CS 1440/2440: Linear Programming

A quick overview
What is it?

- Method used to solve **constrained optimization** problems
- Linear: because it can be expressed entirely using **linear relationships**!
- Applications and famous examples:
  - Graph problems
  - Game Theory (!!!)
  - The Knapsack Problem
- An example:
  - A factory produces cars and trains. Producing a car takes 30 units of metal and 20 hours of work. Producing a train takes 100 units of metal and 30 hours of work. Each car sells for $1,000 and each train sells for $2,500.
  - You, the factory owner, have access to at most 1,000 units of metal and 500 hours of labour. You must decide how many cars and trains should be produced to maximize profit.
A factory produces cars and trains. Producing a car takes 30 units of metal and 20 hours of work. Producing a train takes 100 units of metal and 30 hours of work. Each car sells for $1,000 and each train sells for $2,500.

You, the factory owner, have access to at most 1,000 units of metal and 500 hours of labour. You must decide how many cars and trains should be produced to maximize revenue.

Decision variables: \( c \) (no. of cars), \( t \) (no. of trains)

Objective function: \( \text{max}(\text{Revenue} = 1000c + 2500t) \)

Constraints:
\[
\begin{align*}
30c + 100t & \leq 1000 \\
20c + 30t & \leq 500 \\
0 & \leq c, 0 & \leq t
\end{align*}
\]

- constraint on metal
- constraint on labour
- non-negativity constraints
Solving using graphs

Theorem:

The optimal solution to the problem is one of the corners of the feasible region.

The simplex algorithm uses this property to find the optimal solution.
Solving using computers

Linear programs can be solved in polynomial time!

Here’s some Python code to solve this problem:

```python
from scipy.optimize import linprog

obj = [-1000,-1500]
A = [[30, 100], [20, 30]]
b = [1000,500]
x0_bounds = (0, None)
x1_bounds = (0, None)

res = linprog(obj, A, b,
             bounds=(x0_bounds, x1_bounds))
print(res)
```

Some other tools for solving linear programs: MATLAB, R (lpSolve)
In general...

- You have an objective function $f$ of $n$ variables and their respective coefficients.
- You have $m$ constraints, where $n \leq m$.
- The problem can also be written using vectors and matrices:
  - Objective function: Maximize $c^T x$
  - Constraints: $A x \leq b$
  - Where
    - $c$ and $x$ are $n$ by 1 vectors
    - $A$ is an $m$ by $n$ matrix of coefficients
    - $b$ is an $m$ by 1 vector of constants (constraint bounds)
More general stuff...

- Can there be more than one optimal solution?
  - Yes! This is when one or more constraints are parallel to the objective function
  - In this case, there are infinitely many optimal solutions
- Can there be no solutions?
  - Yes- this is usually when two or more solutions contradict each other (infeasible)
  - It may also be the case that the feasible region is infinite (unbounded)
- Can there be equality constraints?
  - Yes- though we handle these a little differently from inequalities
  - Introduce a “slack” variable
    - $p + q = c$, $0 \leq p$, $0 \leq q$
    - $p + q + s \leq c$, $0 \leq p$, $0 \leq q$, $0 \leq r$
Checking in

Any questions so far?
Duality
What is duality?

Every linear program can be rewritten in a form known as its dual. The original problem is called the primal.

Primal

- Maximize: \( f = c^T x \)
- Subject to: \( Ax \leq b, \ x \geq 0 \)
- Gives the optimum quantity of each variable needed to maximize the objective function
- Revenue maximization

Dual:

- Minimize: \( f' = b^T y \)
- Subject to: \( A^T y \geq c \)
- Gives an upper bound on the optimal solution to the primal problem, if one exists.
- Shadow price minimization
Revenue we would gain by increasing the RHS of a constraint by one unit

The shadow prices of primal constraints are the optimal values of decision variables in the dual

In the example above, increasing the constraint on metal by 1 increases the revenue by 20.5 units

Increasing the constraint on labour by 1 increases the revenue by 22.5 units

This kind of analysis is called sensitivity analysis
Continuing with examples

The automobile manufacturing example from before has the dual:

Instead of maximizing the revenue, we now minimize the resource use for a given level of output.

Decision variables: \( m \) (units of metal), \( l \) (units of labour)

Objective function: \( \text{min} \{ \text{Resource use} = 1000m + 500l \} \)

Constraints:

\[
30m + 20l \geq 1000
\]

\[
100m + 30l \geq 2500
\]

\[
0 \leq m, 0 \leq l
\]

- constraint on no. of cars
- constraint on no. of trucks
- non-negativity constraints
Integer (Linear) Programming

- When there’s an additional constraint in which the decision variables have to be integers
  - You can’t have 0.67 cars or 1.8 people!
- Integer linear programming is NP hard, which means we don’t have an efficient (polynomial runtime) algorithm
More examples!

- Bipartite matching problem
- Maximum flow graphs (the dual of this problem is the minimum-cut problem!)
- Graph colouring (integer program)
- Rock-paper-scissors!!!
So far, we’ve seen games where players lose nothing by playing.

In zero-sum games, the payoffs of all players sum to zero. This means that some players have negative payoffs.

Can you think of an example we’ve already seen?
Given Player 2 plays R, P, and S with probabilities $0 \leq r_2, p_2, s_2 \leq 1$, with what probabilities should Player 1 play R, P, and S?

Write the problem as a linear program (and try to solve it if you can!)
Theorems of Duality

Strong Theorem of Duality:

If there is an optimal solution \( y^* \) to the dual, then an optimal solution \( x^* \) to the primal exists. Further,

\[
f'(y^*) = f(x^*)
\]

The minimax algorithm is just an application of this theorem.

Weak Theorem of Duality (Extra):

For any feasible solution \( y \) in the dual and \( x \) in the primal

\[
f(x) \leq f'(y)
\]

Implications:

- If the primal is infeasible, the dual is unbounded
- If the primal is unbounded, the dual is infeasible
Acknowledgements

This presentation was prepared drawing from older materials for this course, including a presentation with similar content by George Spahn. The images and Python code come from this presentation too.

The GIF was originally shared on giphy.com