The Simplex Algorithm for LP, and an Open Problem
Linear Programming: General Formulation

\[
\begin{align*}
\text{max } & \quad c^T x \\
\text{subject to } & \quad Ax \leq b \\
& \quad x \geq 0
\end{align*}
\]

- Inputs: real-valued $m \times n$ matrix $A$, and vectors $c$ in $\mathbb{R}^n$ and $b$ in $\mathbb{R}^m$
- Output: $n$-dimensional vector $x$
- There is one constraint per row of $A$, $m$ constraints in total
The Simplex Algorithm

let v be any vertex of the feasible region
while v is not optimal
  find neighbor v' of v with a better objective value
  (or determine that the feasible region is unbounded)
  set v = v'
output v

• *Vertex*: a point at the intersection of n or more facets, i.e., (n-1)-dimensional hyperplanes
• *Edge*: the intersection of (n-1) facets
• *Neighbours*: a pair of vertices connected by an edge
The Simplex Algorithm
The Simplex Algorithm

Algorithmic implementation issues:

• How do we find an initial feasible \( v \)? (Or determine that the feasible region is empty)

• How do we determine whether \( v \) is optimal?

• How do we find a neighbor \( v' \) of \( v \) with better objective value?
The Simplex Algorithm

Algorithmic implementation issues:
• How do we find an initial feasible \( v \)? (Or determine that the feasible region is empty)
• How do we determine whether \( v \) is optimal?
• How do we find a neighbor \( v' \) of \( v \) with better objective value?

We'll start with simplifying assumptions:
• Our feasible \( v \) is at the origin
• Also, \( v \) is not degenerate: it is the intersection of exactly \( n \) facets
Algorithmic implementation issues:

• How do we find an initial feasible \( v \)? (Or determine that the feasible region is empty)

• How do we determine whether \( v \) is optimal?

• How do we find a neighbor \( v' \) of \( v \) with better objective value?

Now let’s remove our assumptions:

• \( v \) is not at the origin

• \( v \) is degenerate
The Simplex Algorithm

If \( v \) is not at the origin, we simply shift our coordinate system!

For each of the \( n \) linearly independent facets defining \( v \), let \( y_i = b_i - a_i x \)

- Describe all of the constraints and objective function in terms of the \( y_i \)'s rather than the \( x_i \)'s
- The constraint "\( a_i x \leq b_i \)" becomes "\( y_i \geq 0 \)"; as a result \( v \) shifts to 0
What if $v$ is degenerate?

- Pick any $n$ of the facets. Then apply the algorithm just as before. That is, relax one of the current $n$ facets, identify a new facet that blocks improvement, and replace the relaxed one with this new blocking one. Perhaps surprisingly, we will not run into an infinite loop! (Proof omitted)
An Open Problem: Stochastic Games

• Generalization of Markov Decision Processes

• Applications in economics, evolutionary biology, computer networks
Stochastic Games

• Consider a directed graph with Decision, Random, and Sink nodes as before, but also now with Adversary nodes.
• This models a zero sum game involving two players: Decider and Adversary.
• A policy D of the Decider is simply a choice of one edge from each decision node, and a policy A of the Adversary is one edge from each Adversary node.
Stochastic Games

• Decider wins if the “win” sink node n is reached, otherwise Adversary wins, i.e. when the “lose” sink node n-1 is reached

• Again assume that any run of the process halts with probability 1, for any policies of the players

• We want to find a policy D that maximizes Decider's chance of winning, assuming that Adversary uses a policy A that minimizes Decider's chance of winning
Stochastic Games

Let $w_{D,A}(i)$ be the probability that Decider wins if policies D and A are used and the game starts at node i

- A min-max theorem holds for stochastic games: for any node i,

$$\max_D \min_A w_{D,A}(i) = \min_A \max_D w_{D,A}(i)$$

- "Mixed" policies that assign probabilities to edges from Decision and Adversary nodes are no better than the “pure” policies we've defined
Stochastic Games

• Optimal policies are independent of the starting node: there are policies D* and A* such that for all nodes i,

\[ w_{D^*,A^*}(i) = \max_D \min_A w_{D,A}(i) = \min_A \max_D w_{D,A}(i) \]

• With these optimal policies, value \( w_{D^*,A^*}(i) \) of node i is
  – the \textit{max} of its children’s values if i is a Decision node
  – the \textit{min} of its children’s values if i is an Adversary node
  – the \textit{average} of its children’s values if i is a Random node
  – 0 if i is the “lose” sink node n-1 and 1 if i is the “win” sink node n
Exercise: For a \textit{fixed} policy $A$ of the Adversary, how can we find an optimal policy $D$ of the Decider in polynomial time?

Exercise: For a fixed policy $D$ of the Decider, how can we find an optimal policy $A$ of the Adversary in polynomial time?
Exercise: How to find optimal policies of both players (not necessarily in polynomial time)?
Exercise: How to find optimal policies of both players (not necessarily in polynomial time)?

- **Exhaustive search**, using min-max theorem:
  \[
  \max_D \min_A w_{D,A}(i) = \min_A \max_D w_{A,D}(i)
  \]

- For each fixed D, find the optimal A, i.e. the A that minimizes \( \min_A w_{D,A}(i) \) for all i. Choose the D* that maximizes these quantities.
Exercise: How to find optimal policies of both players (not necessarily in polynomial time)?

*Extended Policy iteration*:  
- Starting with an arbitrary policy D, find the optimal A  
- Then switch the decision of D at a node if it improves the winning probabilities  
- Repeat until no further improvement is possible  
- Once again, this "local search" approach yields globally optimal policies
Stochastic Games

No polynomial time algorithm is known to find optimal policies of stochastic games

Consider the decision problem: Does Decider win with probability at least 1/2 from a given start node?

The decision problem is in NP (why?)

AND... the complement of the decision problem is in NP!

Thus the problem is in NP intersection co-NP, and thus unlikely to be NP-complete
LP Wrap-up

• Linear Programming algorithms are useful for solving a broad array of optimization problems, with many available solvers that use interior point methods or simplex