 675: Topics on Approximation Algorithms and Approximability **Fall 2013**

Lecture 6,7 (Sep 24 and 26,2013):Bin Packing, Facility Location, K-Center

Lecturer: Mohammad R. Salavatipour *Scribe: Rohit Sivakumar and old scribes*

6.1 Bin Packing

The problem statement for *Bin Packing* is as follows:

Bin Packing:

- **Input:** A set of n items with rational sizes in the interval $(0, 1]$.
- **Goal:** Find the minimum number of bins (of size = 1), into which all the n items can be packed.

Theorem 6.1 *It is NP-hard to find an α-approximate solution to bin packing within a factor $\alpha < \frac{3}{2}$.*

Proof: Consider the NP-hard problem of Partition:

- *Input:* Given a set $S = \{s_1, s_2, \cdots, s_n\}$ of items with values $v(s_i) \in (0, 1]$.
- *Problem Statement:* Can we partition S into two subsets S' and $S - S'$ such that $\sum_{x \in S'} v(x) = \sum_{y \in S - S'} v(y)$.

Consider an instance I of this problem with weights normalized such that $\sum_{s \in S} v(s) = 2$. Let this instance be formulated as an instance I' of Bin Packing. If all elements of I fit into 2 bins, this is a Yes-solution to the partition problem with the optimal partition being the contents of each of the two bins. Conversely, if I is a valid Yes solution to the Partition problem, then we can argue that the Bin Packing problem has a valid solution with two bins, the contents of the bins being the optimal partition, I. It follows from the above argument that any instance I' bin packing is a Yes solution (with number of bins equal to 2) if and only if I is a Yes-instance of Partition. Hence, any α-approximation algorithm with $\alpha < \frac{3}{2}$ that provides a bin packing solution can be made to solve the NP-hard problem of Partition in polynomial time. Such an algorithm can't exist as P is not known to be equal to NP. ■

6.2 First Fit Greedy Algorithm

First fit is a straight forward greedy strategy for bin-packing. As per this algorithm, bins are arranged in a sequential order and items are processed in an arbitrary sequence, with each item being placed in the first bin that it fits into.

FirstFit Greedy Algorithm

For $i \leftarrow 1$ to n **do**
Let j be the first bin that i can fit into
Place the item i in bin j

Theorem 6.2 *The cost of a first-fit solution is at most* $2 * OPT + 1$*, where OPT is the cost of the optimal solution.*

Proof: The proof stems from the observation that at most one bin remains less than half-full at any given instant. This is because the first-fit algorithm does not allow placing of an item with size less than $\frac{1}{2}$ into a new bin if there already exists a bin which is less than half full. Thus, if First Fit uses $FF(I)$ bins, at least $FF(I) - 1$ bins are more than half full. Therefore $\frac{FF(I)-1}{2} \leq \sum_{i=1}^{n} s_i \leq OPT(I)$. ■

Theorem 6.3 ***(Johnson '73)*** *If the n items to be packed are sorted in non-decreasing order by size, then* $FF(I) \leq \frac{11}{9} S_I + 4$*, where* S_i *refers to the sum of sizes of all items.*

6.3 An Asymptotic PTAS

Now let us introduce an asymptotic PTAS for Bin Packing. First, suppose all the items are small, say $< \epsilon$ in size, then by FF, there can be at most 1 bin which is less than $1-\epsilon$ full. Then $(FF(I) - 1)(1 - \epsilon) \leq \sum_{i=1}^{n} s_i \leq OPT(I)$. Therefore $FF(I) \leq (1 + 2\epsilon)(OPT(I))+1$. Let us consider a case where all items are large and prove the following lemma.

Lemma 6.4 *Let* $\epsilon > 0$ *be a fixed constant and k be a fixed positive Integer. Suppose that all items are atleast as large as* ϵ *(i.e:* $\forall i \in S : s_i \geq \epsilon$*) and there are at most k distinct sizes of items. Then there is a polynomial time algorithm for bin-packing.*

Proof: As bins are of unit sizes and items are atleast as large as ϵ, no more than $m = \lfloor \frac{1}{\epsilon} \rfloor$ items can go into a bin. Let us define *Bin Type* is a configuration of different sizes of items that will be packed into bins. Since items come in at most k sizes and no more than m items can be placed each bin, the total number of bin types can be calculated as the number of non-negative integral solutions to the equation

$$x_1 + x_2 + \cdots + x_k \leq m$$

where x_is represent the number of items of type i to be placed in the bin. By combinatorial argument, this can be computed to be at most

$$\binom{m+k}{m}$$

which is a large constant. Therefore the number of packings of $R = \binom{m+k}{m}$ bin types will be of the order $O(n^R)$ which can be enumerated to find the optimal bin packing solution.

■

Now let us stick with the assumption that all items are of size atleast ϵ, but we do not classify the items into a constant number of sizes.

Lemma 6.5 *If all items have a size of atleast ϵ for some fixed constant $\epsilon > 0$, then there is a PTAS for bin packing.*

Proof: Let I_L be an instance of the bin packing problem. First, arrange the items in non-decreasing order of size and partition the items into k groups ($k = \lceil \frac{1}{\epsilon^2} \rceil$), each having at most $Q = \lfloor n\epsilon^2 \rfloor$ items. Let us call these groups $G_1, G_2, ..., G_k$. Observe that the size of every item in group G_i is greater than (or equal to) the size of every item in G_{i-1} for $1 < i \leq k$.

We now build an instance I' of this problem in the following way: For each item j in G_i, we round up the size of j to that of the maximum sized element in G_i. This new instance has at most k different item sizes. From Lemma 6.4, we can find an optimal packing for I'.

Claim 6.6 $OPT(I') \leq (1+\epsilon)OPT(I_L)$

It is easy to note that proving the above claim is sufficient to prove lemma 6.5. In order to prove the above claim, we start from the original instance I_L of the problem and build a new instance I'' by rounding down the size of every element in the group to the size of the least sized element in that group. Note that $OPT(I'') \leq OPT(I_L)$. Observe that a packing for instance I'' yields a packing for all but Q largest items of I_L. By placing one item in each bin, these Q items can be placed in Q different bins. Therefore,

$$OPT(I') \leq OPT(I'') + Q \leq OPT(I) + Q$$

As each item in the instance I_L has a size at least ϵ, $OPT(I) \geq n\epsilon$. Furthermore, $Q = \lfloor n\epsilon^2 \rfloor \leq \epsilon OPT$. Plugging this result in the above equation,

$$OPT(I') \leq (1+\epsilon)OPT(I)$$

■

Theorem 6.7 *For any fixed $\epsilon > 0$, there is an approximation algorithm whose solution costs at most* $(1+2\epsilon)OPT+1$.

Proof: Given an instance I, let I' be an instance of the bin packing problem on the set of large items (i.e: $s_i > \epsilon$). Using lemma 6.5, we can find a packing of I' using at most $(1+\epsilon)OPT(I')$ bins. Next, we adopt the first-fit greedy algorithm to pack the items in $I - I'$, utilizing the bins that were used in the solution for instance I' and creating new bins if needed.

We can argue that, if no new bins are created while packing the smaller elements (i.e: $s_i < \epsilon$), then optimum solution remains $(1+\epsilon)OPT(I') \leq (1+\epsilon)OPT(I)$. On the other hand, if new bins were created while running the first-fit algorithm on the smaller bins, then there can be at most one bin which is less than $(1-\epsilon)$ full, thereby bounding the overall cost by $(1+2\epsilon)OPT(I)+1$ as seen in section 6.2. Hence, the overall solution has a cost $\leq (1+2\epsilon)OPT(I)+1$. ■

6.4 Uncapacitated Facility Location Problem

Given a metric graph $G = (V, E)$. There are a set of clients $D \subseteq V$, each having a demand to be served and a set of facilities $F \subseteq V$, each having an opening cost f_i. Note that G is a weighted graph and the edges c_{ij} denote the cost of going from j to i. i.e: if the client at j wants to get service at a facility located at i. The cost functions c_{ij} are known to satisfy triangle inequality. Now we would like to open facilities at a set of locations $F' \subseteq F$ to meet the demands of the clients, keeping in mind that the total cost, $\sum_{i \in F'} f_i + \sum_{j \in D} (min_{i \in F'} c_{ij})$ has to be minimized.

We are going to develop an IP/LP formulation for the problem of metric uncapacitated facility location. Let us declare the variables $y_i \in \{0, 1\}$ for each facility $i \in F$ that denotes if any facility i has been opened. Another binary variable x_{ij} indicates if client j is served by facility i. The following will be the LP formulation for the primal.

$$\begin{aligned} minimize \sum_i f_i y_i + \sum_{i,j} c_j x_{ij} & \\ subject\ to \sum_i x_{ij} = 1 & \qquad \forall j \in D, \\ y_i - x_{ij} \geq 0 & \qquad \forall j \in D, i \in F \\ x_{ij}, y_i \geq 0 & \qquad \forall j \in D, i \in F \end{aligned} \tag{6.1}$$

The dual for this problem can be formulated as:

$$\begin{aligned} maximize \sum_j v_j & \\ subject\ to \sum_j w_{ij} \geq f_i & \qquad \forall i \in F, \\ v_j - w_{ij} \leq c_{ij} & \qquad \forall j \in D, i \in F \\ w_{ij}, v_j \geq 0 & \qquad \forall j \in D, i \in F \end{aligned} \tag{6.2}$$

Here, v_j can be assumed to be the total cost the client j is charged, $j \in D$, which includes w_{ij}, the cost contributed by client j to open the facility i, $i \in F, j \in F$.

Lemma 6.8 *If (x^*, y^*) and (v^*, w^*) are the optimal solutions to the primal and dual problems respectively, then $x^*_{ij} > 0$ implies $c_{ij} \leq vj$.*

Proof: The proof can be derived from the complementary slackness condition: $x^*_{ij} > 0$ *implies* $v_j - w_{ij} = c_{ij}$. Since $w_{ij} \geq 0$, we have $c_{ij} \leq vj$. ■

Let (x^*, y^*) be an optimal solution to the primal LP. For each client $j \in D$, we define the first neighborhood of j as $N(j) = \{i \in F | x^*_{ij} > 0\}$ and the second neighborhood of j as $N^2(j) = \{k \in D | \exists i \in N(j); x^*_{ik} > 0\}$. We also define $C_j = \sum_{i,j} c_j x^*_{ij}$ as the total cost for serving the client j in the optimal solution of the LP. We are going to show that the following algorithm is a 3-approximation for the uncapacitated metric facility location problem.

3-Approximation Algorithm for Facility Location

1. Solve the Linear Program. Let (x^*, y^*) and (v^*, w^*) be the optimal solutions.
2. $C \leftarrow D$
3. $K \leftarrow 0$
4. **while** $C \neq \emptyset$ **do**
5. $K \leftarrow K + 1$
6. Choose $j_k \in C$ which mimimizes $v_j^* + C_j$
7. Choose $i \in N(j_k)$ with probability $x_{ij_k}^*$
8. Assign j_k and all unassigned clients in $N^2(j_k)$ to i
9. $C \leftarrow C \setminus \{j_k \cup N^2(j_k)\}$

Theorem 6.9 *The above algorithm is a 3-approximation algorithm for uncapacitated facility location.*

Proof: Consider an arbitrary iteration k of the above algorithm. The expected cost of opening a facility at this instant is $\sum_{i \in N(j_k)} f_i x_{ij_k}^* \leq \sum_{i \in N(j_k)} f_i y_i$ because of the LP constraint $x_{ij} \leq y_i$.

Moreover, when we pick the client j_k and assign all unassigned clients in $N^2(j_k)$ to a facility $i \in N(j_k)$, we form a partition of the facilities and purge all clients connected to facility i from being parsed in the $(k+1)^{th}$ iteration. Thus, any facility i is chosen at most once in the lifetime of the algorithm. The total expected cost of opening facilities over all iterations is at most:

$$\sum_k \sum_{i \in N(j_k)} f_i y_i^* \leq \sum_{i \in F} f_i y_i^*$$

Now that we have bounded the cost of opening new facilities, let us try to bound the connection costs. Consider an iteration k. The expected cost of assigning j_k to a facility i is $\sum_{i \in N(j_k)} c_{ij_k} x_{ij_k}^* = C_{j_k}$.

For any other client $l \in N^2(j_k)$ that is assigned to the facility i, the expected cost of serving l is at most:

$$c_{hl} + c_{hj_k} + \sum_i + \sum_i c_{ij_k} x_{ij_k}^*$$

Using results from Lemma 6.8, $c_{hl} \leq v_l^*$ and $c_{hj_k} \leq v_{j_k}^*$. Consequently, the cost of serving l,

$$c_{il} \leq v_l^* + v_{j_k}^* + C_{j_k}$$

Since we pick j_k as the minimizer of $v_p^* + C_p$ in Step-6 of the algorithm, $v_{j_k}^* + C_{j_k} \leq v_l^* + C_l$.

Therefore, the expected cost of serving the client l,

$$c_{il} \leq 2v_l^* + C_l$$

Hence, the total expected costs over all such connections is at most:

$$\sum_{i \in F} f_i y_i^* + \sum_{j \in D} (2v_l^* + C_j)$$

Since the value of optimal primal LP is $\sum_{i \in F} f_i y_i^* + \sum_{j \in D} C_j$ and the optimal dual LP has an optimal solution of $\sum_{j \in D} 2v_l^*$, the total expected cost over all connections can be bounded by 3 times OPT.

■

There are many different clustering problems one can consider. Some of the well studied and classical problems are the following. We will focus on the first two and present approximation algorithms for them.

k**-center**:

- **Input:** A metric graph, $G = (V, E)$ with edge costs c_{ij} satisfying triangle inequality and a positive integer $k \geq 1$.
- **Goal:** To find a subset $S \subseteq V$ of k centers to be open ($|S| = k$) and assign each node to the nearest center in S such that minimizes $\max_{v \in V} d(v, S)$, where $d(u, S) = \min_{s \in S} d(u, s)$.

k**-median**:

- **Input:** A metric graph, $G = (V, E)$ with edge costs c_{ij} satisfying triangle inequality and a positive integer $k \geq 1$.
- **Goal:** To find a subset $S \subseteq V$ of k centers to open and assign each node to the neares open center whil minimizing $\sum_{v \in V} d(v, S)$, where $d(u, S) = \min_{s \in S} d(u, s)$.

k**-min-sum-radii**:

- **Input:** A metric graph, $G = (V, E)$ with edge costs c_{ij} satisfying triangle inequality and a positive integer $k \geq 1$.
- **Goal:** To find a subset $S \subseteq V$ of k centers and assign a radius r_i for each center $c_i \in S$ to form a cluster (containing nodes within distance r_i from c_i) which minimizes the sum of radiis of the clusters. Here, radii is defined as the maximum distance between the cluster center and any other vertex in the cluster.

6.5 Greedy Algorithm for k-center

A 2-approximation greedy solution to the k-center problem is as follows:

2-Approximation Algorithm for k-Center

1. Start from an arbitrary vertex $v \subset V$; $S \leftarrow \{v\}$
2. **while** $|S| < k$ **do**
3. $\quad u \leftarrow \max_u d(u, S)$ – pick the vertex farthest from all vertices in S
4. $\quad S \leftarrow S \cup \{u\}$
5. return S

Theorem 6.10 *The above algorithm is a 2-approximation for k-center.*

Proof: Suppose S^* is an optimal k-center for a given graph $G = (V, E)$ and let $V_1^*, V_1^* \cdots V_k^*$ be the clusters associated with this solution. Let r^* be the maximum radius of the k clusters.

Claim 6.11 $\forall u, v \in V_i^*; d(u, v) \leq 2r^*$.

Suppose the solution returned by our algorithm, S, has one vertex from each cluster V_i^*, then it clearly has a cost that is less than $2r^*$. If S does not have one vertex from each V_i^*, then by pegion hole principle, there exists atleast one cluster V_j^* that contains two or more elements of S. Let us call these elements p and p'. Since the maximum radius in the optimum solution is r^* and the vertices p, p' are contained in the same cluster, d(p, p'), the minimum distance between p and p' is at most $2r^*$.

Without loss of generality, let us assume that our algorithm picked p' after p. Since our solution picked p' after p, p' would have been the farthest vertex from any of the cluster centres during the iteration it was picked. Furthermore, as $d(p, p') \leq 2r^*$, the maximum radius of the k cluster, $\max_{v \in V} d(v, S)$ can be at most $2r^*$.

■

Theorem 6.12 *There is no α-approximation algorithm for the k-center problem for $\alpha < 2$ unless $P = NP$.*

Proof: Consider the dominating set problem, which is NP-complete. In the dominating set problem, we are given a graph $G = (V, E)$ and an integer k, and we must decide if there exists a set $S \subseteq V$ of size k such that each vertex is either in S, or adjacent to a vertex in S. Given an instance of the dominating set problem, we can define an instance of the k-center problem by setting the distance between adjacent vertices to 1, and nonadjacent vertices to 2: there is a dominating set of size k if and only if the optimal radius for this k-center instance is 1. Furthermore, any α-approximation algorithm with $\alpha < 2$ must always produce a solution of radius 1 if such a solution exists, since any solution of radius $\alpha < 2$ must actually be of radius 1. So such algorithm would solve the dominating set problem in polytime, which is impossible, unless $P = NP$. ■