4810-1183: Approximation and Online Algorithms with Applications
Lecture Note 1: Course Overview, Optimization Models, Linear Programming

What do we aim at this class?

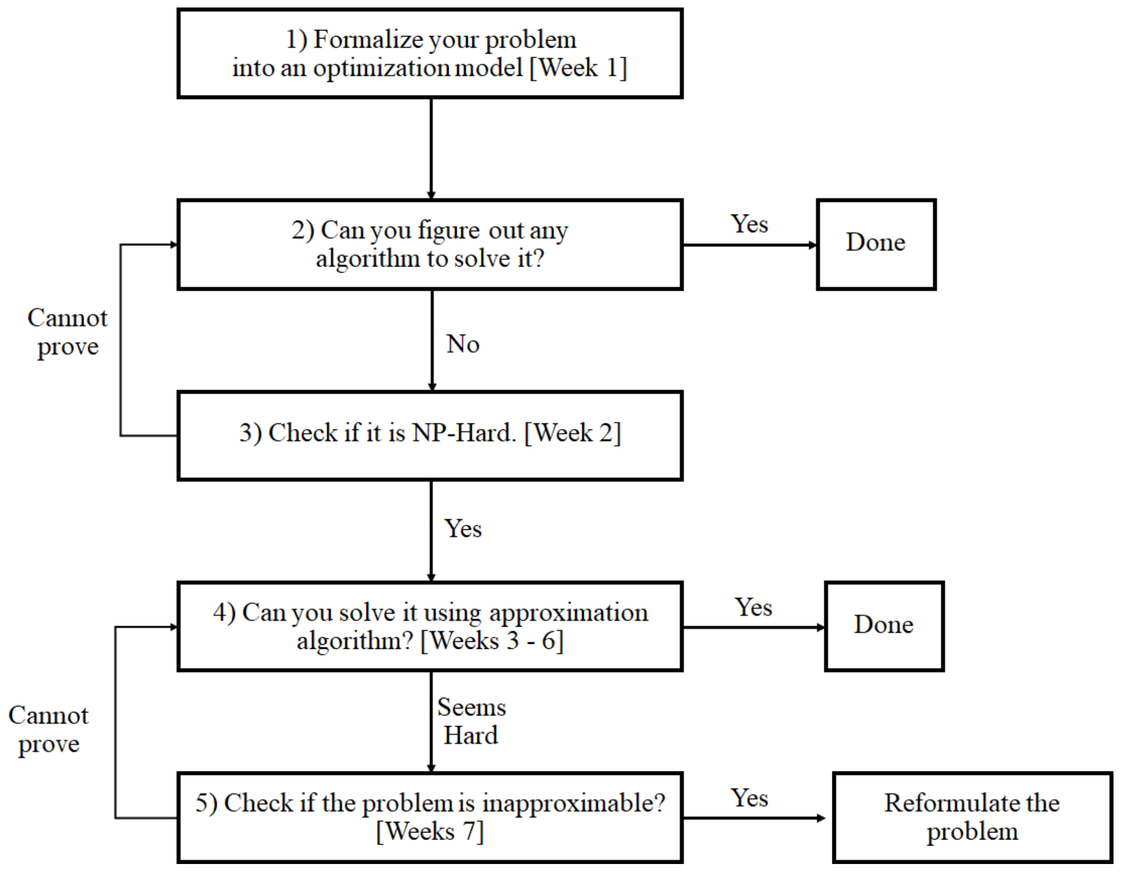
During many undergrad courses in CS, students have a chance to join several “application-oriented courses” such as communication networks, databases, OS, or machine learning. On the other hand, students have a chance to learn several theoretical CS concepts such as computational complexity, hardness, and data structures. However, for some of the students, the “application-oriented courses” and the theoretical CS concepts might look totally separated as they are in totally different world.

At the graduate level, we expect students to produce cutting-edge research results by merging everything they learn. However, I have heard that the merging is not very easy as things are taught separately.

Because of that, in this class, I will guide you to the beginning. I will raise a lot of problems in the application-oriented courses, and will show you how to solve it using theoretical CS. As what some have learnt during the undergrad course might not be enough for the application. I will try to give some more theoretical background, in particular on the hardness, approximation, and online algorithms. For those who have already learnt the concept, that is going to be your reminder.

How to solve a problem in IST?

There are many patterns to problems in information science and technology. The pattern in the below picture is one of the most common patterns.


1) Formalize your problem into an optimization model [Week 1]
2) Can you figure out any algorithm to solve it?
Yes
Done
No
Cannot prove
3) Check if it is NP-Hard. [Week 2]
Yes
4) Can you solve it using approximation algorithm? [Weeks 3 - 6]
Yes
Done
Seems Hard
Cannot prove
5) Check if the problem is inapproximable? [Weeks 7]
Yes
Reformulate the problem

In my opinion, the most important and the hardest step of the whole flowchart is the first step. You have to transform what you have into some mathematical formulas with the clear goal here. That is not easy to do, and more arts than science. During this course, I will try to have you learn this skill by a lot of examples.

Step 2 is when you try to solve your mathematical model using several algorithmic toolbox you have learnt during undergraduate courses, such as divide and conquer, dynamic program, or linear programming. Because we believe that most of IST problems "cannot be solved", we would like to skip this step in this course.

When you want to claim that a problem cannot be solved, there is a way to state that formally, called "NP-Hard". To make everyone believe that a problem cannot be solved, people usually try to prove that the problem is NP-hard. That is Step 3, which we will discuss on next week.

There are several ways to cope with the cannot-be-solved problem. The most conventional way is to use approximation algorithms (Step 4), which is the main focus of this course.

Approximation algorithms can solve almost all problems, but there is a case that we cannot find any approximation algorithms for our problems. It is possible to claim that it is the case formally. That would be done in Step 5. If approximation algorithms are not possible, we might try to get back to Step 1, and reformulate the problem there.

What is online algorithm?

Another main topic of this class is online algorithm, where you have to produce some "output" while you have not got a whole input. The most well-known problem is called "secretary problem". We want to have just one secretary, and we have a number of applications for the job. We do an interview at all applicants, and, just after the interview, we have to tell the applicants immediately if they are hired. If we hire one secretary, we cannot fire him. So, we cannot interview more applicants, although there might be some with a better capability.

Those kind of questions happen a lot in IST, and we will try to address those problems in this course.

Optimization Models

Now, it is the time to work on the first step in our flow chart. We have a problem in IST and we want to solve the problem using some theoretical CS concepts. To kickstart a whole process, we have to answer the following 4 questions:

1) What do we have in real world? What is important for our problem? [input]
2) What do you want to have? [output]
3) What the output can be, what it cannot be? [constraint]
4) What is the most desirable output? [objective function]

After we have some answers for the 4 questions, we have to transform them into some mathematical formulations. Then, we can devise theoretical tools for those mathematical formulations.

Let us try to illustrate you how it works by a toy example (which is unfortunately not an IST problem). Consider the following problems:

- We can eat only Gyu-don and Melon-pan
- Gyu-don and Melon-pan contains different amount of carbohydrates and proteins

This is what you have in real world. It is your task to find the "best" way to consume Gyu-don and Melon-pan. We can formulize by more than one ways, but let try to minimize our cost of living. To do that, we might formulize:

Input: We may want to look in Google to find how much carbohydrates and proteins that Gyu-don and Melon-pan give. Also, we may look there how much carbohydrates and proteins we should take every days. We can go to some places to check the price of Gyu-don and Melon-pan.

Output: We may want to find how much Gyu-don and Melon-pan should we take in each day.

Constraint: We may want to have enough proteins and carbohydrates in every days.

Objective Function: We want to find amount that minimize the cost we have to pay in each day.

After we have some rough ideas on our optimization models, we will now formulate it into some mathematical formulations. One way to do is as follows:

Input:	Carbohydrate that a Gyu-don gives $a_{c,g} \in \mathbb{R}_{\geq 0}$
	Carbohydrate that a Melon-pan gives $a_{c,m} \in \mathbb{R}_{\geq 0}$
	Protein that a Gyu-don gives $a_{p,g} \in \mathbb{R}_{\geq 0}$
	Protein that a Melon-pan gives $a_{p,m} \in \mathbb{R}_{\geq 0}$
	Amount of carbohydrates that we should have per day $b_c \in \mathbb{R}_{\geq 0}$
	Amount of proteins that we should have per day $b_p \in \mathbb{R}_{\geq 0}$
	Price of a Gyu-don $c_g \in \mathbb{R}_{\geq 0}$
	Price of a Melon-pan $c_m \in \mathbb{R}_{\geq 0}$
Output:	Amount of Gyu-don we take in one day $x_g \in \mathbb{R}_{\geq 0}$
	Amount of Melon-pan we take in one day $x_m \in \mathbb{R}_{\geq 0}$
Constraint:	$a_{c,g}x_g + a_{c,m}x_m \geq b_c$
	$a_{p,g}x_g + a_{p,m}x_p \geq b_p$
Objective Function:	Minimize $c_g x_g + c_m x_m$

We can remove the detail from the real world without losing any information.

Input:	$a_{c,g}, a_{c,m}, a_{p,g}, a_{p,m}, b_c, b_p, c_g, c_m \in \mathbb{R}_{\geq 0}$
Output:	$x_g, x_m \in \mathbb{R}_{\geq 0}$
Constraint:	$a_{c,g}x_g + a_{c,m}x_m \geq b_c$
	$a_{p,g}x_g + a_{p,m}x_p \geq b_p$
Objective Function:	Minimize $c_g x_g + c_m x_m$

The above is what we will call as "optimization model" in this course. When we can arrive to something in this form, we are ready to take off. We can apply several algorithmic and theoretical techniques to solve the problem. As most of you can guess, we can use a technique called as "linear programming" solve the problem.

Linear Programming (LP)

LP is a very powerful algorithmic toolbox. In fact, it is shown that "LP can solve all problems that can be solved [1]". The formulation is in the following form:

Input: Matrix $\mathbf{A}$, vectors $\mathbf{b}, \mathbf{c}$

Output: Vector $\mathbf{x}$

Constraint: $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$

Objective Function: Minimize $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x}$

It is known that we can use a method called as "the interior-point method" to solve LP [2]. However, a heuristic algorithm, which might not be able to solve LP, called as "simplex". There are a lot of libraries developed to solve LPs. In my experiments, CPLEX®, developed by IBM, is one of the fastest.

Let get back to our previous example. We can reformulate the Gyu-don-Melon-pan problem to the following:

Input: $\mathbf{A} = \begin{bmatrix} a_{c,g} & a_{c,m} \\ a_{p,g} & a_{p,m} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_c \\ b_p \end{bmatrix}, \mathbf{c} = \begin{bmatrix} c_g \\ c_m \end{bmatrix}$

Output: $\mathbf{x} = \begin{bmatrix} x_g \\ x_m \end{bmatrix}$

Constraint: $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$

Objective Function: Minimize $\mathbf{c}^{T} \cdot \mathbf{x}$

After the reformulation, it is clear that we can use LP algorithm to solve the problem. This course will give you examples how problems are reformulated. We hope that we will get the skill by taking this course.

What do I mean for "solve"?

Suppose that the size of our input file is n bits. In this course, we will say that "we can solve an optimization model" if we can find an "algorithm of which the running time is no more than a polynomial function of n". We will call such kind of algorithms as "polynomial-time algorithm" during this course.

The polynomial-time algorithms do not always "solve" a problem in practice. Consider the case when the running time is somewhere around n^{1000}, which is still a polynomial function of n. The running time we need for 10-bit input is 10^{1000}, which is more than the number of atoms in the world. However, in our experience, when one can find algorithm with the n^{1000} running time, we usually can reduce the running time to n^5 or n^6.

In a last few year, when we are in big data era, the input size n becomes very large. The number is sometimes even larger than 10^{12}. When we have such a large input, we cannot even scan everything we have there. The algorithm with running time = n is not acceptable anymore, and we have to find an algorithm with even smaller running time.

Because of that, in big data era, we have to find an alternative definition for "solve". Those theories are still young and we still have a lot to explore. We will sometimes mention them during this course.

I also want to mention that big data is an opportunity for online algorithms. Because we cannot scan a whole input, we have to output the best things we "learn" from the partial information. Online algorithms can do that, so it is fit to the setting.

References

[1] S. Miyano, S. Shiraishi, and T. Shoudai, “*A List of P-Complete Problems*”, Kyushu University, RIFIS-TR-CS-17, 1990.

[2] N. Karmarkar, "*A new polynomial-time algorithm for linear programming*", Combinatorica, Vol. 4, No. 4. Pages 373-395. 1984.