,7

CMPUT 675: Approximation Algorithms **Fall 2011**

Lecture 6,7 (Sept 27 and 29 , 2011): Bin Packing, MAX-SAT

Lecturer: Mohammad R. Salavatipour *Scribe: Weitian Tong*

6.1 Bin Packing Problem

Recall the bin packing problem:

Bin Packing:

- Input: Given a set of n items each having a rational size s_i in $(0, 1]$.
- Output: Find the minimum number of bins (of unit size) into which all the items can be packed.

Theorem 1 *There is no α-approximation algorithm with $\alpha < \frac{3}{2}$ for Bin Packing unless $P = NP$.*

Proof. Consider the following NP-hard problem.

Partition:

Input: set of items $S = \{1, ..., n\}$ with size $(0 \leq s_i \leq 1) \in Q^+$.

Question: Can we partition S into two parts S' and $S - S'$ such that $\sum_{i \in S} S_i = \sum_{j \in S - S'} S_j$?

Let I be an instance of partition. Scale all S_i's such that $\sum S_i = 2$ and let this instance I' be the input to Bin Packing. If all items of I' fit into 2 bins, since their total sum is 2, both bins must be full and therefore I is a yes instance (the partition is given by the items in 2 bins for I'). On the other hand, if I is a Yes instance, then the corresponding partition implies that the set of items in each part can be fit into one bin for the corresponding instance I'. Therefore, the set of items of I' can be fit into 2 bins if and only if I is a Yes instance. So if we can distinguish between 2 and ≥ 3 for I' then we can decide between Yes and No for I. Therefore, there is no better than $\frac{3}{2}$-approximation for bin packing unless P=NP.

■

6.1.1 The obvious greedy algorithm: First Fit

There is a very straightforward greedy approximation algorithm, called *First Fit*. *First Fit* processes the items in arbitrary order. For each item, it attempts to place the item in the first bin that can accommodate the item. If no bin is found, it opens a new bin and puts the item within the new bin. First Fit implements easily in $O(n^2)$ time. With proper data structures it can run faster.

FirstFit Algorithm

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

Theorem 2 *First Fit is a 2-approximation algorithm.*

Proof. This is due to a simple observation that at any given time, it is impossible for 2 bins to be at most half full. The reason is that if at some point a bin was at most half full, meaning it has at least a half empty space, the algorithm will not open a new bin for any item whose size is at most $\frac{1}{2}$. Only after the bin fills with more than $\frac{1}{2}$ or if an item with a size larger than $\frac{1}{2}$ arrives, the algorithm may open a new bin. Thus, if $First\ Fit$ uses $FF(I)$ bins, at least $FF(I) - 1$ bins are more than half full. Then $\frac{FF(I)-1}{2} < \sum_{i=1}^{n} s_i \leq OPT(I)$, and we get $FF(I) \leq 2 * OPT(I)$ ■

Theorem 3 (Johnson'73) *Suppose the n items have been sorted in descending order by size. $FF(I) \leq \frac{11}{9}OPT(I) + 4$.*

6.1.2 An asymptotic PTAS

Now we will introduce an asymptotic PTAS for *Bin Packing*. First, suppose all the items are small, say less than ϵ, then by FF all but at most one bin are at least $(1-\epsilon)$-full. Then $(1-\epsilon)(FF(I)-1) < \sum_{i=1}^{n} s_i \leq OPT(I)$, we get $FF(I) \leq (1+\epsilon)OPT(I) + 1$. Now let's consider the situation when all items are large. First we prove the following lemma for a slightly more special case.

Lemma 1 *Let $\epsilon > 0$ be fixed constant and g be a fixed positive integer. Suppose all items are large that is at least ϵ, and there are only g distinct sizes. Then we can solve this bin packing problem exactly in polynomial time.*

Proof. Since every bin is of unit size and item size is at least ϵ, at most $m = \lfloor \frac{1}{\epsilon} \rfloor$ items can be packed in each bin. Define bin type is a configuration of different sizes of items that will be packed into bins. Then the number of different bin types is no more than $g^m (R \triangleq g^m)$, which is a constant. Since there are n items and each item has size less than bin size, at most n bins are needed. Remembering that it only matters how many bins of each configuration there are, not what order they are in. If x_i denotes the number of bins with the ith configuration, then the nonequivalent solutions correspond to nonnegative integral solutions to the equation

$$x_1 + x_2 + \cdots + x_R \leq n,$$

of which there are at most

$$\binom{n+R}{R}$$

solutions by a classic combinatorial argument. Since $\binom{n+R}{R}$ is $O(n^R)$, we can solve this problem in polynomial time by enumerating all the possible solutions and finding out the optimal solution. ■

Now what if all the items are large but there are not a constant number of distinct sizes? we use the same trick of scaling (as in the knapsack problem) to bring down the number of distinct sizes to a constant at a loss of small factor in the approximation ratio.

Lemma 2 *If all the items are large (i.e. $s_i > \epsilon$) for a fixed constant $\epsilon > 0$, then there is a $(1+\epsilon)$-approximation algorithm.*

Proof. Let I_L be the given instance. Consider the large items in non-increasing order of their sizes. For some k (which will be set to $\lfloor \epsilon^2 n \rfloor$), put every k consecutive items into one group, i.e. we are partitioning the large items into groups of size k each. So we get $g = \lceil \frac{n}{k} \rceil = \lceil 1/\epsilon^2 \rceil$ groups: $G_1, G_2, \ldots, G_g$ and every item in group G_i is greater than (or equal to) every item in group G_{i+1}, for every $1 \leq i < g$.

We build a new instance I' in the following way. First discard G_1. For G_1, we will need at most need k bins to pack those items, so we can easily fit all those items into k bins at the end. Round up the items in group G_i to the largest value in that group (for every $2 \leq i \leq g$). We call this instance I'. So we get $g-1$ groups and all the items in each group have the same size.

Lemma 3 *The cost of* $\mathrm{OPT}(I') \leq \mathrm{OPT}(I_L)$.

Proof. To prove this lemma we build another instance, called I'' from I_L. We can discard all the items in the last group $G_{\leq g}$ from I_L and round all the items in group G_i (where $1 \leq i < g$) down to the smallest value in that group. Now we call this instance I''. Clearly, $\mathrm{OPT}(I'') \leq \mathrm{OPT}(I_L)$. Also, both I' and I'' have the same number of groups and the same number of items in each group. Furthermore, all the items in the i'th group of I' (which are coming from G_{i+1} of I_L) are smaller than all the items in the i'th group of I'' (which are coming from G_i of I_L). Therefore: $\mathrm{OPT}(I') \leq \mathrm{OPT}(I'')$. Combining these two inequalities we get the lemma.

■

Note that we can compute the optimum solution for I' since it has only a constant $g = \lceil 1/\epsilon^2 \rceil$ number of bins. This will take at most $\mathrm{OPT}(I_L)$ bins according to the above lemma. Also we can pack all the items of groups of $G_2, \ldots, G_g$ of I_L according to the packing of I'. We use an extra at most $k = \lfloor \epsilon^2 n \rfloor$ bins to pack the items in G_1. Thus, we pack all of I_L using at most $\mathrm{OPT}(I_L) + \lfloor \epsilon^2 n \rfloor$ bins. Since all items have size at least ϵ thus $\mathrm{OPT}(I_L) \geq \epsilon n$ which implies $\lfloor \epsilon^2 n \rfloor \leq \epsilon \mathrm{OPT}(I_L)$. Thus we use at most $(1+\epsilon)\mathrm{OPT}(I_L)$ bins. ■

Theorem 4 *For any fixed* $\epsilon > 0$*, we can find an approximation algorithm which produces a solution using at most* $(1+\epsilon)\mathrm{OPT} + 1$ *bins.*

Proof. Let I be the given instance and partition it into two part: I_L containing large items with size larger than ϵ and I_S containing the rest items. Then we have the following algorithm:

APTAS for Bin Packing

> Split all items into smalls ($\leq \frac{\epsilon}{2}$) and larges ($> \frac{\epsilon}{2}$): I_S and I_L.
> Let $k = \lceil \epsilon^2 n \rceil$,
> Construct I' as above from I_L.
> Solve the problem on I' exactly; let us say b is the optimal solution for the instance I'.
> Pack items in G_1 into k bins (because there are only k items in G_1).
> Use FF algorithm to pack the small items.

If the FF phrase does not open any new bin, then the number of bins is at most $(1+\epsilon)\mathrm{OPT}(I') \leq (1+\epsilon)\mathrm{OPT}(I)$ by lemma 2. If the FF phase does open new bins, then at the end, all but the last bin must be more than $1-\epsilon$ full. Recall the discussion in the beginning of this subsection: $FF(I) \leq (1+\epsilon)\mathrm{OPT}(I) + 1$. Therefore, in either case our algorithm uses no more than $(1+\epsilon)\mathrm{OPT}(I) + 1$ bins. ■

6.2 Maximum Satisfiability Problem

The Maximum Satisfiability problem (MAX-SAT), which is an optimization version of SAT, consists of finding a truth assignment that satisfies the maximum number of clauses in a CNF formula. Sometimes, we also consider a variant of MAX-SAT, called weighted MAX-SAT. In weighted MAX-SAT, every clause has a weight and the problem consists of finding a truth assignment in which the sum of weights of violated clauses is minimal.

MAX-SAT:

- Input: A boolean formula in CNF over n variables $x_1, \ldots, x_n$ and a weight w_i for each clause C_i, $1 \leq i \leq m$
- Output: Find a truth assignment to variables such that it maximizes the sum of weights of satisfied clauses.

There are some special subproblem of Weight Maximum Satisfiability problem.

- Special case when $w_i = 1, i = 1, \ldots, m$: maximize the number of satisfied clauses.
- MAX-KSAT: every clause has at most k literals.
- MAX-EKSAT: each clause contains exactly k literals.

Theorem 5 *Max-k-SAT is NP hard for any $k \geq 2$.*

Note that 2-SAT is polynomially solvable and k-SAT (for $k \geq 3$) is NP-hard.

Today, we will see 3 approximation algorithms for Max-SAT. The first one is good when the sizes of clauses are large. Then we show how to improve upon this algorithm. The third algorithm (seen next lecture) will be good when the clauses are small. At the end we show how the combination of the first and third algorithms yields a better approximation algorithm.

6.2.1 Simple Randomized Algorithm

This is perhaps the most obvious randomized (and maybe the dumbest possible randomized) algorithm. Flip a fair coin for every variable (independently) to choose the value True or False for that variable, i.e. set it to True/False with probability of $\frac{1}{2}$ and return this truth assignment. We call this the simple randomized algorithm.

Theorem 6 (Johnson '74) *Simple Randomized Algorithm is a $\frac{1}{2}$-approximation algorithm for MAX-SAT.*

Proof. Let τ be the truth assignment and let boolean variables Y_j indicates whether clause C_j is satisfied.

$$Y_j = \begin{cases} 1, & C_j \text{ is satisfied} \\ 0, & \text{otherwise} \end{cases}$$

Let the random variable W denote the weight of the satisfied clauses, and let W_j be the contribution to W from any particular clause C_j. Thus we have $E[W] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ is satisfied}] = \sum_{j=1}^{m} w_j E[Y_j]$. Since clause C_j is satisfied unless all of its literals are false, $\Pr[C_j \text{ is satisfied}] = 1 - \frac{1}{2^{|C_j|}}$. One the other hand, $|C_j| \geq 1$ indicates $\frac{1}{2^{|C_j|}} \leq \frac{1}{2}$. Thus, $E[W] \geq \frac{1}{2} \sum_{j=1}^{m} w_j \geq \frac{1}{2} OPT$.

■

Note: If all clauses have size k is at least 3, then this is a $(1 - \frac{1}{2^k})$-approximate algorithm which is fairly good. For example, when $k = 3$, we get a $\frac{7}{8}$-approximation algorithm for MAX-3SAT. And the following theorem shows that this is essentially best possible.

Theorem 7 (Hastad'97) *There is no α-approximation algorithm with $\alpha < \frac{7}{8}$ for MAX-3SATs unless $P = NP$.*

6.2.2 De-randomization Using the Method of Conditional Expectation

In fact, it is possible to achieve the approximations we showed not just in expectation but in a deterministic way using the method of conditional expectation. This is a general technique developed by Erdös and Spencer and can be used for many other problems. For this problem, we will use the following important property:

Lemma 4 *Suppose we have assigned the first i boolean variables $x_1 = a_1, \ldots, x_i = a_i$. Then we can compute the expected value of solution in polynomial time.*

Proof. Observe that we can calculate the expectation of W conditioned on any partial set of assignments to the variables if a literal is false, then remove it from all the clauses in which it appears; if it is true, then ignore the clauses which contain it, as they are already satisfied. Then the conditional expectation of W is the unconditioned expectation of W in the reduced set of clauses plus the weight of the already satisfied clauses. Thus let f' be the formula over variables $x_{i+1}, \ldots, x_n$ obtained from original formula f by substituting values of $x_1, \ldots, x_i$ and simply we can compute the expected value of f'. ■

This lemma suggests the following simple algorithm.

— Consider x_1 for each of two assignment of $x_1 = T$ or F, compute the expected value of the solution. If $E[W|x_1 = T] > E[W|x_1 = F]$, then we assign $x_1 = T$, otherwise, $x_1 = F$. Now set $x_1 = \tau$ as above and write the expected value of W as a weighted average of conditional expectations.

$$\begin{aligned} \mathrm{E}[W] &= \mathrm{E}[W|x_1 = T]\,\mathbf{Pr}(x_1 = T) + E[W|x_1 = F]\,\mathbf{Pr}(x_1 = F) \\ &= \frac{1}{2}\left(E[W|x_1 = T] + E[W|x_1 = F] \right) \end{aligned}$$

Then we have $E[W|x_1 = \tau] \geq \mathrm{E}[W] \geq \frac{1}{2}OPT$ So if we go through all the variables and always choose the assignment that gives a larger expected W, eventually we will find some assignment to all the variables such that the value of W is at least $\frac{OPT}{2}$.

6.2.3 A better algorithm using biased coins

Now we introduce a better algorithm using biased coins. First, assume that all 1-clauses in a given instance contain no negated literals. We set each $x_i = T$ with probability p ($\geq \frac{1}{2}$ to be defined). Then we return the truth assignment as the solution of the algorithm. If C_j is a 1-clause, it is satisfied with prob p. If C_j is a ≥ 2-clause then let α be the number of negated variables in C_j, and β be the number of positive variables in C_j. So $\Pr[C_j \text{ is satisfied}] = 1 - p^\alpha \cdot (1-p)^\beta \geq 1 - p^{\alpha+\beta} \geq 1 - p^2$ (where we have used the fact $p \geq 1 - p$). This implies that:

Lemma 5 $\Pr[C_j \textit{ is satisfied}] \geq \min\{p, 1 - p^2\}$.

Now, set $p = 1 - p^2$, which means $p = \frac{(\sqrt{5}+1)}{2} \cong 0.618$. Thus,

$$\mathrm{E}[W] = \sum_j w_j \mathbf{Pr}[C_j \text{ is satisfied}] \geq p \sum_j w_j \geq p\mathrm{OPT}$$

and we get an p-approximation algorithm.

What if some of the 1-clauses have positive literals and some have negative literals? If for some variable x_i only its negated version appears in 1-clauses, say for example $C_j = \overline{x_i}$, then we can define another boolean variable $x_i' = \overline{x_i}$ and replace all occurrences of $\overline{x_i}$ with x_i' and all occurrence of x_i with $\overline{x_i'}$. If for some literal x_i both its negated version and non-negated version appear in 1-clauses, e.g. $C_j = x_i$ and $C_\ell = \overline{x_i}$, without loss of generality and using renaming of variables as above, we can assume that $w_j \geq w_l$. So let N be the set of 1-clauses which contain only one negated literal and U be all other clauses. By the assumption above note that $\mathrm{OPT} \leq \sum_{j \in U} w_j$. Now if we use the same randomized algorithm with probability p, then:

$$\begin{aligned}
\mathrm{E}[W] &= \sum_j w_j \mathbf{Pr}[C_j \text{ is satisfied}] \\
&\geq \sum_j w_j \mathbf{Pr}[C_j \text{ is satisfied}] - \sum_{j \in N} w_j \mathbf{Pr}[C_j \text{ is satisfied}] \\
&= \sum_{j \in U} w_j \mathbf{Pr}[C_j \text{ is satisfied}] \\
&\geq p\mathrm{OPT}.
\end{aligned}$$

6.2.4 Randomized Rounding for Max-SAT

In this section we present an LP rounding algorithm for Max-SAT and show that it has ratio at most $1 - \frac{1}{e}$. The algorithm works better if the size of clauses is smaller. This algorithm is based on an IP/LP formulation of Max-SAT and LP-rounding. First we show how to formulate the problem as an IP/LP.

Let P_j (N_j) be the indices of variables in clauses C_j that are in positive (negative) form. For every x_i, we have an indicating variable y_i which is set to 1 (0) iff x_i is set to Ture (False). Also, for every clause C_j, we have a variable z_j which is 1 iff C_j is satisfied. Then the Max-SAT problem can be stated as:

$$\begin{array}{ll}
\text{maximize} & \sum w_j z_j \\
\text{subject to} & \forall j : \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \\
& \forall j : z_j \in \{0, 1\} \\
& \forall i : y_i \in \{0, 1\}
\end{array}$$

The LP-relaxation is:

$$\begin{array}{ll}
\text{maximize} & \sum w_j z_j \\
\text{subject to} & \forall j : \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \\
& \forall j : 0 \leq z_j \leq 1 \\
& \forall i : 0 \leq y_i \leq 1
\end{array}$$

Randomized-Rounding for Max-SAT:

- Solve the LP; let (y^*, z^*) be the optimal solution
- For each x_i, set it to True with probability y_i^*
- Let $\widehat{x}$ (vector) be the integer solution obtained.

Theorem 8 (Goemans-Williamson'94) *The Rand-Rounding is a* $(1-\frac{1}{e})$*-approximation algorithm.*

Proof. We will use the following two facts:

Fact 1: Arithmetic geometry means inequality For any sequence of non-negative integers$a_1, a_2, \ldots, a_k$:

$$\left(\frac{a_1+a_2+\cdots+a_k}{k}\right)^k \geq a_1 a_2 \ldots a_k.$$

Fact 2: If $f(x)$ is concave over $[0,1]$ (i.e. $(f''(x) \leq 0)$), and $f(0)=0$, $f(1)=\alpha$, then the function f is lower bounded by the line going through origin $(0,0)$ and $(1,\alpha)$.

Figure 6.4: f(x)

Let w_j be the weight contributed by C_j to W. For each clause C_j, it is not satisfied if and only if none of the literals are True. Because x_i is set to True with probability y_i^*, the clause is true with probability

$$\begin{aligned}
\mathbf{Pr}[C_j \text{ is satisfied}] &= 1-\prod_{i\in P_j}(1-y_i^*)\prod_{i\in N_j} y_i^* \\
&\geq 1-\left(\frac{\sum_{i\in P_j}(1-y_i^*)+\sum_{i\in N_j} y_i^*}{k}\right)^k \\
&= 1-\left(1-\frac{\sum_{i\in P_j} y_i^*+\sum_{i\in N_j}(1-y_i^*)}{k}\right)^k \\
&\geq 1-\left(1-\frac{Z_j^*}{k}\right)^k.
\end{aligned} \tag{6.1}$$

Consider the function $g(z)=1-\left(1-\frac{z}{k}\right)^k$, since $g(0)=0$, $g(1)=1-\left(1-\frac{1}{k}\right)^k$ and it is a concave function, we have $g(z)\geq 1-\left(1-\frac{1}{k}\right)^k z$ according to second property as introduced above. Thus,

$$\mathbf{Pr}[C_j \text{ is satisfied}] \geq 1-\left(1-\frac{1}{k}\right)^k Z_j^* \geq \left(1-\frac{1}{e}\right) Z_j^*.$$

And

$$\mathrm{E}[W] = \sum_j w_j \mathbf{Pr}[C_j \text{ is satisfied}] \geq \left(1-\frac{1}{e}\right)\mathrm{OPT}_{LP}.$$

■

Using the method of conditional expectation as applied in the analysis of Simple Randomized Algorithm, we can also de-randomize this problem and get a same result as the Rand Rounding Algorithm gets.

Note that $1 - \frac{1}{e} \approx 0.632$ which is greater than 0.618 in algorithm 2. Also, if all clauses have size at most k and k is relatively small, then the approximation ratio of this algorithm is $1 - (1 - \frac{1}{k})^k > 1 - \frac{1}{e}$. So we get better approximation factor for smaller k's while the simple randomized algorithm gives better approximation factor for larger k's. So it seems reasonable to run both algorithms and return the better solution. This is the main idea of our 3rd algorithm which gives a $\frac{3}{4}$-approximation ratio.

6.2.5 A $\frac{3}{4}$-approximation algorithm

Johnson's Simple Randomized Algorithm works well if clauses are large while Rand Rounding Algorithm works well if clauses are small.

Suppose we flip a coin and based on the outcome ($a = 0$ or $a = 1$) we run algorithm 1 (simple randomized) or algorithm 3 (randomized-rounding).

The better of the two algorithm:

- Flip a fair coin.
- Use Simple Randomized Algorithm and Rand Rounding Algorithm with a probability of $\frac{1}{2}$ respectively.

We can also run the deterministic version of each of the two algorithms and return the better of the two. Clearly the solution is no worse than the expected value of the randomized version described above. We show the following lemma which proves the approximation ratio of this algorithm is 3/4.

Lemma 6 $\mathrm{E}[w_j] \geq \frac{3}{4} w_j Z_j^*$.

Proof. Let's assume that C_j has k variables and define $\alpha_k = 1 - \frac{1}{2^k}$ and $\beta_k = 1 - (1 - \frac{1}{k})^k$. From the analysis of simple randomized algorithm and the randomized rounding algorithm we know that:

$$\mathrm{E}[W_j | a = 0] \geq (1 - \frac{1}{2^k}) w_j \geq \alpha_k w_j z_j^*$$

and

$$\mathrm{E}[W_j | a = 1] \geq \beta_k z_j^* w_j.$$

Therefore, combining these two:

$$\mathrm{E}[W_j] = \mathrm{E}[W_j | a = 0] \Pr[a = 0] + \mathrm{E}[W_j | a = 1] \Pr[a = 1] \geq \frac{1}{2}(\alpha_k + \beta_k) w_j z_j^*$$

Since $\alpha_1 + \beta_1 = \frac{1}{2} + 1 = \frac{3}{2}$, $\alpha_2 + \beta_2 = \frac{3}{4} + \frac{3}{4} = \frac{3}{2}$, and for $k \geq 3$: $\alpha_k = 1 - \frac{1}{2^k} \geq 1 - \frac{1}{8} = 0.875$, $\beta_k = 1 - (1 - \frac{1}{k})^k \geq 1 - \frac{1}{e}$; $\alpha_k + \beta_k \geq \frac{3}{2}$ for all values of k. Therefore,

$$\mathrm{E}[W] = \sum_j \mathrm{E}[W_j] \geq \frac{3}{4} \sum_j w_j z_j^* \geq \frac{3}{4} \mathrm{OPT}.$$

■

The following example shows that the analysis of Algorithm 3 is tight, i.e. the integrality gap of the given LP is at least $\frac{4}{3}$.

Example: Consider the following instance of Max-SAT: $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$, and assume that all the weights are 1. Clearly the cost of OPT is 3. On the other hand, if we set $y_i = 1/2$ and $z_j = 1$ for every i, j we get a feasible fractional solution with weight 4. Therefore, the integrality gap is at least $\frac{4}{3}$.

The best known approximation factor for MAX-SAT is 0.7846 using semi-definite programming. Based on a conjecture (by Uri Zwick), which is supported by experimental results, we can get 0.8331-approximation. Recall that the lower bound (from the hardness of MAX-E3SAT) is 7/8, i.e. we cannot get an $(\frac{7}{8} - \epsilon)$-approximation for any $\epsilon > 0$, unless P=NP.

6.3 Uncapacitated Facility Location Problem

Facility location problem is one of the most well studied problem in approximation algorithms and operation research. There are many different variations of this problem. Here we are going to look at the classical Uncapacitated Facility Location Problem (UFLP).

Uncapacitated Facility Location Problem: Suppose we are given a metric graph $G = (V, E)$. There is a set of clients $D \subseteq V$ each having a demand to be served. There is a set of facilities $F \subseteq V$, each having an opening cost f_i. We assume that G is weighted and c_{ij} is the cost (or distance) of going from j to i, e.g. for a client at location j to get service at a facility at location i. Cost function c_{ij} satisfies the triangle inequality. The goal is to find a subset $F' \subseteq F$ to open and assign each client to nearest opened facility to minimize $\sum_{i \in F'} f_i + \sum_{j \in D} (min_{i \in F'} C_{ij})$.

We are going to write an IP/LP formulation for UFLP. Declare variables $y_i \in \{0, 1\}$ for each facility $i \in F$ opened; variable x_{ij} indicates whether client j is served by facility i. We want to minimize the total cost of opening facilities and the distance cost of each client. So we have the following integer linear program formulation of the metric facility location problem.

$$\begin{array}{lrcl} \text{minimize} & \sum_i f_i y_i + \sum_{i,j} c_{ij} x_{ij} & & \\ \text{subjectto} & \sum_i x_{ij} & = & 1 \ \ \forall j \in D, \\ & y_i & \geq & x_{ij} \ \ \forall j \in D, i \in F, \\ & x_{ij}, y_i & \geq & 0 \end{array}$$

Let us formulate the dual LP too. We can have the following interpretation for the dual variables. Let v_j be the total payment for client j, $j \in D$ and w_{ij} be the cost contributed by client j to open the facility i, $i \in F$, $j \in F$.

$$\begin{array}{lrcl} \text{maximize} & \sum_j v_j & & \\ \text{subjectto} & \sum_j w_{ij} & \geq & f_i \ \ \forall i \in F, \\ & v_j - w_{ij} & \leq & c_{ij} \ \ \forall j \in D, i \in F, \\ & v_i, w_{ij} & \geq & 0 \ \ \forall j \in D, i \in F. \end{array}$$

Lemma 7 *If (x^*, y^*) and (v^*, w^*) are optimal solutions for primal and dual problems respectively, then $x^*_{ij} > 0$ implies $c_{ij} \leq v_j$.*

Proof. The proof is an easy application of the complementary slackness condition: $x^*_{ij} > 0$ implies $v^*_j - w^*_{ij} = c_{ij}$. Thus we have $c_{ij} \leq v_j$ since $w^*_{ij} \geq 0$. ■

Let (x^*, y^*) be an optimal solution for the primal LP. For each client $j \in D$ we define its first neighborhood as: $N(j) = \{i \in F \mid x^*_{ij} > 0\}$ and second neighborhood as $N^2(j) = \{k \in D \mid \exists i \in N(j) \;\; s.t. \;\; x^*_{ik} > 0\}$. We also define $C_j = \sum_i c_{ij} x^*_{ij}$ as the cost of serving client $j \in D$ in the optimum LP. Our goal is to show that the following algorithm is a 3-approximation for UFLP.

Rounding Algorithm for UFLP

1. Solve relaxed LP and its dual problem and get their optimal solutions (x^*, y^*) , (v^*, w^*)
2. $C \leftarrow D$
3. $K \leftarrow 0$
4. while $C \neq \phi$ do
5. $\quad k \leftarrow k + 1$
6. $\quad$ choose $j_k \in C$ which minimizes $C_j + v^*_j$
7. $\quad$ choose $i \in N(j_k)$ with probability $x^*_{ij_k}$
8. $\quad$ assign j_k and all unassigned clients in $N^2(j_k)$ to i
9. $\quad C \leftarrow C \backslash \{j_k\} \cup N^2(j_k)$