$C_j' < \min\{C_j^\sigma, C_k^\sigma\}$ and $C_k' = \max\{C_j^\sigma, C_k^\sigma\}$.

Hence, the sum of the two completion times decreases by rescheduling the two jobs in SRPT order, which means σ is not optimal.

Therefore, any optimal schedule must be an SRPT schedule.

Note, there ~~are~~ may be more than one SRPT schedule, but all have the same value.

Algo. Step 1: Construct a preemptive schedule by SRPT rule. Let C_j^P be the completion time of j in this schedule. Relabel the jobs s.t. $C_1^P \le C_2^P \le \cdots \le C_n^P$.

Step 2: For $j=1 \to n$, schedule the job j non-preemptive and as early as possible after C_j. Denote the schedule by σ^N and let C_j^N be the completion time of job j in this schedule.

Theorem 4.1 The algo. above is a 2-approximation algo.

Proof We have some observations

1) $p_1 + p_2 + \cdots + p_j \le C_j^P$.
2) In the final schedule, between time C_j^P and C_j^N there is no idle time
3) In the final schedule, between time C_j^P and C_j^N there are only jobs $k \le j$.

$\rightarrow \overset{\text{step2}}{C_j^N} \le \overset{\text{step1}}{C_j^P} + \sum_{k\le j} p_k \le C_j^P + C_j^P = 2C_j^P$

$\rightarrow$ Algo. $= \sum_j C_j^N \le 2\sum_j C_j^P \le 2OPT$.

4.2 Weighted sum of completion times.

Notation: For any set of jobs $S \subseteq \{1, 2, \cdots, n\}$, denote $p(S) = \sum_{j\in S} p_j$

Lemma: For any feasible schedule and for any set of jobs $S \subseteq \{1, 2, \cdots, n\}$

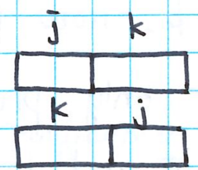
$$\sum_{j\in S} p_j C_j \ge \frac{1}{2} p(S)^2$$

Proof: Let $S = \{1, \cdots, k\}$. Then,

$p_1 C_1 = p_1 p_1$

$p_2 C_2 = p_2 (p_1 + p_2)$

$p_3 C_3 = p_3 (p_1 + p_2 + p_3)$



$C_k \ge C_j + p_k$ OR.

$C_j \ge C_k + p_j$

$\cdots$

$p_k C_k = p_k (p_1 + p_2 + \cdots + p_k)$

$$\sum_j p_j C_j = \frac{1}{2}(p_1 + \cdots + p_k)^2 + \frac{1}{2}[(p_1)^2 + (p_2)^2 + \cdots + (p_k)^2]$$
$$\ge \frac{1}{2}(p_1 + \cdots + p_k)^2$$
$$= \frac{1}{2} p(S)^2$$

(LP) $\min Z = \sum_{j=1}^{n} w_j C_j$

s.t. $C_j \ge r_j + p_j \quad \forall$ jobs j.

$\sum_{j\in S} p_j C_j \ge \frac{1}{2} p(S)^2 \quad \forall S \subseteq \{1, \cdots, n\}$ (Note: 2^n set).

Algo. Step 1: Solve LP.

Step 2: Schedule the jobs non-preemptively and as early as possible in the order $1, 2, \cdots, n$.

Theorem 4.2 The algo. above is a 3-approximation algo.

Proof 1. Polynomial: using the Elipsoid Algo., the LP can be solved in polynomial time.

2. Feasible: As early as possible, so not before release time.

3. Ratio: Denote C_j^*: the completion time of job j in LP solution.

C_j^N: the completion time of job j in the final schedule

Observations 1:

1) $r_j \le C_j^*$
2) In the final schedule, between time r_j and C_j^N, there is no idle time.
3) In the final schedule, before time C_j^N, there are only jobs $k \le j$.

$\rightarrow C_j^N \le r_j + \sum_j p_j \le C_j^* + \sum_j p_j = C_j^* + p(S), j \in S$.

Observation 2:

$\frac{1}{2} p(S)^2 \overset{\text{from LP constraint}}{\le} \sum_{k\in S} p_k C_k^* \overset{C_k^* \le C_j^*}{\le} \sum_{k\in S} p_k C_j^* = C_j^* p(S) \Rightarrow p(S) \le 2C_j^*$

From observation 1&2:

$C_j^N \le 3C_j^*$. Thus $\sum_j w_j C_j^N \le 3\sum_j w_j C_j^* = 3Z^*_{LP} \le 3OPT$.

4.3 Prize-collecting Steiner Tree.

Instance: $G=(V,E)$ and a cost c_e for every edge $e\in E$ and a penalty π_i for every vertex $i\in V$. Also given is a root $r\in V$.

Solution: Tree T containing r. Let $V(T)$ be the vertices in T

Cost: $\sum_{e\in T} c_e + \sum_{i\in V-V(T)} \pi_i$

Goal: Find a solution of minimum cost.

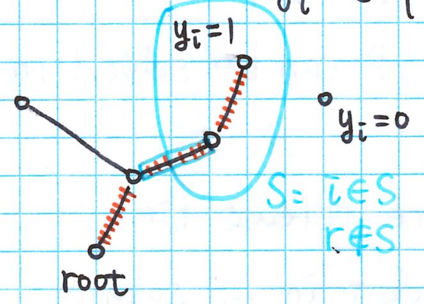
(ILP) $\min Z = \sum_{e\in E} c_e x_e + \sum_{i\in V} \pi_i(1-y_i)$

s.t. $\boxed{① \sum_{e\in\delta(S)} x_e \geq y_i}$ for all i, S with $i\in S\subseteq V-r$

$y_r = 1$ root in tree

$x_e \in \{0,1\}$ $\forall e\in E$

$y_i \in \{0,1\}$ $\forall i\in V$.



if $y_i=1$, then there is a path between i and r.

$\delta(S)$: all edges with exactly one endpoint in S.

if $y_i=1$ then $\sum_{e\in\delta(S)} x_e \geq 1$, $\forall S \subseteq V: (i\in S, r\notin S)$

① if i in tree, then there is at least one edge $e\in\delta(S)$, $\forall S\subseteq V, i\in S, r\notin S$

Algo. Step 0: Let $\alpha\in[0,1]$

Step 1: Solve LP $\rightarrow x^*, y^*, z^*_{LP}$.

Step 2: Let $U=\{i \mid y^*_i \geq \alpha\}$. Construct a Steiner tree T on U.

Lemma 1. The connection cost for the Steiner tree T is $\sum_{e\in T} c_e \leq \frac{2}{\alpha}\sum_{e\in E} c_e x^*_e$

Skip the proof.

Lemma 2. The total penalty cost is $\sum_{i\in V-V(T)} \pi_i \leq \frac{1}{1-\alpha}\sum_{i\in V}(1-y^*_i)\pi_i$.

Proof: $\boxed{\text{If vertex } i \text{ is not in the tree } T, \text{ then } i\notin U \text{ and } y^*_i<\alpha,}$ which implies that $(1-y^*_i)/(1-\alpha) > 1$.

The total penalty cost is

$$\sum_{i\in V-V(T)} \pi_i \leq \sum_{i\notin U}\pi_i \leq \sum_{i\notin U}\frac{1-y^*_i}{1-\alpha}\pi_i = \frac{1}{1-\alpha}\sum_{i\notin U}(1-y^*_i)\pi_i \leq \frac{1}{1-\alpha}\sum_{i\in V}(1-y^*_i)\pi_i.$$

Theorem 4.3 The algo. above with $\alpha = 2/3$ is a 3-approximation algo.

Proof: The sum of connection and penalty cost is at most

$$\frac{2}{\alpha}\sum_{e\in E} c_e x^*_e + \frac{1}{1-\alpha}\sum_{i\in V}(1-y^*_i)\pi_i \leq \max\left\{\frac{2}{\alpha}, \frac{1}{1-\alpha}\right\} z^*_{LP}$$

The minimized ~~max~~ value $\max\left\{\frac{2}{\alpha}, \frac{1}{1-\alpha}\right\}$ is 3 for $\alpha=2/3$.

Therefore, the total cost of solution of the algo. is at most

$$3 z^*_{LP} \leq 3\,OPT.$$

4.4. Uncapacitated facility location.

Instance: Set of point F (facilities) and set of points D (clients). Opening cost f_i for each $i\in F$ and connection cost (assignment cost) c_{ij} for each pair $i\in F, j\in D$. The connection cost is assumed to be metric.

Solution: $F'\subseteq F$

Cost: $\sum_{i\in F'} f_i + \sum_{j\in D}\min_{i\in F'} c_{ij}$.

Goal: Find the solution of minimum cost.

(ILP) $\min Z = \sum_{i\in F} f_i y_i + \sum_{i\in F, j\in D} c_{ij}x_{ij}$

s.t. $\sum_{i\in F} x_{ij} = 1$ $\forall j\in D$ each client connects to a facility.

$x_{ij} \leq y_i$ $\forall i\in F, j\in D$ client only connects to the open facilities

$x_{ij}\in\{0,1\}$ $\forall i\in F, j\in D$ 1 if connected

$y_i\in\{0,1\}$ $\forall i\in F$. 1 if open.

(D) max $\boxed{Z = \sum_{j\in D} v_j}$ most trivial lower bound is to ignore the facility cost entirely: $f_i=0$. The opt is to open all facilities and assign each client to its nearest facility. If we set $v_j = \min_{i\in F} c_{ij}$, then this lower bound is $\sum_{j\in D} v_j$.

s.t. $\boxed{\sum_{j\in D} w_{ij} \leq f_i}$ $\forall i\in F$

w_{ij}: client j's contribution to opening facility i. the total cost of opening facility i is at least the total contribution from every client to opening facility i.

$v_j \leq c_{ij}+w_{ij}$ $\boxed{v_j - w_{ij} \leq c_{ij}}$ $\forall i\in F, j\in D$

$w_{ij}\geq 0$ $\forall i\in F, j\in D$

v_j is free

the total payment by client j is at most the sum of the contribution it makes to opening facility i and the distance it has to travel to i.

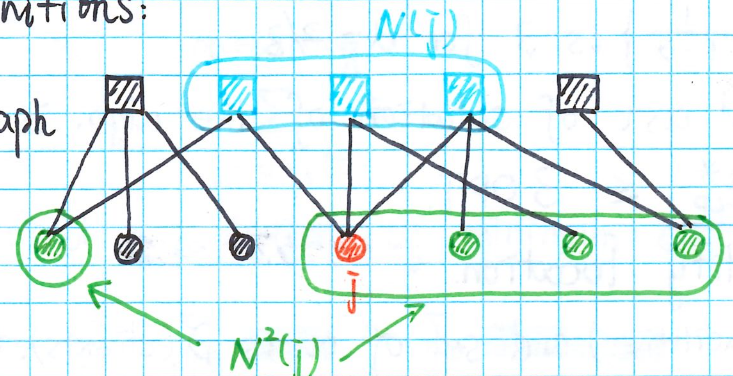
Let x^*, y^* be an optimal primal solution to LP-relaxation and v^*, w^* be an optimal dual solution. By weak duality, $\sum_{j\in D} v_j \leq Z^*_{LP} \leq OPT$.

Lemma If $x^*_{ij} > 0$, then $c_{ij} = v^*_j - w^*_{ij} \le \cancel{c_{ij}}\ v^*_j$

Proof By complementary slackness, $x^*_{ij} > 0$ implies $v^*_j - w^*_{ij} = c_{ij}$.
Since $w^*_{ij} \ge 0$, we have that $c_{ij} \le v^*_j$.

Some definitions:

Support graph



Algo. For $k = 1, 2, \cdots$ until all clients are connected do:

Step 1: Among the unconnected clients, choose client j_k with the smallest value $v^*_{j_k}$.

Step 2: Choose facility $i_k \in N(j_k)$ with the smallest value f_{i_k}.

Step 3: Connect all still unconnected clients in $N^2(j_k)$ to facility i_k.

Theorem 4.4 The algo. above is a 4-approximation algo. for UFL.

Proof: The opening cost for each opened facility i_k is

$$f_{i_k} \overset{①}{=} f_{\boxed{i_k}} \sum_{i \in N(j_k)} x^*_{ij_k} \overset{②}{\le} \sum_{i \in N(j_k)} f_{\boxed{i}}\, x^*_{ij_k} \overset{③}{\le} \sum_{i \in N(j_k)} f_i y^*_i$$

① LP-constraint: $\sum_{i \in F} x_{ij} = 1,\ \forall j \in D$

② The choice of i_k as the cheapest facility in F_k

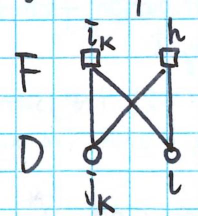
③ LP-constraint: $x_{ij} \le y_{ij},\ \forall i \in F, j \in D$.

The total opening cost is

$$\sum_k f_{i_k} \le \sum_k \sum_{i \in N(j_k)} f_i y^*_i \overset{④}{\le} \sum_{i \in F} f_i y^*_i \le Z^*_{LP} \le OPT.$$

④ $N(j_k)$ partition F: in Step 3, $N(j_k) \cap N(j_{k'}) = \phi$ for any pair $j_k, j_{k'}$.

Consider assigning an unassigned client $l \in N^2(j)$ to facility i_k, where client l neighbors facility h that neighbors client j_k.



The connection cost is

$$c_{i_k l} \overset{⑤}{\le} c_{i_k j_k} + c_{h j_k} + c_{hl} \overset{⑥}{\le} v^*_{j_k} + v^*_{j_k} + v^*_l \overset{⑦}{\le} 3 v^*_l$$

⑤ triangle inequality

⑥ Lemma: complementary slackness

⑦ $v^*_{j_k} \le v^*_l$. Note, both l & j_k were unassigned when j_k was chosen in Step 1.

The total connection cost is at most

$$3 \sum_{l \in D} v^*_l = 3 Z^*_D \le 3 Z^*_{LP} \le 3OPT.$$

Therefore, we conclude that the sum of opening and connection cost is at most $OPT + 3OPT = 4OPT$.

Chapter 5. Random sampling and randomized rounding of LP's.

To show that a randomized algo. is an α-approximation algo., we need to show three things:

1. The algo. runs in polynomial time.
2. The algo. always produces a feasible solution.
3. The expected value is within a factor α of the value of an optimal solution.

5.1 Max Sat and Max cut.

SAT (satisfiability problem): Is there a True/False assignment s.t. all clauses are ~~st~~ satisfied?

Maximum SAT: Maximize the # of satisfied clauses.

Algo. Set all variables independently to true with probability $1/2$.

Theorem 5.1 The algo. above is a $1/2$-approximation algo.

Proof: Let l_j be the # of literals in clause j.

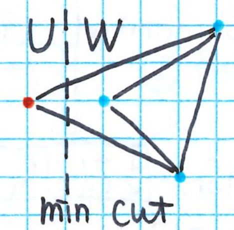
Then, $Pr\boxed{(C_j \text{ is satisfied})} = 1 - (1/2)^{l_j} \ge \frac{1}{2}$

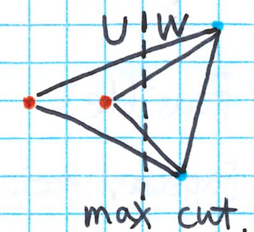
Let W be the total # of satisfied clauses.

Then, $E(W) = \sum_{j=1}^{m} Pr(C_j \text{ is satisfied}) \ge \frac{1}{2} m \ge \frac{1}{2} OPT$.

Remark: If each clause contains at least k literals then from the proof above we see that the approximation guarantee is at least $1 - (1/2)^k$.

Max Cut





Instance	:	~~Gp~~ Graph $G = (V, E)$
Solution	:	$U \subseteq V$
Value	:	Total number of edges between U and $W = V - U$
Goal	:	Maximize value