CS4234: Optimization Algorithms

MiniProject Ideas

Mini-Project 2: Airport Scheduling

One type of combinatorial optimization problem that is immensely important in practice—but that we have not discussed in this class—is scheduling: given a set of tasks, some constraints on how the tasks can be executed, and some resources for executing the tasks, find an efficient plan for executing all the tasks.

In this mini-project, you will choose a specific scheduling problem and focus on the following questions:

- How do you device a good schedule for your problem? How well does it work on real data?
- How fast you can you find a good schedule?
- What happens when the schedule changes? How easily and how quickly can you find a new schedule?
- If you relax the original schedule (e.g., leaving some slack) can you handle changes to the schedule better? Think about the relationship between optimization (i.e., finding an optimal schedule) and fragility (i.e., how it breaks when there are disruptions, e.g., delays).

For example, consider the problem of scheduling gates at a busy airport. Changi airport has three terminals with 92 gates, servicing approximately 1,000 flights per day. When a plane lands or takes off, it must be assigned to a gate for passengers to board or debark. Every day, each flight needs to be assigned to a gate. At the same time, we want to mimimize the number of gates needed during the day.

For example, as you may have noticed, planes are often late. And when a plane is late, gates need to be reassigned. Worse, if there are not enough gates to handle the delays, it may lead to further delays as planes wait for gates to be free! Can you devise a schedule that tolerates delays?

You can find flight data for US flights from: `http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236`. You will want to choose several airport that have a large number of incoming/outgoing flights. Design your algorithms to be as scalable as possible; ideally use the largest data set that you can.

In this context, you might look at the following issues:

- Using real data, determine how best to assign flights to gates. (What metric are you optimizing?) For an airplane taking off from an airport, it needs a gate for 90 minutes before the scheduled takeoff time and for 30 minutes after the scheduled take-off time. For an airport landing at an airport, you need the gate from the scheduled landing time until 45 minutes afterwards. Give results from your experiments.
- Can you prove that your solution is optimal?
- How does your solution scale? Is there any relationship between the number of flights and the number of gates needed? (For example, according to the data from different airports, if I want to build an airport to handle twice as many flights, do I typically need twice as many gates?)

- What happens if there are delays? When a take-off is delayed, you need the gate longer, from 90 minutes before the scheduled time (or the real time?) until 10 minutes after the real take-off. When a landing is delayed, you need the gate until 45 minutes after the actual landing time. If you assign the airplanes from the real data using their scheduled times, but then add random delays to the flights, how many gate collisions occur due to delays? What if you use real data on flight delays in the data set? (Gate collisions happen when the lateness of one flight causes a gate usage to overlap another.)

- Assume that when there is a gate collision, a plane is reassigned to a new (empty) gate. If there are no available gates, then we will have to use an *overflow* gate, i.e., an extra gate that increases the total number of gates. That is, we may now need more gates in order to cope with the reassignments. Can you develop an algorithm for reassigning planes when there is a gate collision (i.e., rescheduling the planes and assigning them to new gates). For the delays in the dataset, how many gates do we need? How does the number of extra gates needed scale with the delays? (If different days have different amounts of delay, you may be able to observe this.)

- Moving planes to new gates is expensive and disruptive. Can you design an algorithm that assigns airplanes to gates in such a way as to minimize the later reassignments? Create the initial assignment using extra gates, leaving as much slack in the schedule as possible. Show, via experiment, that this minimizes the number of gate reassignments. Prove that your assignment can handle a certain level of delay.

There are several other airport scheduling problems you might look at (e.g., scheduling gates to minimize the distance that passengers need to travel when changing planes). Or you may choose a different scheduling problem altogether.

Regardless of the problem, try to find some real data, determine how to come up with an optimal schedule, and then think about what happens when the schedule changes. How much trouble do the changes cause? What can you do about them? How can you minimize the number of changes? Can you leave extra slack in your schedule?