Chapter 20

Approximation Algorithms using Linear Programming

By Sariel Har-Peled, December 10, 2013[1]

Version: 0.2

20.1 Weighted vertex cover

Consider the Weighted Vertex Cover problem. Here, we have a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, and each vertex $\mathsf{v} \in \mathsf{V}$ has an associated cost c_v. We would like to compute a vertex cover of minimum cost – a subset of the vertices of G with minimum total cost so that each edge has at least one of its endpoints in the cover. This problem is (of course) NP-Hard, since the decision problem where all the weights are 1, is the Vertex Cover problem, which we had shown to be NPC.

Let us first state this optimization problem is an integer programming. Indeed, for any $\mathsf{v} \in \mathsf{V}$, let define a variable x_v which is 1 if we decide to pick v to the vertex cover, and zero otherwise. The restriction that x_v is either 0 or 1, is written formally as $x_\mathsf{v} \in \{0, 1\}$. Next, its required that every edge $\mathsf{vu} \in \mathsf{E}$ is covered. Namely, we require that $x_\mathsf{v} \vee x_\mathsf{u}$ to be TRUE. For reasons that would be come clearer shortly, we prefer to write this condition as a linear inequality; namely, we require that $x_\mathsf{v} + x_\mathsf{u} \geq 1$. Finally, we would like to minimize the total cost of the vertices we pick for the cover; namely, we would like to minimize $\sum_{\mathsf{v} \in \mathsf{V}} x_\mathsf{v} \mathsf{c}_\mathsf{v}$. Putting it together, we get the following integer programming instance:

$$\begin{aligned} \min \quad & \sum_{\mathsf{v} \in \mathsf{V}} \mathsf{c}_\mathsf{v} x_\mathsf{v}, & & \\ \text{such that} \quad & x_\mathsf{v} \in \{0, 1\} & \forall \mathsf{v} \in \mathsf{V} & \\ & x_\mathsf{v} + x_\mathsf{u} \geq 1 & \forall \mathsf{vu} \in \mathsf{E}. & \end{aligned} \tag{20.1}$$

Naturally, solving this integer programming efficiently is NP-Hard, so instead let us try to relax this optimization problem to be a LP (which we can solve efficiently, at least in practice[2]). To do this, we need to relax the integer program. We will do it by allowing the variables x_v to get real values between 0 and 1. This is done by replacing the condition that $x_\mathsf{v} \in \{0, 1\}$ by the constraint $0 \leq x_\mathsf{v} \leq 1$.

[2] And also in theory if the costs are integers, using more advanced algorithms than the **Simplex** algorithm.

The resulting LP is

$$\begin{aligned} \min \quad & \sum_{\mathsf{v}\in\mathsf{V}} \mathsf{c}_\mathsf{v} x_\mathsf{v}, & & \\ \text{such that} \quad & 0 \le x_\mathsf{v} & \forall \mathsf{v} \in \mathsf{V}, & \\ & x_\mathsf{v} \le 1 & \forall \mathsf{v} \in \mathsf{V}, & (20.2) \\ & x_\mathsf{v} + x_\mathsf{u} \ge 1 & \forall \mathsf{vu} \in \mathsf{E}. & \end{aligned}$$

So, consider the optimal solution to this LP, assigning value $\widehat{x_\mathsf{v}}$ to the variable X_v, for all $\mathsf{v} \in \mathsf{V}$. As such, the optimal value of the LP solution is

$$\widehat{\alpha} = \sum_{\mathsf{v}\in\mathsf{V}} \mathsf{c}_\mathsf{v} \widehat{x_\mathsf{v}}.$$

Similarly, let the optimal integer solution to integer program (IP) Eq. (20.1) denoted by x^I_v, for all $\mathsf{v} \in \mathsf{V}$ and α^I, respectively. Note, that any feasible solution for the IP of Eq. (20.1), is a feasible solution for the LP of Eq. (20.2). As such, we must have that

$$\widehat{\alpha} \le \alpha^I,$$

where α^I is the value of the optimal solution.

So, what happened? We solved the relaxed optimization problem, and got a fractional solution (i.e., values of $\widehat{x_\mathsf{v}}$ can be fractions). On the other hand, the cost of this fractional solution is better than the optimal cost. So, the natural question is how to turn this fractional solution into a (valid!) integer solution. This process is known as ***rounding***.

To this end, it is beneficial to consider a vertex v and its fractional value $\widehat{x_\mathsf{v}}$. If $\widehat{x_\mathsf{v}} = 1$ then we definitely want to put it into our solution. If $\widehat{x_\mathsf{v}} = 0$ then the LP consider this vertex to be useless, and we really do not want to use it. Similarly, if $\widehat{x_\mathsf{v}} = 0.9$, then the LP considers this vertex to be very useful (0.9 useful to be precise, whatever this "means"). Intuitively, since the LP puts its money where its belief is (i.e., $\widehat{\alpha}$ value is a function of this "belief" generated by the LP), we should trust the LP values as a guidance to which vertices are useful and which are not. Which brings to forefront the following idea: Lets pick all the vertices that are above a certain threshold of usefulness according to the LP solution. Formally, let

$$S = \left\{ \mathsf{v} \;\middle|\; \widehat{x_\mathsf{v}} \ge 1/2 \right\}.$$

We claim that S is a valid vertex cover, and its cost is low.

Indeed, let us verify that the solution is valid. We know that for any edge vu, it holds

$$\widehat{x_\mathsf{v}} + \widehat{x_\mathsf{u}} \ge 1.$$

Since $0 \le \widehat{x_\mathsf{v}} \le 1$ and $0 \le \widehat{x_\mathsf{u}} \le 1$, it must be either $\widehat{x_\mathsf{v}} \ge 1/2$ or $\widehat{x_\mathsf{u}} \ge 1/2$. Namely, either $\mathsf{v} \in S$ or $\mathsf{u} \in S$, or both of them are in S, implying that indeed S covers all the edges of G.

As for the cost of S, we have

$$\mathsf{c}_S = \sum_{\mathsf{v}\in S} \mathsf{c}_\mathsf{v} = \sum_{\mathsf{v}\in S} 1 \cdot \mathsf{c}_\mathsf{v} \le \sum_{\mathsf{v}\in S} 2\widehat{x_\mathsf{v}} \cdot \mathsf{c}_\mathsf{v} \le 2 \sum_{\mathsf{v}\in\mathsf{V}} \widehat{x_\mathsf{v}} \mathsf{c}_\mathsf{v} = 2\widehat{\alpha} \le 2\alpha^I,$$

since $\widehat{x_\mathsf{v}} \ge 1/2$ as $\mathsf{v} \in S$.

Since α^I is the cost of the optimal solution, we got the following result.

Theorem 20.1.1. *The Weighted Vertex Cover problem can be 2-approximated by solving a single LP. Assuming computing the LP takes polynomial time, the resulting approximation algorithm takes polynomial time.*

What lessons can we take from this example? First, this example might be simple, but the resulting approximation algorithm is non-trivial. In particular, I am not aware of any other 2-approximation algorithm for the weighted problem that does not use LP. Secondly, the ***relaxation*** of an optimization problem into a LP provides us with a way to get some insight into the problem in hand. It also hints that in interpreting the values returned by the LP, and how to use them to do the rounding, we have to be creative.

20.2 Revisiting Set Cover

In this section, we are going to revisit the Set Cover problem, and provide an approximation algorithm for this problem. This approximation algorithm would not be better than the greedy algorithm we already saw, but it would expose us to a new technique that we would use shortly for a different problem.

Set Cover

Instance: $(S, \mathcal{F})$
S - a set of n elements
$\mathcal{F}$ - a family of subsets of S, s.t. $\bigcup_{X \in \mathcal{F}} X = S$.
Question: The set $\mathcal{X} \subseteq F$ such that $\mathcal{X}$ contains as few sets as possible, and $\mathcal{X}$ covers S.

As before, we will first define an IP for this problem. In the following IP, the second condition just states that any $s \in s$, must be covered by some set.

$$\begin{aligned} \min \quad & \alpha = \sum_{U \in \mathcal{F}} x_U, & \\ \text{s.t.} \quad & x_U \in \{0,1\} & \forall U \in \mathcal{F}, \\ & \sum_{U \in \mathcal{F}, s \in U} x_U \geq 1 & \forall s \in S. \end{aligned}$$

Next, we relax this IP into the following LP.

$$\begin{aligned} \min \quad & \alpha = \sum_{U \in \mathcal{F}} x_U, & \\ & 0 \leq x_U \leq 1 & \forall U \in \mathcal{F}, \\ & \sum_{U \in \mathcal{F}, s \in U} x_U \geq 1 & \forall s \in S. \end{aligned}$$

As before, consider the optimal solution to the LP: $\forall U \in \mathcal{F}$, $\widehat{x_U}$, and $\widehat{\alpha}$. Similarly, let the optimal solution to the IP (and thus for the problem) be: $\forall U \in \mathcal{F}$, x_U^I, and α^I. As before, we would try to use the LP solution to guide us in the rounding process. As before, if $\widehat{x_U}$ is close to 1 then we should pick U to the cover and if $\widehat{x_U}$ is close to 0 we should not. As such, its natural to pick $U \in \mathcal{F}$ into the cover by randomly choosing it into the cover with ***probability*** $\widehat{x_U}$. Consider the resulting family of sets $\mathcal{G}$.

Let Z_S be an indicator variable which is one if $S \in \mathcal{G}$. We have that the cost of $\mathcal{G}$ is $\sum_{S\in\mathcal{F}} Z_S$, and the expected cost is

$$\mathbf{E}\Big[\text{cost of } \mathcal{G}\Big] = \mathbf{E}\left[\sum_{S\in\mathcal{F}} Z_S\right] = \sum_{S\in\mathcal{F}} \mathbf{E}\Big[Z_S\Big] = \sum_{S\in\mathcal{F}} \mathbf{Pr}\Big[S \in \mathcal{G}\Big] = \sum_{S\in\mathcal{F}} \widehat{x_S} = \widehat{\alpha} \leq \alpha^I. \tag{20.3}$$

As such, in expectation, $\mathcal{G}$ is not too expensive. The problem, of course, is that $\mathcal{G}$ might fail to cover some element $s \in S$. To this end, we repeat this algorithm

$$m = 10\lceil \lg n \rceil = O(\log n)$$

times, where $n = |S|$. Let $\mathcal{G}_i$ be the random cover computed in the ith iteration, and let $\mathcal{H} = \cup_i \mathcal{G}_i$. We return $\mathcal{H}$ as the required cover.

The solution $\mathcal{H}$ covers S. For an element $s \in S$, we have that

$$\sum_{U\in\mathcal{F}, s\in U} \widehat{x_U} \geq 1, \tag{20.4}$$

and consider the probability that s is not covered by $\mathcal{G}_i$, where $\mathcal{G}_i$ is the family computed in the ith iteration of the algorithm. Since deciding if the include each set U into $\mathcal{G}_i$ is done independently for each set, we have that the probability that s is not covered is

$$\begin{aligned}
\mathbf{Pr}\Big[s \text{ not covered by } \mathcal{G}_i\Big] &= \mathbf{Pr}\Big[\text{ none of } U \in \mathcal{F}, \text{ such that } s \in U \text{ were picked into } \mathcal{G}_i\Big] \\
&= \prod_{U\in\mathcal{F}, s\in U} \mathbf{Pr}\Big[U \text{ was not picked into } \mathcal{G}_i\Big] \\
&= \prod_{U\in\mathcal{F}, s\in U} (1 - \widehat{x_U}) \\
&\leq \prod_{U\in\mathcal{F}, s\in U} \exp(-\widehat{x_U}) = \exp\left(-\sum_{U\in\mathcal{F}, s\in U} \widehat{x_U}\right) \\
&\leq \exp(-1) \leq \frac{1}{2},
\end{aligned}$$

by Eq. (20.4). As such, the probability that s is not covered in all m iterations is at most

$$\left(\frac{1}{2}\right)^m < \frac{1}{n^{10}},$$

since $m = O(\log n)$. In particular, the probability that one of the n elements of S is not covered by $\mathcal{H}$ is at most $n(1/n^{10}) = 1/n^9$.

Cost. By Eq. (20.3), in each iteration the expected cost of the cover computed is at most the cost of the optimal solution (i.e., α^I). As such the expected cost of the solution computed is

$$\mathsf{c}_{\mathcal{H}} \leq \sum_i \mathsf{c}_{B_i} \leq m\alpha^I = O\Big(\alpha^I \log n\Big).$$

. Putting everything together, we get the following result.

Theorem 20.2.1. *By solving an LP one can get an $O(\log n)$-approximation to set cover by a randomized algorithm. The algorithm succeeds with high probability.*

20.3 Minimizing congestion

Let G be a graph with n vertices, and let π_i and σ_i be two paths with the same endpoints $\mathsf{v}_i, \mathsf{u}_i \in \mathsf{V}(\mathsf{G})$, for $i = 1, \ldots, t$. Imagine that we need to send one unit of flow from v_i to u_i, and we need to choose whether to use the path π_i or σ_i. We would like to do it in such a way that no edge in the graph is being used too much.

Definition 20.3.1. Given a set X of paths in a graph G, the ***congestion*** of X is the maximum number of paths in X that use the same edge.

Consider the following linear program:

$$\begin{aligned}
\min \quad & w && \\
\text{s.t.} \quad & x_i \geq 0 && i = 1, \ldots, t, \\
& x_i \leq 1 && i = 1, \ldots, t, \\
& \sum_{\mathsf{e} \in \pi_i} x_i + \sum_{\mathsf{e} \in \sigma_i} (1 - x_i) \leq w && \forall \mathsf{e} \in E.
\end{aligned}$$

Let $\widehat{x_i}$ be the value of x_i in the optimal solution of this LP, and let $\widehat{w}$ be the value of w in this solution. Clearly, the optimal congestion must be bigger than $\widehat{w}$.

Let X_i be a random variable which is one with probability $\widehat{x_i}$, and zero otherwise. If $X_i = 1$ then we use π to route from v_i to u_i, otherwise we use σ_i. Clearly, the congestion of e is

$$Y_{\mathsf{e}} = \sum_{\mathsf{e} \in \pi_i} X_i + \sum_{\mathsf{e} \in \sigma_i} (1 - X_i).$$

And in expectation

$$\begin{aligned}
\alpha_{\mathsf{e}} = \mathbf{E}\Big[Y_{\mathsf{e}}\Big] &= \mathbf{E}\left[\sum_{\mathsf{e} \in \pi_i} X_i + \sum_{\mathsf{e} \in \sigma_i} (1 - X_i)\right] = \sum_{\mathsf{e} \in \pi_i} \mathbf{E}\Big[X_i\Big] + \sum_{\mathsf{e} \in \sigma_i} \mathbf{E}\Big[(1 - X_i)\Big] \\
&= \sum_{\mathsf{e} \in \pi_i} \widehat{x_i} + \sum_{\mathsf{e} \in \sigma_i} (1 - \widehat{x_i}) \leq \widehat{w}.
\end{aligned}$$

Using the Chernoff inequality, we have that

$$\mathbf{Pr}\Big[Y_{\mathsf{e}} \geq (1 + \delta)\alpha_{\mathsf{e}}\Big] \leq \exp\left(-\frac{\alpha_{\mathsf{e}}\delta^2}{4}\right) \leq \exp\left(-\frac{\widehat{w}\delta^2}{4}\right).$$

(Note, that this works only if $\delta < 2e - 1$, see Theorem 20.3.4). Let $\delta = \sqrt{\frac{400}{\widehat{w}} \ln t}$. We have that

$$\mathbf{Pr}\Big[Y_{\mathsf{e}} \geq (1 + \delta)\alpha_{\mathsf{e}}\Big] \leq \exp\left(-\frac{\delta^2\widehat{w}}{4}\right) \leq \frac{1}{t^{100}},$$

which is very small. In particular, if $t \geq n^{1/50}$ then all the edges in the graph do not have congestion larger than $(1 + \delta)\widehat{w}$.

To see what this result means, let us play with the numbers. Let assume that $t = n$, and $\widehat{w} \geq \sqrt{n}$. Then, the solution has congestion larger than the optimal solution by a factor of

$$1 + \delta = 1 + \sqrt{\frac{20}{\widehat{w}} \ln t} \leq 1 + \frac{\sqrt{20 \ln n}}{n^{1/4}},$$

which is of course extremely close to 1, if n is sufficiently large.

Theorem 20.3.2. *Given a graph with n vertices, and t pairs of vertices, such that for every pair (s_i, t_i) there are two possible paths to connect s_i to t_i. Then one can choose for each pair which path to use, such that the most congested edge, would have at most $(1+\delta)\mathrm{opt}$, where opt is the congestion of the optimal solution, and $\delta = \sqrt{\frac{20}{\widehat{w}} \ln t}$.*

When the congestion is low. Assume that $\widehat{w}$ is a constant. In this case, we can get a better bound by using the Chernoff inequality in its more general form, see Theorem 20.3.4. Indeed, set $\delta = c \ln t / \ln \ln t$, where c is a constant. For $\mu = \alpha_{\mathsf{e}}$, we have that

$$\begin{aligned}\mathbf{Pr}\Big[Y_{\mathsf{e}} \geq (1+\delta)\mu\Big] &\leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} = \exp\left(\mu\Big(\delta - (1+\delta)\ln(1+\delta)\Big)\right) = \exp\left(-\mu c' \ln t\right) \\ &\leq \frac{1}{t^{O(1)}},\end{aligned}$$

where c' is a constant that depends on c and grows if c grows. We thus proved that if the optimal congestion is $O(1)$, then the algorithm outputs a solution with congestion $O(\log t / \log\log t)$, and this holds with high probability.

20.3.0.1 The Chernoff Bound — General Case

We remind the reader about the Chernoff inequality that we used in the above analysis.

Here we present the Chernoff bound in a more general settings.

Problem 20.3.3. Let $X_1, \ldots X_n$ be n independent Bernoulli trials, where

$$\mathbf{Pr}\Big[X_i = 1\Big] = p_i, \qquad \mathbf{Pr}\Big[X_i = 0\Big] = 1 - p_i,$$
$$Y = \sum_i X_i, \qquad \text{and} \qquad \mu = \mathbf{E}\Big[Y\Big].$$

We are interested in bounding the probability that $Y \geq (1+\delta)\mu$.

Theorem 20.3.4 (Chernoff inequality). *For any $\delta > 0$,*

$$\mathbf{Pr}\Big[Y > (1+\delta)\mu\Big] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}.$$

Or in a more simplified form, for any $\delta \leq 2e - 1$,

$$\mathbf{Pr}\Big[Y > (1+\delta)\mu\Big] < \exp\Big(-\mu\delta^2/4\Big),$$

and

$$\mathbf{Pr}\Big[Y > (1+\delta)\mu\Big] < 2^{-\mu(1+\delta)},$$

for $\delta \geq 2e - 1$.

Theorem 20.3.5. *Under the same assumptions as the theorem above, we have*

$$\mathbf{Pr}\Big[Y < (1-\delta)\mu\Big] \leq \exp\left(-\mu\frac{\delta^2}{2}\right).$$