Linear Programming

The definitions of LP, and other pieces of the material appear in CLRS Chapter 29
In the diet problem, we will have to compute two values $x$ and $y$.

- $x$ indicates how many *bananas* we plan to consume daily.
- $y$ indicates how many *oranges* we plan to consume daily.

The goal is to find a healthy diet that is as cheap as possible.
Define: (amount consumed per day)
- types of foods: \{x-bananas, y-oranges\}
- \(j\) – types of vitamins \((1 \leq j \leq n)\).
- \(x\) – number of oranges we recommend daily
- \(y\) – number of bananas we recommend daily
  // these are the only unknown we have to compute.
- \(a_{ij}\) – the amount of vitamin \(i\) in a unit of food \(j\).
  \((j=1\text{ for bananas, }j=2\text{ for oranges})\). These are given constants.
- \(c_1\) – the cost of an banana.
- \(c_2\) – the cost of an orange
- \(b_i\) – minimal daily required amount of vitamin \(j\), for the diet to be healthy, \(i=1..n\)

Minimize: minimize the cost of a healthy diet

\[
C((x, y)) = c_1 x + c_2 y
\]

\[
\begin{align*}
a_{11}x + a_{12}y & \geq b_1 \\
& \vdots \\
a_{n1}x + a_{n2}y & \geq b_n
\end{align*}
\]
Linear Programming – The Geometry

\[
a_1x + a_2y \geq b
\]

\[
a_1x + a_2y \leq b
\]
Linear Programming – The Geometry

- Each constraint defines a half-space region in \( d \)-dimensional space.
- The feasible region is the (convex) intersection of these half-spaces.

- We will treat the case \( d = 2 \), where each constraint defines a half-plane.
- The equation \( y = ax + b \) defines a line, which we could also write as \( (-a)x + (1)y = b \). Pointed one side of this line forms a half-plane.

\[
\begin{align*}
    a_1x + a_2y & \geq b \\
    a_1x + a_2y & \leq b
\end{align*}
\]
Linear Programming – The Geometry

- Each constraint defines a half-space region in $d$-dimensional space.
- The *feasible region* is the (convex) intersection of these half-spaces.

- We will treat the case $d = 2$, where each constraint defines a *half-plane*.
- The equation $y = ax + b$ defines a line, which we could also write as $(-a)x + (1)y = b$. Pointed one side of this line forms a half-plane.

$$a_1x + a_2y \geq b$$

$$a_1x + a_2y \leq b$$
Each constraint defines a half-space region in $d$-dimensional space.

The feasible region is the (convex) intersection of these half-spaces.

We will treat the case $d = 2$, where each constraint defines a half-plane.

The equation $y = ax + b$ defines a line, which we could also write as $(-a)x + (1)y = b$. Pointed one side of this line forms a half-plane.

$$a_1 x + a_2 y \geq b$$
$$a_1 x + a_2 y \leq b$$
Each constraint defines a half-space region in $d$-dimensional space.

The **feasible region** is the (convex) intersection of these half-spaces.

We will treat the case $d = 2$, where each constraint defines a **half-plane**.

The equation $y = ax + b$ defines a line, which we could also write as $(-a)x + (1)y = b$. Pointed one side of this line forms a half-plane.

\[
a_1x + a_2y \geq b
\]
\[
a_1x + a_2y \leq b
\]
Each constraint defines a half-space region in $d$-dimensional space.

The feasible region is the (convex) intersection of these half-spaces.

We will treat the case $d = 2$, where each constraint defines a half-plane.

The equation $y = ax + b$ defines a line, which we could also write as $(-a)x + (1)y = b$. Pointed one side of this line forms a half-plane.

$$a_1 x + a_2 y \geq b$$

$$a_1 x + a_2 y \leq b$$
Each constraint defines a half-space region in $d$-dimensional space.

The feasible region is the (convex) intersection of these half-spaces.

We will treat the case $d = 2$, where each constraint defines a half-plane.

The equation $y = ax + b$ defines a line, which we could also write as $(-a)x + (1)y = b$. Pointed one side of this line forms a half-plane.

\[
\begin{align*}
a_1 x + a_2 y &\geq b \\
a_1 x + a_2 y &\leq b
\end{align*}
\]
A shape $S$ is convex if for every two points $p,q$ inside the shape, the segment connecting these points is also inside the shape.

If $A$ and $B$ are two convex shapes, then their intersection $A \cap B$ (namely, points that belong both to $A$ and to $B$) is also convex.

A half-plane is convex.

Conclusion: Intersection of half-planes is convex.
A shape $S$ is convex if for every two points $p,q$ inside the shape, the segment connecting these points is also inside the shape.

If $A$ and $B$ are two convex shapes, then their intersection $A \cap B$ (namely, points that belong both to $A$ and to $B$) is also convex.

A half-plane is convex.

Conclusion: Intersection of half-planes is convex.
A shape $S$ is convex if for every two points $p,q$ inside the shape, the segment connecting these points is also inside the shape.

If $A$ and $B$ are two convex shapes, then their intersection $A \cap B$ (namely, points that belong both to $A$ and to $B$) is also convex.

A half-plane is convex.

Conclusion: Intersection of half-planes is convex.
The solution to the linear program is a point in the feasible region that is extreme in the direction of the target function.

**Theorem:** Any bounded linear program that is feasible has a solution, which is a vertex of the feasible region.
The solution to the linear program is a point in the feasible region that is extreme in the direction of the target function.

Theorem: Any bounded linear program that is feasible has a solution, which is a vertex of the feasible region.
The solution to the linear program is a point in the feasible region that is extreme in the direction of the target function.

Theorem: Any bounded linear program that is feasible has a solution, which is a vertex of the feasible region.

Proof: Convexity …
The solution to the linear program is a point in the feasible region that is extreme in the direction of the target function.

Theorem: Any bounded linear program that is feasible has a solution, which is a vertex of the feasible region.

Proof: Convexity …
More Geometry

- The solution to the linear program is a point in the feasible region that is extreme in the direction of the target function.

- **Theorem:** Any bounded linear program that is feasible has a solution, which is a vertex of the feasible region.

- **Proof:** Convexity …
Degenerate Cases
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
  - Not unique
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
  - Not unique
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
  - Not unique
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
  - Not unique
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
  - Not unique
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
  - Not unique
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
  - Not unique
Degenerate Cases

- The feasible region may be:
  - Empty
  - Unbounded

- The solution may be:
  - Not unique
The Simplex Algorithm
The Simplex Algorithm
The Simplex Algorithm
The Simplex Algorithm
The Simplex Algorithm

- Assume WLOG that the cost function points “downwards”.
- Construct (some of) the vertices of the feasible region.
- Walk edge by edge downwards until reaching a local minimum (which is also a global minimum).
- In $\mathbb{R}^d$, the number of vertices might be $\Theta(n \lfloor d/2 \rfloor)$.
Linear Programming - Example

Define: (amount amount consumed per day)
- $i$ – types of foods ($1 \leq i \leq d$).
- $j$ – types of vitamins ($1 \leq j \leq n$).
- $x_i$ – the amount of food of type $i$ consumed per day.
- $a_{ji}$ – the amount of vitamin $j$ in one unit of food $i$.
- $c_i$ – the number of calories in one unit of food $i$.
- $b_j$ – minimal required amount of vitamin $j$.

Constraints (we need to consume some minimal amount of each vitamin):

Minimize: the total number of calories consumed:

$$C(x) = c_1 x_1 + c_2 x_2 + \boxed{?} + c_d x_d$$
Linear Programming - Example

- Define: (amount amount consumed per day)
  - $i$ – types of foods ($1 \leq i \leq d$).
  - $j$ – types of vitamins ($1 \leq j \leq n$).
  - $x_i$ – the amount of food of type $i$ consumed per day.
  - $a_{ji}$ – the amount of vitamin $j$ in one unit of food $i$.
  - $c_i$ – the number of calories in one unit of food $i$.
  - $b_j$ – minimal required amount of vitamin $j$.

- Constraints (we need to consume some minimal amount of each vitamin):
  \[
  a_{11}x_1 + a_{12}x_2 + \ldots + a_{1d}x_d \geq b_1
  \\
  a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nd}x_d \geq b_n
  \]

- Minimize: the total number of calories consumed:
  \[
  C(x) = c_1x_1 + c_2x_2 + \ldots + c_dx_d
  \]
LP History

- Mid 20th century: Simplex algorithm, time complexity $\Theta( n \lfloor d/2 \rfloor )$ in the worst case.
- 1980’s (Khachiyan) ellipsoid algorithm with time complexity $\text{poly}(n,d)$.
- 1980’s (Karmakar) interior-point algorithm with time complexity $\text{poly}(n,d)$.
- 1984 (Megiddo) – parametric search algorithm with time complexity $O( C_d n )$ where $C_d$ is a constant dependent only on $d$. E.g. $C_d = 2^{d^2}$.
- The holy grail: An algorithm with complexity independent of $d$.
- In practice the simplex algorithm is used because of its linear expected runtime.
O(n log n) 2D Linear Programming

- **Input:**
  - \( n \) half planes.
  - Cost function that WLOG “points down”.

- **Algorithm:**
  - Partition the \( n \) half-planes into two groups.
    - \( S \) are all halfplanes contain the point \((0, \infty)\)
    - \( S' \) all other halfplanes contain the point \((0, -\infty)\)
  - Sort them by slopes
  - Compute the upper envelop \( U(S) \) and the lower envelop \( L(S') \)
    (using question from hw1)
  - Scan simultaneously from left to right, and Compute intersection of two envelopes - they can intersect only at 2 points (why).
  - Evaluate cost function at each vertex.
O(n^2) Incremental Algorithm

- The idea:
  - Start by intersecting two halfplanes.
  - Add halfplanes one by one and update optimal vertex by solving one-dimensional LP problem on new line *if needed.*
Incremental Algorithm - Notation

Cost function to minimize: \( c(x,y) = y \).
Returns the lowermost point in feasible region
Incremental Algorithm - Notation

The $i$th half plane is the line that defines the feasible region after $i$ constraints. The optimal vertex of $C_i$ is $h_i$. Costs

Cost function to minimize: $c(x,y) = y$. Returns the lowermost point in feasible region.
Incremental Algorithm - Notation

Cost function to minimize: \( c(x,y) = y \).

Returns the lowermost point in feasible region.
Incremental Algorithm - Notation

Cost function to minimize: \( c(x,y)=y \).
Returns the lowermost point in feasible region.
Incremental Algorithm - Notation

Cost function to minimize: $c(x,y)=y$. Returns the lowermost point in feasible region.
Incremental Algorithm - Notation

Cost function to minimize: \( c(x,y) = y \). Returns the lowermost point in feasible region.
Cost function to minimize: \( c(x,y)=y \). Returns the lowermost point in feasible region.
Theorem:
1. if $v_{i-1} \in h_i$, then $v_i = v_{i-1}$.  // O(1) check, nothing to do
2. if $v_{i-1} \notin h_i$, then it is sufficient to look for $v_i$ on $l_i$ (rather than the whole plane)

Conclusion: If there is no solution on $l_i$, then there is no solution at all. The feasible region is empty.

Proof:
1. Trivial. Otherwise $v_i$ would not have been optimum before.
2. Assume that $v_i$ is not on $l_i$. $v_i$ must be in $C_{i-1}$. By convexity, also the segment $v_iv_{i-1}$ is in $C_{i-1}$.

Consider point $v_j$ - the intersection of $v_iv_{i-1}$ with $l_i$. $v_j$ is in both $C_{i-1}$ and $C_i$, and is better than $v_i$.

Contradiction.
Basic Algorithm

1. Check if $v_{i-1} \in h_i$. If yes, then $v_i = v_{i-1}$. // O(1),

ELSE

2. // $v_i$ must be on the line $l_i$. This is inherently the same problem,
   // but in 1-dim rather than 2-dim. We call it 1D LP

3. If the 1dim problem does not have a solution on $l_i$. - stop.
   There is no solution anywhere else.

ELSE

set $v_i$ to find this solution (see next slides).
Finding $v_i$ given $l_i$ and $\{h_1, h_2, \ldots, h_{i-1}\}$

(one-dimensional LP)

- Intersect each $h_j$ ($j<i$) with $l_i$, generating $i-1$ rays representing (unbounded) intervals.
- Compute the intersection of these $i-1$ intervals. Takes $O(i)$ time.
- If the intersection is empty then report no solution, else report the lowest point.
Worst case, each new constrain $h_n$ forces solving a new 1DLP

$$T(n) = \sum_{i=3}^{n} O(i) = O(n^2)$$
Incremental Algorithm – O(n) Randomized Version

- Exactly like the deterministic version, only the order of the lines is random.

- **Theorem:** The expected runtime of the random incremental algorithm (over all $n!$ permutations of the input constraints) is $O(n)$. 
Probability Analysis

Backward analysis

- **Question**: When given a solution after $i$ half-planes, what is the probability that the *last* half-plane affected the solution?
  - (assume first no three lines shares a point)

- **Answer**: Exactly $2/i$, because a change can occur only if the last halfplane inserted is one of the two halfplanes thru $v_i$.
  - (note that $v_i$ depends on the $i$ halfplanes, but not on their order)
Probability Analysis

Backward analysis

Question: When given a solution after \( i \) half-planes, what is the probability that the last half-plane affected the solution?

(assume first no three lines shares a point)

Answer: Exactly \( 2/i \), because a change can occur only if the last halfplane inserted is one of the two halfplanes thru \( v_i \).

(note that \( v_i \) depends on the \( i \) halfplanes, but not on their order)
Complexity Analysis

Handwave: What is the probability that in the i’th stage we face a violation with $h_i$?

Answer: $2/i$, since only 2 halfplanes determine $v_i$.

So we spend $O(i)$ with probability $2/i$, or $O(1)$ with probability $(i-2)/i$. So the average time in this stage is on average $O(1)$.
False Claim:

The probabilistic analysis is for the average input. Hence there exist bad sets of constraints for which the algorithm’s expected runtime is more than $O(n)$, and there exist good sets of constraints for which the algorithm’s expected runtime is less than $O(n)$. 
False Claim:
The probabilistic analysis is for the average input. Hence there exist bad sets of constraints for which the algorithm’s expected runtime is *more* than $O(n)$, and there exist good sets of constraints for which the algorithm’s expected runtime is *less* than $O(n)$.

True Claim:
The probabilistic analysis is valid for *all* inputs. The expected complexity is over all *permutations* of this input.
Now the input is a collection of half-spaces \( \{h_1 \ldots h_n\} \).

Now \( l_i \) is the plane bounding \( h_i \). (notations are analogous to the 2D case).

We will define \( v_3 \) as the intersection of the planes \( l_1, l_2 \) and \( l_3 \).

We insert the other halfspaces \( \{h_4 \ldots h_n\} \) at a random order, and update \( v_i \) according to the following Theorem:

**Theorem:**

1. if \( v_{i-1} \in h_i \), then \( v_i = v_{i-1} \). // O(1) check, nothing to do

2. if \( v_{i-1} \not\in h_i \), then the solution (if exists) is on \( l_i \).

run \( v_i = 2DLP( h_1 \cap l_i, h_2 \cap l_i, h_3 \cap l_i, \ldots, h_{i-1} \cap l_i ) \).

Terminates if there is no solution (that is, \( C_i=\emptyset \))
LP in 3D and higher dimension

In 3D, the worst case running time is $\Theta(n^3)$ \textit{(prove)}.

However, the expected running time is $O(n)$. In general, the running time in $d$-dimension is $O(d! \, n)$. That is, linear in any fixed (and small) dimension.