02-613 Week 12

Algorithms and Advanced Data Structures

Aidan Jan

November 21, 2025

The Simplex Method

Suppose we have a system we want to optimize under some constraints. Say, maximize $z = x_1 + x_2$ under the constraints

$$\begin{aligned} x_1 + x_2 &\leq 3 \\ x_1 - 3x_2 &\geq -1 \\ x_2 &\leq 3 \end{aligned}$$

Additionally, let all $x_i \geq 0$.

To use the simplex method, we first need to get these constraints into the **standard form**, which is to maximize $\vec{c} \cdot \vec{x}$, subject to $A\vec{x} \leq \vec{b}$ and $\vec{x} \geq 0$.

In our case, we need to do some rewriting. First, we have a greater than sign, which is not allowed. Therefore, we can convert it to a $\leq$ by multiplying both sides of the equation by -1.

$$\begin{aligned} x_1 + x_2 &\leq 3 \\ -x_1 + 3x_2 &\leq 1 \\ x_2 &\leq 3 \end{aligned}$$

Now, we add slack variables $S1, S2, S3$. The purpose of doing this is that [FILL rest of example]

The Simplex Algorithm

1. Write linear program with slack variables and get an initial feasible solution
2. Choose variable v in objective with positive coefficient.
3. Choose strictest constraint (will be one with negative coefficient on pivot)
4. Rewrite strictest constraint with v on the left hand side and subs in others
5. If all coefficients in the objective function are negative, then we are done. Otherwise, return to step 2.

Pivot Rules

- Largest coefficient in objective
- Largest increase after pivot
- Random

- Steepest change
- Bland's Rule - choose pivot v with lower index (This prevents infinite loops by always choosing the smallest index.)

NP Completeness

The phrase 'NP-Hard' tends to be used interchangeably with 'non-polynomial'. However, they are not the same thing.

Decision vs. Optimization Problems

- **Decision:** A problem that asks for a True/False solution.
- **Optimization:** A problem that asks for minimizing or maximizing some objective function.

Conveniently, any optimization problem can be turned into a decision problem with an extra parameter. For example:

- **Optimization:** Given graph $G = (V, E)$, and w weight on all edges, find the spanning tree with minimum total weight.
- **Decision:** Does there exist a spanning tree with at most $\leq k$ weight?

We refer to decision questions as 'problems'. Now we can define P and NP.

P vs. NP

Class P is a set of problems. A problem $Q \in P$, if there exists an algorithm A_Q such that, for all instances I of Q, $A_Q(I)$ runs in polynomial($|I|$) steps and

- If I is a yes instance $A_Q(I) = \text{yes}$
- If I is a no instance $A_Q(I) = \text{no}$

Class NP is a set of problems. A problem $Q \in NP$ if a **non-deterministic** Turing machine can return YES on a YES instance in polynomial time. A problem $Q \in NP$ if evidence of a Yes-instance can be verified in polynomial time. More formally, NP is a set of problems, Q for when there exists an algorithm $C_Q(\cdot)$ such that for all I of Q:

- If I is YES, then $C_Q(I, S) = \text{Yes}$, for some S
- If I is NO, then $C_Q(I, S) = \text{No}$, for all S
- Additionally, $C_Q(\cdot)$ runs in polynomial time.

This implies that problems can be both P *and* NP. $P \subset NP$.

Example: NP-Hard

Consider the traveling salesman decision problem. Given distances d_{ij} between n cities and a number k, is there a way to visit all cities exactly once, with total length of less than or equal to k?

- If given a valid certificate and a k-value (e.g., a YES instance), it is provable in polynomial time.
- Finding a valid certificate (e.g., an instance that satisfies the problem), is not possible though. Therefore, it is an NP-hard question.

Theorem: $P \subseteq NP$

Proof: Suppose $x \in P$ There exists a polynomial alorithm $A_x(\cdot)$. To show $X \in NP$, need efficient certifier $C_x(I, S) = A_x(I)$. Therefore, $P \subseteq NP$. Now, does $P = NP$? It turns out this is a hard problem in itself, and no proof exists for it.

Problem Reductions

A reduction from problem X to Y (written as $X \leq_P Y$), it means for instance I_X of X, create a new instance $(I_X)_Y$ in polynomial time, such that $\{(I_X)_Y$ is a yes instance $\longleftrightarrow I_X$ is a YES-instance$\}$, and Y is **at least as hard** as X.

- This implies that if there exists no polynomial time solution for X, then there must also not exist a polynomial time solution for Y.