Warwick Mathematics Exchange

MA252

Linear Programming

2023, May 29th

Desync, aka The Big Ree

Contents

Introduction

This is a supplementary document to go alongside the main MA252 *Combinatorial Optimisation* module document.

Disclaimer: I make *absolutely no guarantee* that this document is complete nor without error. This document was written during the 2022 academic year, so any changes in the course since then may not be accurately reflected. Also, this module doesn't have any notes, so the structure of this document is based on (non-note) material from previous years.

Notes on formatting

Due to the nature of this module, I will be mixing mathematical and programming notation, a *lot*. obj.flag represents an instance attribute, flag, attached to an object, obj. In algorithm blocks, single equality (=) or left arrow ($\leftarrow$) represents variable assignment while double equality (==) represents an equality check. Setting a variable to "[ ]" indicates a *list* or an *array* being instantiated.

New terminology will be introduced in *italics* when used for the first time. Named theorems will also be introduced in *italics*. Important points will be **bold**. Common mistakes will be <u>underlined</u>. The latter two classifications are under my interpretation. YMMV.

Content not taught in the course will be outlined in the margins like this. Anything outlined like this is not examinable, but has been included as it may be helpful to know alternative methods to solve problems.

The table of contents above, and any inline references are all hyperlinked for your convenience.

History

First Edition: 2023-05-29*
Current Edition: 2023-06-01

Authors

This document was written by R.J. Kit L., a maths student. I am not otherwise affiliated with the university, and cannot help you with related matters.

Please send me a PM on Discord @Desync#6290, a message in the WMX server, or an email to Warwick.Mathematics.Exchange@gmail.com for any corrections. (If this document somehow manages to persist for more than a few years, these contact details might be out of date, depending on the maintainers. Please check the most recently updated version you can find.)

If you found this guide helpful and want to support me, you can buy me a coffee!

(Direct link for if hyperlinks are not supported on your device/reader: ko-fi.com/desync.)

*Storing dates in big-endian format is clearly the superior option, as sorting dates lexicographically will also sort dates chronologically, which is a property that little and middle-endian date formats do not share. See ISO-8601 for more details. This footnote was made by the computer science gang.

1 Linear Programming

A *linear program* is a problem of the form,

"Minimise $\mathbf{c} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$, subject to the *constraint* $\mathbf{Ax} \leq \mathbf{b}$,
where $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$, and the inequality is considered componentwise."

The vector $\mathbf{c}$ is then called the *cost vector* or the *objective function.*

The set,

$$\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} \leq \mathbf{b}\}$$

is called the *feasible region* of the linear program, as it contains every value of $\mathbf{x}$ that satisfies the constraint. The objective is then to find a value of $\mathbf{x}$ in the feasible region that yields a minimum value for $\mathbf{c} \cdot \mathbf{x}$.

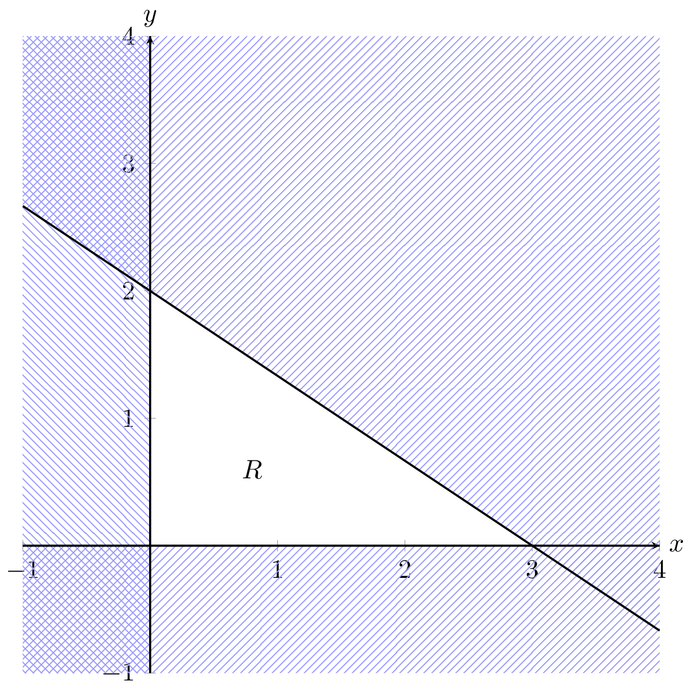
Example. Minimise $\begin{bmatrix} 1 \\ 2 \end{bmatrix} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^2$ satisfiying,

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 2 & 3 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix}$$

Writing $\mathbf{x} = [x,y]$, we can expand out the constraint into the system of equations,

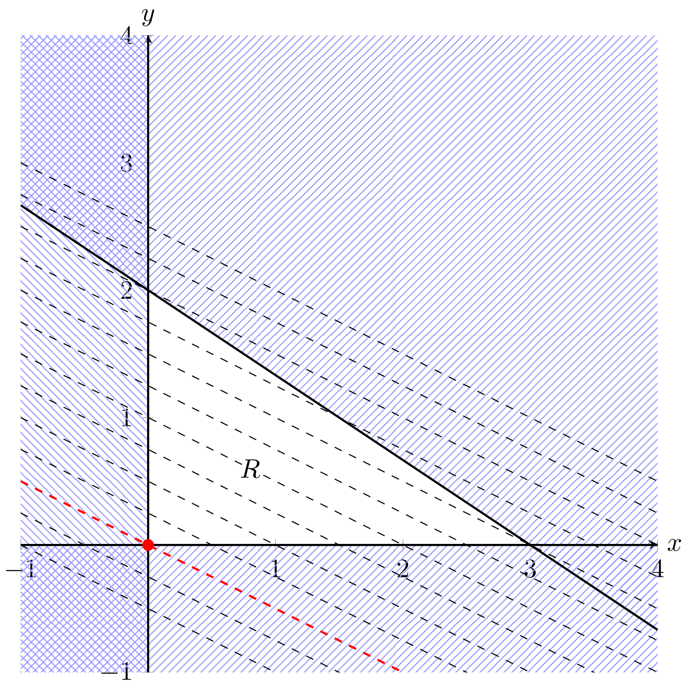
$$\begin{aligned} -x &\leq 0 \\ -y &\leq 0 \\ 2x + 3y &\leq 6 \end{aligned}$$

which define the feasible region R,



As a tip, it may be helpful to shade the unwanted region for each inequality so the feasible region is the only unshaded area left. Otherwise, you might have difficulty deciding which part is shaded by every inequality, especially for larger constraint matrices.

The objective function to minimise is then $x + 2y$, and we can picture various values of this function by looking at the lines $x + 2y = k$ for various values of k:



with k decreasing further down. Clearly, the objective function is minimised at the point $\mathbf{x} = (0,0)$, with value 0.

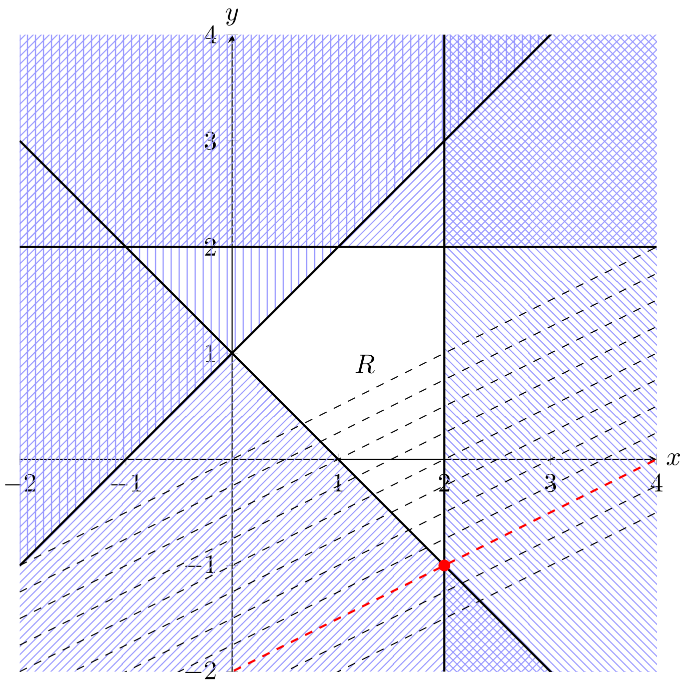
Example. Minimise, $\begin{bmatrix} -1 \\ 2 \end{bmatrix} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^2$ satisfiying,

$$\begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} -1 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

Again, writing $\mathbf{x} = [x,y]$, we can expand out the constraint into the system,

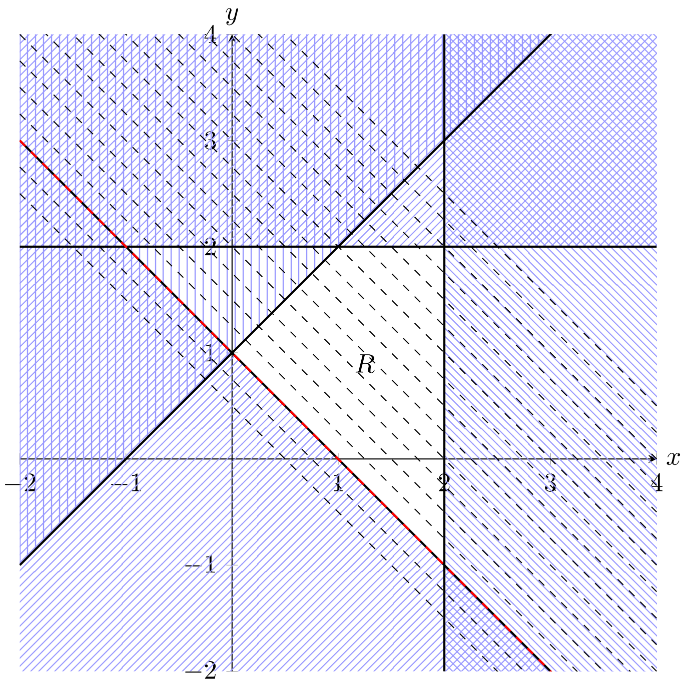
$$\begin{aligned} -x - y &\leq -1 \\ -x + y &\leq 1 \\ x &\leq 2 \\ y &\leq 2 \end{aligned}$$

with the objective function $-x + 2y = k$.



Again, the value of k decreases lower down, so the minimum is achieved at $(2, -1)$ with value -4.

Now, suppose the objective function was instead $x + y$:



Now, we have a whole line of minimal solutions, and any point on the line $x + y = 1$ between $x = 0$ and $x = 2$ is a minimal solution.

1.1 Polyhedra

A *hyperplane* is a subset of $\mathbb{R}^n$ of the form,

$$\{\mathbf{x} \in \mathbb{R}^n : \mathbf{n} \cdot \mathbf{x} = \mathbf{b}\}$$

where $\mathbf{n}$ is the normal vector of the plane, while a *halfspace* is a subset of the form,

$$\{\mathbf{x} \in \mathbb{R}^n : \mathbf{n} \cdot \mathbf{x} \leq \mathbf{b}\}$$

so the hyperplane given by the equality is the boundary of this set. Notice that we can write a dot product as,

$$H = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{n}^\top \mathbf{x} = \mathbf{b}\}$$

and we can then consider $\mathbf{n}$ as a $1 \times n$ matrix.

A *polyhedron* is the intersection of finitely many halfspaces, and a *polytope* is a bounded polyhedron, or equivalently, a polytope is the convex hull of a finite set.

The *face* of a polyhedron $P = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ that minimises some vector $\mathbf{c} \in \mathbb{R}^n$ is defined by,

$$\text{face}_{\mathbf{c}}(P) = \{\mathbf{x} \in P : \forall \mathbf{y} \in P, \mathbf{c} \cdot \mathbf{x} \leq \mathbf{c} \cdot \mathbf{y}\}$$

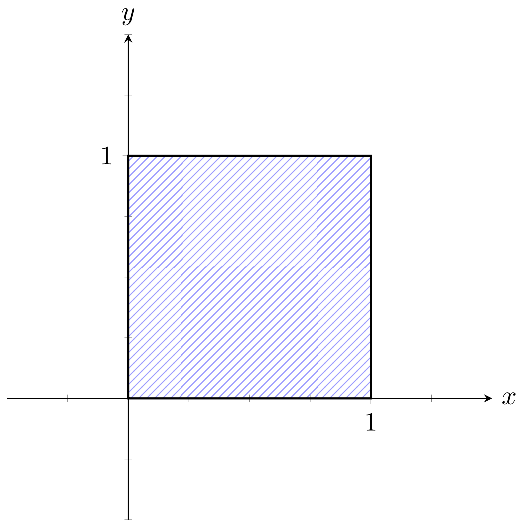
Example. Consider the polyhedron given by,

$$P = \big\{(x,y) \in \mathbb{R}^2 : 0 \leq x \leq 1, 0 \leq y \leq 1\big\}$$

What is the face minimising:

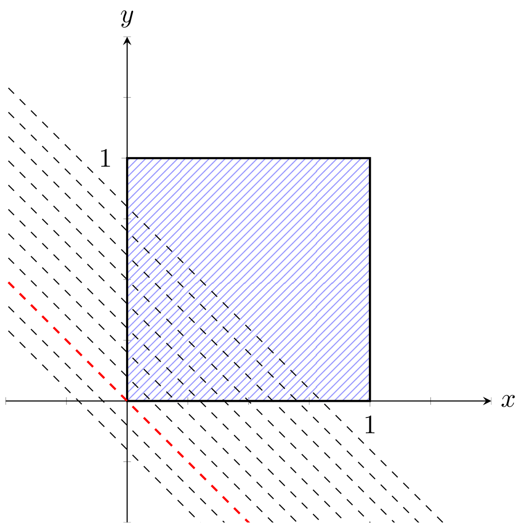
1. $\mathbf{c} = [1,1]$;
2. $\mathbf{c} = [-1,2]$;
3. $\mathbf{c} = [0,1]$?

P is given by,



$$\begin{aligned}\text{face}_{[1,1]}(P) &= \big\{(x,y) \in P : \forall (x',y') \in P, [1,1] \cdot [x,y] \leq [1,1] \cdot [x',y']\big\} \\ &= \big\{(x,y) \in P : \forall (x',y') \in P, x + y \leq x' + y'\big\}\end{aligned}$$

So, $\text{face}_{[1,1]}(P)$ is the set of points where the line $x + y = k$ intersects P for a minimal value of k:

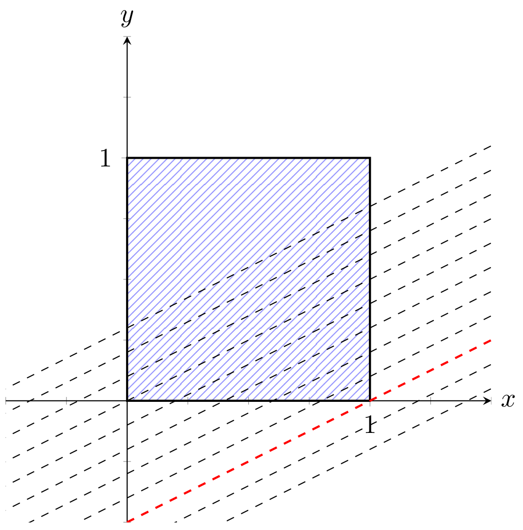


the minimal valued line intersects the polygon at (0,0), giving $\text{face}_{[1,1](P)} = \{(0,0)\}$.

Next, take $\mathbf{c} = [-1,2]$.

$$\begin{aligned}\text{face}_{[-1,2]}(P) &= \{(x,y) \in P : \forall (x',y') \in P, [-1,2] \cdot [x,y] \le [-1,2] \cdot [x',y']\} \\ &= \{(x,y) \in P : \forall (x',y') \in P, -x + 2y \le -x' + 2y'\}\end{aligned}$$

This time, we look at the line $-x + 2y = k$,

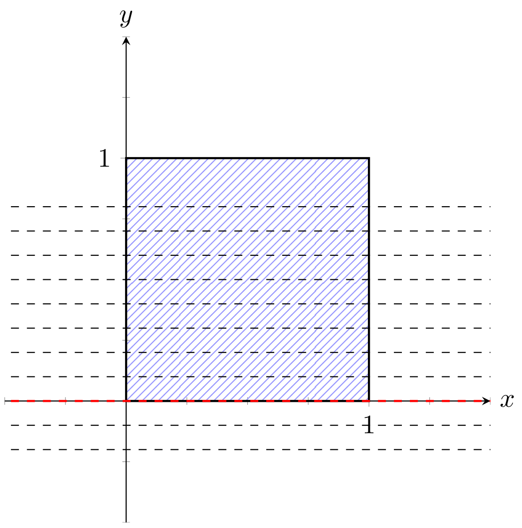


and now the minimal intersection point is at (1,0), so $\text{face}_{[-1,2]}(P) = \{(1,0)\}$

Next, take $\mathbf{c} = [0,1]$.

$$\begin{aligned}\text{face}_{[0,1]}(P) &= \{(x,y) \in P : \forall (x',y') \in P, [0,1] \cdot [x,y] \le [0,1] \cdot [x',y']\} \\ &= \{(x,y) \in P : \forall (x',y') \in P, y \le y'\}\end{aligned}$$

So the line is now $y = k$,



Now, the minimal intersection points are an entire line segment, so $\text{face}_{[-1,2]}(P) = \big\{(x,0) : 0 \leq x \leq 1\big\}$.

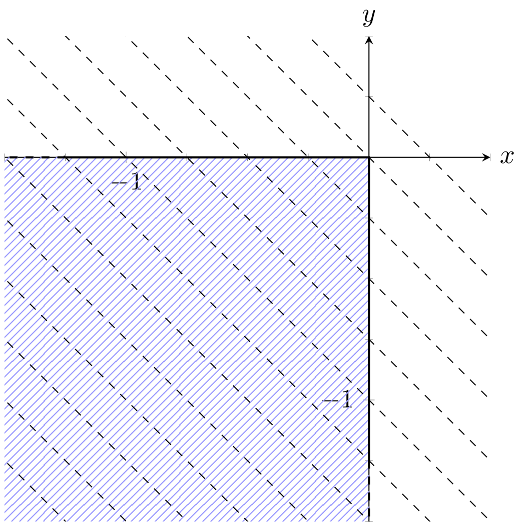
Example. Consider the polyhedron given by,

$$Q = \big\{(x,y) \in \mathbb{R}^2 : x \leq 0, y \leq 0\big\}$$

What is the face minimising $\mathbf{c} = [1,1]$?

$$\begin{aligned} \text{face}_{[1,1]}(Q) &= \big\{(x,y) \in Q : \forall (x',y') \in Q, [1,1] \cdot [x,y] \leq [1,1] \cdot [x',y']\big\} \\ &= \big\{(x,y) \in Q : \forall (x',y') \in Q, x + y \leq x' + y'\big\} \end{aligned}$$

We look at the line $x + y = k$:



so the minimum does not exist, and $\text{face}_{[1,1]}(Q) = \emptyset$.

We have been solving these problems by finding minimal points, so we can also characterise faces as,

$$\begin{aligned} \text{face}_{\mathbf{c}}(P) &= \{\mathbf{x} \in P : \forall \mathbf{y} \in P, \mathbf{c} \cdot \mathbf{x} \leq \mathbf{c} \cdot \mathbf{y}\} \\ &= \left\{\mathbf{A}\mathbf{x} \leq \mathbf{b} : \mathbf{c} \cdot \mathbf{x} = \min_{\mathbf{y} \in p} \mathbf{c} \cdot \mathbf{y}\right\} \end{aligned}$$

and we can see that $\text{face}_{\mathbf{c}}(P)$ is a polyhedron if the minimum exists, and is otherwise empty.

The linear program,

"Minimise $\mathbf{c} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{Ax} = \mathbf{b}$."

is equivalent to,

"Find $\mathbf{y} \in \text{face}_{\mathbf{c}}(P)$ where $P = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} \leq \mathbf{b}\}$ and compute $\mathbf{c} \cdot \mathbf{y}$."

The *linear span* of a polyhedron P is the subspace,

$$\text{span}(\{\mathbf{x} - \mathbf{y} : \mathbf{x}, \mathbf{y} \in P\})$$

of $\mathbb{R}^n$. The *dimension* of P is the dimension of its linear span.

Faces of dimension 0 are called *vertices*, and faces of dimension 1 are called *edges*.

1.2 Standard Form

Recall that a linear program is a problem of the form,

"Minimise $\mathbf{c} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{Ax} \leq \mathbf{b}$."

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$.

A linear program is in *standard form* if it can be written as,

"Minimise $\mathbf{c} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{Ax} = \mathbf{b}$, and $\mathbf{x} \geq \mathbf{0}$."

where $\mathbf{A} \in \mathbb{R}^{d \times n}$ and $\mathbf{b} \in \mathbb{R}^d$.

We can convert any linear program into standard form:

Algorithm 1 Linear Program Standard Form

```
1: Split each component $x_i$ of $\mathbf{x}$ into $x_i = x_i^+ - x_i^-$.
2: Add new variables to the inequality constraints to give $\mathbf{Ax} + \mathbf{s} = \mathbf{b}$.
3: Change the components $c_i$ of the cost vector $\mathbf{c}$ into $c_i = (c_i^+, -c_i^-)$, and set any components
   corresponding to slack variables to $c_s = 0$.
```

Example. Transform the following problem into standard form:

Minimise $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^2$ such that,

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

The constraint matrix gives,

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

Perform the replacements on the components, and add slack variables to remove the inequalities to obtain,

$$\begin{aligned} x_1^+ - x_1^- + x_2^+ - x_2^- + s_1 &= 1 \\ x_1^+ - x_1^- - x_2^+ + x_2^- + s_2 &= 1 \\ -x_1^+ + x_1^- + x_2^+ - x_2^- + s_3 &= 1 \\ -x_1^+ + x_1^- - x_2^+ + x_2^- + s_5 &= 1 \end{aligned}$$

which can be written in matrix form as,

$$\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 1 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 & 1 & 0 \\ -1 & 1 & -1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1^+ \\ x_1^- \\ x_2^+ \\ x_2^- \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Then, the original cost vector gives the constraint $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{x} = x_1$, so our new cost vector is,

$$\mathbf{c} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Example. Transform the following problem into standard form:

Minimise $\begin{bmatrix} 2 \\ 3 \end{bmatrix} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^2$ such that,

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

The constraint matrix gives,

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

so,

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 & 0 \\ -1 & 1 & -1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1^+ \\ x_1^- \\ x_2^+ \\ x_2^- \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

with cost vector given by,

$$\mathbf{c} = \begin{bmatrix} 2 \\ -2 \\ 3 \\ -3 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

To solve linear programs, we often look for faces of the feasible region. The standard form makes it easier to find vertices as solutions.

A vector $\mathbf{y} \in \mathbb{R}^n$ is a *basic solution* of a linear program if $\mathbf{A}\mathbf{y} = \mathbf{b}$ and the columns of A corresponding to non-zero entries of $\mathbf{y}$ are linear independent. That is, if $\mathbf{A} = [\mathbf{A}_1|\mathbf{A}_2|\cdots|\mathbf{A}_n]$ and $\mathbf{y} = [y_1,y_2,\ldots,y_n]$, then $\{A_i : y_i \neq 0\}$ is a linearly independent set.

A *basic feasible solution* is a basic solution $\mathbf{y} \in \mathbb{R}^n$ such that $\mathbf{y} \geq \mathbf{0}$.

We will write $\mathbf{A}_i$ for the ith column of a matrix $\mathbf{A}$, and $\mathbf{a}_i$ for the ith row of $\mathbf{A}$. That is,

$$\mathbf{A} = [\mathbf{A}_1|\mathbf{A}_2|\cdots|\mathbf{A}_n] = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_d \end{bmatrix}$$

Example. Minimise $\begin{bmatrix} 1 \\ 2 \\ 0 \\ -1 \end{bmatrix} \cdot \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^4$ such that $\mathbf{x} \geq \mathbf{0}$, and,

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 & -2 \\ 0 & 1 & 1 & -2 \end{bmatrix}}_{\mathbf{A}} \mathbf{x} = \underbrace{\begin{bmatrix} 3 \\ -2 \end{bmatrix}}_{\mathbf{b}}$$

$\mathbf{y} = [5,0,-2,0]$ is a basic solution since $\mathbf{A}\mathbf{y} = \mathbf{b}$ and the first and third columns of $\mathbf{A}$, are linearly independent, but it is not a basic feasible solution since $\mathbf{y}$ contains a negative component.

$\mathbf{y} = [5,0,0,1]$ is a basic feasible solution since $\mathbf{A}\mathbf{y} = \mathbf{b}$ and the first and last columns of $\mathbf{A}$, are linearly independent, and $\mathbf{y} \geq \mathbf{0}$.

To construct a basic solution, we choose d linearly independent columns $(\mathbf{A}_i)_{i\in\mathcal{I}}$ with $\mathcal{I} \subseteq [n]$ and $|\mathcal{I}| = d$, and set $y_i = 0$ for each $i \notin \mathcal{I}$. Augment these columns together into a $d\times d$ matrix $\mathbf{B} = [\mathbf{A}_{i_1}|\mathbf{A}_{i_2}|\cdots|\mathbf{A}_{i_d}]$.

Because the $(\mathbf{A}_i)$ are linearly independent, $\mathbf{B}$ is invertible, so $\mathbf{B}\mathbf{y} = \mathbf{b}$ has a unique solution given by

$$\mathbf{y}_\mathbf{B} = \mathbf{B}^{-1}\mathbf{b} \in \mathbb{R}^d$$

where the coordinates in $\mathbb{R}^d$ are indexed by $\mathcal{I}$. Then, we set,

$$\mathbf{y} = \begin{cases} (\mathbf{y}_B)_i & i \in \mathcal{I} \\ 0 & i \notin \mathcal{I} \end{cases}$$

That is, we invert a matrix made of linearly independent columns of $\mathbf{A}$, then add in a 0 entry to the solution vector wherever we skipped a column from $\mathbf{A}$.

Example. Consider the polygon defined by,

$$P = \left\{(x,y) \in \mathbb{R}^n : -x + y \leq 1, x + y \leq 3, x \geq 0, y \geq 0\right\}$$

This can be written in matrix form as,

$$\begin{bmatrix} -1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

noting that we did not have to split x and y as we are already given $x \geq 0$ and $y \geq 0$.

One linearly independent set is given by $\mathbf{A}_3$ and $\mathbf{A}_4$, so,

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and,

$$\begin{aligned} \mathbf{y_B} &= \mathbf{B}^{-1}\mathbf{b} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 3 \end{bmatrix} \end{aligned}$$

Since we skipped the first two columns, we have,

$$\mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 3 \end{bmatrix}$$

Because all the components are positive, this is a basic feasible solution.

Another linearly independent set is given by $\mathbf{A}_1$ and $\mathbf{A}_4$, so,

$$\mathbf{B} = \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix}$$

and,

$$\begin{aligned} \mathbf{y_B} &= \mathbf{B}^{-1}\mathbf{b} \\ &= \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \\ &= \begin{bmatrix} -1 \\ 4 \end{bmatrix} \end{aligned}$$

Columns 2 and 3 were omitted from $\mathbf{B}$, so we have,

$$\mathbf{y} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 4 \end{bmatrix}$$

This time, we have a negative component, so this basic solution is *not* feasible.

Theorem 1.1. *A vector* $\mathbf{v} \in \mathbb{R}^n$ *is a basic feasible solution of a linear program if and only if it is a vertex of the corresponding polyhedron defined by* $P = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$.

Theorem 1.2. *Either* $\min_{\mathbf{x} \in P} \mathbf{c} \cdot \mathbf{x} = -\infty$, *or there is a basic feasible solution* $\mathbf{y}$ *with* $\mathbf{c} \cdot \mathbf{y} \leq \mathbf{c} \cdot \mathbf{x}$ *for all* $\mathbf{x} \in P$.

2 The Simplex Algorithm

2.1 Geometric Simplex

Recall that an edge of a polyhedron is a one dimensional face. If a vector $\mathbf{y} \in P$ lies on an edge E, but does not lie on a vertex, then, up to scaling, there is a unique vector $\mathbf{d}$ such that $\mathbf{y} + \lambda \mathbf{d} \in E$ for sufficiently small λ. That is, $\mathbf{d}$ is the *direction vector* pointing along the edge.

Walking along an edge like this, we can eventually reach a vertex (assuming the edge is not a half-line that goes infinity in some direction). Because basic feasible solutions are vertices of polyhedra, this suggests an algorithm to find optimal basic feasible solutions:

1. Find a vertex of P.
2. Walk along edges of P, moving to vertices that reduce $\mathbf{c} \cdot \mathbf{x}$.
3. Stop when this isn't possible anymore.

Let $\mathbf{y} \in \mathbb{R}^n$ be a basic feasible solution obtained from the $d \times d$ matrix $\mathbf{B}_{\mathcal{I}} = [\mathbf{A}_{i_1} | A_{i_2} | \cdots | A_{i_d}]_{i \in \mathcal{I}}$ consisting of columns of $\mathbf{A}$, indexed by $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| = d$. Recall that $y_i = 0$ for all $i \notin \mathcal{I}$ by construction.

Denote by $\mathbf{y}_{\mathcal{I}}$ the vector obtained by restricting $\mathbf{y}$ to coordinates in $\mathbf{I}$. Then, $\mathbf{B}_{\mathcal{I}} \mathbf{y}_{\mathcal{I}} = \mathbf{b}$.

Let $k \notin \mathcal{I}$. We will look for a vector $\mathbf{d}$ with $\mathbf{d}_j = 0$ for all $\mathbf{j} \in \mathcal{I} \cup \{k\}$, $d_k = 1$, and $\mathbf{y} + \lambda \mathbf{d}$ is feasible for some $\lambda > 0$.

Then,

$$\begin{aligned} \mathbf{A}(\mathbf{y} + \lambda \mathbf{d}) &= \mathbf{b} \\ \mathbf{A}\mathbf{y} + \lambda \mathbf{A}\mathbf{d} &= \mathbf{b} \\ \mathbf{b} + \lambda \mathbf{A}\mathbf{d} &= \mathbf{b} \\ \lambda \mathbf{A}\mathbf{d} &= \mathbf{0} \end{aligned}$$

So $\mathbf{A}\mathbf{d} = \mathbf{0}$, and we can rewrite this as,

$$\mathbf{0} = \sum_{j=1}^{n} \mathbf{A}_j d_j$$

Recall that $d_j = 0$ for all $j \notin \mathcal{I} \cup \{k\}$, and $d_k = 1$, so,

$$\begin{aligned} &= \sum_{j \in \mathcal{I}} A_j d_j + A_k \\ &= \mathbf{B}_{\mathcal{I}} \mathbf{d}_{\mathcal{I}} + \mathbf{A}_k \end{aligned}$$

so $\mathbf{d}_{\mathcal{I}} = -\mathbf{B}_{\mathcal{I}}^{-1} \mathbf{A}_k$, giving,

$$d_j = \begin{cases} -\left(\mathbf{B}^{-1} \mathbf{A}_k\right)_j & j \in \mathcal{I} \\ 1 & j = k \\ 0 & j \notin \mathcal{I} \cup \{1\} \end{cases}$$

This vector $\mathbf{d}$ is called the *kth basic direction at* $\mathbf{y}$, and it depends on the choice of $\mathcal{I}$ and k.

Note that it is not always possible to find $\lambda > 0$ such that $\mathbf{y} + \lambda \mathbf{d}$ is feasible. That is, that $\mathbf{y} + \lambda \mathbf{d} \geq \mathbf{0}$.

A basic feasible solution $\mathbf{y}$ is *degenerate* if $|\{i : y_i \neq 0\}| < d$, and is *nondegenerate* otherwise.

Theorem 2.1. *If $\mathbf{y}$ is a nondegenerate basic feasible solution, then the kth basic direction is feasible.*

If this is the case, then $\mathbf{d}$ points along an edge, and we can move along it. Furthermore,

$$\mathbf{c} \cdot (\mathbf{y} + \lambda \mathbf{d}) = \mathbf{c} \cdot \mathbf{y} + \lambda \mathbf{c} \cdot \mathbf{d}$$

so the cost decreases if and only if $\mathbf{c} \cdot \mathbf{d} < 0$. We also have,

$$\begin{aligned} \mathbf{c} \cdot \mathbf{d} &= \sum_{j=1}^{n} c_j d_j \\ &= \sum_{j \in \mathcal{I}} c_j d_j + c_k \\ &= \mathbf{c}_{\mathcal{I}} \cdot \mathbf{d}_{\mathcal{I}} + c_k \\ &= \mathbf{c}_{\mathcal{I}} \cdot (-\mathbf{B}_{\mathcal{I}}^{-1} \mathbf{A}_k) + c_k \\ &= c_k - \mathbf{c}_{\mathcal{I}}^{\top} \mathbf{B}_{\mathcal{I}}^{-1} \mathbf{A}_k \end{aligned}$$

The *reduced cost* in direction k with respect to a basic feasible direction corresponding to $\mathcal{I}$ is given by $\bar{c}_i = c_i - \mathbf{c}_{\mathcal{I}}^{\top} \mathbf{B}_{\mathcal{I}}^{-1} \mathbf{A}_k$, and the *reduced cost vector* is given by $\bar{\mathbf{c}} = (\bar{c}_i)_{i=1}^{n}$

Note that if $\mathbf{y}$ is degenerate, then $\lambda^{\bullet}$ may be zero, and $\mathbf{y}' = \mathbf{y}$.

Theorem 2.2. *Let $\mathbf{y}$ be a basic feasible solution corresponding to $\mathcal{I} \subseteq [n]$, and let $\bar{\mathbf{c}}$ be the reduced cost vector. Then,*

- *If $\bar{\mathbf{c}} \geq \mathbf{0}$, then $\mathbf{y}$ is optimal.*
- *If $\mathbf{y}$ is optimal and nondegenerate, then $\bar{\mathbf{c}} \geq 0$.*

Algorithm 2 Geometric Simplex

```
1: Find a basic feasible solution $\mathbf{y}$ corresponding to $\mathcal{I} \subseteq [n]$.
2: Compute the reduced cost vector given by $\bar{c}_j = c_j - c_{\mathcal{I}}^{\top} \mathbf{B}_{\mathcal{I}}^{-1} \mathbf{A}_j$, where $c_j$ is the $j$th component of the
   cost vector $\mathbf{c}$. If $\bar{\mathbf{c}} \geq 0$, then $\mathbf{y}$ is optimal, and we may stop.
3: Choose $k \notin I$ with $\bar{c}_i < 0$, and compute the $k$th basic direction $\mathbf{d}$, given by
       $d_j = \begin{cases} -\left(\mathbf{B}^{-1}\mathbf{A}_k\right)_j & j \in \mathcal{I} \\ 1 & j = k \\ 0 & j \notin \mathcal{I} \cup \{1\} \end{cases}$
   If $\mathbf{d} \geq 0$, then the optimal cost is $-\infty$, and we may stop.
4: If $d_j < 0$ for some $j$, then let $\lambda^{\bullet} = \min_{d_j < 0} \frac{-y_j}{d_j}$, and let $\ell$ be the value of $j$ that achieves this
   minimum. That is, $\lambda^{\bullet} = \frac{-y_\ell}{d_\ell}$.
5: Set $\mathbf{y} = \mathbf{y} + \lambda^{\bullet}\mathbf{d}$ and $\mathcal{I} = (\mathcal{I} \setminus \{\ell\}) \cup \{k\}$. Go to step 2.
```

Example. TO DO (I want to die.)

See spchee's notes (partial example).

2.2 Graph Optimisation Problems

The following problems on graphs can all be expressed as linear programs:

1. MST (Minimum Spanning Tree);

2. SHORTEST-PATH;
3. MAX-FLOW;
4. MAXIMUM-MATCHING;

TO DO. See lecture 21 notes or spchee's notes.

2.3 Simplex Tableau

This algorithm is rather involved, so we begin with an annotated worked example to give an overview of the method.

Example. Minimise $P = \begin{bmatrix} 3 \\ -1 \end{bmatrix} \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^2$ subject to,

$$\begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ r \\ s \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix}$$

and x,y,r,$s \geq 0$. (Here, r and s are slack variables.)

The *initial tableau* is written as follows:

Basic variable	x	y	r	s	Value
r	2	1	1	0	12
s	1	4	0	1	8
P	3	-1	0	0	0

The first row of the table shows the first constraint, the second row shows the second constraint, and the final *objective row* shows the objective function.

The "Basic variable" column indicates the variables that are not currently at zero. We start at the vertex (0,0), so $x = y = 0$.

Any variables in a simplex variable that are not basic variables have the value 0.

If $x = y = 0$, then $r = 12$ from the first row of the constraint matrix, and similarly, $s = 8$. We currently therefore have,

$$\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 12 \\ 8 \end{bmatrix}$$

as our basic feasible solution with total value $P = 0$.

We scan the objective row of the tableau for the most negative number. This gives the *pivot column*. In this case, the pivot column is the y column.

For each other row, we then calculate a θ *value*, each given by dividing the value entry by the pivot entry.

Basic variable	x	y	r	s	Value	θ value
r	2	1	1	0	12	$12/1 = 12$
s	1	4	0	1	8	$8/4 = 2$
P	3	-1	0	0	0	

Next, we select the row containing the smallest positive θ value to be the *pivot row.*

Basic variable	x	y	r	s	Value	θ value
r	2	1	1	0	12	12
s	1	4	0	1	8	2
P	3	-1	0	0	0	

The entry at the intersection is then the *pivot.* We divide the values in the pivot row by the pivot, and replace the basic variable in the pivot row with the variable in the pivot column. In this case, s is replaced with y:

Basic variable	x	y	r	s	Value	Row operation
r	2	1	1	0	12	
y	$\frac{1}{4}$	1	0	$\frac{1}{4}$	2	$R2 \div 4$
P	3	-1	0	0	0	

Now use the pivot row to eliminate the pivot term from every other row:

Basic variable	x	y	r	s	Value	Row operation
r	$\frac{7}{4}$	0	1	$-\frac{1}{4}$	10	$R2 - R1$
y	$\frac{1}{4}$	1	0	$\frac{1}{4}$	2	
P	$\frac{13}{4}$	0	0	$\frac{1}{4}$	2	$R3 + R2$

There are no negative values in the objectie row, so the solution is optimal. We read the entries in the value column for each variable, to obtain $x = 0$, $y = 2$, $r = 10$, and $s = 0$, recalling that any variable not listed in the first column is 0. This gives the vector,

$$\mathbf{x} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

We also have $P = -2$ (the simplex tableau is a maximisation method, so we've actually maximised the negative of P, so we need to negate the final value).

Algorithm 3 Simplex Tableau

```
1: Draw the tableau(x) with a basic variable column on the left, one column for each variable (including
   slack variables), and a value column. Add a row for each constraint, and the bottom row for the
   objective function.
2: Enter the coefficients of the variables in the appropriate cells to form the initial tableau.
3: Find the most negative entry in the objective row to obtain the pivot column.
4: Calculate the $\theta$ values for each of the constraint rows, where $\theta$ is the value term divided by the pivot
   term.
5: Select the row with the smallest positive $\theta$ value to be the pivot row.
6: The element in the pivot row and pivot column is the pivot.
7: Divide the pivot row by the pivot, and change the basic variable in the first column to the variable
   at the top of the pivot column.
8: Use the pivot row to eliminate the pivot variable from other rows.
9: Repeat steps 3 to 8 until there are no negative values in the objective row.
10: The tableau is now optimal, and the non-zero values can be read off using the basic variable and
    value columns. If the objective function is to be minimised, take the negative of the objective value.
```