Linear Programming I

In this lecture, we describe a very general problem called *linear programming* that can be used to express a wide variety of different kinds of problems. We can use algorithms for linear programming to solve the max-flow problem, solve the min-cost max-flow problem, find minimax-optimal strategies in games, and many other things. We will primarily discuss the setting and how to code up various problems as linear programs (LPs).

Objectives of this lecture

In this lecture, we will cover

- The definition of linear programming and simple examples.
- Using linear programming to solve max flow and min-cost max flow.
- Using linear programming to solve for minimax-optimal strategies in games.

1 Introduction

In recent lectures we have looked at the following problems:

- Maximum bipartite matching
- Maximum flow (more general than bipartite matching).
- Min-Cost Max-flow (even more general than plain max flow).

Today, we'll look at something even more general that we can solve algorithmically: **linear programming**. Linear Programming is important because it is so expressive: many, *many* problems can be coded up as linear programs (LPs). This especially includes problems of allocating resources and business supply-chain applications. In business schools and Operations Research departments there are entire courses devoted to linear programming. There are also commercial software packages charging tens of thousands of dollars per license for solving linear programs (okay they also solve more general problems too, but linear programming is the basis for most of them)! Today we will mostly say what they are and give examples of encoding problems as LPs. We will only say a tiny bit about algorithms for solving them.

2 Definition of Linear Programming

Formally, a linear programming problem is specified as follows.

Definition: Linear program

Given:

- n **real-valued** *variables* $x_1, \dots, x_n$.
- A linear *objective function*. e.g., $2x_1 + 3x_2 + x_3$.
- m *linear inequalities* in these variables (equalities are OK too).

  e.g., $3x_1 + 4x_2 \leq 6$, or $0 \leq x_1 \leq 3$, etc.

Goal:

- Find values for the x_i's that satisfy the constraints and maximize or minimize the objective function.

Remark: No strict inequalities

Linear programs **can not** contain strict inequalities, e.g., $x_1 < 3$ or $x_2 > 5$. Why? Suppose we wrote maximize x_1 such that $x_1 < 3$, then there does not exist an optimal solution.

Remark: Variables are real-valued

This is super important to remember!! Linear program variables take on real values, i.e., the variables can not be guaranteed to take integer values!

An LP may also come without an objective function, in which case we simply wish to find any value for the x_i's that satisfy all of the constraints. Such an LP is called a "feasibility problem". You can think of this as a special case of a linear program where the objective value is just a constant (e.g., zero). We can write either minimization problems or maximization problems.

A set of x_i's that satisfies all of the constraints is called a *feasible solution*. Not all linear programs have a solution; it may be impossible to find any x_i's that satisfy the constraints. Alternatively, it may be the case that there is no optimal objective value because there exists feasible solutions of arbitrarily high value. We can classify all LPs this way into three categories.

Definition: Classification of LPs

Every LP falls into one of three categories:

- **Infeasible** (there is no point that satisfies the constraints)
- **Feasible and Bounded** (there is a feasible point of maximum objective function value)
- **Feasible and Unbounded** (there are feasible points of arbitrarily large value)

An algorithm for LP should classify the input LP into one of these categories, and find the optimum feasible point when the LP is feasible and bounded.

3 Modeling problems as Linear Programs

3.1 An Operations Research Problem

Here is a typical Operations-Research kind of problem (stolen from Mike Trick's course notes): Suppose you have 4 production plants for making cars. Each works a little differently in terms of labor needed, materials, and pollution produced per car:

	labor	materials	pollution
plant 1	2	3	15
plant 2	3	4	10
plant 3	4	5	9
plant 4	5	6	7

Suppose we need to produce at least 400 cars at plant 3 according to a labor agreement. We have 3300 hours of labor and 4000 units of material available. We are allowed to produce 12000 units of pollution, and we want to maximize the number of cars produced. How can we model this?

To model a problem like this, it helps to ask the following three questions in order: (1) what are the variables, (2) what is our objective in terms of these variables, and (3) what are the constraints. Let's go through these questions for this problem.

Variables: x_1, x_2, x_3, x_4, where x_i denotes the number of cars at plant i.
Objective: maximize $x_1 + x_2 + x_3 + x_4$.
Constraints:

$$\begin{aligned} x_i &\geq 0 \qquad \text{for all } i \\ x_3 &\geq 400 \\ 2x_1 + 3x_2 + 4x_3 + 5x_4 &\leq 3300 \\ 3x_1 + 4x_2 + 5x_3 + 6x_4 &\leq 4000 \\ 15x_1 + 10x_2 + 9x_3 + 7x_4 &\leq 12000 \end{aligned}$$

Note that we are not guaranteed the solution will be integral. For problems where the numbers we are solving for are large (like here), it is usually not a very big deal because you can just round them down to get an almost-optimal solution. However, we will see problems later where it *is* a very big deal.

3.2 Modeling Maximum Flow

We can model the max flow problem as a linear program too. In fact, when we wrote down the definitions of the max flow problem, like the capacity and conservation constraints, and defined the net s-t flow, we pretty much did write down a linear program already!

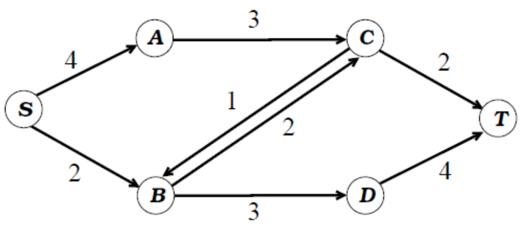
Variables: Set up a variable f_{uv} for each edge (u, v), representing $f(u, v)$

Objective: maximize $\sum_{u \in V} f_{su} - \sum_{u \in V} f_{us}$. (the net s-t flow)

Constraints:

- For all edges (u, v), $0 \leq f_{uv} \leq c(u, v)$. (capacity constraints)
- For all $v \notin \{s, t\}$, $\sum_{u \in V} f_{uv} = \sum_{u \in V} f_{vu}$. (flow conservation)

For instance, consider this example:



In this case, our LP is: maximize $f_{sa} + f_{sb}$ subject to the constraints:

Capacity	**Conservation**
$0 \leq f_{sa} \leq 4$	$f_{sa} = f_{ac}$
$0 \leq f_{ac} \leq 3$	$f_{sb} + f_{cb} = f_{bc} + f_{bd}$
$0 \leq f_{ct} \leq 2$	$f_{ac} + f_{bc} = f_{cb} + f_{ct}$
$0 \leq f_{sb} \leq 2$	$f_{bd} = f_{dt}$
$0 \leq f_{bd} \leq 3$	
$0 \leq f_{cb} \leq 1$	
$0 \leq f_{bc} \leq 2$	
$0 \leq f_{dt} \leq 4$	

3.3 Modeling Minimum-cost Max Flow

Recall that in min-cost max flow, each edge (u, v) has both a capacity $c(u, v)$ and a cost $\$(u, v)$. The goal is to find out of all possible maximum s-t flows the one of least cost, where the cost of a flow f is defined as

$$\sum_{(u,v) \in E} \$(u, v) f_{uv}.$$

Approach #1: Maximize flow first One simple way to do this is to first solve for the maximum flow f, ignoring costs. Then, set up a new linear program and add the constraint that flow must equal f, i.e.

$$\sum_{u \in V} f_{su} - \sum_{u \in V} f_{us} = f$$

(plus the original capacity and flow conservation constraints), then set the objective to minimize the cost

$$\text{minimize} \sum_{(u,v)\in E} \$(u,v) f_{uv}$$

Note that we have used a minimization objective function rather than a maximization this time. This is allowed. If you want to stick strictly to maximization problems, then you can maximize the negative of the cost instead, as this will give you the same solution.

Approach #2: Combine the two objectives The reason that min-cost max-flow is annoying to write as an LP is that it kind of has two objectives, to maximize flow then to minimize cost, and we can't directly encode two objectives into an LP. However, in *some* cases, it is possible to combine two objectives into one (but not always).

To simplify the presentation, lets add two new variables, one representing the net flow and another representing the total cost:

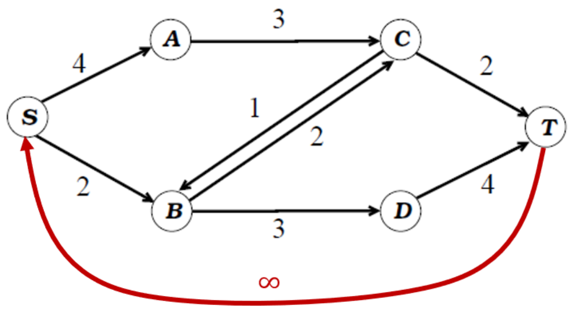
$$f = \sum_{u\in V} f_{su} - \sum_{u\in V} f_{us},$$
$$C = \sum_{(u,v)\in E} \$(u,v) f_{uv}$$

How can we represent the idea of maximizing the flow while tiebreaking by costs using these variables? One way is to multiply the cost by a very small constant ε, small enough that it is definitely less than the difference between the values of any two feasible flows. Then, we can use it to tiebreak since it will only ever matter if two feasible flows have the same value! So our objective function becomes:

$$\text{maximize}\, f - \varepsilon C$$

Approach #3: Reduce to "Minimum-cost circulation" As our last idea, instead of trying to encode the problem with two objective functions, we can instead reduce the problem to a similar one that only has one objective, and model that problem as a linear program.

To do so, lets forget about "producing" flow at s and "consuming" it at t and imagine that instead we take all of the flow into t and "circulate" it back to s, forming a big cycle of flow instead.



A *circulation* is the same thing as a flow, except that we now require that s and t also satisfy the conservation constraint. In this problem, there is no such concept as "net-flow" anymore, because the net flow should always be zero, so we can't directly ask about maximum flow. However, if we have costs on the edges, then we can write an LP for a *minimum-cost circulation*, which finds a circulation of minimum possible cost.

Variables: Set up a variable f_{uv} for each edge (u,v), representing $f(u,v)$

Objective: minimize $\sum_{(u,v)\in E} \$(u,v) f_{uv}$

Constraints:

- For all edges (u,v), $0 \le f_{uv} \le c(u,v)$. (capacity constraints)
- For **all** v, $\sum_{u\in V} f_{uv} = \sum_{u\in V} f_{vu}$. (flow conservation, **includes s and t!**)

To reduce minimum-cost max flow to a minimum-cost circulation problem, we draw the graph as above where we connect an infinite-capacity edge from t to s, and then give that edge a very negative cost. By giving (t,s) a very negative cost, the solution is encouraged to send as much flow as possible along (t,s), which we can observe is equivalent to maximizing the s-t flow! There are other variants of the minimum-cost circulation problem but we won't cover those.

3.4 2-Player Zero-Sum Games

Suppose we are given a 2-player zero-sum game with n rows and n columns, and we want to compute a minimax optimal strategy. For instance, perhaps a game like this (say payoffs are for the row player):

	column player		
	20	-10	5
row player	5	10	-10
	-5	0	10

Let's see how we can use linear programming to solve this game. Informally, we want the variables to be the things we want to figure out, which in this case are the probabilities to put on our different choices $p_1, \ldots, p_n$. These have to form a legal probability distribution, and we can describe this using linear inequalities: namely, $p_1 + \ldots + p_n = 1$ and $p_i \ge 0$ for all i.

Our goal is to maximize the worst case (minimum), over all columns our opponent can play, of our expected gain. This is a little confusing because we are maximizing a minimum. However, we can use a trick: we will add one new variable v (representing the minimum), put in *constraints* that our expected gain has to be at least v for every column, and then define our objective to be to maximize v. Assume our input is given as an array m where m_{ij} represents the payoff to the row player when the row player plays i and the column player plays j. Putting this all together we have:

Variables: $p_1, \ldots, p_n$ and v.
Objective: Maximize v.
Constraints:

- $p_i \geq 0$ for all $1 \leq i \leq n$,
- $\sum_{i=1}^{n} p_i = 1$. (the p_i form a probability distribution)
- $\sum_{i=1}^{n} p_i m_{ij} \geq v$ for all columns $1 \leq j \leq m$

Exercises: Linear Programming Fundamentals

Problem 1. Our second approach to modeling minimum-cost maximum flow as an LP was to introduce a small ε constant and use it to make the maximum flows be tiebroken by costs. What would be a suitable value for ε that would guarantee that this method works?

Problem 2. Suppose you have an algorithm that can tell you whether an LP is feasible, but does not give you the optimal objective value. Describe a simple method for determining the optimal objective value using the feasibility algorithm as a black box.