Introduction to Linear Programming

Mitchell Jones

December 19, 2014

Outline

- What are linear programs?
- Why are they useful?
- Solving linear programs
- Integer linear programs
- A few examples

What is linear programming (LP)?

- Nothing to do with programming in an engineering sense :(
- Related to modelling a problem as a mathematical “program”
- Given some constraints, we want to maximise or minimise some objective goal
- Linear programming can model a wide range of problems related to computer science

Example problems

Some example problems that can be solved with LP..

1. Knapsack
2. Max flow/min cut
3. Minimum spanning tree
4. Vertex cover
5. Connected vertex cover
6. Independent set
7. Facility location
8. Job-Machine scheduling

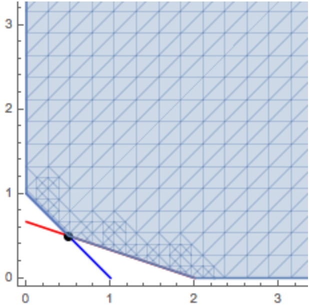
A simple optimisation example

- Minimise x^2 subject to $x \geq 0$
- A trivial example, but what about adding more variables?
- Minimise $x^2 + y^2$ subject to $x \geq 1$, $y \leq 1$
- Now extend this to n variables..

Linear programming example

- The last example involved *quadratic* objectives (quadratic programming), linear programming involves linear terms
- The following linear program has 2 variables and 2 constraints

$$\begin{array}{lrcl} \text{min} & x_1 + x_2 & & \\ \text{subject to} & x_1 + x_2 & \geq & 1 \\ & x_1 + 3x_2 & \geq & 2 \\ & x_1, x_2 & \geq & 0 \end{array}$$



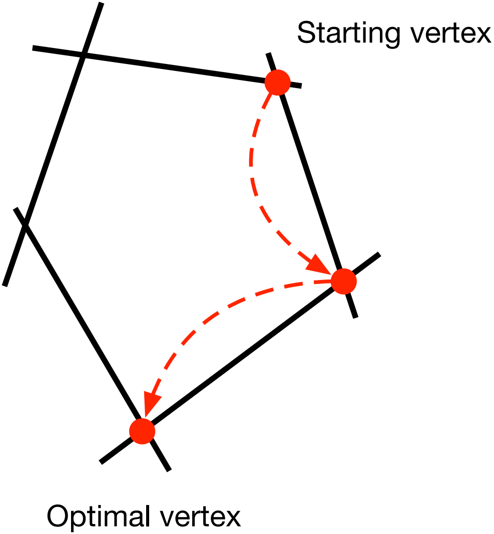
Standard Form

- More generally, we express a linear program with n variables and m constraints in matrix form
- $\min \mathbf{c} \cdot \mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$
- Where $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$
- In our previous example: $\mathbf{c} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$

Great, but how do I solve them?

- Use an algorithm called SIMPLEX
- Observation: The optimal solution always lies on the vertex of a polyhedra
- High-level idea: Start at an arbitrary vertex, and jump from vertex to vertex until an optimal solution is found

Visualisation of SIMPLEX

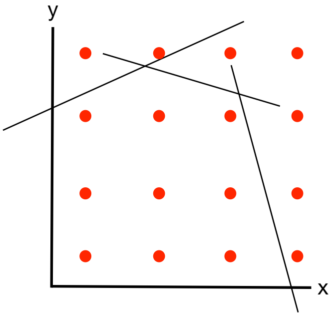


Discrete Optimisation

- Still an optimisation problem, but now given a discrete (finite) set of possible feasible values
- Want to find the 'cheapest' feasible value
- Formally: Given a set of feasible solutions $\mathcal{F}$ and an objective function $f : \mathcal{F} \to \mathbb{R}$, we want to solve $\min_{x \in \mathcal{F}} f(x)$

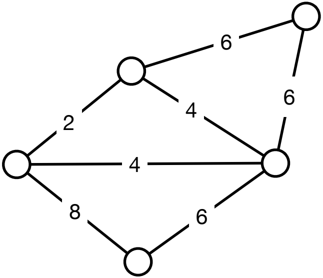
Integer Programming

- Similar to linear programs, however instead of a solution $\mathbf{x}$ being real values, they are integer values (i.e. $\mathbf{x} \in \mathbb{Z}_{+}^{n}$)
- Our set of feasible solutions now are in the format of a grid
- Unfortunately solving an integer linear program is NP-hard
- Just because we can model a problem as an integer linear program, doesn't meant it's NP-hard in general

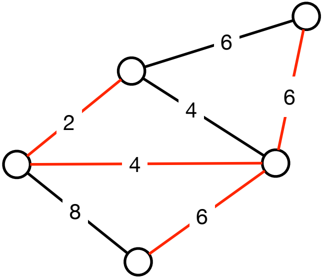


The Minimum Spanning Tree (MST) problem

- Given an undirected graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}$ on each edge, select an acyclic subset of edges such that we span the graph
- Want to minimise of the sum of the selected edge's weights



A graph G



Possible (minimum) spanning tree

Modelling the MST Problem as an IP

- By our formal definition:
  - $\mathcal{F} = \{T \subseteq E : (V, T) \text{ is a tree}\}$
  - Want to minimise $f(T) = \sum_{e \in T} w_e$
- Intuition: For each edge $e \in E$, create a variable $x_e \in \{0, 1\}$
- If $x_e = 1$ then this indicates we wish to include e in the spanning tree
- The set of edges we pick must be acyclic and we only want to pick $|V| - 1$ edges

The IP

$$
\begin{array}{llcll}
\min & \sum_{e\in E} w_e x_e & & & \\
\text{subject to} & \sum_{e\in E} x_e & = & |V|-1 & \\
 & \sum_{e\in E[S]} x_e & \leq & |S|-1 & \forall \emptyset \subset S \subseteq V \\
 & x_e & \in & \{0,1\} & \forall e \in E
\end{array}
$$

Where $E[S]$ is the *edge set* of a subset $S \subseteq V$. It is the set of all edges in the component S.

How does this relate to linear programming?

- For certain types of integer programs, we can *relax* program
- By relaxing the IP, we turn it into a LP
- Instead of $x_e \in \{0, 1\}$, let $1 \geq x_e \geq 0$ (i.e. can take any value between 0 and 1)
- Turns out every vertex (and therefore every solution) of this polyhedron is integral!
- Therefore we can also use SIMPLEX to solve the MST problem

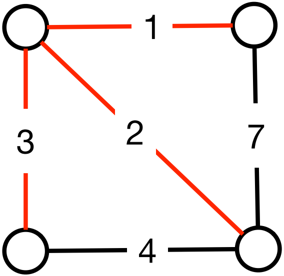
Final MST LP

$$
\begin{array}{llcll}
\min & \sum_{e \in E} w_e x_e & & & \\
\text{subject to} & \sum_{e \in E} x_e & = & |V| - 1 & \\
& \sum_{e \in E[S]} x_e & \leq & |S| - 1 & \forall \varnothing \subset S \subseteq V \\
& x_e & \geq & 0 & \forall e \in E
\end{array}
$$

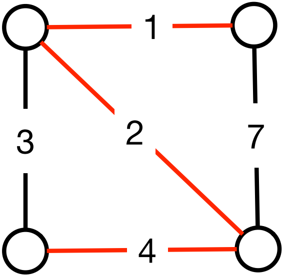
Degree Bounded MST

- Of course we can solve MST in Polynomial time (e.g. Kruskal's algorithm), so why bother?
- Modelling the MST problem as an IP allows us to solve more difficult versions of the problem
- Let's impose another constraint: The degree of each vertex $v \in V$ in the spanning tree cannot be more than a fixed integer Δ

Example when $\Delta = 2$



Optimal MST,
does not satisfy
degree constraint



Optimal MST,
does satisfy
degree constraint

The IP for Degree Bounded MST

Can reuse our old IP, and impose this new constraint for each vertex in G:

$$\begin{array}{llcll}
\min & \sum_{e \in E} w_e x_e & & & \\
\text{subject to} & \sum_{e \in E} x_e & = & |V| - 1 & \\
& \sum_{e \in E[S]} x_e & \leq & |S| - 1 & \forall \varnothing \subset S \subseteq V \\
& \sum_{e \in \delta(u)} x_e & \leq & \Delta & \forall u \in V \\
& x_e & \in & \{0, 1\} & \forall e \in E
\end{array}$$

Where $\delta(u) = \{(u, v) \in E\}$ is the set of neighbouring edges to a node u

The IP for Degree Bounded MST (cont.)

- The optimisation version of the Degree Bounded MST problem is NP-hard
- By only adding a single extra constraint for each vertex, the complexity of the problem has blown up
- This is what makes Linear/Integer programming so powerful, can model problems of varying complexity

What we've covered

- Standard represention of a LP
- Core-idea behind SIMPLEX
- Integer programs
- Two applications of IP

Thanks for listening!

Questions?