CSG399 Home work 1

Xin Dong
College of Computer Science
Northeastern University
xindong@ccs.neu.edu

October 14, 2007

1 Stochastic matrices and LP duality

Let

$$\mathbf{M} = \left(\begin{array}{c} m_{ij} \end{array} \right)$$

Consider below LP:
$min\ 0$
$s.t.$
$(I-M)\overline{x} = 0$
$\overline{1}^T\overline{x} = 1$
$\overline{x} \geq \overline{0}$
Rewrite the above LP:
$min\ 0$
$s.t.$

$$\begin{pmatrix} I-M \\ \overline{1}^T \end{pmatrix} \overline{x} = \begin{pmatrix} \overline{0} \\ 1 \end{pmatrix}$$

$\overline{x} \geq 0$
Consider the dual of the above LP:
max

$$\overline{y}^T \begin{pmatrix} \overline{0} \\ 1 \end{pmatrix}$$

$s.t.$

$$\overline{y}^T \begin{pmatrix} I-M \\ \overline{1}^T \end{pmatrix} \leq \overline{0}^T$$

Claim 1: The dual is feasible.
Proof of claim 1:
Let

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \\ y_{n+1} \end{pmatrix}$$

And rewrite the dual as

max y_{n+1}
s.t. $(y_1, ..., y_n, y_{n+1})A \leq \overline{0}^T$
That is equivalent to (according to the definition of A)
$max\ y_{n+1}$
$s.t.$
$(y_1, ..., y_n)(M-I) + y_{n+1}\overline{1}^T \leq \overline{0}^T$
That is
$max\ y_{n+1}$
s.t.
$y_{n+1}\overline{1}^T \leq (y_1, ..., y_n)(I-M)$
The above LP is feasible because we can just let all y_i except y_{n+1} equal to 1 and y_{n+1} equal to 0.
Claim 2: The dual is bounded.
Proof of claim 2: Let y_k is the minimum value of all y_i i = 1,...,n.
Then we have
$y_{n+1} \leq y_k - \sum_{i=1}^{n} y_i m_{ik}$
$\leq y_k - \sum_{i=1}^{n} y_k m_{ik}$
$= y_k - y_k \sum_{i=1} m_{ik}$
=0

Because the dual is feasible and bounded, the primal is feasible and bounded. It is simple to check that the solution π of the primal satisfies $M\pi = \pi$ and π is a probability vector.

2 Currency trading

2.1 a

Let N be the number of currency and T be the number of trading. Let $C_{ki}, k = 1, \cdots, T, i = 1, \cdots, N$ be the number of the i^{th} currency after the k^{th} trading. Let $C_{0i}, i = 1, \cdots N$ be the number of the i^{th} currency. Let Dollar be the first currency and Euros be the last currency. Obviously, $C_{01} = D$ and $C_{0i} = 0, i = 2, \cdots, N$. Let $t_{kij}, k = 1, \cdots, T, i, j = 1, \cdots, N$ be the number of the i^{th} currency that is changed to the j^{th} currency in the k^{th} trading. The LP for determining the collection of trades that will yield the maximum around of Euros for D US Dollars is:

$max\ C_{TN}$
$s.t.$
$C_{01} = D$
$C_{0i} = 0, i = 2, \cdots, N$
$t_{kij} \geq 0$
$t_{kij} < l_{ij}$
$C_{k-1\ i} \geq \Sigma_j t_{kij}$
$C_{ki} = C_{k-1i} - \Sigma_j t_{kij} + \Sigma_j t_{kji} r_{ji}$

The constraint before the last one says that you must have enough i^{th} currency in the trading.

The last constraint means that the amount of i^{th} currency after k^{th} trading is equal to the amount before trading minus the amount that is changed to other currencies plus the amount that is changed to i^{th} currency.

2.2 b

Proof: Suppose there are some cycles in the solution of the above problem that achieves the objective. We show that this solution can be changed to the one that achieves the objective without any cycle. The way is to remove the cycle one by one.

Consider any cycle from C_{mi} to $C_{m+n\ i}$ in this solution such that $t_{m+k\ p_k q_k} > 0$, $k = 1, \cdots, n$, $p_1 = q_n = i$ and $q_k = p_{k+1} \neq i, k = 1, \cdots, n-1$.

First unify all $t_{m+k\ p_k q_k} > 0$, $k = 1, \cdots, n$ to the i^{th} currency as $t'_{m+k\ iq_k} = t_{m+k\ p_k q_k} / \Pi_{j=1}^{k-1} r_{p_j q_j}$. Let k' is the number such that $t_{min} = t'_{m+k'\ iq_{k'}} = min(t'_{m+k\ iq_k}, k = 1, \cdots, n)$.

Then modify the solution by changing $t_{m+k\ p_k q_k} > 0$, $k = 1, \cdots, n$ as $t'_{m+k\ p_k q_k} = t_{m+k\ p_k q_k} - t_{min} \Pi_{j=1}^{k-1} r_{p_j q_j}$. Obviously $t'_{m+k'\ p_{k'} q_{k'}} = 0$. By doing so, we actually push back some tradings that lead to the i^{th} currency.

Furthermore, modify the solution by changing $C_{ki}, k = m+n, \cdots, T$ as $C'_{ki} = C_{ki} + t_{min} - t_{min} \Pi_{j=1}^{n-1} r_{p_j q_j} r_{p_n q_n} = C_{ki} + t_{min}(1 - \Pi_{j=1}^{n} r_{p_j q_j}) \geq C_{ki}$.

The result solution is in accordance with the constraint of the original problem. In addition, it does not decrease the value of C_{Tj}. So the result solution achieves our objective given the original one does so. By the above way, we remove the considered cycle. It is not difficult to see that we do not produce any new trading because we only push back some original tradings. So no new cycle is produced. Repeat the same way for each cycle one by one. Finally, we can remove all cycles while still achieve the objective.

2.3 c

Proof: Suppose there exist $C_{Tj} \neq 0$ for some $j \neq 1, N$ in the solution of the above problem that achieves the objective. By section b, we suppose the solution is acyclic.

Consider any $C_{Tj} \neq 0, j \neq 1, N$, because initially we have only Dollars, there must be some trading chains that start from Dollars to the j^{th} currency. Following the method in section b, we must be able to remove any chain of this kind. In addition, no new trading is produced. Repeat the same way for each chain one by one. After all chains that lead Dollars to the j^{th} currency are removed, the result solution must satisfy $C_{Tj} = 0$.

Repeat the above process for each $C_{Tj} \neq 0, j \neq 1, N$ one by one. Finally, we obtain the solution satisfies $C_{Tj} = 0, j = 2, \cdots N-1$.

In each step of the above process, C_{TN} does not change. So the result solution achieves the objective.

3 Analysis of an LP rounding algorithm for unweighted set cover

Hooman's algorithm does not yield an $O(logn)$ approximation even for uniform costs. Without loss of generality, suppose the elements of universe are numbered by $1, \cdots, n$. Let $k = \sqrt{n}$. In order to explain the counterexample clearly, assume $n = k^2$. Construct the set of sets in this universe as follows.

$S_1 = \{1, \cdots, k\}$
$S_2 = \{2, \cdots, k+1\}$
.
.
.
$S_k = \{k, \cdots, 2k-1\}$
$S_{k+1} = \{k+1, \cdots, 2k\}$
.
.
.
$S_{ik+j} = \{ik+j, \cdots, (i+1)k+j-1\}$
.
.
.
$S_{(k-1)k+1} = \{(k-1)k+1, \cdots, n\}$
$S_{(k-1)k+2} = \{(k-1)k+2, \cdots, n, 1\}$
.
.
.
$S_{(k-1)k+j} = \{(k-1)k+j, \cdots, n, 1, \cdots, j-1\}$

In the above construction, $j = 1, \cdots, k$. Two properties in the above construction are

1. Each element in universe appears in k sets.
2. The size of each set is k.

Let all $C(S_i) = 1, i = 1, \cdots, n$.

By property 2, we see that the optimal solution is to make $x_{ik+j} = 0$ for $j \neq k$ and $x_{ik} = 1$. The cost for the optimal solution is k.

Consider LP:

$min\ \Sigma x_S$
$s.t.$
$\Sigma_{S\in C: e\in S} x_S \geq 1\ \forall e \in U$
$x_S \geq 0\ \forall S \in C.$

Claim 1: The solution of the above problem satisfies that $min\ \Sigma x_S \geq k$.

Proof of claim 1: By the observation of the constraints, $\Sigma_{j=1}^{n} x_{S_{ik+j}} \geq 1$ for all $i = 0, \cdots, k-1$.

So $min\ \Sigma x_S = min\ \Sigma_{i=0}^{k-1}\Sigma_{j=1}^{n} x_{S_{ik+j}} \geq min\ \Sigma_{i=0}^{k-1} 1 = min\ k = k$.

Claim 2: The lowest boundary k is reachable by let all $x_{S_i} = 1/k$ for all $i = 1, \cdots, n$.

Proof of claim 2: It is simple to verify.

Consider this solution x^*: $x_{S_i} = 1/k, i = 1, \cdots, n$.

Sort the sets in nonincreasing order of x^*_S values, the possible sequence Σ can still be $S_1, S_2, \cdots, S_n$.

However, the smallest prefix of Σ that covers all the elements is $S_1, S_2, \cdots, S_{(k-1)k+1}$. So the cost is $(k-1)k+1$ following the Hooman's algorithm. It is $O(\sqrt{n})$ approximation rather than $O(logn)$ approximation.

4 A greedy algorithm for uncapacitated facility location

Proof of complexity: The greedy algorithm for this problem is at each step, pick the set with the smallest ratio of cost to newly covered elements, or, where U' is the set of elements covered currently by the solution, find the set:

$$\min_{s\in S} \frac{c(s)}{|s - U'|}$$

Let S_1 denote the first set selected by the greedy algorithm, and let OPT denote the optimal solution. We claim that $c(S_1)/|S_1|$ is at most $c(OPT)/n$. This is because if every set S of the optimal solution had $c(S_i)/|S_i|$ exceed OPT/n, then the total cost of the optimal solution, which equals $\Sigma_{s\in OPT} c(s) \geq \Sigma_{s\in OPT} c(OPT)|s|/n = c(OPT)\Sigma_{s\in OPT}|s|/n \geq c(OPT)$. Similarly if S_2 is the second set selected by the greedy algorithm, then $c(S_2)/|S_2 - S_1|$ is at most $c(OPT)/(n-|S_1|)$. In general, if S_i is the i^{th} set selected by the greedy algorithm, then $c(S_i)/|S_i - \bigcup_{j<i} S_j|$ is at most $c(OPT)/(n - |\bigcup_{j<i} S_j|)$. Summing over all i, the cost of the greedy algorithm is at most

$c(OPT)\Sigma_i|S_i - \bigcup_{j<i} S_j|/(n - |\bigcup_{j<i} S_j|) \leq c(OPT)\Sigma_i 1/(n-i) \leq c(OPT)H_n$.

Implementation in polynomial time: We manage to find

$$\min_{X_{jS}} \frac{c(X_{jS})}{|X_{jS} - U'|}$$

in polynomial time.

In order to find the minimal ratio, fix j and consider any $c(X_{jS})$ for $S \subseteq V$. We have $c(X_{jS}) = f_j + \Sigma_{i\in X_{jS}} d_i c_{ij} = f_j + \Sigma_{i\in X_{jS}\cap U'} d_i c_{ij} + \Sigma_{i\in X_{jS}-U'} d_i c_{ij}$.

Then $\frac{c(X_{jS})}{|X_{jS}-U'|} = (f_j + \Sigma_{i\in X_{jS}\cap U'} d_i c_{ij} + \Sigma_{i\in X_{jS}-U'} d_i c_{ij})/|X_{jS} - U'| \geq (f_j + \Sigma_{i\in X_{jS}-U'} d_i c_{ij})/|X_{jS} - U'| = (f_j + \Sigma_{i\in X_{jS}-U'} d_i c_{ij})/|X_{jS} - U' - U'|$. To find the minimal value, we need only check the set of form $X_{jS} - U'$, i.e., $S \cap U' = \phi$.

In those sets that do not intersect with U', fix the size of X_{jS}, $c(X_{jS})$ reaches minimum only when we pick X_{jS} that is closest to facility j, i.e., the minimal value of $\frac{c(X_{jS})}{|X_{jS}-U'|} = \frac{c(X_{jS})}{|X_{jS}|}$ is achieved when we pick the first $|X_{jS}|$ clients after all clients $i \notin U'$ are sorted in nonincreasing order of the $d_i c_{ij}$ value.

From the above analysis, the polynomial time algorithm to find

$$\min_{X_{jS}} \frac{c(X_{jS})}{|X_{jS} - U'|}$$

is

v = $+\infty$
rj = $+\infty$
rk = ϕ
for each $j \in F$
  sort all $i \notin U'$ in nonincreasing order of the $d_i c_{ij}$ value
   for $k = 1$ to $|V - U'|$
    if $((f_j + \Sigma_{i\leq k} d_i c_{ij})/k <$ v)
     v = $(f_j + \Sigma_{i\leq k} d_i c_{ij})/k$
     rj = j
     rk = the set of first k elements in the ordered $V - U'$
return rj and rk

5 Multiset multicover problem

We introduce some denotations on the set in the finite universe U:

Element e occurs in a set s with arbitrary multiplicity, whose number is denoted by $m(s; e)$.

Extend m to a denotation for the set s and the element from U as $m(s; e) = 0$ iff $e \notin s$. Then the extended $m(s; e)$ can be used to define s.

Define $s_1 \bigcup s_2$ as $m(s_1 \bigcup s_2; e) = m(s_1; e) + m(s_2; e)$. Define $s_1 \bigcap s_2$ as $m(s_1 \bigcap s_2; e) = min(m(s_1; e), m(s_2; e))$. Define $s_1 - s_2$ as $m(s_1 - s_2; e) = max(0, m(s_1; e) - m(s_2; e))$. Define $|s| = \Sigma_{e\in U} m(s; e)$.

Define the set A as $m(A; e) = r_e$, i.e., the objective set. The greedy algorithm for this problem is at each step, pick the set with the smallest ratio of cost to newly covered elements, or, where U' is the set of elements covered currently by the solution, find the set:

$$\min_{s \in S} \frac{c(s)}{|s \bigcap (A - U')|}$$

Let S_1 denote the first set selected by the greedy algorithm, and let OPT denote the optimal solution. We claim that $c(S_1)/|S_1|$ is at most $c(OPT)/|A|$. The reason is if every set S of the optimal solution had $c(S_i)/|S_i|$ exceed $OPT/|A|$, then the total cost of the optimal solution, which equals $\Sigma_{s \in OPT} c(s) \geq \Sigma_{s \in OPT} c(OPT)|s|/|A| = c(OPT)\Sigma_{s \in OPT}|s|/|A| \geq c(OPT)$. Similarly if S_2 is the second set selected by the greedy algorithm, then $c(S_2)/|S_2 \bigcap (A - S_1)|$ is at most $c(OPT)/(|A| - |S_1|)$. In general, if S_i is the i^{th} set selected by the greedy algorithm, then $c(S_i)/|S_i \bigcap (A - \bigcup_{j<i} S_j)|$ is at most $c(OPT)/(|A| - |\bigcup_{j<i} S_j|)$. Summing over all i, the cost of the greedy algorithm is at most

$c(OPT)\Sigma_i |S_i \bigcap (A - \bigcup_{j<i} S_j)|/(|A| - |\bigcup_{j<i} S_j|) \leq c(OPT)\Sigma_i 1/(|A| - i) \leq c(OPT)H_{|A|}$.

6 Selecting optimal views for a data warehouse

Some results come from the paper: Implementing Data Cubes Efficiently by Harinarayan, V.; Rajaraman, A.; Ullman, J. With some changes.

Here we are given a lattice $< V, \preceq >$ of views $v_1, v_2, \cdots, v_n$. The number of rows (cost) for each view is $r(v_i), i = 1, \cdots, n$. and a lattice $< Q, \preceq >$ of queries $q_1, q_2, \cdots, q_n$. That is isomorphic to $< V, \preceq >$.

For a sublattice $< P, \preceq >$ of $< V, \preceq >$ (precomputed views, include the top element T of $< V, \preceq >$), the cost C of $< Q, \preceq >$ corresponding to $< P, \preceq >$ is determined as the following.

For $q_i \in < Q, \preceq >$, find the youngest ancestor $A(v_i, P)$ of v_i in $< P, \preceq >$, the cost $C(q_i)$ of q_i is $r(A(v_i, P))$.

$C = \Sigma_{q_i \in Q} r(A(v_i, P))$ denoted as $C(< P, \preceq >)$.

Below we no longer make difference between q_i and v_i.

The LP for the optimal views for a data warehouse is described as following.

T is the top element of $< V, \preceq >$. Without loss of generality, let $T = v_n$.

$r(v)$ is the cost of view v.

Let $x_i, i = 1, \cdots, n$ denotes whether $v_i \in P$.

Let $y_{ij}, i, j = 1, \cdots, n$ denotes whether query i can be answered by j.

All $x_i, y_{ij} \in \{0, 1\}$.

In our case, if $v_i \preceq v_j$ then $y_{ij} \leq x_j$. That means query i can be answered by precomputed j if $x_j = 1$.

Otherwise $y_{ij} = 0$. That means i can never be answered by j no matter whether j is precomputed. Because we are maximizing the decrease, we relax the condition as $y_{ij} \leq 0$.

In addition, we require $\Sigma_j y_{ij} = 1\, \forall i$. The reason is that we are maximizing the decrease. Under the constraint that we can allow one precomputed j to answer i for each i, we won't to include decreases from other ancestors of i. For the same reason as the above, we relax it as $\Sigma_j y_{ij} \leq 1$

$max\ \Sigma_{ij} y_{ij}(r(T) - r(v_j))$
$s.t.$
$\Sigma_i x_i = k + 1$
$x_n = 1$
$y_{ij} \leq x_j\ \forall v_i \preceq v_j$; $y_{ij} \leq 0$ otherwise
$\Sigma_j y_{ij} \leq 1\ \forall i$

The greedy algorithm of the above problem: After selecting some set S of views, the benefit of view v relative to S, which we denote $B(v, S)$, is defined as follows.

1. For each $w \preceq v$, define the quantity B_w by:

(a) Let u be the view of least cost in S such that $w \preceq u$. Note that since the top view is in S, there must be at least one such view in S.

(b) If $r(v) < r(u)$, then $B_w = C(v) - C(u)$. Otherwise, $B_w = 0$.

2. Define $B(v, S) = \Sigma_{w \preceq v} B_w$.

The greedy algorithm for selecting optimal views is:

$S = \{T\}$
for i=1 to k do begin
  select that view v not in S such that $B(v, S)$ is maximized;
  $S = S \bigcup \{v\}$;
end;
resulting S is the greedy selection;

To analyze the complexity of the above greedy algorithm, we introduce some denotations. Let $v_i, i = 1, \cdots, k$ be the k views selected in order by the above algorithm. Let a_i be the corresponding benefit achieved by the selection of v_i. Let $w_i, i = 1, \cdots, k$ be the optimal set of k views. We just pick an order. Let b_i be the corresponding benefit achieved by w_i. Define $A = \Sigma_{i=1}^{k} a_i$ and $B = \Sigma_{i=1}^{k} b_i$.

Along with more and more views are selected, the improvement increases. For any view v, we can order the improvements in the same sequence that the views are chosen. For any view v, compare the improvements sequence of greedy algorithm with that of optimal solution. We can attribute each piece in optimal solution to some pieces in greedy algorithm. Define x_{ij} to be the sum over all views u in the lattice of the amount of the benefit b_i that is attributed to v_j.

First, the sum of the pieces of the benefit of v_j that is attributed to can't exceed the benefit of v_j. That is $\Sigma_{i=1}^{k} x_{ij} \leq a_j$.

Second, from our greedy algorithm, we have $\forall i, b_i \leq a_i$. Furthermore, $\forall i : b_i - x_{i1} \leq a_2$. The reason is that the benefit w_i brings to the competition for second choice is b_i minus the amount of benefit of w_i that was covered by v_1; the latter is x_{i1}. The benefit a_2 must be at least as great as any competing choice. Generalizing the above , for each j we have $\forall i : b_i - x_{i1} - \cdots x_{i,j-1} \leq a_j$.

So we get the family of inequalities:

$B \leq ka_1$

$B \leq ka_2 + a_1$

.

.

.

$B \leq ka_k + a_1 + \cdots + a_{k-1}$

Consider any $e_i \geq 0, i = 1, \cdots, k$. We have

$e_1 B \leq e_1 k a_1$

$e_2 B \leq e_2 k a_2 + e_2 a_1$

.

.

.

$e_k B \leq e_k k a_k + e_k a_1 + \cdots + e_k a_{k-1}$

Sum left and right respectively, we have

$(\Sigma_i e_i) B \leq$

$(e_1 k + e_2 + \cdots e_k) a_1 +$

$(e_2 k + e_3 + \cdots e_k) a_2 +$

.

.

.

$(e_{k-1} k + e_k) a_{k-1} +$

$e_k k a_k$

Let $e_k = 1/k, e_{k-1} = (1 - 1/k)/k, \cdots e_i = (1 - e_{i+1} - \cdots - e_k)/k = (1 - 1/k)^{k-i}/k, i = k, \cdots, 1$. We make coefficients of a_i in the right be all 1. The left side is $(1/k + (1 - 1/k)/k + \cdots + (1 - 1/k)^{k-1}/k)B$.

So $(1 - (1 - 1/k)^k)B \leq A$. The greedy algorithm is 1/(1 - 1/e)-approximation when k is large enough.

7 Approximation factor for the LP deterministic rounding algorithm

Some results come from lecture notes: FacLocFiltering.pdf. Change step 2 of algorithm 1, let $r_i = \frac{4}{3}\Sigma_j x_{ij}^* \times c_{ij}$.

Claim 1 is changed to: $\forall i \in V, \Sigma_{j \in B_i} y_j^* \geq \frac{1}{4}$.

Lemma 1 is changed to: Let LP_F denote the facility cost of the LP solution. Then, $\Sigma_{j \in F'} f_j \leq 4 \times LP_F$.

Lemma 2 is changed to: Let LP_i be the service cost of client in the LP solution. Then, $\forall i \in V, c_{i\sigma(i)} \leq 4 \times LP_i$.

Let LP_S denote the service cost of the LP solution. Then Lemma 2 gives us $\Sigma_i c_{i\sigma(i)} \leq 4 \times LP_S$.

All above results can be proved by the same way in the lecture notes except some changes on the coefficients when estimating the left side or the right of inequalities.

Then it is almost trivial to achieve new Theorem 2: $TotalCost \leq 4 \times (LP_F + LP_S)$.

8 Another 4-approximation LP-based algorithm for uncapacitated facility location

(a) Proof: We want to show that $c_{k\sigma(k)} \leq 3v_i^*$. According to our algorithm, there are two cases:

1. $\sigma(k) = j_k$. Then $x_{kj_k}^* > 0$. By complementary slackness conditions, $v_k^* - w_{kj_k}^* = c_{kj_k}$. So $v_k^* \geq c_{kj_k} = c_{k\sigma(k)}$.

2. $\exists j, k$ such that $x_{ij}^* > 0$, $x_{kj}^* > 0$ and $\sigma(i) = j_k$. Then $c_{i\sigma(i)} = c_{ij_k} \leq c_{ij} + c_{jk} + c_{kj_k} = c_{ij} + c_{kj} + c_{kj_k}$.

Because $x_{ij}^* > 0$, $v_i^* - w_{ij}^* = c_{ij}$ by complementary slackness conditions. Similarly, $v_k^* - w_{kj}^* = c_{kj}$. In addition, $c_{kj_k} \leq v_k^*$ by the discussion of case 1. So, $c_{ij} + c_{kj} + c_{kj_k} \leq v_i^* + v_k^* + v_k^*$. Because v_k^* is the least in U, $v_k^* \leq v_i^*$. Then we have $c_{i\sigma(i)} \leq 3v_i^*$.

(b) Proof: Consider any k where j_k is added to F in step 2 of our algorithm. We have $f_{j_k} = min\{f_j : x_{kj}^* > 0\} \leq \Sigma_{j:x_{kj}^*>0} f_j x_{kj}^* \leq \Sigma_{j:x_{kj}^*>0} f_j y_j^*$. Assume k' is added to F after some time. We know that $\forall j$ if $x_{k'j}^* > 0$, then $x_{kj}^* = 0$. Otherwise k' will be removed from U after k is handled. So $\{j : x_{kj}^* > 0\} \bigcap \{j : x_{k'j}^* > 0\} = \Phi$. Then we have $\Sigma_k f_{j_k} \leq \Sigma_k \Sigma_{j:x_{kj}^*>0} f_j y_j^* \leq \Sigma_j f_j y_j^*$.

(c) Proof: $COST = \Sigma_k f_{j_k} + \Sigma_i c_{i\sigma(i)} \leq \Sigma_j f_j y_j^* + 3\Sigma_i v_i^* = \Sigma_j f_j y_j^* + 3(\Sigma_j f_j y_j^* + \Sigma_{ij} x_{ij}^* c_{ij}) \leq 4(\Sigma_j f_j y_j^* + \Sigma_{ij} x_{ij}^* c_{ij})$.

9 Hardness of the non-metric k-median problem

Some results come from the paper: Inapproximability of the Asymmetric Facility Location and k-Median Problems by Aaron Archher. With some changes.

Proof: Suppose we have polynomial-time algorithm PKM with an approximation ratio $c^n (c > 1)$. We are given set cover problem $< S, E >$ where E is a ground set and S a collection of subsets of E. Here let $m = |S|$ and $n = |E|$. We can construct the k-Median problem below in polynomial time:

Let $N = m + n$ and $M = c^N m$.

Let $F = S$ and $C = S \bigcup E$. Let $c_{i_1 i_2} = 1$ for $i_1, i_2 \in F$. For $i \in E, j \in S$, let $c_{ij} = 0$ whenever j covers i, and $c_{ij} = M$ otherwise. We still assume demand is all 1. (Unit demand). This forms a k-Median problem given any k.

Consider the algorithm below:

for k=1 to m

run PKM

if the cost of $PKM < M$

return k and facilities chosen

return -1

By observation, if t is the size of the optimal set cover, then the k-Median problem with input t has optimal solution with the cost $(m - t)$. Meanwhile, the k-median problem with any input that is less than t has optimal solution with the cost that is at least M.

So if t is the size of the optimal set cover, our algorithm will loop until t because the cost of PKM is at least M for $k < t$. Then our algorithm will return at k because at that time, the cost of PKM is at most $c^N(m - k) < M$.

On the other hand, if our algorithm returns t and facilities chosen, then we know the size of the optimal set cover is not in $[1, t-1]$. Otherwise, the cost of PKM is at most $c^N(m-k) < M$ at some $k \in [1, t-1]$ and our algorithm should return at that time.

Finally, if the cost of $PKM < M$ at some t, the facilities chosen is a cover of size t. The reason is no client with cost M. So all client in E is assigned to one facility with cost 0. That is $\forall i \in E, \exists j \in F$, j covers i. So the facilities chosen (the number is t) covers E.

So we get a polynomial-time algorithm for set cover problem. That is $P = NP$ because set cover problem is $NP - Complete$.

10 Maximum satisfiability and randomized rounding

Some results come from: The Probabilistic Method by Sariel Har-peled. With some changes

(a) Proof: Let I_{true} be the assignment that all variables are true and I_{false} be the assignment that all variables are false. Consider any clause $C_j, j = 1, \cdots, m$. C_j is satisfied by either I_{true} or I_{false}. The reason is for any variable $v_i, i = 1, \cdots, n$ in C_j, if it appears in the positive, C_j must be satisfied by I_{true}. If it appears in the negative, C_j must be satisfied by I_{false}.

So the total number of clauses satisfied by either I_{true} or I_{false} is at least m. Then, we have either the number of clauses satisfied by I_{true} or the number of clauses satisfied by I_{false} is at least $m/2$. The algorithm has an approximation factor of 2.

(b) For any clause C_j, let $z_j \in \{0, 1\}$ be a variable indicating whether C_j is satisfied or not. Similarly, let $x_i = 1$ if the v_i variable is being assigned the value $TRUE$. Let C_j^+ be indices of the variables that appear in C_j in the positive and C_j^- be indices of the variables that appear in C_j in the negative. Clearly, to solve MAX-SAT, we need to solve:

$max\ \Sigma_{j=1}^m z_j$

$s.t.$

$x_i, z_j \in \{0, 1\}\ \forall i, j$

$\Sigma_{i \in C_j^+} x_i + \Sigma_{i \in C_j^-}(1 - x_i) \geq z_j\ \forall j$

We relax this into the following linear program:

$max\ \Sigma_{j=1}^m z_j$

$s.t.$

$x_i, z_j \in [0, 1]\ \forall i, j$

$\Sigma_{i \in C_j^+} x_i + \Sigma_{i \in C_j^-}(1 - x_i) \geq z_j\ \forall j$

(c) Proof: Consider any clause C_j with k literals $l_t, t = 1, \cdots, k$. For any l_t, if it is the positive appearance of some variable v_{i_t}, then l_t is satisfied with the probability $x_{i_t}^*$. If it is the negative appearance of some variable v_{i_t}, then l_t is satisfied with the probability that is equal to the probability that x_{i_t} is assigned $FALSE$, that is $1 - x_{i_t}^*$.

So C_j is not satisfied with the probability $\Pi_{i \in C_j^+}(1 - x_i^*)\Pi_{i \in C_j^-} x_i^*$. Then C_j is satisfied with the probability $Pr[C_j]$

$= 1 - \Pi_{i \in C_j^+}(1 - x_i^*)\Pi_{i \in C_j^-} x_i^*$

$\geq 1 - ((\Sigma_{i \in C_j^+}(1 - x_i^*) + \Sigma_{i \in C_j^-} x_i^*)/k)^k$

$= 1 - ((k - (\Sigma_{i \in C_j^+} x_i + \Sigma_{i \in C_j^-}(1 - x_i)))/k)^k$

$\geq 1 - ((k - z_j^*)/k)^k$

$= 1 - (1 - z_j^*/k)^k$

$\geq (1 - (1 - 1/k)^k)z_j^*$

$\geq (1 - (1 - 1/L)^L)z_j^*$ where L is the maximum number of literals in a clause.

Therefore, the expected number of clauses satisfied is $\Sigma_j Pr[C_j] \geq \Sigma_j(1 - (1 - 1/L)^L)z_j^* = (1 - (1 - 1/L)^L)\Sigma_j z_j^* \geq (1 - (1 - 1/L)^L)OPT$.

(d) Proof: In order to analyze the algorithm, we consider the more detailed description.

$v = \Phi$ (v is the assignment of variables: 0 is false; 1 is true)

while $|v| < n$

Assume v has the form $(v_1, \cdots, v_l)$

$v_0 = (v_1, \cdots, v_l, 0)$

$v_1 = (v_1, \cdots, v_l, 1)$

if $(E_{v_0} < E_{v_1})$

$v = v_1$

else

$v = v_0$

Assume that the randomized algorithm in c choose v_0 with probability p_0 and v_1 with probability p_1. Then, $E_v = p_0 E_{v_0} + p_1 E_{v_1} \leq max_i E_{v_i}$. And so, by choosing the v_i that maximizes the expectation E_{v_i}, we are still guaranteed a good approximation. Repeat this step over and over, the value of the guaranteed expectation does not decrease.