Rounding Techniques in Approximation Algorithms

Lecture 1: Introduction to the Relax and Round Framework

Lecturer: Nathan Klein

1 Introduction

In this course, we will study approximation algorithms. These are efficient algorithms that give approximate solutions to optimization problems.

Approximation Algorithm

An α-approximation for an optimization problem is a polynomial time algorithm which produces a solution of cost within a factor of α of the optimal solution *for every instance.*

So, for a minimization problem, the solution S returned by the algorithm must have the property $c(S) \leq \alpha \cdot c(OPT)$, where $c(S)$ is the cost of S and OPT is an optimal solution. For a maximization problem, we will have $c(S) \geq \alpha \cdot c(OPT)$.

We care about approximation algorithms for three reasons:

1. We can use approximation algorithms to get solutions of reasonable quality to NP-Hard problems. If we want algorithms that work in the worst case and the size of our input is large, this is the only option unless P=NP. Thus, these are useful algorithms in practice.

2. Working on approximation algorithms helps us understand and categorize NP-Hard problems. A problem with a 1.01 approximation such as knapsack is clearly different from a problem with no $n^{0.99}$ approximation (unless P=NP) such as max clique.

3. While not the focus of this course, approximation algorithms can be used for problems that are not NP-Hard. A 2-approximation running in time $O(n)$ may be preferable to an exact algorithm running in time $O(n^2)$ when a massive amount of data is involved.

Plus, the field is tied to many areas of theoretical computer science and math, some of which we will see in this course.

1.1 Complexity Classes

For each optimization problem, we would like to obtain an approximation algorithm with ratio α and prove that it is NP-Hard to approximate better than α. For many problems we are far from achieving this goal. However, for many problems we know at least whether they are in PTAS, APX, or in neither.

PTAS and APX

1. A PTAS (Polynomial Time Approximation Scheme) is an approximation algorithm with ratio $1+\epsilon$ for any $\epsilon > 0$, i.e. running time polynomial for every fixed $\epsilon > 0$ (for a maximization problem, the ratio would be $1-\epsilon$). For example, a PTAS may have running time $n^{2^{1/\epsilon}}$. A problem with a PTAS is in the complexity class PTAS.
2. A problem is in APX (Approximable) if there is an approximation algorithm with ratio $\alpha \in O(1)$.

A problem is called APX-Hard if there is a PTAS preserving reduction from all problems in APX to that problem: in other words, if a PTAS for the given problem would imply a PTAS for all others. Many problems we work with will be APX-Hard. Such a problem has no PTAS unless P=NP.

You may sometimes see a theorem claiming that a problem is hard to approximate better than some α under the Unique Games Conjecture (UGC). Showing UGC-Hardness is not as strong as showing NP-Hardness but it is a good substitute and identifies a clear barrier for progress, since the UGC is a famous and well-studied open problem.

2 Vertex Cover and Relax-and-Round

In the Vertex Cover problem, we are given a graph $G = (V, E)$ and the goal is to select a minimum cardinality set of vertices $S \subseteq V$ such that every edge is adjacent to at least one vertex in S. We will use this problem to introduce approximation algorithms and the Relax-and-Round framework.

Question: what is a natural algorithm for this problem? The first thing you might try is the *greedy* algorithm: take the vertex of largest degree, put it in the cover and delete it and its edges from the graph. Iterate until the graph has no edges. It turns out there are examples where this is a $\Omega(\log n)$ approximation. You will prove this on the first homework.

2.1 A 2-approximation

Despite this $\Omega(\log n)$ bound for greedy, it turns out there is an easy 2-approximation: pick any uncovered edge and put both vertices in the cover. Iterate until no edges remain.

Fact 2.1. *The algorithm which iteratively picks both endpoints of an uncovered edge and puts them in the cover is a 2-approximation.*

Proof. Consider the uncovered edges $e_1, \ldots, e_k$ picked by the algorithm. They must form a matching, since if any two edges e_i and e_j for $j > i$ shared an endpoint, then e_j would have been covered when the two endpoints of e_i were put in the cover. Furthermore, the solution has size $2k$.

The optimal vertex cover by definition must contain at least one vertex adjacent to every edge. However every vertex is adjacent to at most one edge among $e_1, \ldots, e_k$. Therefore the optimal matching has size at least k. Since the algorithm outputs a matching of size $2k$, this is a 2-approximation. □

2.2 Relax and Round

Now consider the variant of Vertex Cover where the vertices have costs and the goal is to minimize the cost of the cover. The same 2-approximation does not work, and it is not obvious how to fix it. We will use the relax and round framework to make this problem easy for us. The framework has three ingredients.

Step 1: Model the problem as an Integer Linear Program (ILP). For vertex cover, this is the following, where we have a variable x_v for each vertex v indicating whether or not it is in the cover and the cost of vertex v is c_v:

$$\begin{aligned} \min \quad & c(x) \\ \text{s.t.} \quad & x_u + x_v \geq 1 \quad \forall e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \qquad\qquad \forall v \in V \end{aligned} \tag{1}$$

Here $c(x) = \sum_{v \in V} c_v x_v$. This is an ILP because it consists of a linear objective function, linear constraints, and the variables are required to be integers.

This is clearly an equivalent problem to Vertex Cover, so it's still NP-Hard, and it doesn't seem like we've gained anything. That's why we do this next step:

Step 2: Relax the ILP into a Linear Program (LP). This just means we remove the requirement that the variables are integers.

$$\begin{aligned} \min \quad & c(x) \\ \text{s.t.} \quad & x_u + x_v \geq 1 \quad \forall e = \{u, v\} \in E \\ & 0 \leq x_v \leq 1 \qquad\qquad \forall v \in V \end{aligned} \tag{2}$$

The result is an LP. It turns out *all LPs* can be solved in polynomial time by the ellipsoid method so long as the set of constraints has a separation oracle, as was proved by Khachiyan in 1979.

Definition 2.2 (Separation Oracle)**.** *A separation oracle for a linear program is a polynomial time procedure that given a point $x \in \mathbb{R}^n$ either certifies that x satisfies all constraints or finds a constraint that x violates.*

In other words, the set of constraints in our linear program should either (i) have polynomial size or (ii) given a possible solution x, we should be able to determine in polynomial time if all constraints are met and, if not, which constraint is not met. In the case of vertex cover, we have only polynomially many constraints (just $|E| + 2n$) so we can solve the LP in polynomial time.

So, great: now we have some solution x to this LP. We also know that $c(x) \leq c(OPT)$. This is because OPT is a feasible solution to the LP and we found the cheapest LP solution. But, x has fractional coordinates. What do we do to get an actual cover? In comes the final part:

Step 3: Round the fractional solution to an integer one. This is what we will spend most of the course on. For Vertex Cover, if we want a 2-approximation, this turns out to be very easy. Think a moment and see if you can figure it out. Remember you can use that $c(x) \leq c(OPT)$.

The idea is just to let the cover consist of all vertices v with $x_v \geq \frac{1}{2}$. This is called **threshold rounding** and is arguably the simplest form of rounding.

Fact 2.3. *Applying a threshold rounding at $\frac{1}{2}$ is a 2-approximation for vertex cover.*

Proof. Let S be the cover returned by the threshold rounding. Consider any constraint $x_u + x_v \geq 1$ for some $e = (u, v)$. Either x_u or x_v must be at least $\frac{1}{2}$, as otherwise $x_u + x_v < 1$. Therefore, either u or v is in S, demonstrating that S is a feasible vertex cover.

To analyze the cost of S, consider:

$$c(S) = \sum_{v:x_v \geq \frac{1}{2}} c_v \leq \sum_{v \in V} 2x_v c_v = 2c(x) \leq 2c(OPT),$$

as desired. □

We can also study the "strength" of this relaxation by measuring how far a fractional solution can be from an integer one in the worst case.

Integrality Gap

The **integrality gap** of an ILP (and its associated LP) is the worst-case value of $\frac{c(OPT_{ILP})}{c(OPT_{LP})}$ over all instances.

Sometimes you will also see the integrality gap of an LP defined as the worst-case value of $\frac{c(OPT)}{c(OPT_{LP})}$ over all instances (with no reference to an ILP). But this is equivalent so long as the ILP is equivalent to the original problem.

Upper bounding the integrality gap can be accomplished in several ways, but designing an approximation algorithm is clearly one such way:

Fact 2.4. *Given a feasible solution x to an LP, if a rounding algorithm A always produces a feasible solution to the corresponding ILP of cost at most $\alpha \cdot c(x)$, then the integrality gap of the ILP is at most α.*

So, we know that the integrality gap of the polytope given by (1) is at most 2. Lower bounding the integrality gap is usually done using an example.

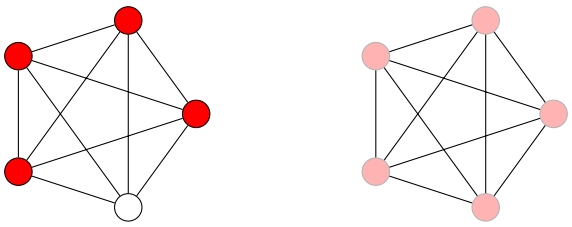

Figure 1: On the left is an optimal vertex cover where the four red nodes have $x_v = 1$ and the white node is set to 0. On the right is an optimal fractional vertex cover where all nodes have $x_v = 1/2$.

It turns out that there is also a lower bound of 2: a complete graph has an optimal integral solution of value $n - 1$, while it has an optimal fractional solution of value $\frac{n}{2}$, as demonstrated in Fig. 1. Therefore, the value of the optimal solution to (2) may be only half as large as value of the integer optimal solution (which would be faithfully returned by (1)).

2.3 Summary for Vertex Cover

So, the situation is quite good for vertex cover. We have an approximation algorithm with ratio 2, and we know the LP we used has an integrality gap of 2, so if we want to use $c(x)$ as a lower bound we cannot do better than 2. It turns out we cannot get an approximation algorithm with ratio better than 2 under the UGC [KR08]. It is NP-Hard to approximate better than $\sqrt{2} \approx 1.414$ [KMS23].

For most problems, we are farther from knowing the truth. Often, we do not even know the integrality gap for the natural ILP.

2.4 A Note on Integrality Gaps

The conventional wisdom is that if you are basing an approximation algorithm off of an LP relaxation of an ILP, it is difficult to show that the approximation ratio is better than the integrality gap. Why is this the case? Well, usually the approach is as above: given an LP solution x, construct an integer solution of cost at most $\alpha \cdot c(x)$. But if the integrality gap is β (for a minimization problem), it is simply impossible to find an integer solution of cost less than $\beta \cdot c(x)$ in all cases, so you're stuck with at best a β approximation.

But the weird thing here is that on an instance achieving the integrality gap, actually your approximation algorithm is doing *really well*, because in fact $c(OPT) \approx \beta \cdot c(x)$. So for these instances, if you find a solution of cost at most $\beta \cdot c(x)$, you are really finding a near optimal solution. Thus, "integrality gap instances" (those achieving near the integrality gap) are strange beasts: they show lower bounds for your *proof strategy* while being *really easy instances* for approximation algorithms.

Sometimes, it is possible to exploit this to give guarantees that are based on the LP but go beyond the integrality gap. In other words, an integrality gap instance doesn't have to be the end of the story, even if it often is.

2.5 Strengthening the LP

Even though the ILP is equivalent to the original problem, we can still add constraints that may strengthen the LP to reduce the integrality gap.

For example, for the vertex cover problem, we can write the constraint that $x_u + x_v + x_w \geq 2$ for all triangles $e = \{u, v\}, f = \{v, w\}, g = \{w, u\}$ in the graph. This is a valid constraint, since at least two vertices in any triangle need to be in the cover. Now the integrality gap of some specific instances goes down. For example, it used to be that a triangle with all edges set to $1/2$ was a feasible LP solution of value $3/2$, whereas OPT was 2. Now on this instance, the LP solves it optimally.

Notice that the ILP *didn't need* this extra constraint, because it was implied by the integrality of the variables! We can derive it in the ILP as follows:

$$x_u + x_v \geq 1, x_v + x_w \geq 1, x_u + x_w \geq 1$$

$$2x_u + 2x_v + 2x_w \geq 3$$

$$x_u + x_v + x_w \geq \frac{3}{2}$$

But this means that $x_u + x_v + x_w \geq 2$ since all variables are integers, so the RHS must be an integer.

In some cases, adding constraints like this decreases the integrality gap of the LP. In this specific case, it unfortunately does not. In fact, it was recently shown that no polynomial sized LP has an integrality gap below 2 for Vertex Cover [Baz+19] (however, note that this doesn't rule out exponential sized LPs with separation oracles).

References

[Baz+19] Abbas Bazzi, Samuel Fiorini, Sebastian Pokutta, and Ola Svensson. "No Small Linear Program Approximates Vertex Cover Within a Factor $2 - \epsilon$". In: *Math. Oper. Res.* 44.1 (2019), 147–172. ISSN: 0364-765X. DOI: 10.1287/moor.2017.0918 (cit. on p. 6).

[KMS23] Subhash Khot, Dor Minzer, and Muli Safra. "Pseudorandom sets in Grassmann graph have near-perfect expansion". In: *Annals of Mathematics* 198.1 (2023), pp. 1 –92. DOI: 10.4007/annals.2023.198.1.1 (cit. on p. 5).

[KR08] Subhash Khot and Oded Regev. "Vertex cover might be hard to approximate to within $2 - \epsilon$". In: *J. Comput. Syst. Sci.* 74.3 (2008), 335–349. ISSN: 0022-0000. DOI: 10.1016/j.jcss.2007.06.019 (cit. on p. 5).