Approximation via LP Rounding

Let $G = (V, E)$ be an (undirected) graph. A subset $C \subset V$ is called a *vertex cover* for G if for every edge $(v_i, v_j) \in E$ we have $v_i \in C$ or $v_j \in C$ (or both). In other words, for every edge in E at least one of its endpoints is in C.

3.1 Unweighted Vertex Cover

The unweighted version of the Vertex Cover problem is to find a vertex cover of minimum size for a given graph G. This problem is NP-complete.

Let's try to come up with an approximation algorithm. A natural greedy approach would be the following. Initialize the cover C as the empty set, and set $E' := E$; the set E' will contain the edges that are not yet covered by C. Now take an edge $(v_i, v_j) \in E'$, put one of its two vertices, say v_i, into C, and remove from E' all edges incident to v_i. Repeat the process until $E' = \emptyset$. Clearly C is a vertex cover after the algorithm has finished. Unfortunately the algorithm has a very bad approximation ratio: there are instances where it can produce a vertex cover of size $|V| - 1$ even though a vertex cover of size 1 exists. A small change in the algorithm leads to a 2-approximation algorithm. The change is based on the following lower bound. Call two edges $e, e' \in E$ *disjoint* if they do not have a vertex in common.

Lemma 3.1 *Let $G = (V, E)$ be a graph and let* OPT *denote the minimum size of a vertex cover for G. Let $E^* \subset E$ be any subset of pairwise disjoint edges, that is, any subset such that each pair of edges in E^* is disjoint. Then* $\text{OPT} \geqslant |E^*|$.

Proof. Let C be an optimal vertex cover for G. By definition, any edge $e \in E^*$ must be covered by a vertex in C, and since the edges in E^* are disjoint any vertex in C can cover at most one edge in E^*. □

This lemma suggests the greedy algorithm given in Algorithm 3.1. It is easy to check that the **while**-loop in the algorithm indeed maintains the stated invariant. After the **while**-loop has terminated—the loop must terminate since at every step we remove at least one edge from E'—we have $E \setminus E' = E \setminus \emptyset = E$. Together with the invariant this implies that the algorithm indeed returns a vertex cover. Next we show that the algorithm gives a 2-approximation.

Theorem 3.2 *Algorithm* ApproxVertexCover *produces a vertex cover C such that* $|C| \leqslant 2 \cdot \text{OPT}$, *where* OPT *is the minimum size of a vertex cover.*

Algorithm 3.1 Approximation algorithm for Vertex Cover.

ApproxVertexCover(V, E)
1: $C \leftarrow \emptyset$; $E' \leftarrow E$ ▷ Invariant: C is a vertex cover for $G' = (V, E \setminus E')$
2: **while** $E' \neq \emptyset$ **do**
3: Take an arbitrary edge $(v_i, v_j) \in E'$.
4: $C \leftarrow C \cup \{v_i, v_j\}$.
5: Remove (v_i, v_j), and all other edges with v_i or v_j as an endpoint, from E'.
6: **end while**
7: **return** C

Proof. Let E^* be the set of edges selected in line 3 over the course of the algorithm. Then C consists of the endpoints of the edges in E^*, and so $|C| \leqslant 2|E^*|$. Moreover, any two edges in E^* are disjoint because as soon as an edge (v_i, v_j) is selected from E' all edges in E' that share v_i and/or v_j are removed from E'. The theorem now follows from Lemma 3.1. □

3.2 Weighted Vertex Cover

Now let's consider a generalization of Vertex Cover, where each vertex $v_i \in V$ has a weight $\mathit{weight}(v_i)$ and we want to find a vertex cover of minimum total weight. We call the new problem Weighted Vertex Cover. The first idea that comes to mind to get an approximation algorithm for Weighted Vertex Cover is to generalize *ApproxVertexCover* as follows: instead of selecting an arbitrary edge (v_i, v_j) from E' in line 3, we select the edge of minimum weight (where the weight of an edge is defined as the sum of the weights of its endpoints). Unfortunately this doesn't work: the weight of the resulting cover can be arbitrarily much larger than the weight of an optimal cover. Our new approach will be based on linear programming.

(Integer) linear programming. In a linear-programming problem we are given a linear *cost function* of d real variables $x_1, \ldots, x_d$ and a set of n linear *constraints* on these variables. The goal is to assign values to the variables so that the cost function is minimized (or: maximized) and all constraints are satisfied. In other words, the Linear Programming problem can be stated as follows:

$$\begin{aligned} \text{Minimize} \quad & c_1 x_1 + \cdots + c_d x_d \\ \text{Subject to} \quad & a_{1,1} x_1 + \cdots + a_{1,d} x_d \leqslant b_1 \\ & a_{2,1} x_1 + \cdots + a_{2,d} x_d \leqslant b_2 \\ & \vdots \\ & a_{n,1} x_1 + \cdots + a_{n,d} x_d \leqslant b_n \end{aligned}$$

There are algorithms—for example the so-called *interior-point methods*—that can solve linear programs in time polynomial in the input size.[1] In practice linear programming is often done

[1] Here the input size is measured in terms of the number of bits needed to describe the input, so this is different from the usual notion of input size.

with with famous *simplex method*, which is exponential in the worst case but works quite well in most practical applications. Hence, if we can formulate a problem as a linear-programming problem then we can solve it efficiently, both in theory and in practice.

There are several problems that can be formulated as a linear-programming problem but with one twist: the variables $x_1, \ldots, x_d$ can not take on real values but only integer values. This is called INTEGER LINEAR PROGRAMMING. (When the variables can only take the values 0 or 1, the problem is called 0/1 LINEAR PROGRAMMING.) Unfortunately, INTEGER LINEAR PROGRAMMING and 0/1 LINEAR PROGRAMMING are considerably harder than LINEAR PROGRAMMING. In fact, INTEGER LINEAR PROGRAMMING and 0/1 LINEAR PROGRAMMING are NP-complete. However, formulating a problem as an integer linear program can still be useful, as shall see next.

Approximating via LP relaxation and rounding. Let's go back to WEIGHTED VERTEX COVER. Thus we are given a weighted graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, and we want to find a minimum-weight vertex cover. To formulate this as a 0/1 linear program, we introduce a variable x_i for each vertex $v_i \in V$; the idea is to have $x_i = 1$ if v_i is taken into the vertex cover, and $x_i = 0$ if v_i is not taken into the cover. When is a subset $C \subseteq V$ a vertex cover? Then C must contain at least one endpoint for every edge (v_i, v_j). This means we must have $x_i = 1$ or $x_j = 1$. We can enforce this by introducing for every edge $(v_i, v_j) \in E$ the constraint $x_i + x_j \geqslant 1$. Finally, we wish to minimize the total weight of the cover, so we get as a cost function $\sum_{i=1}^{n} \mathit{weight}(v_i) \cdot x_i$. To summarize, solving the weighted vertex-cover problem corresponds to solving the following 0/1 linear-programming problem.

$$\begin{aligned} \text{Minimize} \quad & \textstyle\sum_{i=1}^{n} \mathit{weight}(v_i) \cdot x_i & & \\ \text{Subject to} \quad & x_i + x_j \geqslant 1 & & \text{for all edges } (v_i, v_j) \in E \\ & x_i \in \{0, 1\} & & \text{for } 1 \leqslant i \leqslant n \end{aligned} \tag{3.1}$$

As noted earlier, solving 0/1 linear programs is hard. Therefore we perform *relaxation*: we drop the restriction that the variables can only take integer values and we replace the integrality constraints (3.1) by

$$0 \leqslant x_i \leqslant 1 \qquad \text{for } 1 \leqslant i \leqslant n \tag{3.2}$$

This linear program can be solved in polynomial time. But what good is a solution where the variables can take on any real number in the interval $[0, 1]$? A solution with $x_i = 1/3$, for instance, would suggest that we put 1/3 of the vertex v_i into the cover—something that does not make sense. First we note that the solution to our new relaxed linear program provides us with a lower bound.

Lemma 3.3 *Let* $\text{OPT}_{\text{relaxed}}$ *denote the value of an optimal solution to the relaxed linear program described above, and let* OPT *denote the minimum weight of a vertex cover. Then* $\text{OPT} \geqslant \text{OPT}_{\text{relaxed}}$.

Proof. Any vertex cover corresponds to a feasible solution of the linear program, by setting the variables of the vertices in the cover to 1 and the other variables to 0. Hence, the optimal solution of the linear program is at least as good as this solution. (Stated differently: we already argued that an optimal solution of the 0/1-version of the linear program corresponds

to an optimal solution of the vertex-cover problem. Relaxing the integrality constraints clearly cannot make the solution worse.) □

The next step is to derive a valid vertex cover—or, equivalently, a feasible solution to the 0/1 linear program—from the optimal solution to the relaxed linear program. We want to do this in such a way that the total weight of the solution does not increase by much. This can simply be done by *rounding*: we pick a suitable threshold τ, and then round all variables whose value is at least τ up to 1 and all variables whose value is less than τ down to 0. The rounding should be done in such a way that the constraints are still satisfied. In our algorithm we can take $\tau = 1/2$—see the first paragraph of the proof of Theorem 3.4 below—but in other applications we may need to use a different threshold. We thus obtain the algorithm for Weighted Vertex Cover shown in Algorithm 3.2.

Algorithm 3.2 Approximation algorithm for Weighted Vertex Cover.

ApproxWeightedVertexCover(V, E)

1: Solve the relaxed linear program corresponding to the given problem:

$$\begin{array}{lll} \text{Minimize} & \sum_{i=1}^{n} \mathit{weight}(v_i) \cdot x_i & \\ \text{Subject to} & x_i + x_j \geqslant 1 & \text{for all edges } (v_i, v_j) \in E \\ & 0 \leqslant x_i \leqslant 1 & \text{for } 1 \leqslant i \leqslant n \end{array}$$

2: $C \leftarrow \{v_i \in V : x_i \geqslant 1/2\}$

3: **return** C

Theorem 3.4 *Algorithm* ApproxWeightedVertexCover *is a 2-approximation algorithm.*

Proof. We first argue that the set C returned by the algorithm is a vertex cover. Consider an edge $(v_i, v_j) \in E$. Then $x_i + x_j \geqslant 1$ is one of the constraints of the linear program. Hence, the reported solution to the linear program—note that a solution will be reported, since the program is obviously feasible by setting all variables to 1—has $\max(x_i, x_j) \geqslant 1/2$. It follows that at least one of v_i and v_j will be put into C.

Let $\textsc{opt}_{\text{relaxed}} := \sum_{i=1}^{n} \mathit{weight}(v_i) \cdot x_i$ be the total weight of the optimal solution to the relaxed linear program. By Lemma 3.3 we have $\textsc{opt} \geqslant \textsc{opt}_{\text{relaxed}}$. Using that $x_i \geqslant 1/2$ for all $v_i \in C$, we can now bound the total weight of C as follows:

$$\begin{aligned} \textstyle\sum_{v_i \in C} \mathit{weight}(v_i) &\leqslant \textstyle\sum_{v_i \in C} \mathit{weight}(v_i) \cdot 2x_i \leqslant 2 \sum_{v_i \in C} \mathit{weight}(v_i) \cdot x_i \\ &\leqslant 2 \textstyle\sum_{i=1}^{n} \mathit{weight}(v_i) \cdot x_i = 2\textsc{opt}_{\text{relaxed}} \leqslant 2 \cdot \textsc{opt} \end{aligned}$$

□

The integrality gap. Note that, as always, the approximation ratio of our algorithm is obtained by comparing the obtained solution to a certain lower bound. Here–and this is essentially always the case when LP relaxation is used–the lower bound is the solution to the relaxed LP. The worst-case ratio between the solution to the integer linear program (which models the problem exactly) and its relaxed version is called the *integrality gap*. For approximation algorithms based on rounding the relaxation of an integer linear program,

one cannot prove a better approximation ratio than the integrality gap (assuming that the solution to the relaxed LP is used as the lower bound).