Lecture 24: LP-based algorithm for vertex cover

Rajat Mittal *
IIT Kanpur

Today we will see an application of linear programming. The well-known problem of vertex cover is NP-hard. A very simple algorithm shows that we can get a 2-approximation for this problem. Using linear programming, we will come up with the algorithm which gives this 2-approximation and is optimal assuming Unique Games Conjecture (UGC).

1 Vertex cover

Given an undirected graph $G = (V, E)$, the vertex cover problem asks for the subset $S \subseteq V$, s.t., all the edges have at least one vertex in $S$. We want to minimize the number of elements in $S$. In the weighted version, there is a weight given on every vertex ($w_v$, $\forall v \in V$) and the task is to minimize the weight of $S$, $w(S) = \sum_{v \in S} w_v$. We can without loss of generality assume that $\sum_v w_v = 1$.

Clearly this is an optimization problem. It is known to be NP-hard and so not expected to have a polynomial time algorithm. We are interested in finding an approximation algorithm for this problem. Vertex cover has applications in Biology, monitoring etc. and is a special case of set-cover problem.

![Vertex Cover Example](image)

Fig. 1. Example of a vertex cover with cover shown by solid dots.

2 Integer programming formulation

An integer program is an optimization problem where variables are constrained to be a set of integers. The vertex cover problem can be framed as an integer programming problem.

Given an undirected graph $G = (V, E)$, which has $|V| = n$ vertices and $|E| = m$ edges, introduce variable $y_v$ for every vertex $v \in V$. Say $y_v = 1$ if the vertex is in $S$ and $y_v = 0$ otherwise. Suppose $w_v$ is the weight of the vertex $v$, then the task is to minimize the weight of elements in $S$.

⋆ The content of these notes is influenced from the article "On the optimality of a Class of LP-Based Algorithms" by Kumar et.al.
The constraint is that every edge should have at least one vertex in \( S \). This can be expressed by saying, \( \forall \{u,v\} \in E, \ y_u + y_v \geq 1 \). So the integer program is,

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} w_v y_v \\
\text{s.t.} & \quad y_u + y_v \geq 1 \quad \forall \{u,v\} \in E \\
& \quad y_v \in \{0,1\} \quad \forall \ v \in V 
\end{align*}
\]

(1)

In general, NP hard problems can be formulated as integer programs. Hence, we can’t hope to solve integer programs efficiently (in general). The idea here is to convert this integer program into a linear program and then convert the solution of the obtained LP into an integer solution (\( \{0,1\} \)) again.

**Exercise 1.** Write the maximum independent set problem as an integer program.

### 3 LP relaxation and rounding

How can we convert this integer program into an LP. The *relaxation* of an optimization program is another optimization program which ideally is easier to solve and and every solution of original program is also a solution of the new program with the same (or related) objective value.

In the case of Eqn. 1 we change the domain of \( y_v \)'s to be \([0,1]\) instead of integers \( \{0,1\} \).

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} w_v y_v \\
\text{s.t.} & \quad y_u + y_v \geq 1 \quad \forall \{u,v\} \in E \\
& \quad y_v \in [0,1] \quad \forall \ v \in V 
\end{align*}
\]

(2)

The only change is that the \( y_v \)'s are real numbers between 0 and 1 instead of integers now. It is clear that any solution of Eqn. 1 will still be a solution of Eqn. 2 with same objective value. Hence the objective value of the LP is at least the value of the integer program. The LP is called the relaxation of the integer program.

**Exercise 2.** Show the statements given above and also show that Eqn. 2 is an LP.

For relaxations, there is an important concept of *integrality gap*. For a minimization problem, the integrality gap is the maximum possible ratio between the LP optimum and the integral optimum (optimum of the integer program). Here maximum is taken over all possible instances of the problem. For example, if the problem is vertex cover, every different graph \( G \) will give us a different instance of a linear program (say \( LP(G) \)) and an integer program (say \( IP(G) \)). Then the integrality gap is,

\[
\sup_G \frac{\text{Opt}(IP(G))}{\text{Opt}(LP(G))}
\]

Since the LP can be solved in polynomial time (with some precision), we can obtain the optimal solution of Eqn. 2. The goal now is to convert these fractional solutions into integer (\( \{0,1\} \)) solutions. The process of converting fractional solutions into integers is called *rounding*.

**Exercise 3.** What kind of rounding techniques can be possible?

Suppose the solution of LP gives \( y^* \) as the optimal solution. One simple rounding procedure could be, if \( y^*_v \geq \frac{1}{2} \) then we include the vertex in the vertex cover otherwise skip it. The corresponding integer solution, say \( z \), is

\[
z_v = \begin{cases} 
1 & \text{if } y^*_v \geq \frac{1}{2} \\
0 & \text{if } y^*_v < \frac{1}{2} 
\end{cases}
\]

It is easy to show the following two properties.
• $z$ is a feasible solution.
• The objective value of $z$, call it $Ob(z) = \sum v w v z_v$, is at most twice the objective value of $Ob(y^*)$.

This gives us a 2-approximate algorithm for vertex cover.

$$Ob(z) \leq 2 Ob(y^*) = 2 Opt(LP) \leq 2 Opt(IP)$$

Here $Opt(LP), Opt(IP)$ denote the optimal values of the linear program (Eqn. 2) and the integer program (Eqn. 1).

**Exercise 4.** Show that the last inequality follows because LP is the relaxation of the integer program.

**Exercise 5.** Show that we can never obtain an approximation guarantee better than the integrality gap by the above method.

4 Another LP formulation

Note that the way we defined LP relaxation and rounding are not fixed. It turns out that some LP formulations might be better than others and some rounding algorithms might be better than other rounding techniques. We will take a look at another LP formulation of vertex cover,

$$\min \sum_{v \in V} w v y_v$$

s.t. $(y_u, y_v) \in Conv((1, 0), (0, 1), (1, 1))$ \forall $\{u, v\} \in E$

$y_v \in [0, 1] \forall v \in V$ (3)

**Exercise 6.** Show that this is a relaxation of Eqn. 1.

It turns out that this is the same LP as Eqn. 2 (exercise). So this LP also gives a 2-approximation.

**Exercise 7.** Show that the integrality gap of this LP is 2.

We look at this LP because it can be generalized to a large class of problems known as strict monotone constraint satisfaction problems. The rounding algorithm we are going to give below can be extended to the entire class and can be shown to be optimal assuming UGC. For further details look at the article by Kumar et.al., "On the Optimality of LP-Based Algorithms".

This article is similar to another remarkable result by Prasad Raghavendra, which shows that a certain semidefinite programming based algorithm is optimal for max-cut kind of constraint satisfaction problems assuming UGC.

4.1 Rounding algorithm

The first step of the algorithm is to discretize the solution of the LP, Eqn. 3 Divide $[0, 1]$ in $m$ buckets; $[0, 1/m], [1/m, 2/m]$ and so on. For the solution of Eqn. 3 say $y^*$, assign every $x_v$ to be the right extreme of the bucket in which $y^*_v$ lies. For example, if $y^*_v$ lies between $3/m$ and $4/m$ then $x_v = 4/m$. This step keeps the solution feasible and close to optimal.

**Exercise 8.** Show these two things.

• The new solution is still feasible.
• The objective value of the new solution is at most the previous solution plus $1/m$. 
The rounding algorithm works on this discrete solution. We need to assign these discrete values to either 0 or 1. Suppose elements in the same bucket are assigned the same value, i.e., if one element of the bucket is assigned 1 (0) then all elements of the bucket are assigned 1 (respectively 0). Then we can just go over all possible assignments to buckets (of 0 or 1), every such assignment will give us an integer solution. Output the one which has minimum weight among the feasible assignments. Since the number of buckets are constant $m$, the time taken, $2^m$, is still low.

We know that the integrality gap of the relaxations is $\gamma = 2$. This algorithm gives us a $\gamma$ approximation. Suppose for a graph $G$, $Opt(G)$ denotes the value of the minimum vertex cover and $Al(G)$ denotes the value of the solution given by algorithm.

**Theorem 1.** The solution given by this algorithm (say $Al(G)$) will not be more than $\gamma(Op(G) + 1/m)$.

**Proof.** Consider another graph $G' = (V', E')$, where every bucket has a vertex. There is an edge between two vertices $i, j$ if $G$ has an edge $u, v$ whose values $y_u$ ($y_v$) fall in buckets $i$ ($j$) respectively. The weight of the vertex in $G'$ (a bucket) is the sum of vertices in $G$ which fall in this bucket. The algorithm searches over all possible assignments of graph $G'$ and hence output the best solution for $G'$. So, $Al(G) = Opt(G')$. Using the definition of integrality gap,

$$Al(G) = Opt(G') \leq \gamma LP(G') \leq \gamma (LP(G) + 1/m) \leq \gamma (Opt(G) + 1/m)$$