1 Weighted Vertex Cover

Input: An undirected graph $G(V, E)$ with each vertex i having a weight w_i.
Output: To compute a vertex cover of minimum weight.

Suppose we create a variable x_i for each vertex i such that $x_i = 1$ if vertex i is included in the vertex cover, $x_i = 0$ otherwise. Then the problem can be expressed as the following Integer Program(IP).

$$\begin{aligned} \text{minimize} \quad & z = \sum w_i \cdot x_i \\ \text{s.t.} \quad & x_i + x_j \geq 1, \quad \text{for each}(i, j) \in E \\ & x_i = \{0, 1\}, \quad \forall i \end{aligned} \tag{1.1}$$

Solving the above IP optimally is equivalent to solving the corresponding vertex cover instance optimally, ie $z_{IP} = \texttt{opt}$. However, solving an IP optimally is a NP-hard problem. But we can relax the constraint, $x_i = \{0, 1\}$ and let $x_i \geq 0$ instead. This is called *LP-relaxation* of the IP.

$$\begin{aligned} \text{minimize} \quad & z = \sum w_i \cdot x_i \\ \text{s.t.} \quad & x_i + x_j \geq 1, \quad \text{for each}(i, j) \in E \\ & x_i \geq 0, \quad \forall i \end{aligned} \tag{1.2}$$

Note that the constraint $x_i \leq 1$ is redundant, since any optimal solution with $x_i > 1$ can safely make $x_i = 1$ and still satisfy all the constraints but have lower objective function value.

Every feasible solution to the IP is a feasible solution to LP, ie $z_{LP} \leq z_{IP}$. In other words, the cost of the LP solution is a lower bound on the cost of the IP solution. We can thus solve the LP of equation (1.2) optimally and obtain a vertex cover. More specifically, let, $X^* = \langle x_1^*, x_2^*, \ldots, x_n^* \rangle$ be the solution of equation (1.2). We can then construct a solution X^a for the IP of equation (1.1) as follows: for each x_i^*, if $x_i^* \geq 1/2$ we set $x_i^a = 1$, else $x_i^a = 0$. This assignment satisfies all constraints of equation (1.1). Thus we include those vertices in the vertex cover whose $x_i^a = 1$.

Theorem 1.1. *The above approach gives a 2-approximation guarantee.*

Proof.

$$\begin{aligned} z_{IP}^a &= \sum w_i x_i^a \\ &\leq \sum w_i (2x_i^*) \quad \text{(since at the most we are doubling each } x_i^*\text{)} \\ &= 2 \cdot \sum w_i x_i^* \\ &= 2z_{LP} \\ &\leq 2z_{IP} \end{aligned}$$

□

2 Job Scheduling to minimize sum of completion times

Input: A set J of n jobs with each job i having release time r_i and processing time p_i.
Output: A non-preemptive schedule of jobs such that $\sum C_i$ is the least, where C_i is the completion time of job i.

If all jobs are available at the start of the algorithm($r_i \leq 0$) or all of them are released at the same time, then by using *shortest job first (SJF)* rule, we get the optimal schedule. Also, if preemption was permitted, then *shortest remaining time first (SRTF)* rule gives the optimal solution. Since every non-preemptive schedule can be a preemptive one, the cost of the preemptive schedule is a lower bound on the cost of the optimal solution, ie

$$\sum C_i^P \leq \texttt{opt}$$

We use "information" provided by SRTF rule to help us in our algorithm for non-preemptive scheduling. More concretely, we schedule the jobs in J according to SRTF rule and then arrange them in non-decreasing order of their completion times, C_i^P. Without loss of generality, let $C_1^P \leq C_2^P \leq \ldots \leq C_n^P$. Therefore, in our schedule, we schedule job 1 first, then wait for release time of job 2 if needed, schedule job 2 and so on. If C_i^N is the completion time of the i^{th} job in our schedule, the cost of our solution is $\sum C_i^N$.

Lemma 2.1. *For each job $j = 1, 2, \ldots, n$, we have $C_j^N \leq 2C_j^P$.*

Proof. Consider job j in the preemptive schedule. Clearly, it must have had to wait until the maximum release time of all jobs that came before it, ie

$$C_j^P \geq \max_{k=1,2,\ldots,j} r_k \tag{2.1}$$

Also, assuming all jobs from $k = 1, 2, \ldots, j$ are scheduled without any idle time on the machine,

$$C_j^P \geq \sum_{k=1}^{j} p_k \tag{2.2}$$

In the non-preemptive schedule, the last we can complete job j would be to wait until $\max_{k=1,2,\ldots,j} r_k$ and then schedule all jobs consecutively, without any idle time on the machine(since all of them would be available). Thus,

$$C_j^N \leq \max_{k=1,2,\ldots,j} r_k + \sum_{k=1}^{j} p_k \tag{2.3}$$

Using results from (2.1) and (2.2) we get $C_j^N \leq 2C_j^P$. □

Theorem 2.2. *Our approach gives a 2-approximation guarantee.*

Proof.

$$\sum C_i^N \leq \sum 2C_i^P = 2 \cdot \sum C_i^P \leq 2 \cdot \texttt{opt}$$

□

3 Job Scheduling to minimize sum of weighted completion times

Input: A set J of n jobs with each job i having release time r_i, processing time p_i and weight w_i.
Output: A non-preemptive schedule of jobs such that $\sum w_i C_i$ is the least, where C_i is the completion time of job i.

If we use the above approach of scheduling jobs preemptively using SRTF rule, we can easily see that this rule will not give an optimal solution when all the jobs *do not* have the same weight. An approach we use is to model the problem as a linear program. If C_i is the completion time of job i, and $p(S) = \sum_i p_i$ for a set of jobs S, then,

$$\begin{aligned}
\text{minimize} \quad & z = \sum w_i C_i \qquad (3.1)\\
\text{s.t.} \quad & C_i \geq r_i + p_i, \quad \forall i \\
& \sum_{i \in S} p_i C_i \geq \frac{1}{2} p(S)^2, \quad \forall S \subseteq J
\end{aligned}$$

The above linear program is an LP-relaxation of the original problem and an optimal solution to it will give a lower bound on the optimal solution to our original problem. Additionally, we have an exponential number of constraints since the second constraint applies on all $S \subseteq J$.[1] The set of constraints is by no means complete, but we try to choose a set of *good* constraints which will help us get a good solution for the LP. We now look at the rationale behind the second constraint: The primary constraint we have is our machine can process only

[1]Solving such an LP would be seen later in the course.

one job at a time. Keeping this in mind, suppose all jobs in S are available and we schedule them one after the other without any idle time, we get,

$$
\begin{aligned}
\sum_{i\in S} p_i C_i &= p_1 C_1 + p_2 C_2 + \ldots + p_k C_k \\
&= p_1 p_1 + p_2(p_1 + p_2) + \ldots + p_k(p_1 + p_2 + \ldots + p_k) \\
&= p_1^2 + p_2^2 + \ldots + p_k^2 + \sum_{1\le a<b\le k} p_a p_b \\
&= \frac{1}{2}(2p_1^2 + 2p_2^2 + \ldots + 2p_k^2 + \sum_{1\le a<b\le k} 2p_a p_b) \\
&= \frac{1}{2}(p_1^2 + p_2^2 + \ldots + p_k^2) + \frac{1}{2}(p_1^2 + p_2^2 + \ldots + p_k^2 + \sum_{1\le a<b\le k} 2p_a p_b) \\
&= \frac{1}{2}\sum p_i^2 + \frac{1}{2}(\sum p_i)^2 \\
&\le \frac{1}{2}(\sum p_i)^2 \\
&= \frac{1}{2}p(S)^2
\end{aligned}
$$

If all the jobs in S are not available or there is some idle time, then the term on the LHS only increases.

We solve the linear program of equation (3.1) optimally and let C^* be the solution returned. Without loss of generality, suppose $C_1^* \le C_2^* \le \ldots \le C_n^*$. Our non-preemptive algorithm then schedules job 1 first, followed by job 2 (waiting for its release time if needed), and so on. Let C_i^N be the completion time of the i^{th} job in our schedule.

Lemma 3.1. *For each job $j = 1, 2, \ldots, n$, we have $C_j^N \le 3C_j^*$.*

Proof. By a similar reasoning as in equation (2.3) we get,

$$C_j^N \le \max_{k=1,2,\ldots,j} r_k + \sum_{k=1}^{j} p_k \tag{3.2}$$

The first constraint of the LP gives,

$$C_j^* \ge \max_{k=1,2,\ldots,j} r_k \tag{3.3}$$

To bound $\sum_{k=1}^{j} p_k$, we cannot use a relation on the lines of equation (2.2). To see this, assume the input was $p = \langle 2, 2, 2\rangle$, $r = \langle 2, 2, 2\rangle$ and $w = \langle 1, 2, 3\rangle$. Then the LP would give a solution $C^* = \langle 4, 4, 4\rangle$. Note that this solution satisfies all the constraints and minimizes z. But $C_3^* \ngeq \sum_{k=1}^{3} p_k$.

To get a bound on $\sum_{k=1}^{j} p_k$, we use the second constraint. Let S be the set of jobs from 1 to j. Then,

$$\sum_{k\in S} p_k C_k^* \ge \frac{1}{2}(\sum_{k=1}^{j} p_k)^2$$

Since j is the job that finished last in S, C_j is the highest. Therefore,

$$C_j^* \cdot \sum_{k\in S} p_k \ge \sum_{k\in S} p_k C_k^* \ge \frac{1}{2}(\sum_{k=1}^{j} p_k)^2$$

Thus, we get,

$$2C_j^* \ge \sum_{k=1}^{j} p_k \tag{3.4}$$

Using results from (3.3) and (3.4) in (3.2) we get $C_j^N \le 3C_j^*$. □

Theorem 3.2. *Our approach gives a 3-approximation guarantee.*

Proof.

$$\sum w_i C_i^N \le \sum w_i (3C_i^*) = 3 \cdot \sum w_i C_i^* \le 3 \cdot \texttt{opt}$$

□

4 Uncapacitated Facility Location

Input: A set F of facilities and a set D of clients such that associated with each facility i we have a facility opening cost f_i and for each pair of facility i and client j we have a connection cost c_{ij}, ie cost of assigning client j to facility i. The costs c_{ij} obey the triangle inequality.
Output: We wish to open a set of facilities $F' \subseteq F$ such that the total facility opening cost and connection cost is minimized, ie $\sum_{i\in F'} f_i + \sum_{j\in D} \min_{i\in F'} c_{ij}$ is the least.

Let us define an integer program for our problem. For each facility i we introduce a variable y_i; if facility i is open, $y_i = 1$, else $y_i = 0$. Similarly, for each pair of facility i and client j, we introduce a variable x_{ij}; if client j is assigned to facility i, $x_{ij} = 1$, else $x_{ij} = 0$. We can now write our objective function as,

$$\text{minimize} \quad \sum_{i\in F} y_i f_i + \sum_{i\in F, j\in D} x_{ij} c_{ij}$$

Our first constraint is each client should be assigned to exactly one facility. We express this as,

$$\sum_{i\in F} x_{ij} = 1, \quad \forall j \in D$$

Our next constraint specifies that if a client j is assigned to some facility i, it should be open. We express this as,

$$y_i - x_{ij} \geq 0, \quad \forall i \in F, j \in D$$

Finally, we require that $y_i = \{0,1\}$ and $x_{ij} = \{0,1\}$. As usual, we obtain the LP-relaxation of our integer program by replacing, $y_i = \{0,1\}$ with $y_i \geq 0$ and $x_{ij} = \{0,1\}$ with $x_{ij} \geq 0$. The complete linear program is outlined below,

$$\text{minimize} \quad z = \sum_{i\in F} y_i f_i + \sum_{i\in F, j\in D} x_{ij} c_{ij} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i\in F} x_{ij} = 1, \quad \forall j \in D \tag{4.2}$$

$$y_i - x_{ij} \geq 0, \quad \forall i \in F, j \in D \tag{4.3}$$

$$y_i \geq 0, \quad \forall i \in F$$

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

Let us obtain the dual of the above linear program. We create a variable α_j for each constraint of type (4.2) and a variable β_{ij} for each constraint of type (4.3).

$$\text{maximize} \quad z' = \sum_{j\in D} \alpha_j \tag{4.4}$$

$$\text{s.t.} \quad \sum_{j\in D} \beta_{ij} \leq f_i, \quad \forall i \in F \tag{4.5}$$

$$\alpha_j - \beta_{ij} \leq c_{ij}, \quad \forall i \in F, j \in D \tag{4.6}$$

$$\alpha_j \geq 0, \quad \forall j \in D$$

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

If z^* is the primal optimal solution and z'^* is the dual optimal solution, by strong duality theorem we have, $z^* = z'^*$. Thus each is a lower bound on the optimal solution for our original problem, ie $z^* = z'^* \leq \texttt{opt}$. In the analysis of the problem, we will use the primal optimal solution to bound the facility opening cost and the dual optimal solution to bound the connection cost.

Suppose (y_i^*, x_{ij}^*) and $(\alpha_j^*, \beta_{ij}^*)$ are the primal and dual optimal solutions respectively. For a client j, we define its neighborhood $N(j)$ to be the set of all facilities with whom the client gets "fractionally" serviced. More precisely, $N(j) = \{i \in F | x_{ij}^* > 0\}$. By the complementary slackness condition,

$$x_{ij}^* > 0 \implies \alpha_j^* - \beta_{ij}^* = c_{ij} \implies c_{ij} \leq \alpha_j^*$$

Thus for each client j, if we open a facility $i \in N(j)$, the connection cost c_{ij} is not more than α_j^*. The total connection cost therefore is not more than $\sum_{j\in D} \alpha_j^*$, the dual optimal solution. *The total connection cost is thus upper bounded by* $\texttt{opt}$.