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 introduce 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:

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

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.

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

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.

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. There are many amazing ways to do this. 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. (For a minimization problem, this is therefore the largest value, and for a maximization problem, the smallest value.)

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

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.

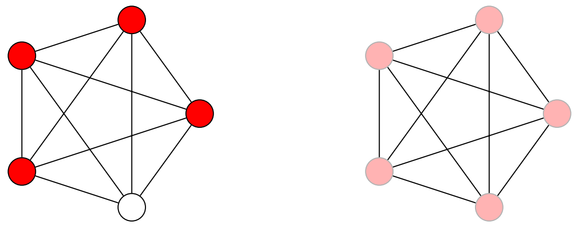

Figure 1: On the left is an optimal vertex cover where the four red nodes have $x_v = 1$ and the white node is set to 0. On the right is an optimal fractional vertex cover where all nodes have $x_v = 1/2$.

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. It is NP-Hard to approximate better than $\sqrt{2} \approx 1.414$.

For most problems, we will be farther from knowing the truth. Often, we will not even be able to exactly determine the integrality gap.

3 Summarizing

We will now step back a bit and reformulate the relax and round approach.

Definition 3.1 (Convex Polyhedron/Polytope)**.** *A convex polyhedron is the intersection of finitely many half-spaces of the form $a^T x \geq b$ for $a \in \mathbb{R}^n, b \in R$. A convex polytope is a bounded convex polyhedron. See Fig. 2.*

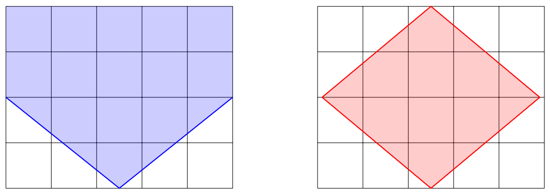

Figure 2: On the left is a convex polyhedron defined by two half-planes, and on the right is a convex polytope defined by four half-planes.

Here is an example of the relax-and-round framework geometrically. Suppose the marked points in $\mathbb{Z}^2$ are the feasible solutions to our optimization problem O. Then the blue polytope P is a valid ILP for the problem since $P \cap \mathbb{Z}^2$ is the set of feasible solutions for O.

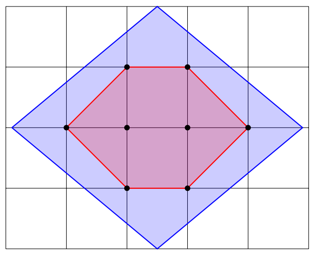

Figure 3: In blue is the feasible region of four linear inequalities over $\mathbb{R}^2$, and in red the convex hull of the same feasible region over $\mathbb{Z}^2$. In general we cannot optimize over the red polytope due to its complexity. So instead we optimize over the blue polytope, which is simpler and has the same integer points as the red set.

What we really would like to find is a point in the convex hull of the set of feasible integer solutions minimizing the objective function. Call this convex hull P' (the red polytope). Suppose we could find such a point $x \in P'$ in polynomial time minimizing $c(x)$ over all $x \in P'$. Then, by the following theorem we would have solved the optimization problem O optimally.

Theorem 3.2 (Carathéodory's Theorem)**.** *Let x lie in the convex hull of a set $P \subseteq \mathbb{R}^n$. Then x can be written as the convex combination of at most $n+1$ points of P.*

Using this, given a point $x \in P'$, we can write x as a convex combination of feasible integer solutions. But then at least one of the integer points costs no more than x, because x is their average. Thus:

Fact 3.3. *Unless $P = NP$, we cannot optimize over the convex hull of the feasible integer points in polynomial time for NP-Hard optimization problems.*

The intuition here is that for these NP-Hard problems, P' can be extremely complicated. And yet, for almost every combinatorial optimization problem we care about, a much “simpler" polytope P exists that we *can* optimize over in polynomial time for which $P \cap \mathbb{Z}^n = P'$. This is the key to the relax and round framework, and gives us the following:

Fact 3.4. *Suppose $P \cap Z^n = P'$. Then, if x is a point in P minimizing $c(x)$, and OPT is a point in P' minimizing $c(OPT)$,*

$$c(x) \leq c(OPT)$$

In other words, $c(x)$ is a lower bound on the cost of the optimal solution.

Of course, we will usually have $P \neq P'$, as in Fig. 3. This means that (i) our point is not in $\mathbb{Z}^n$ but instead has many fractional coordinates, and (ii) we may have $c(x) < c(OPT)$. This is the price we pay for simplifying the polytope. This leads to a question we will be very interested in throughout: *how well does P approximate P'?* Formally, we study the integrality gap.

Finding an ILP with the property that all of its integer points are feasible solutions is usually straightforward. In other words, it is generally not hard to construct P such that $P \cap \mathbb{Z}^n = P'$. However, note that some care should be taken to find the right ILP. There are many ILPs with this property, and some may be much better than others. We will explore this more later in the course.