Lecture 2: Review, Linear Programming Relaxations

Today we will talk about expressing combinatorial problems as *mathematical programs*, specifically Integer Linear Programs (ILPs). We then see what happens if we *relax* the integrality condition, obtaining linear programs (LPs). Then, we introduce the paradigm of designing approximation algorithms by *rounding* LP relaxations, i.e., obtaining a "good" solution to the ILP from a solution to the LP.

Disclaimer: These lecture notes are informal in nature and are not thoroughly proofread. In case you find a serious error, please send email to the instructor pointing it out.

Integer Linear Programs

Consider the following optimization problem in n variables $x_1, x_2, \ldots, x_n$:

$$\begin{aligned} \text{maximize } & f(x_1, x_2, \ldots, x_n) \text{ subject to} \\ & g_1(x_1, x_2, \ldots, x_n) \leq 0, \\ & g_2(x_1, x_2, \ldots, x_n) \leq 0, \\ & \vdots \\ & g_m(x_1, x_2, \ldots, x_n) \leq 0, \\ & x_i \in \{0, 1\} \text{ for all } i. \end{aligned}$$

We call such an optimization problem an *Integer Program* (IP).[1] Integer programs are highly expressive, even for a restricted class of functions f, g_i. One particular sub-class of IPs we study are the so-called Integer *Linear* Programs (ILP), in which the functions f and g_i are all linear.

In this case each constraint $g_i(x_1, \ldots, x_n) \leq 0$ can be written as $A_i^T x \leq b_i$, for some $A_i \in \mathbb{R}^n$. If we denote by A the $m \times n$ matrix whose ith row is A_i and by b the vector whose ith entry is b_i, we can write the constraints in the matrix form

$$Ax \leq b.$$

(Throughout the course, inequality for vectors simply means entry-wise inequality).

Expressibility. Despite the linearity restriction, many interesting problems are easily expressible as ILPs. We will see below how to write the Set Cover and Max-Cut problems (defined in the last class) as ILPs. An implication of this is that solving ILPs is NP-hard in general.

If it is NP-hard, why do we care about formulating problems as ILPs? There are two answers. The first is that there are many heuristics for solving ILPs (there is an extensive literature, comparable to that on SAT solvers), and readily available software that can solve ILPs with 1000s of variables in a few seconds. Thus in many practical applications, it is useful to formulate a problem as an ILP and try using a solver. The second reason, from the theory side, is that some formulations can lead to approximation algorithms. This is the focus of today's lecture.

[1] Some authors prefer to replace the constraints $x_i \in \{0, 1\}$ with somewhat more general ones.

Example: Set Cover

Recall the set cover problem, we have topics $T_1, \ldots, T_n$, and people $P_1, \ldots, P_m$, and each person is an expert on a subset of the topics. The goal is to pick the smallest number of people, among whom there is an expert on *every* T_i.

This can easily be phrased as an ILP. Suppose we have 0/1 variables $x_1, \ldots, x_m$, where x_i indicates if person P_i is picked. Then for any j, the constraint that at least one expert on T_j is picked, can be written as

$$\sum_{i:P_i \text{ expert on } T_j} x_i \geq 1.$$

The solution must satisfy this constraint for all $j \in [n]$.[2] Subject to these constraints, we wish to minimize $\sum_{i=1}^{m} x_i$. This is the objective function for the ILP.

Formulation as an ILP need not always be so straight-forward, as we illustrate now.

Example: Max-Cut

Recall that in the Max-Cut problem, we have a graph $G = (V, E)$, and the goal is to partition V into V_1 and V_2 so as to maximize the number of edges in the *cut*, $E(V_1, V_2)$.

We need to partition V into two sets, thus a natural choice of variables is to have one for each vertex, $x_u = 0$ if $u \in V_1$, and 1 otherwise. All partitions are allowed, so we do not have any constraints other that $x_u \in \{0, 1\}$ for all u.

How do we now write the objective value? For an edge uv, we need some way of figuring out if x_u and x_v are *unequal*. This is easy to do if we allow quadratic functions – $(x_u - x_v)^2$ captures precisely this. However it is not possible to write it as a linear function of x_u and x_v. Can we have other choices of variables using which we can express the objective as a linear function?

There are a couple of ways. The first is to introduce new variables, y_{uv}, one for each *pair* of vertices. The goal is to add linear constraints on y_{uv} such that for any x_u, x_v that are 0/1, y_{uv} is forced to take the value $x_u \cdot x_v$. (Again, we are only allowed to use *linear* constraints.) In the homework, we see that the following conditions suffice:

$$y_{uv} \in \{0, 1\}; \qquad y_{uv} \leq x_u; \qquad y_{uv} \leq x_v; \qquad 1 - x_u - x_v + y_{uv} \geq 0.$$

In fact, these type of constraints are a simple example of the so-called *lift and project* methods, which we will see later in the course.

Now, once have such variables y_{uv}, we can write the objective function as $x_u + x_v - 2y_{uv}$ (this is precisely the same as $x_u^2 + x_v^2 - 2x_u x_v$, since $x_u^2 = x_u$ for 0/1 variables).

[2] Note the standard notation $[n] = \{1, 2, \ldots, n\}$.

Example: Max-Cut continued on next page...

An aside. (not absolutely necessary to follow this) For the Max-Cut problem, it turns out there is another way to write an ILP relaxation. We can have variables y_{uv} that are supposed be 1 if the vertices u, v are on *different sides* of the cut, and 0 if they are on the same side. With this choice of variables, the objective is easy, we maximize the sum over edges uv of y_{uv}. But what constraints do we place on the y_{uv}? It turns out that the so-called *triangle* constraints, along with *metric* constraints suffice:

$$\text{for all } u, v, w, \qquad y_{uv} + y_{vw} + y_{uw} \leq 2,$$
$$\text{for all } u, v, w, \qquad y_{uv} + y_{vw} \geq y_{uw}.$$

For a solution of the *intended* form (as described above), these condition hold. The first condition holds because if we pick any three vertices, at least two of them are on the same side, so at least one of the three terms on the LHS is 0. The second constraint can also easily be verified. The tricky thing is to show that any 0/1 variables y_{uv} that satisfy the constraints are actually of the intended kind (for some partition).

Linear Programs

Finally, we turn to a more tractable set of problems, called *linear programs*, or LPs. These are optimization problems in which we have have linear constraints and objective (as in ILPs), but we do not have any binary constraints (i.e. the x_i are allowed to be arbitrary real numbers, as opposed to $x_i \in \{0, 1\}$).

To be more precise, a linear program is the following problem:

$$\text{maximize } c^T x \quad \text{subject to} Ax \leq b,$$

where as before, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and A is an $m \times n$ matrix. Once the binary restriction on the x_i's is removed, the problem has a very geometric interpretation.

Geometric interpretation. Let us think about the set of *feasible* points, i.e., the points in $x \in \mathbb{R}^n$ that satisfy the constraints $Ax \leq b$. For any constraint $A_i x \leq b_i$, the set of feasible points x is a *half-space* (as a simple example, think of the $n = 2$ case, and the constraint $x_1 + 2x_2 \leq 1$). Thus if we want points that satisfy all the constraints, we are interested in the *intersection* of the m half-spaces given by $A_i x \leq b_i$, for $i \in [m]$. Such an intersection is called a *polytope*.

For details, we refer to Section 7.1 in: https://www.cs.berkeley.edu/∼vazirani/algorithms/chap7.pdf

The chapter also contains a brief history of LPs, and a short description of the simplex algorithm. The main fact we will use about LPs is that *they can be solved efficiently* (i.e., in polynomial time). The first efficient algorithm was given by Khachiyan [1979] — considered one of the big achievements of modern algorithms research.

Linear programs as a proxy for ILP

Let us now get back to our main focus of designing approximation algorithms. For the remainder of the lecture, let us focus on the Set Cover problem.

Recall the ILP relaxation for Set Cover:

Linear programs as a proxy for ILP continued on next page...

$$\text{minimize} \quad \sum_i x_i \quad \text{subject to}$$
$$\text{for all } j \in [n]: \quad \sum_{i \text{ expert on } T_j} x_i \geq 1, \text{ and}$$
$$x_i \in \{0,1\}.$$

What is a natural *LP* that one could use as a proxy for this ILP? The simplest idea is to replace the constraints $x_i \in \{0,1\}$ with $0 \leq x_i$ and $x_i \leq 1$ for all i! Such a transformation is generally referred to as *relaxation* (we are relaxing the integer constraints).

Suppose we denote the *feasible set* of the ILP by $\mathcal{F}_{\text{ILP}}$ (i.e., it is the set of x that satisfy all the constraints of the ILP). Similarly, denote the feasible set of the relaxation by $\mathcal{F}_{\text{relax}}$. Since the only difference between the ILP and the relaxation are the constraints $x_i \in \{0,1\}$ (which are replaced by $0 \leq x_i \leq 1$), we have $\mathcal{F}_{\text{ILP}} \subseteq \mathcal{F}_{\text{relax}}$.

Thus the minimum value of the objective over $\mathcal{F}_{\text{ILP}}$ (called $OPT(\mathcal{F}_{\text{ILP}})$) is greater than or equal to the minimum value of the objective over $\mathcal{F}_{\text{relax}}$ (called $OPT(\mathcal{F}_{\text{relax}})$).

The **integrality gap** of an ILP formulation of a problem (in our case, Set Cover) is defined to be the maximum over all inputs[a] of the ratio:

$$\frac{OPT(\mathcal{F}_{\text{ILP}})}{OPT(\mathcal{F}_{\text{relax}})}.$$

[a] As was the case for the approximation ratio, sometimes the integrality gap is a function of the input size, in which case the maximum is over inputs of that size.

Let us see a simple input for Set Cover in which $OPT(\mathcal{F}_{\text{ILP}})$ is strictly larger than $OPT(\mathcal{F}_{\text{relax}})$. Consider the following:

- We have 3 topics: A, B, C, and three people.
- The first person is an expert on A, B, the second on B, C, and the third on C, A.

Now, in the ILP formulation, we have three variables x_1, x_2, x_3 for the three people. The constraints are now:

$$x_1 + x_3 \geq 1; \qquad x_2 + x_3 \geq 1; \qquad x_1 + x_2 \geq 1; \qquad x_i \in \{0,1\}.$$

It is easy to see that to satisfy the constraints, we must set at least two of the x_i to 1. Thus the min value of $x_1 + x_2 + x_3$ is 2.

Now let us see what happens to the relaxation. Here, the constraint $x_i \in \{0,1\}$ is replaced with $0 \leq x_i \leq 1$. Let us set $x_i = 1/2$ for all i. Now, it is easy to see that all the constraints are satisfied! And for this solution, we have $x_1 + x_2 + x_3 = 3/2$, thus the objective value is only 1.5.

Thus the integrality gap, just considering this instance, is at least $2/1.5 = 4/3$. As we increase the number of topics n, we can construct examples in which the gap is larger – in fact the integrality gap can be as bad as $\Omega(\log n)$.

"Rounding" fractional solutions

In summary, the relaxation of an ILP always gives an *efficiently solvable* optimization problem. The catch is that the solution we get could be fractional, so it might not help us solve the ILP we started with.

However, in some cases, it is possible to *round* the fractional solution to the relaxation, and get a feasible solution to the ILP without much change in the objective value. This is a general, and a powerful paradigm in approximation algorithms.

LP relaxations for approximation algorithms

- Start with an ILP formulation for the problem
- Relax the integrality constraints, obtaining a linear program
- Solve it to obtain a (possibly fractional) solution x (this is polynomial time)
- "Round" the solution x into a solution for the ILP. Two things need to be ensured:
  1. the resulting solution satisfies all the constraints of the ILP
  2. the objective value does not change too much (the factor it changes by determines the approximation factor)

The last step, i.e., how we round the fractional solution x, is where all the ingenuity lies. We will now see a very simple procedure, which works for the set cover problem.

Randomized rounding

Recall the ILP for Set Cover (start of Page 4 of the notes). In the relaxation, we replace the constraint $x_i \in \{0,1\}$ with $0 \le x_i \le 1$ for all i. Now suppose we start with a solution x for the relaxation. The goal is to come up with a solution to the ILP, such that the two conditions above are satisfied (solution satisfies all constraints and objective value does not increase too much).

We consider the following rounding procedure. For some $\alpha \ge 1$ (to be picked shortly), we set:

$$Y_i = \begin{cases} 1 & \text{with probability } \min\{\alpha x_i, 1\} \\ 0 & \text{otherwise} \end{cases}.$$

Note that this is a randomized procedure that comes up with a 0/1 valued vector Y. Note also that each i is *rounded* independently of the other i. Y is now our candidate solution to the ILP. Let us evaluate the probability of it satisfying all the constraints, and the cost of the solution.

Constraints. We want to show that *none* of the constraints are violated with high probabililty. As in the coupon collector analysis, we can start by showing that the probability that any single constraint is violated is $\ll 1/n$, and then take a union bound.

Consider the constraint corresponding to some topic T_j:

$$\sum_{i \text{ expert on } T_j} Y_i \ge 1. \tag{1}$$

For convenience, let us write $i \sim T_j$ to denote "i expert on T_j". Now, by assumption, the fractional solution x satisfies the above constraint, i.e., $\sum_{i \sim T_j} x_i \geq 1$. What is the probability that (1) is not satisfied?

It is precisely the probability that *none* of the Y_i is 1. Because the rounding is done independently for different i, this probability is:

$$\prod_{i \sim T_j} (1 - \max\{\alpha x_i, 1\}).$$

If one of the αx_i terms is ≥ 1, the probability is zero, so the constraint is certainly satisfied. Thus we may assume that $0 \leq \alpha x_i < 1$. In this case, we can use the inequality $1 - z \leq e^{-z}$ as in the last lecture, to conclude that the probability above is at most

$$\prod_{i \sim T_j} e^{-\alpha x_i} = e^{-\alpha \sum_{i \sim T_j} x_i} \leq e^{-\alpha}.$$

In the last step, we used the fact that $\sum_{i \sim T_j} x_i \geq 1$. We would like to make this quantity $1/n^2$. Thus it suffices to pick $\alpha = 2 \log n$. Now we can use the union bound, to conclude that the probability that *none* of the constraints are violated is at most n times the above, which is still $< 1/n$.

Objective value. Now we need to see what the rounding above does to the objective value, namely $\sum_{i=1}^{m} Y_i$. In expectation, we have (since for each i, $\mathbb{E}[Y_i] = \alpha x_i$, and expectation is linear)

$$\mathbb{E}[\sum_i Y_i] = \sum_i \alpha x_i = \alpha \cdot \sum_i x_i.$$

The last term is the objective value of the fractional solution, which as we have seen (since we have a minimization problem), is smaller than or equal to the optimum solution to the ILP. Thus if we denote by OPT the optimum objective value for the ILP, we have

$$\mathbb{E}[\sum_i Y_i] \leq \alpha \cdot \text{OPT}.$$

So *in expectation*, the objective value of the solution we produced is at most an α factor larger than the optimum objective value. We would like to say that our algorithm does well with high probability, not simply in expectation. Here is where we use one of the standard probability inequalities.

Theorem 1 (Markov's Inequality)**.** *Let X be a random variable that always takes non-negative values. Then for any $t > 0$, we have*

$$\Pr[X > t \cdot \mathbb{E}[X]] \leq \frac{1}{t}.$$

The proof is very simple, and is left as an exercise.

Using this inequality with $t = 4$, we obtain that

$$\Pr[\sum_i Y_i > 4\alpha \cdot \text{OPT}] \leq 1/4.$$

Overall success probability. We have shown that the probability that none of the constraints is violated is at least $1 - \frac{1}{n}$, and that the probability that $\sum_i Y_i \leq 4\alpha \cdot \text{OPT}$ is at least $3/4$. Thus, the probability that *both* these events occur is at least $1 - \left(\frac{1}{4} + \frac{1}{n}\right) > 2/3$, for $n > 12$. Thus the rounding procedure gives a 4α

approximate solution, with probability $> 2/3$.

This gives an alternate way to obtain an $O(\log n)$ approximation algorithm for the Set Cover problem.