
SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" University of Rome
Faculty of Information Engineering,
Informatics and Statistics
Department of Computer Science

# Optimization

Lecture notes integrated with the book "Understanding and Using Linear Programming.", J. Matoušek, B. Gärtner

*Author*
Simone Bianco

May 28, 2024

# Contents

# Information and Contacts

Personal notes and summaries collected as part of the *Optimization* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
**https://github.com/Exyss/university-notes**. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:


- Email: **bianco.simone@outlook.it**
- LinkedIn: **Simone Bianco**


The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

**Suggested prerequisites:**

Preventive learning of material related to the *Linear Algebra* and *Algorithms 2* course is recommended

**Licence:**

# 1 Flow networks

## 1.1 Networks and flows

**Definition 1: Graph**

A **graph** is a mathematical structure $G = (V, E)$, where $V$ is the set of **vertices** in $G$ and $E \subseteq V \times V$ is the set of **edges** that link two vertices in $G$.

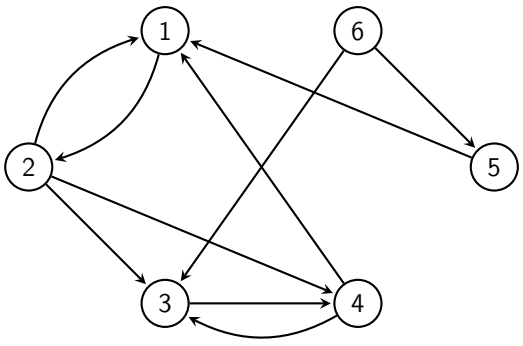
We will assume that each graph is *simple*, meaning that there are no multiple edges between the same nodes and no loops (that being an edge from a vertex to itself).

From now on, we will assume that $|V(G)| = n$ and $|E(G)| = m$.

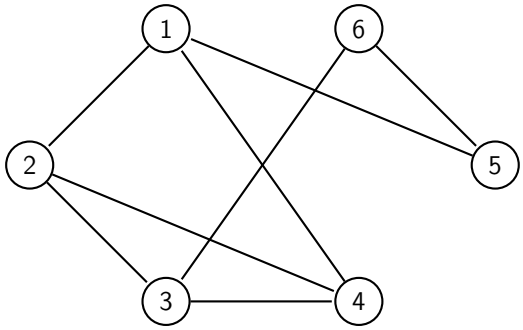
A graph can be **directed** or **undirected**. In a directed graph we consider the edges $(u, v) \in E(G)$ and $(v, u) \in E(G)$ as two different edges, while in an undirected graph they represent the same edge.

**Example:**

**Directed graph**




**Undirected graph**

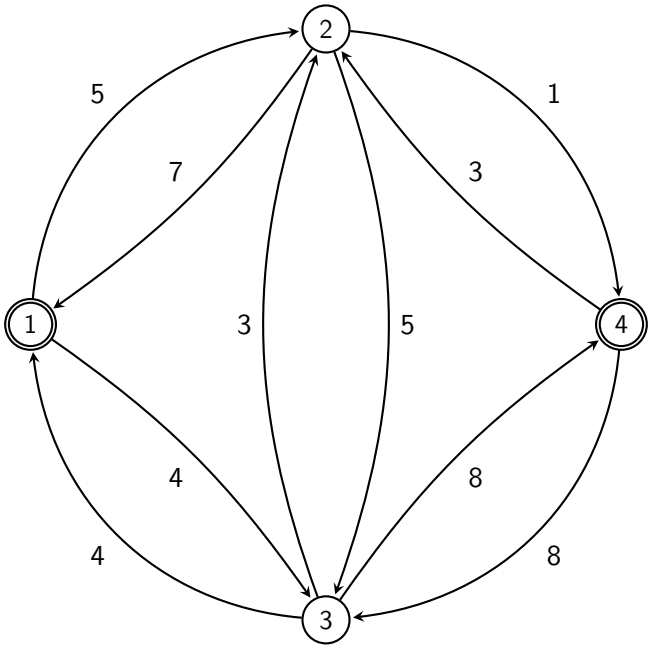


**Definition 2: Network**

A **network** is a mathematical structure $N = (G, s, t, c)$ where:

- $G = (V, E)$ is a directed graph
- $s$ and $t$ are two vertices of $G$ ($s, t \in V(G)$), respectively called the **source** and the **sink**
- $c : E(G) \to \mathbb{R}^+$ is a weight function on the edges called **capacity**
- $(u, v) \in E(G) \implies (v, u) \in E(G)$

**Example:**




*The numbers on the edges represent the capacities of the edges, while the nodes 1 and 4 are chosen respectively as the source and the sink of the network*

Essentially, a network effectively describes a *water system* made up of *pipes* (the edges) that can transport a maximum amount of fluid inside them (the capacity of the edges). In fact, the last property of a network defines the idea of a bi-directional *flow of water* inside the pipes. In particular, we note that each pipe can have a different capacity based on the direction of the flow.

**Definition 3: Flow**

Given a network $N = (G, s, t, c)$, a **flow** is a weight function $f : E(G) \to \mathbb{R}$ on the edges defined by the following properties:

- **Skew-symmetric**: $\forall (u, v) \in E(G) \quad f(u, v) = -f(v, u)$, meaning that the incoming flow in an edge is the inverse of the outgoing flow in the corresponding opposite edge
- **Capacity bounded**: $\forall (u, v) \in E(G) \quad f(u, v) \leq c(u, v)$, meaning that the flow can't be greater than the supported capacity
- **Conservation of flow**: $\forall v \in V(G) - \{s, t\}$ it holds that

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u, v) = \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(v, w)$$

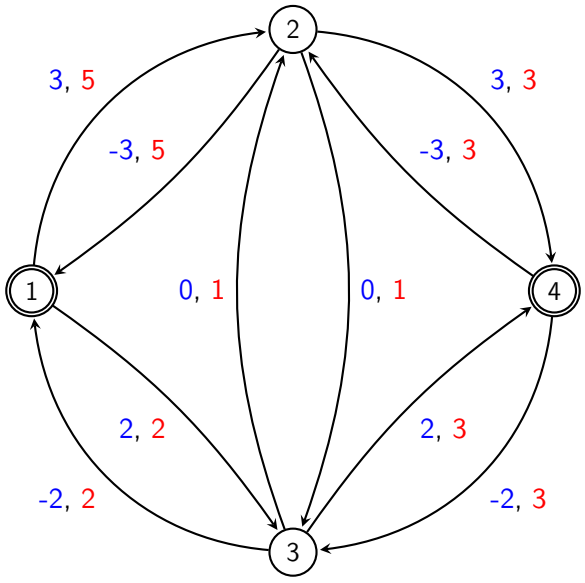
  meaning that the incoming flow in $v$ is the same as the outgoing flow from $v$

**Definition 4: Flow value**

Given a network $N = (G, s, t, c)$ and a flow $f$ on $N$, we define the **value of** $f$, noted by $\text{val}(f)$, as the sum of the flow outgoing from the source $s$ or the sum of the flow incoming to the sink $t$:

$$\text{val}(f) := \sum_{(s,u)\in E(G)} f(s, u) = \sum_{(v,t)\in E(G)} f(v, t)$$

**Example:**




*For each edge, the numbers in blue and red respectively represent its flow and its capacity. The flow value of the given flow is 5.*

**Observation 1: Nullification of flow for middle edges**

Given a network $N = (G, s, t, c)$ and a flow $f$ defined on $G$, for each vertex different from $s$ and $t$ it holds that:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) = \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(v,w) \iff \sum_{(u,v)\in E(G)} f(u,v) = 0$$

*Proof.*

Due to the conservation of flow and the skew-symmetric properties, for all nodes $v \neq s, t$ we know that:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) = \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(v,w) = \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} -f(w,v)$$

which is possible if only if:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) = - \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(w,v) \iff$$

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) + \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(w,v) = 0$$

Again, by the skew-symmetric property we get that:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) + \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(w,v) = 0 \iff$$

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) + \sum_{\substack{(w,v)\in E(G):\\ f(w,v)<0}} f(w,v) = 0$$

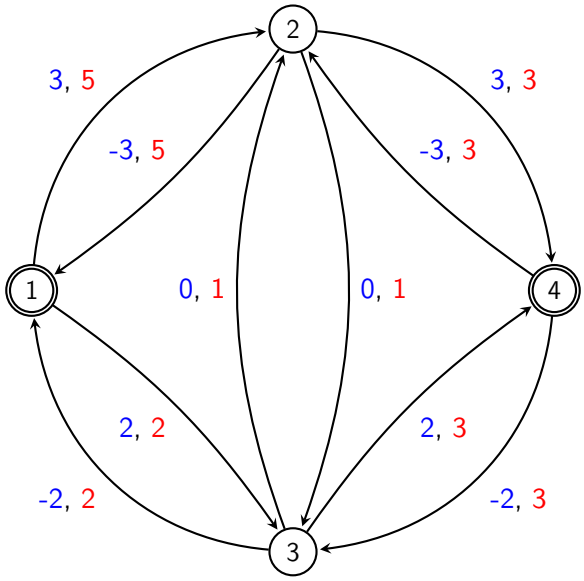
Then, by adding each edge incoming in $v$ that has no flow we conclude that:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) + \sum_{\substack{(w,v)\in E(G):\\ f(w,v)<0}} f(w,v) + \sum_{\substack{(x,v)\in E(G):\\ f(x,v)=0}} f(x,v) = 0 \iff \sum_{(u,v)\in E(G)} f(u,v) = 0$$

□

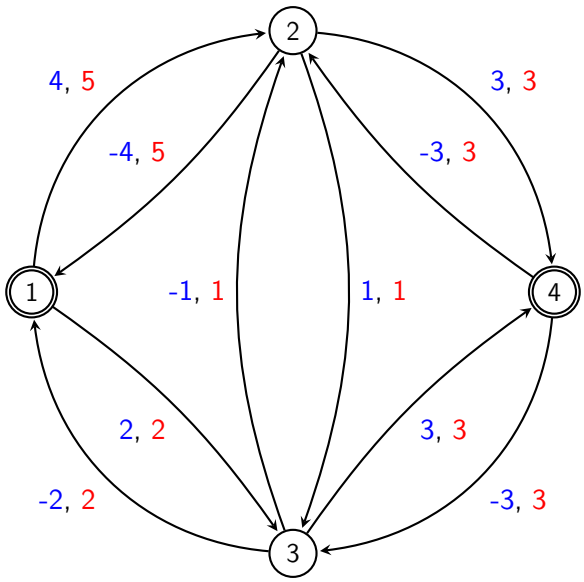
## 1.2 Residual graphs, flow increase and $st$-cuts

Consider the network shown in the last example of the previous section.




We notice that some *pipes* aren't completely *"filled up"*, meaning that their capacity could support a bigger flow. In particular, due to conservation of flow, not all pipes can be filled to the maximum capacity. In fact, we know that flow outgoing from the source must still be equal to the incoming flow of the sink.

Thus, we can increase the flow value by 1 only by using the remaining capacities in the path $1 \to 2 \to 3 \to 4$




*The flow value of the new flow is 6.*

**Definition 5: Residual capacity**

Given a network $N = (G, s, t, c)$ and a flow $f$ on $N$, the **residual capacity** is a weight function $r : E(G) \to \mathbb{R}^+$ defined as:

$$r(u, v) = c(u, v) - f(u, v)$$

**Observation 2**

Given a network $N = (G, s, t, c)$ and a flow $f$ on $N$, we note that:

$$r(u, v) + r(v, u) = c(u, v) + c(v, u)$$

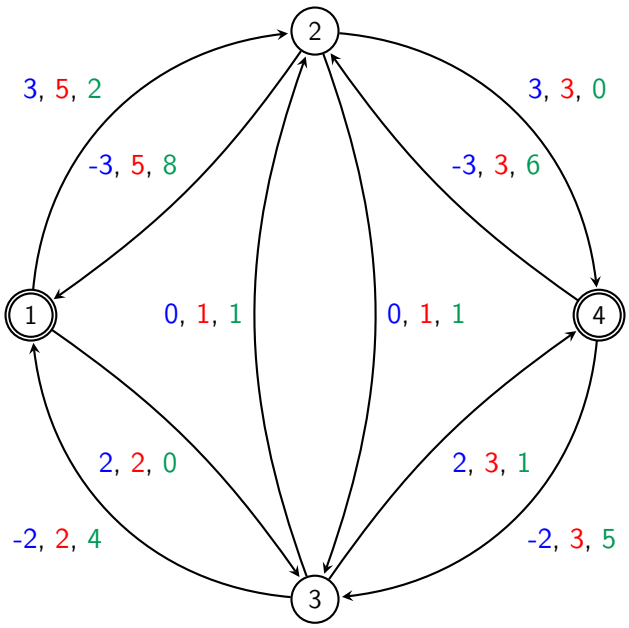
**Definition 6: Residual graph**

Given a network $N = (G, s, t, c)$ and a flow $f$ on $N$, we define $R \subseteq G$ as the **residual graph** of $G$ if:

$$(u, v) \in E(R) \iff r(u, v) > 0$$

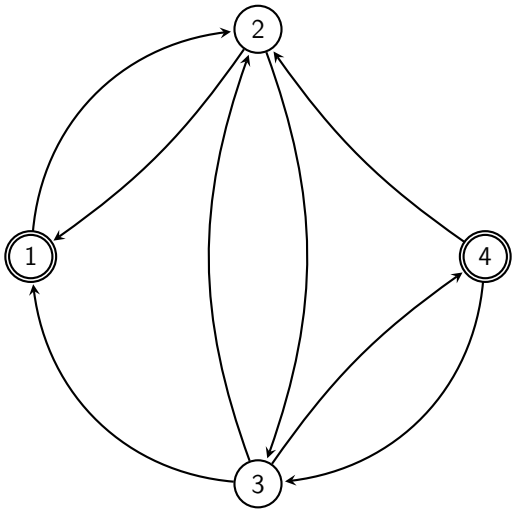
**Example:**

Consider the first graph previously shown with flow value 5. We now add the residual capacities obtained through that flow.




*For each edge, the number in green represents its residual capacity*

Thus, the residual graph is the following:




**Proposition 1: Flow-augmenting path**

Given a network $N = (G, s, t, c)$ and a flow $f$ on $N$, let $R \subseteq G$ be the residual graph of $G$ on $f$ and let $P$ be a direct path $s \to t$ in $R$.

Given $\alpha := \min_{(u,v)\in E(P)} r(u, v)$, we define $f' : E(G) \to R$ as:

$$f'(u,v) = \begin{cases} f(u,v) + \alpha & \text{if } (u,v) \in E(P) \\ f(u,v) - \alpha & \text{if } (v,u) \in E(P) \\ f(u,v) & \text{otherwise} \end{cases}$$

The function $f'$ is a flow for $N$ such that $\text{val}(f') = \text{val}(f) + \alpha$.

*Proof.*

Suppose that $(u, v) \in E(P)$:

- By using the skew-symmetric property of $f$, we get that:

$$f'(u,v) = f(u,v) + \alpha \implies -f'(u,v) = -f(u,v) - \alpha = f(v,u) - \alpha$$

  Also, since $(u, v) \in E(P)$, for $(v, u)$ we get that

$$f'(v,u) = f(v,u) - \alpha = -f'(u,v)$$

- By assumption, we know that $f'(u, v) = f(u, v) + \alpha$. Thus, by definition of $\alpha$ we get that:

$$\alpha \leq r(u,v) = c(u,v) - f(u,v) \implies$$
$$f'(u,v) = f(u,v) + \alpha \leq f(u,v) + c(u,v) - f(u,v) = c(u,v)$$

If $(v, u) \in E(P)$, we can get the same result by repeating the same steps. Otherwise, the flow remains unchanged, meaning that the property is already satisfied. Thus, we conclude that $f'$ is *skew-symmetric* and *capacity bounded*.

Given a vertex $x \in E(G)$, if $x \notin E(P)$ then the flows of its edges are unchanged. If $x \in E(P)$, we procede by splitting the edges in $G$ through $P$:

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0}} f'(u,x) = \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (u,x)\in E(P)}} f'(u,x) + \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (x,u)\in E(P)}} f'(u,x) =$$

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (u,x)\in E(P)}} [f(u,x)+\alpha] + \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (x,u)\in E(P)}} [f(u,x)-\alpha]$$

We notice that the amount of vertices in these sums is the same, implying that each time $\alpha$ gets summed in the first sum it also gets subtracted from the other sum:

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (u,x)\in E(P)}} [f(u,x)+\alpha] + \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (x,u)\in E(P)}} [f(u,x)-\alpha] =$$

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (u,x)\in E(P)}} f(u,x) + \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (x,u)\in E(P)}} f(u,x) = \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0}} f(u,x)$$

By using the same argument, we get that:

$$\sum_{\substack{(x,w)\in E(G):\\ f'(x,w)>0}} f'(x,w) = \sum_{\substack{(x,w)\in E(G):\\ f'(x,w)>0}} f(x,w)$$

Finally, through the the *conservation of flow* of $f$, we conclude that $f'$ also satisfies such property:

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0}} f'(u,x) = \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0}} f(u,x) = \sum_{\substack{(x,w)\in E(G):\\ f'(x,w)>0}} f(x,w) = \sum_{\substack{(x,w)\in E(G):\\ f'(x,w)>0}} f'(x,w)$$

□

**Definition 7: $st-$cut**

Given a network $N = (G, s, t, c)$ and a flow $f$ on $N$, we define a ***st*-cut of** $G$ as a subset $\mathcal{U} \subseteq V(G)$ that makes a partition on $V(G)$ such that $s \in \mathcal{U}, t \notin \mathcal{U}$.

Additionally, the vertices inside of $\mathcal{U}$ are called the *s-part* of the cut, while the vertices outside of $\mathcal{U}$ are called the *t-part* of the cut.

**Proposition 2: Flow of an $st$-cut**

Given a network $N = (G, s, t, c)$, a flow $f$ on $N$ and an $st$-cut $\mathcal{U} \subseteq G$, we have that:

$$\text{val}(f) = \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}, v\notin\mathcal{U}}} f(u,v)$$

*Proof.*

Consider the total sum of the flows of all the vertices in $\mathcal{U}$:

$$\sum_{x\in\mathcal{U}} \sum_{(u,v)\in E(G)} f(u,v)$$

By definition, we know that $s \in \mathcal{U}$. We can separate the flows outgoing from $s$ from the rest of the flows:

$$\sum_{x\in\mathcal{U}} \sum_{(u,v)\in E(G)} f(u,v) = \sum_{(s,w)\in E(G)} f(s,w) + \sum_{x\in\mathcal{U}-\{s\}} \sum_{(u,v)\in E(G)} f(u,v)$$

Due to the Nullification of flow for middle edges, for all vertices $u, v \neq s$ we know that the total flow is qual to 0, implying that:

$$\sum_{(s,w)\in E(G)} f(s,w) + \sum_{x\in\mathcal{U}-\{s\}} \sum_{(u,v)\in E(G)} f(u,v) = \sum_{(s,w)\in E(G)} f(s,w) + 0 = \text{val}(f)$$

We now consider again the initial total sum. We notice that:

$$\sum_{x\in\mathcal{U}} \sum_{(u,v)\in E(G)} f(u,v) = \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}}} f(u,v)$$

Additionally, for any $(u, v)$ if $u, v \in \mathcal{U}$ then $f(u, v)$ cancels out with $f(v, u)$, implying that:

$$\sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}}} f(u,v) = \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}, v\notin\mathcal{U}}} f(u,v)$$

By combining the shown equalities, we conclude the proof.

□

**Definition 8: Capacity of an $st$-cut**

Given a network $N = (G, s, t, c)$, a flow $f$ on $N$ and an $st$-cut $\mathcal{U} \subseteq G$, we define the **capacity of an st− cut on** $G$, noted by $c(V(G)\backslash\mathcal{U})$, as the sum of the capacities of the edges outgoing from *s-part* to the *t-part*.

$$c(V(G)\backslash\mathcal{U}) := \sum_{\substack{(u,v)\in E(G):\\ x\in\mathcal{U}, u\notin\mathcal{U}}} c(u,v)$$

**Lemma 1**

Given a network $N = (G, s, t, c)$, the maximum value of a flow on $G$ at most the minimal capacity of an $st$-cut on $G$:

$$\max_{f\,:\,\text{flow on } G}(\text{val}(f)) \leq \min_{\mathcal{U}\,:\,st-\text{cut on } G}(c(V(G)\backslash\mathcal{U}))$$

*Proof.*

Let $f$ be the flow on $G$ that maximizes $\text{val}(f)$ and let $\mathcal{U} \subseteq G$ be the $st$-cut on $G$ that minimizes capacity. By the Flow of an $st$-cut and by the *capacity bounded* property of $f$, we get that:

$$\text{val}(f) = \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}, v\notin\mathcal{U}}} f(u,v) \leq \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}, v\notin\mathcal{U}}} c(u,v) = c(V(G)\backslash\mathcal{U})$$

□

## 1.3 The Ford-Fulkerson algorithm

**Algorithm 1: The Ford-Fulkerson algorithm**

Given a network $N = (G, s, t, c)$, we define the following algorithm:

**function** FordFulkerson($G$)
  Start with the trivial flow $f$ with all 0s
  **while** True **do**
    Compute the residual graph $R \subseteq G$ on flow $f$
    Find a path $P$ in $R$ from $s \to t$
    **if** $P$ does not exist **then**
      Return $f$
    **else**
      Increase $f$ through the value $\alpha$ obtained by $P$
    **end if**
  **end while**
**end function**

We note that the FordFulkerson($N$) terminates only if the augment value eventually becomes 0, implying that there is no flow-augmenting path to be used. Thus, if the capacities defined by $c$ are all integers, the algorithm always terminates.

Moreover, the flow-augmenting path of each iteration of FordFulkerson($N$) can be found with a simple DFS search, requiring $O(n+m)$, which is also the cost needed for computing the residual graph of each iteration. Thus, the **computational complexity** of the algorithm is $O(k(n + m))$, where $k$ is the maximum number of iterations of the while loop.

It's easy to notice that this value $k$ **depends too much on the shape of the graph** $G$. However, due to the *capacity bounded* property, in the worst case we have that $k$ equals the maximum flow value. We also notice that, technically, the cost of this algorithm is **exponential**: the number of bits required to store $k$ are $\log_2 k$, but the cost relies on $2^{\log_2 k} = k$ iterations.

**Lemma 2**

Given a network $N = (G, s, t, c)$, if the algorithm FordFulkerson($N$) terminates then it returns a flow $f$ such that there exists an $st$-cut $\mathcal{U} \subseteq G$ for which we have that $\text{val}(f) = c(V(G) \backslash \mathcal{U})$

*Proof.*

Let $R$ be the residual graph of $G$ on $f$ = FordFulkerson($N$) and let $r$ be the residual capacity function obtained through $f$. We define $\mathcal{U} \subseteq G$ as:

$$\mathcal{U} = \{x \in V(G) \mid \exists s \to x \text{ in } G'\}$$

Since the algorithm terminates when there is no path from $s \to t$, we know that $t \notin \mathcal{U}$, implying that $\mathcal{U}$ is an $st$-cut of $G$. Thus, by the Flow of an $st$-cut, we have that:

$$\text{val}(f) = \sum_{\substack{(x,v)\in E(G):\\ x\in\mathcal{U}, v\notin\mathcal{U}}} f(x,u)$$

By way of contradiction, we suppose that $\exists (x,v) \in E(G') \subseteq E(G)$ such that $x \in \mathcal{U}, v \notin \mathcal{U}$. By definition of $\mathcal{U}$, we know that $s \to x$ in $G'$, so by adding $(x,v)$ to the path we get that $s \to x \to y$, which contradicts $y \notin \mathcal{U}$.

Thus, such edge can't exists, implying that $\forall (x,v) \in E(G)$ such that $x \in \mathcal{U}, v \notin \mathcal{U}$ it holds that $(x,v) \notin E(G')$, which by definition of residue graph implies that:

$$\forall (x,v) \in E(G) \text{ s.t. } x \in \mathcal{U}, v \notin \mathcal{U} \quad f(x,v) = c(x,v)$$

concluding that:

$$\text{val}(f) = \sum_{\substack{(x,v)\in E(G):\\ x\in\mathcal{U}, v\notin\mathcal{U}}} f(x,u) = \sum_{\substack{(x,v)\in E(G):\\ x\in\mathcal{U}, v\notin\mathcal{U}}} c(x,u) = c(V(G)\backslash\mathcal{U})$$

□

### 1.3.1 The Max-flow/Min-cut theorem

**Theorem 1: Max-flow/Min-cut theorem**

Given a network $N = (G, s, t, c)$, the maximum value of a flow on $G$ at most the minimal capacity of an $st$-cut on $G$:

$$\max_{f\,:\,\text{flow on } G}(\text{val}(f)) = \min_{\mathcal{U}\,:\,st-\text{cut on } G}(c(V(G)\backslash\mathcal{U}))$$

*Proof.*

By the Lemma 1, we already know that:

$$\max_{f\,:\,\text{flow on } G}(\text{val}(f)) \leq \min_{\mathcal{U}\,:\,st-\text{cut on } G}(c(V(G)\backslash\mathcal{U}))$$

We now consider $f' = \textsf{FordFulkerson}(N)$. By the Lemma 2, we know that there exists an $st$-cut $\mathcal{U}' \subseteq G$ such that $\text{val}(f') = c(V(G)\backslash\mathcal{U}')$.

Thus, it's easy to conclude that:

$$\max_{f\,:\,\text{flow on } G}(\text{val}(f)) \geq \text{val}(f') = c(V(G)\backslash\mathcal{U}) \geq \min_{\mathcal{U}\,:\,st-\text{cut on } G}(c(V(G)\backslash\mathcal{U}))$$

□

**Corollary 1: Optimality of Ford-Fulkerson**

The Ford-Fulkerson algorithm returns a flow with **maximum value**

## 1.4 The Edmonds-Karp algorithm

**Algorithm 2: The Edmonds-Karp algorithm**

Given a network $N = (G, s, t, c)$, we define the following algorithm:

**function** EDMONDSKARP($N$)
  Start with the trivial flow $f$ with all 0s
  **while** True **do**
    Compute the residual graph $R \subseteq G$ on flow $f$
    Find the shortest path $P$ in $R$ from $s \to t$
    **if** $P$ does not exist **then**
      Return $f$
    **else**
      Increase $f$ through the value $\alpha$ obtained by $P$
    **end if**
  **end while**
**end function**

**Lemma 3**

Given a network $N = (G, s, t, c)$, if the algorithm EdmondsKarp($N$) terminates then it returns a flow $f$ such that there exists an $st$-cut $\mathcal{U} \subseteq G$ for which we have that $\text{val}(f) = c(V(G) \backslash \mathcal{U})$

(*proof identical to* Lemma 2)

**Corollary 2: Optimality of Edmonds-Karp**

The Edmonds-Karp algorithm returns a flow with **maximum value**

The Edmonds-Keep algorithm looks identical to the Ford-Fulkerson algorithm, except for the type of path searched at each iteration. The idea behind finding the shortest path instead of a random path is to **limit** the number of iterations made by the algorithm by making them rely on the number of nodes instead of the maximum flow value. Thus, the **computational complexity** of the algorithm effectively becomes $O(mn(n + m))$, meaning that it's not an exponential algorithm. The following statements will be necessary to justify this result.

**Observation 3**

Given a graph $G$ and two vertices $x, y \in V(G)$, let $P$ be the shortest path from $x \to y$. Then, for each node $z_i \in V(P)$ the path $P$ contains the shortest path $P_i$ from $x \to z_i$

*Proof.*

For each node $z_1, \dots, z_k \in V(P)$ ($x$ and $y$ included), let $P_i$ be the shortest path from $x \to z_i$. By way of contradiction, suppose that $P_i \not\subseteq P$.

Given an index $i$ such that $1 \leq i \leq k$, let $P_x, P_y \subseteq P$ be the sub-paths that partition $P$ by $z_i$, meaning that $x, z_1, \dots, z_i \in V(P_x)$ and $z_{i+1}, \dots, z_k, y \in V(P_y)$.

Since by assumption $P_i$ is shorter that $P_x$, we get that the path $P_i \cup P_y$ is a path from $x \to y$ that is shorter than $P$, which is a contradiction. Thus, we conclude that for each node $z_1, \dots, z_k \in V(P)$ it holds that $P_i \subseteq P$.

□

**Proposition 3: Disappearing and appearing edges**

Given a network $N = (G, s, t, c)$ and the computation $\mathsf{EdmondsKarp}(N)$, let:

- $f_0, \dots, f_k$ be the series of flows computed, where $f_0$ is the trivial flow
- $R_0, \dots, R_k$ be the series of residue graphs computed, where $R_0$ is obtained with flow $f_i$
- $P_0, \dots, P_k$ be the series of shortest paths from $s \to t$ computed on $R_i$

Then, $\forall u \in V(G)$ and for each index $i = 1, \dots, k$ it holds that:

$$(u, v) \in E(R_i), (u, v) \notin E(R_{i+1}) \implies (u, v) \in E(P_i)$$

and that:

$$(u, v) \notin E(R_i), (u, v) \in E(R_{i+1}) \implies (v, u) \in E(P_i)$$

*Proof.*

If $(x, y) \in E(R_i)$ but $(x, y) \notin E(R_{i+1})$, the edge got removed by the flow-increase of path $P_i$. This can only happen only if $c(x, y) = f_{i+1}(x, y) = f_i(x, y) + \alpha_i$, where $\alpha_i$ is the flow increase obtained by $P_i$, meaning that $(x, y) \in P_i$.

Instead, if $(x, y) \notin E(R_i)$ but $(x, y) \in E(R_{i+1})$, the edge got added by the flow-increase of path $P_i$. This can happen only if $f_i(x, y) \geq f_{i+1}(y, x)$, which in turn can happen only if the flow of the opposite edge $(x, y)$ got increased, meaning that $(y, x) \in E(P_i)$.

□

**Lemma 4: Monotone increasing distance in Edmonds-Keep**

Given a network $N = (G, s, t, c)$ and the computation $\mathsf{EdmondsKarp}(N)$, let:

- $f_0, \ldots, f_k$ be the series of flows computed, where $f_0$ is the trivial flow
- $R_0, \ldots, R_k$ be the series of residue graphs computed, where $R_0$ is obtained with flow $f_i$
- $P_i, \ldots, P_k$ be the series of shortest paths from $s \to t$ computed, where $P_i \subseteq R_i$

Then, $\forall u \in V(G)$ and for each index $i = 1, \ldots, k$ it holds that:

$$\text{dist}_{R_i}(s, u) \leq \text{dist}_{R_{i+1}}(s, u)$$

*Proof.*

By way of contradiction, we assume that there exist some vertices for which the statement doesnt hold, meaning that $\exists u_1, \ldots, u_k \in V(G)$ such that $\text{dist}_{R_i}(s, u_j) > \text{dist}_{R_{i+1}}(s, u_j)$.

Let $v$ be the vertex picked from $u_1, \ldots, u_k$ with minimal distance from $s$ in $R_i$, meaning that:

$$\text{dist}_{R_i}(s, v) = \min_{j=1}^{k} \text{dist}_{R_i}(s, u_j)$$

Consider the shortest path $P$ in $R_{i+1}$ from $s \to v$, that being the path with length $\text{dist}_{R_{i+1}}(s, v)$. Let $u \in V(P)$ be the vertex that precedes $v$ in $P$, meaning that $(u, v) \in E(P)$.

Since $v$ is the vertex from $u_1, \ldots, u_k$ with the shortest distance and since $\text{dist}_{R_{i+1}}(s, u) = \text{dist}_{R_{i+1}}(s, v) - 1$ due to it being the previous vertex of $v$ in $P$, it must hold that $u \notin \{u_1, \ldots, u_k\}$ because otherwise $u$ would be the one with the shortest distance.

Thus, we get that $\text{dist}_{R_i}(s, u) \leq \text{dist}_{R_{i+1}}(s, u)$, which implies that:

$$\text{dist}_{R_i}(s, u) \leq \text{dist}_{R_{i+1}}(s, u) = \text{dist}_{R_{i+1}}(s, v) - 1$$

Suppose now that $(u, v) \in E(R_i)$. Given the shortest path $P'$ in $R_i i$ from $s \to u$ can be extended with $(u, v)$, we get that $\text{dist}_{R_i i}(s, v) \leq \text{dist}_{R_i i}(s, u) + 1$. However, this would imply that:

$$\text{dist}_{R_i}(s, v) \leq \text{dist}_{R_i}(s, u) + 1 \leq \text{dist}_{R_{i+1}}(s, u) + 1 \leq= \text{dist}_{R_{i+1}}(s, v)$$

which contradicts the initial assumption. Thus, it can't be that $(u, v) \in E(R_i)$.

Moreover, since $(u, v) \in E(P) \subseteq E(R_{i+1})$ and $(u, v) \notin E(R_i)$, by Proposition 3 we know that:

$$(u, v) \notin E(R_i), (u, v) \in E(R_{i+1}) \implies (v, u) \in E(P_i)$$

Since $P_i$ is the shortest path $s \to t$ in $R_i$ and $u, v \in V(P_i)$, by Observation 3 we know that $P_i$ also contains the shortest paths from $s \to u$ and $s \to v$ in $R_i$. In particular, the path from $s \to u$ also contains the path $s \to v$, so $\text{dist}_{R_i}(s, v) = \text{dist}_{R_i}(s, u) - 1$.

Combining this with the previous results and the initial assumption, we get that:

$$\text{dist}_{R_i}(s, v) = \text{dist}_{R_i}(s, u) - 1 \leq \text{dist}_{R_{i+1}}(s, u) - 1 = \text{dist}_{R_{i+1}}(s, v) - 2 < \text{dist}_{R_i}(s, v) - 2$$

implying that $0 < -2$, which is impossible, meaning that such vertices $u_1, \ldots, u_k$ can't exists.

□

**Theorem 2: Total iterations of Edmonds-Karp**

The total number of iterations done by the Edmonds-Karp algorithm is $O(mn)$.

This also implies that the algorithm always terminates.

*Proof.*

Given a network $N = (G, s, t, c)$ and the computation $\mathsf{EdmondsKarp}(N)$, let:

- $f_0, \ldots, f_k$ be the series of flows computed, where $f_0$ is the trivial flow
- $R_0, \ldots, R_k$ be the series of residue graphs computed, where $R_0$ is obtained with flow $f_i$
- $P_i, \ldots, P_k$ be the series of shortest paths from $s \to t$ computed, where $P_i \subseteq R_i$

In the following steps, we define an edge $(u, v)$ as *critical* for $R_i$ if $(u, v) \in E(R_i)$ but $(u, v) \notin E(R_{i+1})$. In particular, by Proposition 3 we know that if an edge is critical for $R_i$ then $(u, v) \in P_i$. Furthermore, for each shortest path $P_i$ we know that there is at least one critical edge inside it, that being the edge which defines the flow augmentation value $\alpha_i$ for $P_i$.

Given an edge $(u, v) \in E(G)$, let $\pi(1), \ldots, \pi(\ell)$ be the indexes such that $1 \leq \pi(1) < \pi(2) < \ldots < \pi(\ell) \leq k$ and such that $(u, v)$ is critical for each $\pi(i)$. By Observation 3, we know that for each index $i = 1, \ldots, k$ it holds that:

$$(u, v) \in E(P_{\pi(i)}) \implies \text{dist}_{R_{\pi(i)}}(s, v) = \text{dist}_{R_{\pi(i)+1}}(s, u) + 1$$

meaning that $(u, v) \in E(R_{\pi(i)+1})$.

If $(u, v)$ is critical for both $R_{\pi(i)}$ and $R_{\pi(j)}$ (where $i < j$, meaning that $(u, v)$ disappears both times), then there must be an index $h$ such that $\pi(i) < h < \pi(j)$ where $(u, v)$ reappears, meaning that $(u, v) \notin E(G_h)$ and $h \in E(G_{h+1})$.

Again, by Proposition 3, we know that:

$$(u, v) \notin E(G_h), h \in E(G_{h+1}) \implies (v, u) \in E(P_h) \implies \text{dist}_{R_h}(s, u) = \text{dist}_{R_h}(s, v) + 1$$

Then, since $\pi(i) < h < \pi(j)$, by the Monotone increasing distance in Edmonds-Keep we know that:

$$\text{dist}_{R_{\pi(j)}}(s,u) \geq \text{dist}_{R_h}(s,u) = \text{dist}_{R_h}(s,v) + 1 \geq \text{dist}_{R_{\pi(i)}}(s,u) + 2$$

Thus, between $R_{\pi(i)}$ and $R_{\pi(j)}$ the vertex $u$'s distance increases by at least 2. Thus, since $u$'s final distance can be at most $n-1$, meaning that $\text{dist}_{R_k}(s,u) \leq n$, starting from 0 the distance can be increased by 2 at most $\frac{n}{2}$ times, meaning that $\ell = \frac{n}{2}$.

Since each flow-augmenting shortest path computed by the algorithm implies the existence of a a critical path and since since the number of times each edge can be critical is at most $\ell = \frac{n}{2}$, the number of paths computable is at most $m \times \frac{n}{2}$, which is in $O(mn)$.

□

## 1.5 Applications of the Max-flow/Min-cut theorem

Considering the previous definitions, it's easy to see that for each undirected graph there is an associated network with chosen capacities by simply "splitting" each undirected edge $(u,v)$ into two directed edges $(u,v),(v,u)$. Vice versa, by inverting this transformation process, we can associate an undirected graph to each network. In particular, given an undirected graph $G$, we denote with $\overrightarrow{G}$ the directed graph underling the network $N = (\overrightarrow{G}, s, t, c)$ associated to $G$.

This natural association can be used to prove theorems on undirected graphs through the use of the Max-flow/Min-cut theorem. In particular, we'll consider the problem of **finding the maximum number of pairwise edge-disjoint paths** between two nodes in an undirected graph.

**Lemma 5**

Let $G$ be an undirected graph and let $N = (\overrightarrow{G}, s, t, c)$ be the network associated to $G$ such that all edges have capacity set to 1. If $G$ has at most $k$ pairwise edge-disjoint paths $s \to t$ then the maximum value of a flow $f$ on $N$ is $k$.

*Proof.*

Since $G$ has at most $k$ pairwise edge-disjoint paths $s \to t$, the associated graph $\overrightarrow{G}$ will have $k$ pairwise edge-disjoint paths $s \to t$ and $k$ pairwise edge-disjoint paths $t \to s$ thanks to the "splitting" process.

Then, for each path $s \to t$ in $\overrightarrow{G}$ we can set $f(u,v) = 1$ and $f(v,u) = -1$ for each edge $(u,v)$ in the path. Since each path is disjoint, sending flow to one of them doesn't influence the other paths, implying that one unit of flow can be sent on each one of the $k$ paths and thus that the maximum value of $f$ is $k$.

□

**Definition 9: Support of a flow**

Given a network $N = (\overrightarrow{G}, s, t, c)$ and a flow $f$ on $N$, we define the **support of** $f$ as the subgraph $\overrightarrow{W} \subseteq \overrightarrow{G}$ such that:

$$E(\overrightarrow{W}) = \{(u, v) \in E(\overrightarrow{G}) \mid f(u, v) \neq 0\}$$

**Lemma 6**

Let $G$ be an undirected graph and let $N = (\overrightarrow{G}, s, t, c)$ be the network associated to $G$ such that all edges have capacity set to 1. If the maximum value of a flow $f$ on $N$ is $k$ then the undirected graph $W$ associated to the support graph $\overrightarrow{W}$ of $f$ has at most $k$ pairwise edge-disjoint paths $s \to t$.

*Proof.*

Let $f$ be a flow on $N$ with maximum value, $\overrightarrow{W} \subseteq \overrightarrow{G}$ be the support graph of $f$ and $W$ the undirected graph associated to $\overrightarrow{W}$. By strong induction on the number $m$ of edges in $W$, meaning that $|E(W)| = m$, we show that $W$ has maximum $k$ pairwise edge-disjoint paths $s \to t$.

If $m = 0$, then $\forall (u, v) \in E(G)$ it holds that $f(u, v) = 0$, implying that $\operatorname{val} f = 0$ and that the number of pairwise edge-disjoint paths $s \to t$ in $W$ is 0. We now assume by strong induction that the statement holds for all graphs whose support graph has at most $m$ edges.

Suppose now that $|E(W)| = m + 1$. In $\overrightarrow{W}$, let $P \subseteq \overrightarrow{W}$ be the subgraph such that:

- $V(P) = \{v_0, \ldots, v_h\}$
- $v_0 = s$
- For each index $i = 0, \ldots, h - 1$ it holds that $(v_i, v_{i+1}) \in E(\overrightarrow{W})$
- For each index $i = 0, \ldots, h - 1$ it holds that $(v_i, v_{i+1}) = 1$
- $h$ is as big as possible

In particular, we observe that this subgraph describes the longest possible path starting from $s$ in the whole graph. Such a subgraph can always be found since $v_h$ can be equal to $s$, giving us the trivial zero-edged path $s \to s$.

In case $v_h = t$, the subgraph $P$ corresponds to the longest path $s \to t$ in $\overrightarrow{W}$ and thus the longest path in $W$. Then, considering $f'$ such that:

$$f'(x, y) = \begin{cases} 0 & \text{if } (x, y) \in E(P) \text{ or } (x, y) \in E(P) \\ f(x, y) & \text{otherwise} \end{cases}$$

its easy to see that $f'$ is a flow on the network $(\overrightarrow{W}, s, t, c)$.

Let $\overrightarrow{W'} \subseteq \overrightarrow{W}$ be the support graph of $f'$. By definition of $f'$, we get that $\overrightarrow{W'} = \overrightarrow{W} - E(P)$. Since $P$ contains an edge outgoing from $s$, in $f'$ the flow value of that edge was set to 0, implying that $\operatorname{val} f' = \operatorname{val} f - 1 = k - 1$.

Since $|E(W')| < |E(W)|$, by inductive hypothesis the graph $W'$ has at most $\operatorname{val} f' = k-1$ pairwise edge-disjoint paths $s \to t$. Furthermore, since $\overrightarrow{W'} = \overrightarrow{W} - E(P)$, the path $P$ is disjoint from these $k-1$ paths, concluding that $W$ has $k$ pairwise edge-disjoint paths.

In case $v_h \neq t$, instead, since $f(v_{h-1}, v_h) = 1$, there must exists an edge $(u, v_h) \in E(\overrightarrow{W})$ with flow value $f(u, v_h) = -1$ in order to preserve the conservation of flow for $v_h$. Moreover, this also implies that $f(v_h, u) = 1$.

By way of contradiction, suppose that $u \notin V(P)$. Then, since $f(v_h, u) = 1$, we could extend this path $P$ with $(v_h, u)$, which contradicts the fact that $h$ was chosen as big as possible, meaning that the only possibility is that $u \in V(P)$. In particular, since by definition of $P$ we have that $f(v_{h-1}, v_h) = 1$ and for $u$ we have that $f(u, v_h) = -1$, we know that $u \neq v_{h-1}$.

Given for each index $i = 0, \ldots, k-2$ such that $u = v_i$, let $C$ be the directed cycle $v_i, \ldots, v_h, u$. Then, considering $f''$ such that:

$$f''(x,y) = \begin{cases} 0 & \text{if } (x,y) \in E(C) \text{ or } (x,y) \in E(C) \\ f(x,y) & \text{otherwise} \end{cases}$$

its easy to see that $f''$ is a flow on the network $(\overrightarrow{W}, s, t, c)$.

If $i \neq 0$ then $v_i \neq s$, implying that $\operatorname{val} f'' = \operatorname{val} f = k$. Otherwise, if $i = 0$, we have that $f''(s, v1) = f''(v_h, s) = 0$. Then, since previously we had $f(s, v1) = 1$ and $f(v_h, s) = -1$, the value of the flow hasn't changed, concluding that in both cases we have that $\operatorname{val} f'' = \operatorname{val} f$.

Finally, since $|E(W'')| < |E(W)|$, by inductive hypothesis the graph $W''$ has at most $\operatorname{val} f'' = k$ pairwise edge-disjoint paths $s \to t$, implying that also $W$ has these paths, concluding that in all cases the statement holds for $m + 1$.

□

**Corollary 3**

Let $G$ be an undirected graph and let $N = (\overrightarrow{G}, s, t, c)$ be the network associated to $G$ such that all edges have capacity set to 1. If the maximum value of a flow $f$ on $N$ is $k$ then $G$ has at most $k$ pairwise edge-disjoint paths $s \to t$.

*Proof.*

By the previous lemma, we know that $W \subseteq G$ has $k$ edge-disjoint paths. By way of contradiction, suppose that the number of pairwise edge-disjoint paths in $G$ is greater than $k$. Then, there exists at least another disjoint path $P$ from $s \to t$ in $G$ that isn't in $W$.

Since $P$ is not in $W$, by definition of support graph, each edge in $P$ must have a flow equal to 0. Then, we could send another unit of flow in this path, contradicting the fact that $f$ is the maximum flow in $G$. Thus, the only possibility is that such a path cannot exist.

□

**Theorem 3: Menger's theorem**

Let $G$ be an undirected graph and let $N = (\overrightarrow{G}, s, t, c)$ be the network associated to $G$ such that all edges have capacity set to 1. The maximum value of a flow $f$ on $N$ is exactly equal to the maximum number of pairwise edge-disjoint paths $s \to t$ in $G$.

*(follows from Lemma 5 and Corollary 3)*

# 2 Linear programming

## 2.1 Introduction and interpretation

**Linear programming**, also called *linear optimization*, is a method to achieve the **optimal outcome** (such as maximum profit or lowest cost) in a mathematical model whose requirements and objective are represented by **linear inequalities**. In particular, the objective to be optimized is defined by an **objective function** on variables $x_1, \ldots, x_n$ that must be *maximized* (or *minimized*).

For example, given two variables $x_1, x_2 \in \mathbb{R}$, we want to maximize the sum $x_1 + x_2$ while also respecting the following linear constraints:

$$\begin{aligned} x_1 + 6x_2 &\leq 15 \\ 4x_1 - x_2 &\leq 10 \\ x_2 - x_1 &\leq 1 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Since we have only two variables, these constraints can be interpreted as lines in a cartesian plane defined by $x_1$ and $x_2$: given a constraint $\alpha x_1 + \beta y_1 \leq \gamma$ (or $\alpha x_1 + \beta y_1 \geq \gamma$), the plane gets partitioned by the line $\alpha x_1 + \beta y_1 = \gamma$ and we can consider only the part that **respects the constraint**.

By applying this process for each constraint, only one part of the graph will respect all the constraints. Any point in this area is a **feasible solution** to the problem. Furthermore, the area that contains the feasible solutions is called the **feasible region**.

In particular, we want to find a feasible solution that maximizes (or minimizes) the objective function. These solutions are called **optimal solutions**. We notice now that the objective function $x_1 + 2x_2$ can also be rewritten as a scalar product:

$$x_1 + x_2 = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

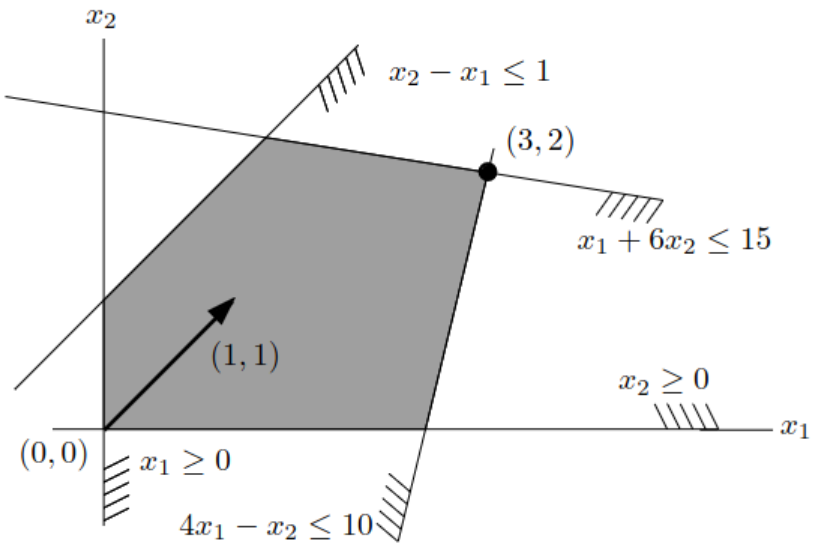



Figure 2.1: The grey area is the feasible region defined by the constraints. The vector $\begin{bmatrix}1 & 1\end{bmatrix}$ shows the direction of the objective function to be maximized

Considering the line described by the vector $\begin{bmatrix}1 & 1\end{bmatrix}$, it's easy to see the optimal solution to the problem is given by the point **furthest from the origin**: the line *perpendicular* to the vector $\begin{bmatrix}1 & 1\end{bmatrix}$ that passes through such point is the optimal solution.

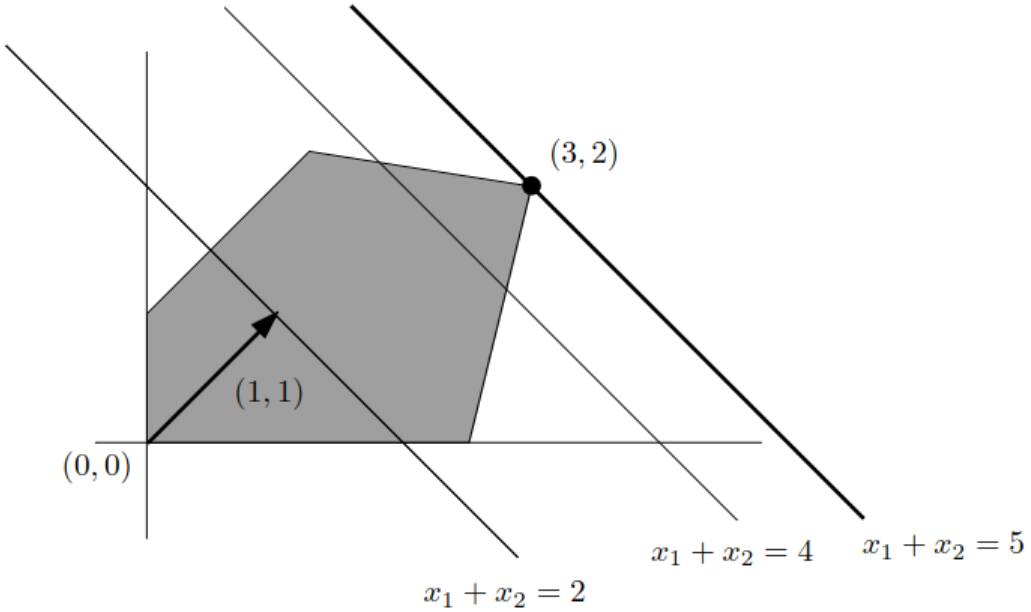



Figure 2.2: The optimal solution is given by the line $x_1 + x_2 = 5$, meaning that 5 is the optimal solution

In this problem, there is **only one optimal solution**, which is the vector $\begin{bmatrix}3 & 2\end{bmatrix}$. However, if vector that defines the direction of the objective function is *perpendicular* to a constraint, we get **infinite optimal solutions** (because in $\mathbb{R}$ a segment has infinite points).

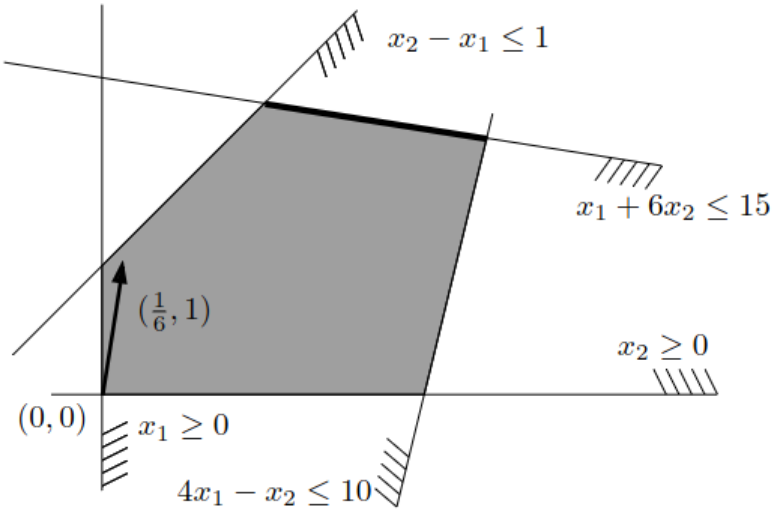
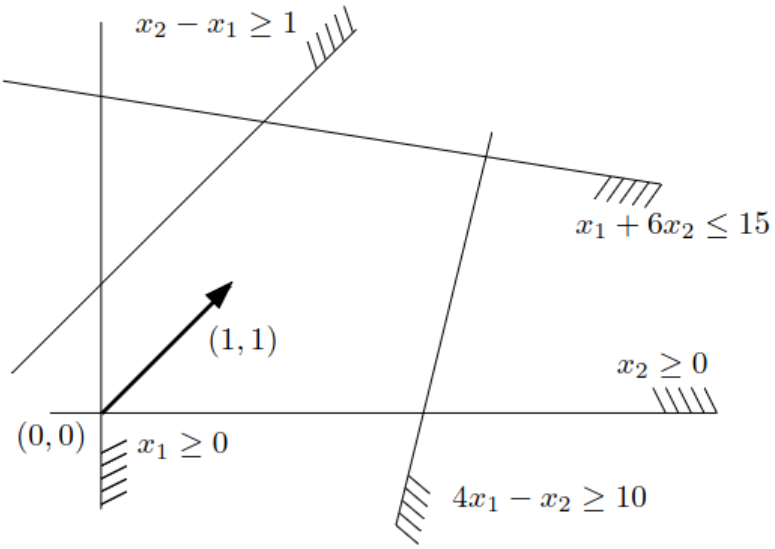



Figure 2.3: If the objective function is $\frac{1}{6}x_1 + x_2$, we get an infinite set of optimal solutions

Instead, if we reverse the directions of the inequalities of some of the constraints, we get that there is no feasible solution. Such programs are called **infeasible**.




Finally, in some situations there may be a feasible solution but no optimal solution. These cases arise when the direction of the objective function is unbounded, meaning that we can always find a solution with an higher (or lower) value. Such programs are called **feasible unbounded**.

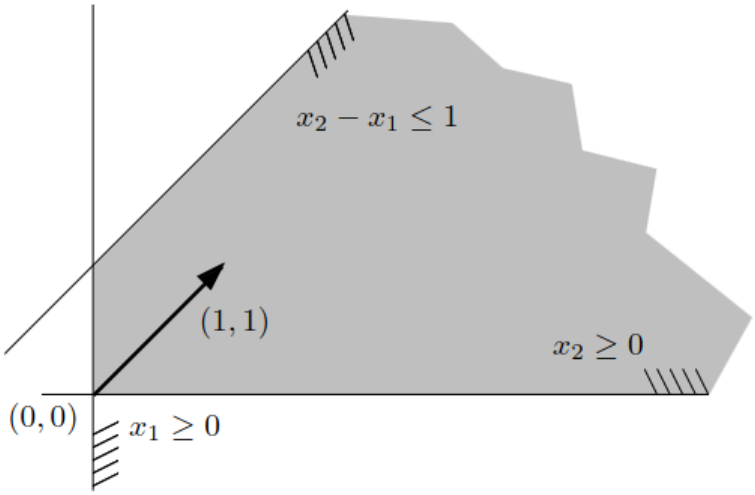



Figure 2.4: If we remove some constraints, the program becomes feasible unbounded

**Proposition 4: Possibilities in a linear program**

For each linear program, only one of the following cases holds:

1. The program is **infeasible**, meaning that there is no feasible solution
2. The program is **feasible unbounded**, meaning that are infinite feasible solutions but no optimal solution
3. The program is **uniquely optimal**, meaning that there are infinite feasible solutions and only one optimal solution
4. The program is **infinitely optimal**, meaning that there are infinite optimal solutions

Throughout the following sections and chapters, we will give proper justifications to formalize this result. Optimization problems are hidden in everyday life. For example, we can resolve the following two problems with linear programming:

- **Diet optimization**:

  Given $n$ types of food, for each $i$ such that $1 \leq n$ we define $x_i$ and $c_i$ respectively as the quantity and the cost of each food type $i$. Considering a set of nutritional constraints, such as minimum and maximum macro-nutrient intakes, we can define the following optimization problem in order to find the best combination of foods for our diet:

$$\min c_1 x_1 + \ldots + c_n x_n$$

$$\begin{aligned} a_{1,1}x_1 + \ldots + a_{1,n}x_n &\leq b_1 \\ &\vdots \\ a_{m,1}x_1 + \ldots + a_{m,n}x_n &\leq b_m \end{aligned}$$

- **Network flow**:

  Given a network $N = (G, s, t, c)$ and a flow $f$ on $N$, we can define a variable $x_{u,v}$ for all $(u, v) \in E(G)$ in order to formulate the max flow value problem as an optimization problem and find the optimal flow by setting $f(u, v) = x_{u,v}$. Trivially, each constraint of the network becomes a constraint of the problem.

## 2.2 Linear programs and Standard form

In this section we give a proper formal definition of the concepts previously shown. Hence, it is assumed that each interpretation is already well-known.

**Definition 10: Linear function**

Let $f : \mathbb{R}^n \to \mathbb{R}$. We say that $f$ is a **linear function** if

$$\forall x, y \in \mathbb{R}^n, \forall \alpha, \beta \in \mathbb{R} \quad f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

**Definition 11: Linear program**

A **linear program**, or *linear optimization problem*, is an optimization problem defined on an linear function $f : \mathbb{R}^n \to \mathbb{R}$ to be maximized, called **objective function**, and constraints on the input vector $x$:

$$\max f(x) = c_1 x_1 + \ldots c_n x_n$$

$$\begin{aligned} a_{1,1}x_1 + \ldots + a_{1,n}x_n &\leq b_1 \\ &\vdots \\ a_{m,1}x_1 + \ldots + a_{m,n}x_n &\leq b_m \end{aligned}$$

Each vector $x$ that respects the constraints is called a **feasible solution** and the set of feasible solutions is called **feasible region**. Each feasible solution that maximizes the objective function is called **optimal solution**.

In the previous section, we noticed how the objective function can also be rewritten as a scalar product:

$$f(x) = c_1 x_1 + \ldots + c_n x_n = \begin{bmatrix} c_1 & \ldots & c_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Thus, we can say that $f(x) = c^T x$, where $c^T = \begin{bmatrix} c_1 & \ldots & c_n \end{bmatrix}$ is the transpose of the **coefficient vector of** $f$. Likewise, we notice that each constraint of the problem corresponds to a row of the linear system $Ax \leq b$. Thus, we can define the following **general**

**formulation of linear programs**:

**Proposition 5: Conversion of linear programs**

Every linear program can be written as a maximization problem in the following general form:

$$\begin{gathered}\max\ c^T x \\ Ax \leq b \\ x \geq 0\end{gathered}$$

Through the following steps:

1. The optimal solution that minimizes $c^T x$ also maximizes $-c^T x$, so we can substitute $\min\ c^T x$ with $\max\ -c^T x$. **Warning**: even thought the solution is the same, the optimal value will have negated sign. This can be fixed by just inverting the negating optimal value.
2. Any constraint of the form $a_1 x_1 + \ldots + a_n x_n \geq b$ is equivalent to the constraint $-a_1 x_1 - \ldots - a_n x_n \leq -b$, so we can substitute the first with the latter
3. Any component variable $x_i$ such that $x < 0$ can be substituted with two non-negative variables $z_i, z_i' \geq 0$ such that $x_i = z_i - z_i'$

Now, we would like to find a way to define the linear program in an **equational form**. In particular, consider the following linear program:

$$\max c_1 x_1 + \ldots + c_n x_n$$

$$\begin{gathered}a_{1,1} x_1 + \ldots + a_{1,n} x_n \leq b_1 \\ \vdots \\ a_{m,1} x_1 + \ldots + a_{m,n} x_n \leq b_m\end{gathered}$$

where $A$ is the matrix formed by the constraints.

We define $m$ variables $s_1, \ldots, s_m$, called **slack variables**, such that for each index $i = 1, \ldots, m$, we define $s_i := b_i - a_{1,1} x_1 + \ldots + a_{1,n} x_n$. By adding the slack variable $s_i$ to the left side of the $i$-th inequality, each inequality becomes an equality.

Thus, at the cost of adding $m$ variables to the problem, we can reformulate the problem as:

$$\max\ c_1 x_1 + \ldots + c_n x_n + 0 \cdot s_1 + \ldots + 0 \cdot s_m$$

$$\begin{gathered}a_{1,1} x_1 + \ldots + a_{1,n} x_n + s_1 = b_1 \\ \vdots \\ a_{m,1} x_1 + \ldots + a_{m,n} x_n + s_m = b_m\end{gathered}$$

Given the $m \times m$ identity matrix $I_m$, we get that:

$$[A \mid I_m] \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ s_1 \\ \vdots \\ s_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Then, by setting $A' := [A \mid I_m]$ and $c' := \begin{bmatrix} c_1 & \dots & c_n & 0 & \dots & 0 \end{bmatrix}$ we get the following linear program in equational form:

$$\begin{gathered} \max\ c'^T x \\ A'x = b \\ x \geq 0 \end{gathered}$$

**Definition 12: Standard form**

A linear program is said to be in **standard form** (or *equational form*) if it is formulated as:

$$\begin{gathered} \max\ c^T x \\ Ax = b \\ x \geq 0 \end{gathered}$$

where $c, x \in \mathbb{R}^n$, $A \in \mathrm{Mat}_{m \times n}(\mathbb{R})$ and $b \in \mathbb{R}^m$.

Through *linear algebra*, we can find numerous results involving linear programs in standard form. In particular, a fundamental result is that the system $Ax = b$ is **invariant under Gaussian row operations**. Thus, we can reduce the number of rows of $A$ through row elimination in order to obtain the **minimal amount of constraints** that define the problem. For these reasons, we are going to assume that each removable row has already been removed, meaning that the **rows** (but always not the columns) of $A$ are assumed to be **linearly independent**. Moreover, since we generally obtain a standard form linear program from a non-standard one and the transformation adds $m$ variables to the problem, we're going to always assume that $m \leq n$.

**Observation 4: Assumptions on standard form LPs**

Given a linear program in standard form $Ax = b$ with $x \geq 0$, we assume that:

- $A \in \mathrm{Mat}_{m \times n}(\mathbb{R})$ and $m \leq n$
- The rows of the matrix $A$ are always linearly independent
- The rank of $A$ is $\mathrm{rank}(A) = \mathrm{colrank}(A) = \mathrm{rowrank}(A) = m$

## 2.3 Basic feasible solutions

Among all the feasible solutions of a linear program, a privileged status is granted to so-called **basic feasible solutions**. First, we'll give a geometric intuition of these types of feasible solutions: a basic feasible solution is a *vertex* (or corner, spike) of the set of feasible solutions.

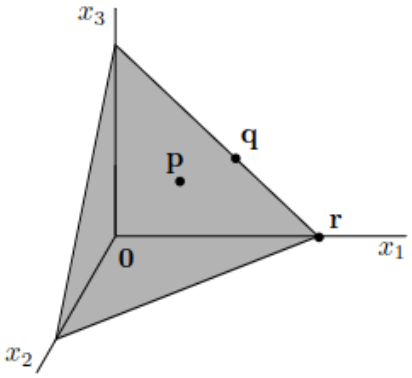



Figure 2.5: A set of feasible solutions

For example, in the set of feasible solutions shown above in grey, among $p$, $q$ and $r$ only the latter is basic. In particular, we notice that the solution $r$ is set to zero on both the $x_2$ and $x_3$ axes. In fact, very roughly speaking, a basic feasible solution is a feasible solution with sufficiently many variables set to zero. This intuition will be justified later both informally and formally.

First of all, we will introduce some new notation: consider a linear program in standard form:

$$\begin{aligned} \max \ & c^T x \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

where $A \in \mathrm{Mat}_{m \times n}$ and we denote with $A = (A^1, \ldots, A^n)$ the column decomposition of $A$. Given a subset of indices $\mathcal{B} \subseteq \{1, \ldots, n\}$ called **basis**, we define the matrix $A_\mathcal{B}$ as the **matrix formed by the columns of** $A$ whose index is inside $\mathcal{B}$, meaning that $A_\mathcal{B} = (A^{i_1}, \ldots, A^{i_k})$ where $i_1, \ldots, i_k \in \mathcal{B}$.

**Example:**

- Suppose that:

$$A = \begin{bmatrix} 1 & 6 & 3 & -3 & 4 \\ 0 & 1 & 3 & 2 & 4 \end{bmatrix}$$

  If $\mathcal{B} = \{2, 4\}$, the matrix $A_\mathcal{B}$ is equal to:

$$A_\mathcal{B} = \begin{bmatrix} 6 & -3 \\ 1 & 2 \end{bmatrix}$$

**Proposition 6**

Given the matrix equation $Ax = b$ and a subset of indices $\mathcal{B} \subseteq \{1, \dots, n\}$, if $|\mathcal{B}| = m$ and $A_{\mathcal{B}}$ is *non-singular*, meaning that $\det(A_{\mathcal{B}}) \neq 0$, then there exists a solution $\overline{x} := (x_1, \dots, x_n) \in \mathbb{R}^n$ such that $A\overline{x} = b$ and $\forall i \notin \mathcal{B}\ \ x_i = 0$.

*Proof.*

If $|B| = m$ and $\det(A_{\mathcal{B}}) \neq 0$ then $A_{\mathcal{B}}$ is a non-singular $m \times m$ matrix with rank $m$. Thus, through the *Rouché-Capelli theorem*, since $A_{\mathcal{B}}$ is also a minor of both $A$ and $(A \mid b)$, we get that $\text{rank}(A) = \text{rank}(A \mid b) = m$, implying that subspace of solutions of $Ax = b$ has rank $n - m$. Moreover, since the columns whose index isn't in $\mathcal{B}$ are linearly dependent by the other columns, this subspace will always contain a solution such that $\forall i \notin \mathcal{B}\ \ x_i = 0$.

□

**Definition 13: Basic feasible solution**

Given the equation $Ax = b$ of a standard form linear program, we say that $\overline{x} \in \mathbb{R}^n$ is a **basic feasible solution (BFS)** if $\overline{x}$ is a feasible solution for which there exists an index set $\mathcal{B} \subseteq \{1, \dots, n\}$ such that:

- $A_{\mathcal{B}}$ is an $m \times m$ non-singular matrix
- $\forall i \notin \mathcal{B}\ \ \overline{x_i} = 0$

Moreover, we say that $\mathcal{B}$ **certifies** $\overline{x}$ and call **basic variable** each variable $x_i$ such that $i \in \mathcal{B}$, while the other variables are called **non-basic variable**.

**Example:**

- Suppose that a standard form linear program is defined by:

$$A = \begin{bmatrix} 1 & 6 & 3 & -3 & 4 \\ 0 & 1 & 3 & 2 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

  If $\mathcal{B} = \{2, 4\}$, the matrix $A_{\mathcal{B}}$ is equal to:

$$A_{\mathcal{B}} = \begin{bmatrix} 6 & -3 \\ 1 & 2 \end{bmatrix}$$

  Furthermore, since $\det(A_{\mathcal{B}}) = 15 \neq 0$ and $|\mathcal{B}| = 2$, we know that there exists a vector $\overline{x}^T = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}$ such that $A\overline{x} = b$. By solving the following equation

$$\begin{bmatrix} 6 & -3 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

  we get that $x_2 = 2$ and $x_4 = 2$. Finally, by setting all the other variables to 0, we get that the vector $\overline{x}^T = \begin{bmatrix} 0 & 2 & 0 & 2 & 0 \end{bmatrix}$ implies that $\overline{x}$ is a BFS.

An important thing to notice is that $\overline{x}$ is a BFS because it also respects the constraint of non-negative solutions since $\overline{x} \geq 0$, meaning that it is indeed a feasible solution. The method that was just shown doesn't always guarantee that the solution to the "reduced equation" is also a feasible solution.

For example, if we consider $\mathcal{B}' = \{1, 2\}$, we get that $\det(A_{\mathcal{B}'}) = -5 \neq 0$ and $|\mathcal{B}'| = 2$, so we know that there exists a vector $\overline{y}^T = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 \end{bmatrix}$ such that $A\overline{y} = b$. However, by solving the following equation

$$\begin{bmatrix} 1 & 6 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

we get that $y_1 = -30$ and $y_2 = 6$, the vector $\overline{y}^T = \begin{bmatrix} -30 & 6 & 0 & 0 & 0 \end{bmatrix}$ doesn't respect the constraint $\overline{y} \geq 0$, meaning that it isn't a feasible solution and by consequence that it isn't a BFS.

**Theorem 4**

Given a feasible solution $\overline{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}$ to a linear program, let $K = \{i \in \{1, \dots, n\} : x_i > 0\}$. It holds that:

$$\overline{x} \text{ is a BFS} \iff \text{Columns of } A_K \text{ are lin. ind.}$$

*Proof.*

*First implication.* Suppose that $\overline{x}$ is a BFS and let $\mathcal{B}$ be the basis through which we obtain $\overline{x}$. Every non-basic variable of $\overline{x}$ is set to 0, meaning that $\forall i \notin \mathcal{B}$ we have that $x_i = 0$. However, since for all the other basic variables $x_j$ it holds that $x_j \geq 0$, these variables could also be set to 0.

Thus, generally we have that $K \subseteq \mathcal{B}$. Since $A_{\mathcal{B}}$ is non-singular, the columns of $A_{\mathcal{B}}$ are linearly independent. Then, since $i \in K \subseteq \mathcal{B}$, each column of the matrix $A_K$ is also linearly independent.

*Second implication.* Suppose that the columns of $A_K$ are linearly independent, implying that $\det(A_K) \neq 0$. If $|K| = m$ then $K$ certifies that $\overline{x}$ is a BFS. If $|K| < m$, instead, we pick $\mathcal{B} \subseteq \{1, \dots, n\}$ such that:

- $K \subseteq \mathcal{B}$
- $A_{\mathcal{B}}$ has linearly independent columns
- $\mathcal{B}$ is as big as possible

Such set $\mathcal{B}$ always exists since, eventually, $K = \mathcal{B}$ is allowed by definition.

Since the columns of $A_{\mathcal{B}}$ are also columns of $A$, since $\mathcal{B}$ is as big as possible and since $A_{\mathcal{B}}$ is has linearly independent columns, it holds that $\text{rank}(A_{\mathcal{B}}) = \text{rank}(A) = m$, implying that $|\mathcal{B}| = m$ and thus that $\mathcal{B}$ certifies that $\overline{x}$ is a BFS.

□

**Observation 5**

A BFS can be certified by more than one basis

**Example:**

- Suppose that a standard form linear program is defined by:

$$A = \begin{bmatrix} 1 & 6 & 3 & -3 & 4 \\ 0 & 1 & 3 & 2 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

  By the previous theorem, the vector $\overline{x}^T = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 \end{bmatrix}$ gives a BFS due to the fact that $K = \{3\}$ and the columns of $A_K$ are clearly linearly independent since it is made of only one column. Moreover, we notice that we can expand the set $K$ to two different basis $\mathcal{B} = \{1, 3\}$ and $\mathcal{B}' = \{2, 3\}$

**Definition 14: Feasible basis**

Given a basis $\mathcal{B}$ of a linear program, we say that $\mathcal{B}$ is **feasible** if it certifies a BFS

**Observation 6**

Given a feasible basis $\mathcal{B}$, there exists only one BFS $\overline{x}$ certified by $\mathcal{B}$

*Proof.*

Let $\mathcal{B} = \{i_1, \dots, i_k\}$. Since $\mathcal{B}$ is a feasible basis, the matrix $A_{\mathcal{B}}$ is made of linearly independent columns, implying that there must be only one solution to the equation

$$A_{\mathcal{B}} \cdot \begin{bmatrix} x_{i_1} \\ \vdots \\ x_{i_k} \end{bmatrix} = b$$

□

**Proposition 7**

A linear program in standard form has at most $\binom{n}{m}$ BFS

*Proof.*

Since each basis must have cardinality $m$ and since $A$ has $n$ columns, the number of possible choises for a basis is $\binom{n}{m}$. Then, even if all the possible basis are feasible basis each one of them corresponds to only one BFS which could also be shared with another basis. Thus, even if no BFS is shared among more than one basis, there can be at most $\binom{n}{m}$ BFSs.

□

### 2.3.1 Above bounded linear programs and BFS

**Definition 15: Above bounded linear program**

Given a linear program in standard form, we say that it's objective function is **bounded above** if $\exists N \in \mathbb{R}$ such that for all feasible solutions $y$ it holds that $c^T y \leq N$.

**Theorem 5**

Given a linear program in standard form, if the objective function has an upper bound then for each feasible solution $y$ there exists a BFS $z$ such that $c^T z \geq c^T y$.

*Proof.*

Let $y$ be a feasible solution and let $z$ be a feasible solution with the maximum amount of zero variables and such that $c^T z \geq c^T y$. Such a feasible solution $z$ always exists since, eventually, $y = z$ is allowed by definition.

By way of contradtiction, suppose that $z$ isn't a BFS and let $K = \{i \in \{1, \ldots, n\} \mid z_i > 0\}$. It's easy to see that the columns of $A_K$ must be linearly dependent: if they weren't, by theorem Theorem 4 we would get that $z$ is a BFS, which is impossible.

Since they are linearly dependent, we know that $\exists \alpha_{i_1}, \ldots, \alpha_{i_k} \neq 0$, where $K = \{i_1, \ldots, i_k\}$, such that

$$\alpha_{i_1} A_K^{i_1} + \ldots + \alpha_{i_k} A_K^{i_k} = 0 \iff A_K \cdot \begin{bmatrix} \alpha_{i_1} \\ \vdots \\ \alpha_{i_1} \end{bmatrix} = 0$$

Since the columns of $A_K$ are also columns of $A$ and since the coefficients $\alpha_{i_1}, \ldots, \alpha_{i_k}$ nullify the linear combination of the columns in $A_K$, by setting to 0 the coefficient of every other column of $A$ we can nullify the linear combination of the columns of $A$.

Then, let $w$ such vector of coefficients, formally defined as:

$$w_j = \begin{cases} a_j & \text{if } j \in K \\ 0 & \text{otherwise} \end{cases}$$

As we just said, we know that $Aw = 0$.

**Claim**: We can assume that the vector $w$ satisfies the following two conditions:

1. $c^T w \geq 0$
2. $\exists j \in K$ such that $w_j < 0$

*Proof of the claim.*

First, we notice that $Aw = 0$, for all $\beta \in \mathbb{R}$ it also holds that $A(\beta w) = 0$.

If $c^T w = 0$ then $c^T w \geq 0$ trivially holds. Moreover, if the second condition doesn't hold for $w$, meaning that $\forall j \in K$ we have that $w_j \geq 0$, we can substitute $w$ with the vector

$-w$, satisfying the second condition since $w \neq 0$. In particular, we notice that such substitution doesn't violate the first condition: if $c^T w = 0$, by linearity of the objective function it holds that $c^T(-w) = -(c^T w) = 0$.

If $c^T w \neq 0$, by picking $w$ or $-w$ we can satisfy the first condition since $c^T w > 0$ or $c^T(-w) > 0$ must hold. Without loss of generality, in the following statements we will assume that $w$ was chosen.

By way of contradiction, suppose that the second condition doesn't hold, meaning that $\forall j \in K$ we have that $w_j \geq 0$.

Given $t \in \mathbb{R}$ such that $t \geq 0$, let $z(t) := z + tw$. We notice that $z(0) = z$ and that

$$A \cdot z(t) = A(z + tw) = Az + tAw$$

Since $z$ is a feasible solution, meaning that $Az = b$, and since $Aw = 0$, we get that $A \cdot z(t) = b$. Furthermore, since $z$ is a feasible solution and since by assumption $\forall j \in K \;\; w_j \geq 0$, for each index $i$ we know that $z(t)_i = z_i + tw_i \geq 0$, concluding that $z(t) \geq 0$ and thus that it is indeed a feasible solution.

By linearity of the objective function, we also know that:

$$c^T \cdot z(t) = c^T(z + tw) = c^T z + tc^T w$$

Since $t \geq 0$ and since $c^T > w$, we also get that $tc^T w > 0$. However, this is valid $\forall t \in \mathbb{R}$, meaning that we can always choose a value $t'$ bigger than the previous for which $c^{t'} w > c^t w$, contradicting the fact that the objective function has an upper bound. Thus, it must be true that $\exists j \in K$ such that $w_j < 0$. □

Once this assumption is made, given $t \in \mathbb{R}$ we define again the vector $z(t) := z + tw$. As before, we notice that $A \cdot z(t) = b$. However, it is no longer always true that $z(t) \geq 0$ since we proved that $\exists j \in K$ such that $w_j < 0$ (that result was given by assuming the contrary). In fact, if $t > \frac{z_i}{-w_i}$ we get that $z(t)_i < 0$.

**Claim**: $\exists t' > 0$ such that $z(t')$ is a feasible solution with more zeros than $z$

*Proof of the claim.* Since $z$ is a feasible solution, we know that $z \geq 0$. Furthermore, we recall that $K$ is defined as $K = \{i \in \{1, \ldots, n\} : z_i > 0\}$, implying that for each $j \notin K$ it must be true that $z_j = 0$.

Thus, since for $j \notin K$ we also know that $w_j = 0$, we get that:

$$z(t)_j = \begin{cases} z_j + tw_j & \text{if } j \in K \\ 0 & \text{otherwise} \end{cases}$$

Since $\exists h \in K$ such that $w_h < 0$, let $t' = \min_{h \in K : w_h < 0} \left( \frac{z_j}{-w_j} \right)$. In particular, notice that by the way $t'$ is defined we get that:

- By choice of $h$, we know that $z_h > 0$ and $w_h < 0$, thus $t' > 0$

- If $z_i = 0$ then $i \notin K$, implying also that $z(t')_i = 0$. However, by choice of $h$, we know that $z_h > 0$
$$z(t')_h = z_h + t'w_h = z_h + \frac{z_j}{-w_j} \cdot w_j = 0$$
thus $z(t')$ always has at least one more zero variable than $z$
- Since $t' > 0$, for each $j \in K$ such that $w_j < 0$ it holds that:
$$t' = \frac{z_h}{-w_h} \leq \frac{z_j}{-w_h} \implies t'w_j \geq z_j$$

Given an index $i$, if $i \notin K$ then we already know that $z(t')_i = 0$. If $i \in K$, instead, by definition of $K$ we get that $z_i > 0$. Thus, we get two additional subcases:

- If $w_i \geq 0$ then $z(t')_i = z_i + t'w_i \geq 0$
- If $w_i < 0$ then $t'w_i < 0$, but we also know that $t'w_i \geq z_i$, implying that $z(t')_i = z_i + t'w_i \geq 0$

Thus in all cases we get that $z(t') \geq 0$. Since it's also true that $A \cdot z(t') = b$, this concludes that $z(t')$ is a feasible solution with more zeros than $z$ □

Finally, by linearity of the objective function and since we showed that $c^T w \geq 0$ and $t' > 0$, we get that

$$c^T \cdot z(t') = c^T(z + t'w) = c^T z + t'c^T w \geq c^T z \geq c^T y$$

Thus, we get that $z(t')$ is a feasible solution with more zeros than $z$ and such that $c^T \cdot z(t') \geq c^T y$, contradicting our initial assumption for which $z$ was chosen with such characteristics. Thus, it must be impossible for $z$ to not be a BFS.

□

**Corollary 4: Feasible solution with maximum zeros**

A BFS is a feasible solution with the **maximum amount of zero variables**. Thus, if there is at least one feasible solution, there also is at least one BFS.

*(follows from the way we proved the previous theorem)*

This corollary only partially justifies the previously mentioned geometric intuitions. In fact, we now know that a BFS has a sufficient amount of zero components. However, this isn't enough to justify that it is a tip of the set of feasible solutions: if we consider the image shown in Fig. 2.5, a point on the segment between the origin and $r$ also has the components $x_2$ and $x_3$ set to zero. In the following sections, we will discuss how these points can't be a BFS due to not being a tip.

**Theorem 6**

Given a linear program in standard form, it holds that:

1. If there is at least one feasible solution and the objective function has an upper bound, there also is at least one **optimal solution**
2. If there is an optimal solution, there also is a **BFS that is optimal**

*Proof.*

1. Suppose that there is at least a feasible solution $y$ and that the objective function has an upper bound. Through the previous theorem, we know that there is a BFS $z$ such that $c^T z \geq c^T y$. Thus, there is at least one BFS.

   Since there are at most $\binom{n}{m}$ BFSs, let $\overline{x}$ be the BFS such that:

   $$c^T \overline{x} = \max_{z \text{ BFS}} (c^T z)$$

   By way of contradiction, suppose that $\overline{x}$ isn't optimal, implying that there is at least another feasible solution $\overline{z}$ such that $c^T \overline{z} > c^T \overline{x}$.

   Again, through the previous theorem we know that there also exists a BFS $\overline{y}$ such that $c^T \overline{y} \geq c^T \overline{z}$. However, this would also imply that $c^T \overline{y} \geq c^T \overline{z} > c^T \overline{x}$, which contradicts the fact that $\overline{x}$ was chosen as the BFS with maximal objective function value. Thus, it must be true that $\overline{x}$ is optimal and thus that there is at least one optimal solution.

2. Suppose that there is an optimal solution $\overline{x}$. Then, by the previous theorem, we know that there is a BFS $\overline{y}$ such that $c^T \overline{y} \geq c^T \overline{x}$.

   However, since $\overline{x}$ is optimal, it's also true that $c^T \overline{x} \geq c^T \overline{y}$. Thus, $c^T \overline{x} = c^T \overline{y}$, concluding that $\overline{x}$ must also be an optimal solution.

□

## 2.4 Geometry behind linear programs

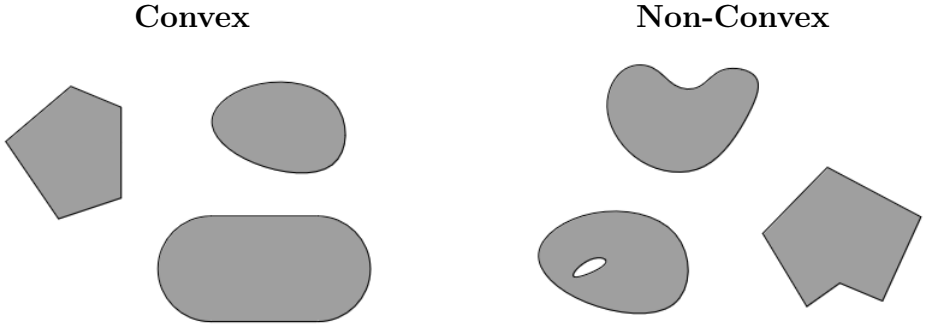
### 2.4.1 Convexity

In this section we will formally define the geometric intuitions behind linear programs. These geometric concepts are fundamental to define methods and algorithms that can solve linear programs efficiently.

**Definition 16: Convex set**

Let $X \subseteq \mathbb{R}^n$. We say that $X$ is **convex** if $\forall x_1, x_2 \in X$ it holds that all the points in the line segment from $x_1$ to $x_2$ are also in $X$.

Formally, this means that $\forall x_1, x_2 \in X$ and $\forall \alpha \in [0,1] \subset \mathbb{R}^n$ it holds that the linear combination $\alpha x_1 + (1-\alpha)x_2$ is also in $X$.




**Proposition 8**

The **feasible region** of every linear program is a **convex set**.

*Proof.*

Suppose that the following inequalities are the constraints of a linear program:

$$\begin{aligned} a_{1,1}x_1 + \ldots + a_{1,n}x_n &\leq b_1 \\ &\vdots \\ a_{m,1}x_1 + \ldots + a_{m,n}x_n &\leq b_m \end{aligned}$$

Let $X$ be the feasible region of the linear program. Let $\overline{x} := (\overline{x_1}, \ldots, \overline{x_1})$ and $\overline{y} := (\overline{y_1}, \ldots, \overline{y_1})$ be two feasible solutions, meaning that $\overline{x}, \overline{y} \in X$.

Given $\gamma \in [0,1] \subset \mathbb{R}^n$, we notice that:

$$\begin{aligned} \gamma(a_{1,1}\overline{x_1} + \ldots + a_{1,n}\overline{x_n}) &\leq \gamma b_1 \\ &\vdots \\ \gamma(a_{m,1}\overline{x_1} + \ldots + a_{m,n}\overline{x_n}) &\leq \gamma b_m \end{aligned} \quad \Longrightarrow \quad \begin{aligned} a_{1,1}(\gamma\overline{x_1}) + \ldots + a_{1,n}(\gamma\overline{x_n}) &\leq \gamma b_1 \\ &\vdots \\ a_{m,1}(\gamma\overline{x_1}) + \ldots + a_{m,n}(\gamma\overline{x_n}) &\leq \gamma b_m \end{aligned}$$

Likewise, we can show that:

$$\begin{gathered} a_{1,1}((1-\gamma)\overline{y_1}) + \ldots + a_{1,n}((1-\gamma)\overline{y_n}) \leq (1-\gamma)b_1 \\ \vdots \\ a_{m,1}((1-\gamma)\overline{y_1}) + \ldots + a_{m,n}((1-\gamma)\overline{y_n}) \leq (1-\gamma)b_m \end{gathered}$$

Thus, by summing the two systems of inequalities, we get that:

$$\begin{gathered} a_{1,1}(\gamma\overline{x_1} + (1-\gamma)\overline{y_1}) + \ldots + a_{1,n}(\gamma\overline{x_n} + (1-\gamma)\overline{y_n}) \leq b_1 \\ \vdots \\ a_{m,1}(\gamma\overline{x_1} + (1-\gamma)\overline{y_1}) + \ldots + a_{m,n}(\gamma\overline{x_n} + (1-\gamma)\overline{y_n}) \leq b_m \end{gathered}$$

concluding that $\gamma\overline{x} + (1-\gamma)\overline{y}$ is also a feasible solution and thus that it's in $X$

□

**Observation 7: Infinite optimal solutions**

Given a linear program, if there are two distinct optimal solutions $\overline{x}, \overline{y}$ then there are infinite optimal solutions.

*Proof.*

Let $\beta$ be the maximum value of the objective function $f$ obtained with the optimal solutions $\overline{x}, \overline{y}$, meaning that $f(\overline{x}) = f(\overline{y}) = \beta$. For each $\alpha \in [0,1] \subset \mathbb{R}^n$, we know that $\alpha\overline{x} + (1-\alpha)\overline{y}$ is a feasible solution due to the feasible region being a convex set.

Furthermore, by linearity of $f$, we notice that:

$$f(\alpha\overline{x} + (1-\alpha)\overline{y}) = \alpha f(\overline{x}) + (1-\alpha)f(\overline{y}) = \alpha\beta + (1-\alpha)\beta = \beta$$

concluding that $\forall\alpha \in [0,1] \subset \mathbb{R}^n$ we get that $\alpha\overline{x}+(1-\alpha)\overline{y}$ is also an optimal solution.

□

**Observation 8: Intersection of convex sets**

Given two convex sets $X, Y \subseteq \mathbb{R}^n$, the intersection $X \cap Y$ is also convex

*Proof.*

Given $u, v \in X \cap Y$, we know that each point in the segment between $u$ and $v$ is both in $X$ and $Y$ due to the convexity of $X$ and $Y$, implying that the segment is also in $X \cap Y$.

□

**Definition 17: Convex hull**

Given $x_1, \ldots, x_n \in \mathbb{R}^n$, a **convex hull** of $x_1, \ldots, x_n$ is the intersection of all the convex sets $X_1, X_2, \ldots \subseteq \mathbb{R}^n$ that contain $x_1, \ldots, x_n$.

By the previous observation, its obvious that the convex hull is also convex. Moreover, intuitively by definition we get that the convex hull is the minimal convex set that contains $x_1, \ldots, x_n$.

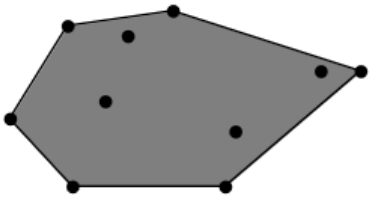


Figure 2.6: Example of convex hull

Even though this definition is intuitive enough, it isn't very constructive. In fact, we can also define the convex hull in terms of *convex combinations*, a generalization of the formal definition of a segment between two points.

**Definition 18: Convex combination**

Given $x_1, \ldots, x_n \in \mathbb{R}^n$, a **convex combination** of $x_1, \ldots, x_n$ is a linear combination $z$ with non-negative coefficients such that the sum of the coefficients equals one.

Formally, it corresponds to a vector $z$ such that:

- $z = \sum_{i=1}^{n} a_i x_i$
- $\sum_{i=1}^{n} a_i = 1$
- $a_1, \ldots, a_n \geq 0$

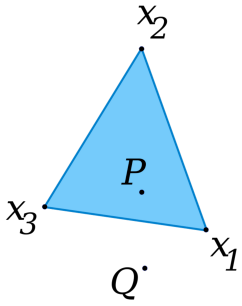



Figure 2.7: The point $P$ is a convex combination of $x_1, x_2, x_3$, while $Q$ is not

**Proposition 9**

Given $x_1, \ldots, x_n \in \mathbb{R}^n$, the convex hull of $x_1, \ldots, x_n$ is equal to the set of convex combinations of $x_1, \ldots, x_n$

*Proof.*

Let $C$ be the convex hull of $x_1, \ldots, x_n$ and let $\tilde{C}$ be the set of convex combinations, meaning that:

$$\tilde{C} = \left\{ \sum_{i=1}^{n} a_i x_i \;\middle|\; \sum_{i=1}^{n} a_i = 1 \text{ and } a_1, \ldots, a_n \geq 0 \right\}$$

Given $u, v \in \tilde{C}$, we know that:

$$u = \sum_{i=1}^{n} a_i x_i \qquad\qquad v = \sum_{i=0}^{n} b_i x_i$$

where $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i = 1$ and $a_1, \ldots, a_n, b_1, \ldots, b_n \geq 0$.

Given $\gamma \in [0, 1] \subseteq \mathbb{R}^n$, let $z := \gamma u + (1 - \gamma)v$. We notice that:

$$\begin{aligned} z &= \gamma u + (1-\gamma)v \\ &= \gamma \sum_{i=1}^{n} a_i x_i + (1-\gamma) \sum_{i=1}^{n} b_i x_i \\ &= \sum_{i=1}^{n} (\gamma a_i + (1-\gamma) b_i) x_i \\ &= \sum_{i=1}^{n} c_i x_i \end{aligned}$$

where $c_i := \gamma a_i + (1 - \gamma)b_i$. Since for each index $i$ we have that $\gamma, (1 - \gamma), a_i, b_i \geq 0$, it's also true that $c_i \geq 0$. Finally, we notice that:

$$\begin{aligned} \sum_{i=1}^{n} c_i &= \sum_{i=1}^{n} (\gamma a_i + (1-\gamma) b_i) \\ &= \gamma \sum_{i=1}^{n} a_i + (1-\gamma) \sum_{i=1}^{n} b_i \\ &= \gamma \cdot 1 + (1-\gamma) \cdot 1 \\ &= 1 \end{aligned}$$

Thus, we conclude that $\tilde{C}$ is convex. In particular, we notice that for each index $i$ it holds that:

$$x_i = 0 \cdot x_1 + \ldots 0 \cdot x_{i-1} + 1 \cdot x_i + 0 \cdot x_{i+1} + \ldots 0 \cdot x_n$$

implying that $x_1, \ldots, x_n \in \tilde{C}$, meaning that $\tilde{C}$ is a convex that contains $x_1, \ldots, x_n$. Then, by definition of convex hull, it must hold that $C \subseteq \tilde{C}$.

Vice versa, we know show that $\tilde{C} \subseteq C$. Given $z \in \tilde{C}$, we know that $z = \sum_{i=1}^{n} d_i x_i$ where $\sum_{i=1}^{n} d_i = 1$ and $d_1, \ldots, d_n \geq 0$. By induction, we show that for each $m \in \mathbb{N}$ such that $1 \leq m \leq n$, if the number of coefficients greater than 0 that compose $z$ equals $m$ then it holds that $z \in C$.

For the base case, it's easy to see that if $m = 1$ then there is only one index $i$ such that $d_i > 0$, while for every other index $j \neq i$ it holds that $d_j = 0$. Then, since the sum of all the coefficients must be equal to one, this can hold only if $d_i = 1$, meaning that $z = x_i$ and thus that $z \in C$ by definition of convex hull.

Assuming that the statement holds for each point in $\tilde{C}$ with $m - 1$ coefficients greater than zero, suppose that $z$ has $m$ coefficients $i_1, \ldots, i_m$ such that $d_{i_1}, \ldots, d_{i_m} > 0$. Given $j \in \{i_1, \ldots, i_m\}$, we notice that:

$$\sum_{i=1}^{n} d_i = 1 \implies \sum_{\substack{i=1,\\ i\neq j}}^{n} d_i = 1 - d_j \implies \sum_{\substack{i=1,\\ i\neq j}}^{n} \frac{d_i}{1 - d_j} = 1$$

For each index $i$, we define $k_i$ as:

$$k_i = \begin{cases} \frac{d_i}{1-d_j} & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

Let $w = \sum_{i=0}^{n} k_i x_i$. We notice that, by definition of each $k_i$, it holds that

$$\sum_{i=0}^{n} k_i = k_j + \sum_{\substack{i=1,\\ i\neq j}}^{n} \frac{d_i}{1 - d_j} = 1$$

Moreover, since $d_{i_1}, \ldots, d_{i_m} > 0$, by definition of each $k_i$ we know that $k_{i_1}, \ldots, k_{j-1}, k_{j+1}, \ldots, k_{i_m} > 0$, meaning that $w$ has $m - 1$ coefficients greater than zero. Thus, by inductive hypothesis we know that $w \in C$.

Finally, we notice that:

$$z = \sum_{i=1}^{n} d_i x_i = d_j x_j + \sum_{\substack{i=1,\\ i\neq j}}^{n} d_i x_i = d_j x_j + (1 - d_j) w$$

meaning that $z$ is part of the segment between $x_j$ and $w$. Thus, since $C$ is convex and since $x_j, w \in C$, it must also be true that $z \in C$. Then, since we showed that this holds for each vector with any possible amount of coefficients with value greater than zero, this holds for all points in $\tilde{C}$, concluding that $\tilde{C} \subseteq C$.

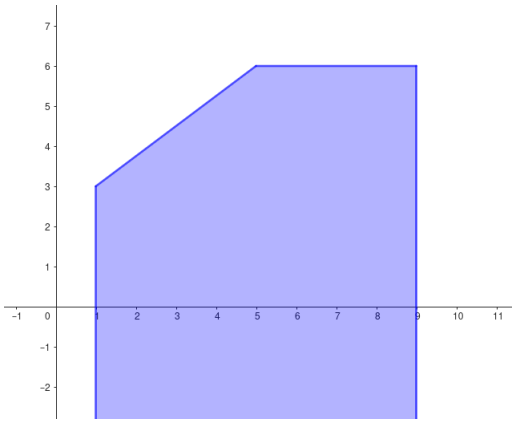
□

After giving a proper definition of convex hull, we can give a clearer picture of what's going on: given $x_1, \ldots, x_n \in \mathbb{R}^n$, assuming that the vectors are *linearly independent*, we get that:

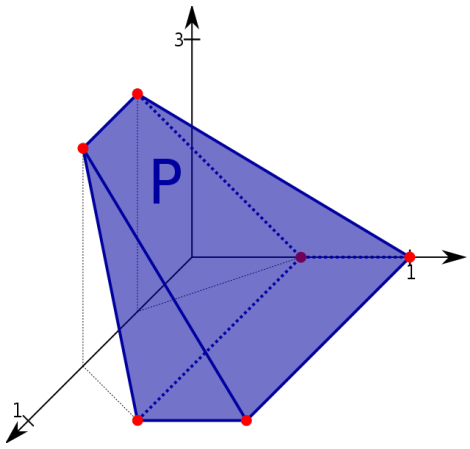
- If $n = 2$ then the set of convex combinations corresponds to a segment between $x_1$ and $x_2$
- If $n = 3$ then the set of convex combinations corresponds to a convex bounded polygon with vertices $x_1, x_2$ and $x_3$
- If $n = 4$ then the set of convex combinations corresponds to a convex bounded polyhedron with vertices $x_1, x_2, x_3$ and $x_4$

More generally, for $n$ vectors we get a $n$ dimensional **convex bounded $n$-polytope**, that being a generalization to $n$ dimensions of the polygon and polyhedron. In fact, each of the listed examples correspond respectively to a $1, 2, 3$ and $4$.

However, in the following sections we will use **different definitions**: we will define any convex unbounded polytope as a *convex polyhedron*, while we will reserve the term *polytope* only for convex bounded polytopes (i.e: the previous figure would just be called a 3 dimensional polytope, even thought in reality it should be called a convex bounded polyhedron).



Convex 2-polyhedron




3-polytope

### 2.4.2 Hyperplanes, Half-Spaces and Polytopes

**Definition 19: Hyperplane**

An **hyperplane** of an $n$ dimensional space is a subspace of dimension $n-1$.

In other words, given $a_1, \ldots, a_n \neq 0 \in \mathbb{R}$, an hyperplane is the set of all solutions to the equation $a_1x_1 + \ldots + a_1x_n = 0$

For example, in a plane (a 2 dimensional space) an hyperplane would correspond to a line passing through the origin of that plane (a 1 dimensional subspace). Likewise, in a 3D-space an hyperplane would correspond to a plane that intersects the origin. Formally, these hyperplanes would be described by the equations $a_1x_1 + a_2x_2 = 0$ (a line) and $b_1x_1 + b_2x_2 + b_3x_3 = 0$ (a plane).

**Definition 20: Affine subspace**

Let $W \subseteq V$ be a subspace of a space $V$. We say that $U \subseteq V$ is an **affine subspace** if $\exists v \in V$ such that:

$$U = \{v + w \mid w \in W\}$$

In other words, an affine subspace is a *translation* of a subspace. Moreover, the subspace $W$ is called the **direction** of $U$.

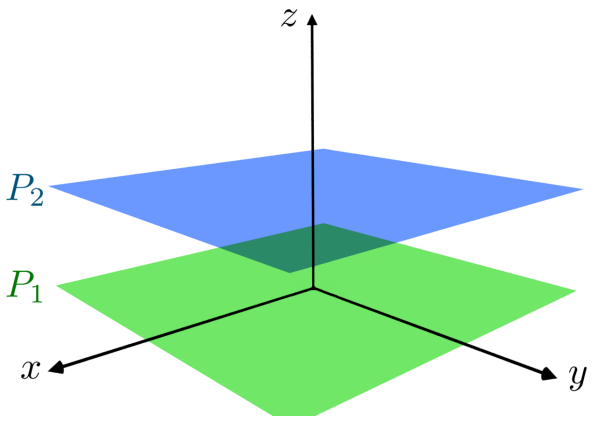



Figure 2.8: $P_2$ is an affine subspace with $P_1$ as its direction

In particular, we notice that an affine subspace is not a subspace since it doesn't satisfy all properties (i.e: an affine subspace hasn't got a 0 vector since it got "translated").

**Definition 21: Affine hyperplane**

An **affine hyperplane** is an affine subspace of dimension $n-1$.

In other words, given $a_1, \ldots, a_n, b \neq 0$, an affine hyperplane is the set of all solutions to the equation $a_1x_1 + \ldots + a_nx_n = b$

Affine hyperplanes ofter get directly called *hyperplanes*. This is due to the fact that every affine hyperplane corresponds to what intend with such concept. For example, in the case of $\mathbb{R}^n$, the line $r : x_2 = x_1 + 6$ can also be rewritten as $r : x_2 - x_1 = 6$ showing that it is in fact an affine hyperplane with direction given by $r' : x_2 - x_1 = 0$.

**Definition 22: Half space**

Given an euclidean space (such as $\mathbb{R}^n$), an **half space** is either of the two parts into which an affine hyperplane divides an affine space.

In particular, given the (affine) hyperplane $a_1x_1 + \ldots + a_nx_n = b$, the space gets split into the two following half spaces:

$$H_1 = \{x \in \mathbb{R}^n \mid a_1x_1 + \ldots + a_nx_n \leq b\}$$

$$H_2 = \{x \in \mathbb{R}^n \mid a_1x_1 + \ldots + a_nx_n \geq b\}$$

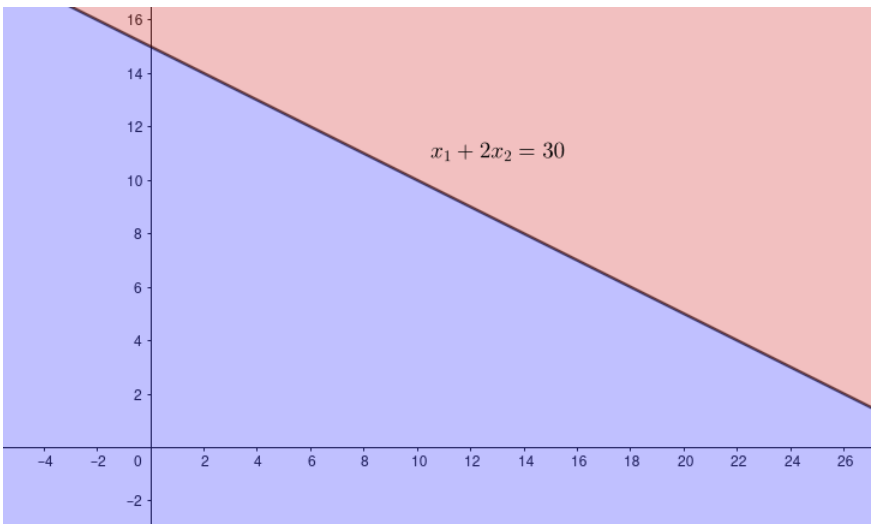



Figure 2.9: The (affine) plane $x_1 + 2x_2 = 30$ splits the plane into two half spaces

**Observation 9**

An hyperplane and its half spaces are all **convex**

*Proof.*

The proof is the same as the proof of Proposition 8, except there is only one equation involved.

□

**Definition 23: Convex polyhedron and Polytopes**

A **convex polyhedron** is the intersection of finitely many half spaces. If the polyhedron is bounded it is called a **polytope**. The dimension of a convex polyhedron is the minimal dimension of the affine spaces that define it.

Given a linear program in standard form, it's easy to see that each equation of the system $Ax = b$ defines an hyperplane that divides $\mathbb{R}^n$ into half spaces. Since each feasible solution must satisfy each of the constraints, it must be in one of the two half spaces defined by each row. In particular, this tells us that the **feasible region** is in fact a **convex polyhedron** (or a *polytope* if it's bounded). This relation formally defines the very initial intuitions given in the first section of this chapter.

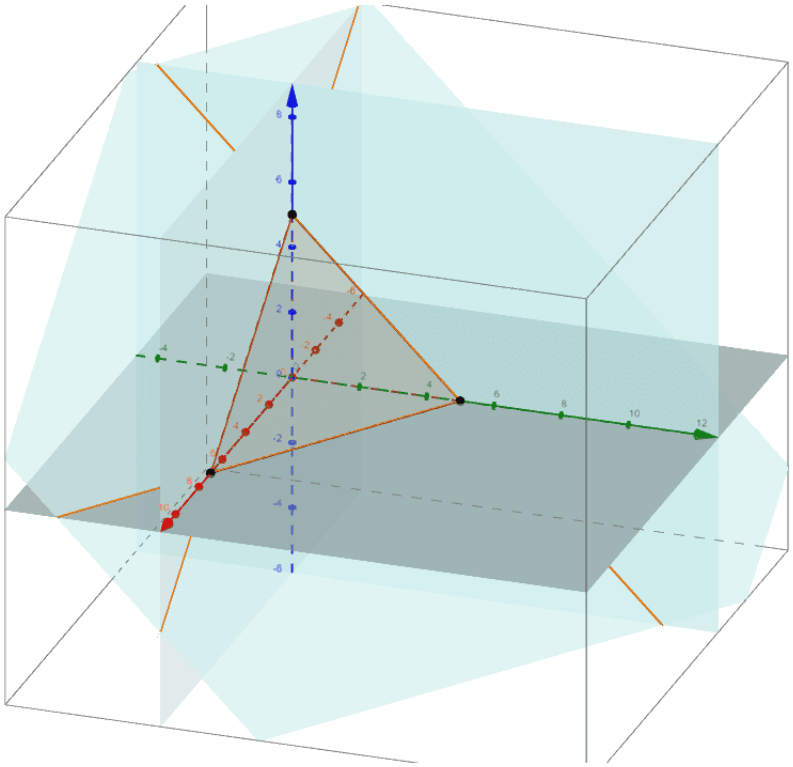


Figure 2.10: The feasible region (red) of a linear program as a polytope generated by the constraint matrix's hyperplanes (light blue)

### 2.4.3 Vertices and Basic Feasible Solutions

We have already mentioned how each basic feasible solution can be viewed as a *vertex* of the feasible region. After defining the geometrical concept of *convex polyhedron* and showing how the feasible region is in fact a convex polyhedra, we are now ready to formally justify this connection between vertices and basic feasible solutions.

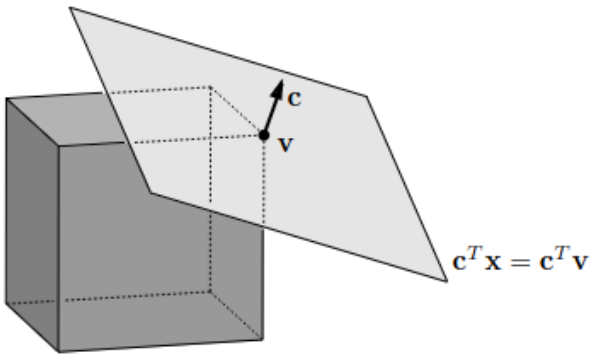
**Definition 24: Face**

A **$k$-dimensional face** of a convex polyhedron $P$ is a subspace $F \subset P$ of dimension $k$ such that there is an (affine) hyperplane $H = \{x \in \mathbb{R}^n \mid c^T x = \beta\}$ for which:

- $F = P \cap H$
- For each $y \in P - F$ it holds that $c^T y < \beta$

In other words, there is an (affine) hyperplane that touches the polyhedron exactly on $F$, where *touch* means that it doesn't divide it.

Given this definition, it's easy to see that a *vertex* corresponds to a 0-dimensional face, while an *edge* is a 1-dimensional face and what is commonly known as *face* is a 2-dimensional face. In particular, if $F$ is a 0-dimensional face then there is only one point $v$ such that $F = \{v\}$. Moreover, since $F = P \cap H$ we also get that $v$ is the only point in $P$ for which $c^T v = \beta$, while for all the other points $y \in P$ it holds that $c^T y < c^T v$.




**Theorem 7: Vertices and BFSs**

Let $P$ be the feasible region of a linear program in standard form. Given $v \in P$, it holds that:

$$v \text{ is a vertex of } P \iff v \text{ is a BFS}$$

*Proof.*

*First implication.*

Suppose that $v$ is a vertex of $P$, meaning that there is an hyperplane $H = \{x \in \mathbb{R}^n \mid c^T x = \beta\}$ such that $\{v\} = P \cap H$ and $\forall y \in P - \{v\}$ it holds that $c^T y < \beta$.

Given the constraint matrix $Ax = b$ of the original linear problem, we consider the following linear problem:

$$\begin{aligned} \max\ & c^T x \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

In particular, we notice that both problems share the same constraint matrix, they also share the same feasible region. However, since $\forall y \in P - \{v\}$ it holds that $c^T y < c^T v = \beta$, we get that $v$ is an optimal solution to this new linear problem (not of the original problem).

As shown in Theorem 6, since $v$ is an optimal solution there must also exist an optimal BFS $v'$. Thus, since $c^T v = \beta = c^T v'$ in order to both be optimal solutions and since $v$ is the only point in $P$ such that $c^T v = \beta$, we get that $v = v'$, meaning that $v$ is an optimal BFS of this new problem.

Let $\mathcal{B}$ be a basis that certifies that $v$ is a BFS for the second problem. Since both problems share the same constraint matrix $Ax = b$, the basis $\mathcal{B}$ certifies that $v$ is also a BFS for the original problem (even though it is not optimal).

*Second implication.*

Suppose that $v$ is a BFS with basis $\mathcal{B} \subseteq \{1, \ldots, n\}$. Let $c$ be the vector defined as:

$$c_i = \begin{cases} 0 & \text{if } i \in \mathcal{B} \\ -1 & \text{if } i \notin \mathcal{B} \end{cases}$$

By definition of basis we have that $v_i > 0$ if $i \in B$ and $v_i = 0$ if $i \notin \mathcal{B}$, we get that $c^T v = 0$ since each component $c_i v_i$ gets nullified by the coefficient $c_i$ or by the value $v_i$. Instead, for all other vectors $y \in P$, we get that $c^T y \leq 0$ since each component $c_j y_j$ such that $j \notin B$ gets negated.

Let $w \in P$ such that $c^T w = 0$. In order for this to be true, due to the way $c$ is defined it must hold that $w_i = 0$ for each index $i \notin \mathcal{B}$ so we get that $\mathcal{B}$ is also a basis for $w$, meaning that $w$ is also a BFS. However, as shown in Observation 6, there can only be one BFS for each feasible basis, meaning that $v = w$ must be true.

Thus, we conclude that $v$ is the only point in $P$ such that $c^T v = 0$, while for all $y \in P - \{v\}$ it holds that $c^T y < 0$, implying that the hyperplane $H = \{x \in \mathbb{R}^n \mid c^T x = 0\}$ concludes that $v$ is a vertex of $P$.

□

**Corollary 5**

The feasible region of a linear program is the **convex hull** of its vertices

## 2.5 Solved exercises

**Problem 1**

Consider the following linear program:

$$\max\ x_1 + 2x_2 - 3x_3 + 7x_5$$

$$\begin{aligned} x_1 + 2x_2 + 2x_3 + x_4 &= 3 \\ x_1 + 2x_2 + 7x_3 + x_5 &= 3 \\ 2x_1 + 4x_2 + 7x_3 + x_6 &= 6 \\ x_1, x_2, x_3, x_4, x_5, x_6 &\geq 0 \end{aligned}$$

Determine and justify which of the following vectors is a basic feasible solution (BFS):

1. $a = (1, 1, 0, 0, 0, 0)$
2. $b = (1, 0, 0, 2, 2, 4)$
3. $c = (0, 0, 0, 3, 3, 6)$

*Solution:*

First, we notice that all the proposed solutions are feasible. By definition, we know that a feasible solution $\overline{x}$ is basic if there exists a basis $\mathcal{B}$ such that $A_{\mathcal{B}}$ is a $m \times n$ non-singular matrix and $\forall i \notin \mathcal{B}$ it holds that $\overline{x_i} = 0$.

1. By way of contradiction, suppose that there is a basis $\mathcal{B}_a$ that certifies $a$. Since $a_1, a_2 > 0$, we have that $\{1, 2\} \subseteq \mathcal{B}_a$. However, the columns $A^1, A^2$ are linearly dependent, meaning that $A_{\mathcal{B}_a}$ is singular, contradicting the fact that $\mathcal{B}_a$ is a feasible basis. Thus, the feasible solution $a$ cannot be basic.
2. By way of contradiction, suppose that there is a basis $\mathcal{B}_b$ that certifies $b$. Since $b_1, b_4, b_5, b_6 > 0$, we have that $\{1, 4, 5, 6\} \subseteq \mathcal{B}_b$, implying that $|B_b| \geq 4$. In order for $\mathcal{B}_b$ to be a feasible basis, $A_{\mathcal{B}_b}$ must be a $m \times n$ matrix, $|\mathcal{B}_b| = m$ must hold. Thus, since in this case we have that $m = 3$ and $|B_b| > 3$, we get a contradiction. Thus, the feasible solution $b$ cannot be basic.
3. Let $\mathcal{B}_c = \{4, 5, 6\}$. We notice that $|\mathcal{B}_c| = 3$ and that $A_{\mathcal{B}_c} = I_m$, implying that it is a non-singular matrix. Thus, we get that $\mathcal{B}_c$ certifies that $c$ is a BFS.

**Problem 2**

Given a general linear program in equational form

$$\begin{aligned} \max\ & c^T x \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

let $\overline{x}$ and $\overline{y}$ be two optimal solutions to the problem. Prove or disprove the following statements:

1. Any convex combination of $\overline{x}$ and $\overline{y}$ is also an optimal solution to the linear program
2. There exists an optimal solution $\overline{z}$ on the line through $\overline{x}$ and $\overline{y}$ which is also basic

*Solution:*

First, we recall that any convex combination of the points $x_1, \ldots, x_n$ is a point $\overline{z}$ such that

$$\overline{z} = \sum_{i=1}^{n} \alpha_i x_i$$

where $\sum_{i=1}^{n} \alpha_i = 1$. Thus, for two points $\overline{x}$ and $\overline{y}$ we have that $\overline{z} = \alpha\overline{x} + (1-\alpha)\overline{y}$ for some value $\alpha \in [0,1] \subseteq \mathbb{R}$. Let the constraints $Ax = b$ be defined as

$$\begin{aligned} a_{1,1}x_1 + \ldots + a_{1,n}x_n &= b_1 \\ &\vdots \\ a_{m,1}x_1 + \ldots + a_{m,n}x_n &= b_m \end{aligned}$$

Since $\overline{x}$ and $\overline{y}$ are feasible, we notice that:

$$\begin{aligned} a_{1,1}\overline{z}_1 + \ldots + a_{1,n}\overline{z}_n &= a_{1,1}(\alpha\overline{x}_1 + (1-\alpha)\overline{y}_1) + \ldots + a_{1,n}(\alpha\overline{x}_n + (1-\alpha)\overline{y}_n) \\ &\vdots \qquad\qquad \vdots \\ a_{m,1}\overline{z}_1 + \ldots + a_{m,n}\overline{z}_n &= a_{m,1}(\alpha\overline{x}_1 + (1-\alpha)\overline{y}_1) + \ldots + a_{m,n}(\alpha\overline{x}_n + (1-\alpha)\overline{y}_n) \end{aligned}$$

$$\begin{aligned} = \alpha(a_{1,1}\overline{x}_1 + \ldots + a_{1,n}\overline{x}_n) + (1-\alpha)(a_{1,1}\overline{y}_1 + \ldots + a_{1,n}\overline{y}_n) &= \alpha b_1 + (1-\alpha)b_1 = b_1 \\ \vdots \qquad\qquad\qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \qquad\qquad \vdots \qquad \vdots \quad \vdots \\ = \alpha(a_{m,1}\overline{x}_1 + \ldots + a_{m,n}\overline{x}_n) + (1-\alpha)(a_{m,1}\overline{y}_1 + \ldots + a_{m,n}\overline{y}_n) &= \alpha b_m + (1-\alpha)b_m = b_m \end{aligned}$$

implying that $\overline{z}$ indeed is a feasible solution. Moreover, by linearity of the objective function we have that:

$$c^T\overline{z} = c^T(\alpha\overline{x} + (1-\alpha)\overline{y}) = \alpha c^T\overline{x} + (1-\alpha)c^T\overline{y} = \alpha k + (1-\alpha)k = k$$

where $k$ is the optimal value of the linear program, concluding that any convex combination $\overline{z}$ is an optimal solution.

To disprove the second point, we ignore the cases where $\overline{x}$ (or $\overline{y}$) is a BFS, since it would obviously imply that the statement is true. Thus, suppose that $\overline{x}$ and $\overline{y}$ are two optimal solutions but none of them is a BFS. By way of contradiction, suppose that there exists an optimal BFS $\overline{w}$ such that $\overline{w} = \alpha\overline{x} + (1-\alpha)\overline{y}$, where $\alpha \in [0,1] \subseteq \mathbb{R}$. We notice that $0 < \alpha < 1$, since otherwise we would have that $\overline{w} = \overline{x}$ or $\overline{z} = \overline{y}$, implying that $\overline{x}$ or $\overline{y}$ is an optimal BFS, which we assumed to not be true.

Since $\overline{w}$ is a BFS if and only if it is a vertex of the polyhedron $P$ of the feasible solutions, there must be an hyperplane $H = \{x \in \mathbb{R}^n \mid d^T x = d^T\overline{w}\}$ such that $\{\overline{w}\} = P \cap H$ and $\forall y \in P - \{\overline{w}\}$ it holds that $d^T y < d^T\overline{w}$, where $d \in \mathbb{R}^n$. In particular, we have that:

$$d^T\overline{w} = d^T(\alpha\overline{x} + (1-\alpha)\overline{y}) = \alpha d^T(\overline{x} - \overline{y}) + d^T\overline{y}$$

However, we notice that this can be true only if $d^T\overline{x} < d^T\overline{w} < d^T\overline{y}$ or if $d^T\overline{y} < d^T\overline{w} < d^T\overline{x}$, implying that $\overline{y} \notin P$ or that $\overline{x} \notin P$ must hold and thus that $\overline{y}$ or $\overline{x}$ aren't feasible solution, which is absurd since they are optimal solutions. Thus, the only possibility is that $\overline{w}$ cannot be a BFS, concluding that the statement is true only if $\overline{x}$ or $\overline{y}$ are a BFS.

To give a geometrical interpretation of the contradiction, we can imagine that the hyperplane splits in two parts the segment of optimal solutions given by the convex combinations of $\overline{x}$ and $\overline{y}$, cutting off one of the two parts from the polyhedron.

# 3

# The Simplex method

## 3.1 Intuition and examples

After analyzing the algebraic and a geometrical properties of the concepts involving a linear program, we know that:

- The feasible region is a *convex polyhedron*
- The vertices of the feasible region correspond to the BFSs of the linear program
- There are at most $\binom{n}{m}$ BFSs in a linear program
- If there is an optimal solution then there is an optimal BFS

Given these results, a simple (*pun intended*) but effective idea is given by the **Simplex method**, extensively researched by George Dantzig: since the number of BFS is finite, we can move from one vertex of the polyhedron to the other increasing the objective function value until we reach an optimal BFS (if there is an optimal solution).

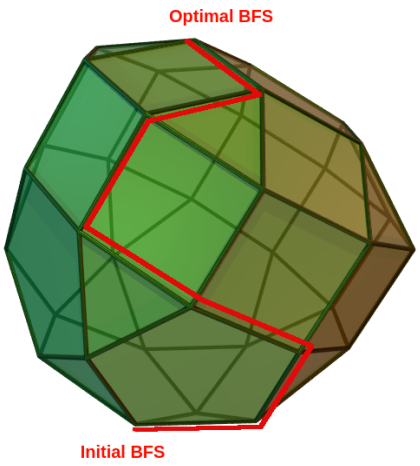



Figure 3.1: Example of execution of the simplex algorithm

The term *Simplex method* comes from the idea that the algorithm, when viewed in the geometry of the columns, is best described as a movement from one simplex to a neighboring one. In fact, a *simplex* is the $n$-dimensional generalization of the triangle and the tetrahedron and, just like in 2 and 3 dimensional spaces, each polyhedron can be described as a set of simplices.

We will now show an example of the application of the Simplex method in order to give a general understanding. Consider the following linear program:

$$\begin{aligned} \max\, & x_1 + x_2 \\ -x_1 + x_2 &\leq 1 \\ x_1 &\leq 3 \\ x_2 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

First, we convert this linear program in standard form by introducing the slack variables $x_1, x_2$ and $x_3$:

$$\begin{aligned} \max\, & x_1 + x_2 \\ -x_1 + x_2 + x_3 &= 1 \\ x_1 + x_4 &= 3 \\ x_2 + x_5 &= 2 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 \end{aligned}$$

We notice that the constraint matrix $A$ associated with this program corresponds to:

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

As discussed, the Simplex method moves from a BFS to another, increasing the value of the objective function. Since we added 3 slack variables, it's easy to see $\mathcal{B}_1 = \{3, 4, 5\}$ is a feasible basis since the columns $A^3, A^4$ and $A^5$ are linearly independent. Now, we procede by writing the following tableau where the constraints of $A$ are rewritten by solving for the basic variables in terms of the non-basic ones:

$$\begin{array}{rcrrr} x_3 & = & 1 & +x_1 & -x_2 \\ x_4 & = & 3 & -x_1 & \\ x_5 & = & 2 & & -x_2 \\ \hline z & = & & +x_1 & +x_2 \end{array}$$

where $z$ is the current value of the objective function. From this tableau, it's easy to see that $s_1^T = \begin{bmatrix} 0 & 0 & 1 & 3 & 2 \end{bmatrix}$ is a valid BFS since each basic variable is independent by the others. In particular, this BFS has an objective function value equal to $z = 0$.

Now, we can increase the value of the objective function by picking a variable in the equation $z = x_1 + x_2$ with **positive coefficient**. Otherwise, if we picked a variable with negative coefficient, the value would decrease. The chosen variable is called **pivot**.

For our example, we will pick $x_2$ as the first pivot. We notice that the constraint $x_3 = 1 + x_1 - x_2$ tells us that we can increase $x_2$ by at most 1 before $x_3$ becomes negative (since $x_1$ will stay fixed to 0), violating the constraint $x_3 \geq 0$. Likewise, the constraint $x_5 = 2 - x_2$ tells us that we can increase $x_2$ by at most 2, while the constraint $x_4 = 3 - x_1$ tells us that we can increase $x_2$ as much as we please.

To ensure that every constraint is still satisfied, we must pick the most restrictive possible increase, so we increase $x_2$ by 1 giving us $x_2 = 1$, while $x_1$ stays fixed to $x_1 = 0$. Thus, we also get that:

- $x_3 = 1 + x_1 - x_2 = 1 + 0 - 1 = 0$
- $x_4 = 3 - x_1 = 3 - 0 = 3$
- $x_5 = 2 - x_2 = 2 - 1 = 1$

These results give us a new feasible solution $s_2^T = \begin{bmatrix} 0 & 1 & 0 & 3 & 1 \end{bmatrix}$. Since now $x_2 > 0$ and $x_3 = 0$, we can effectively say that this method *pivoted* 2 into the basis while removing 3. In fact, the basis $\mathcal{B}_2 = \{2, 4, 5\}$ certifies that $s_2$ is in fact a BFS.

Now, we procede by writing a new tableau with the same logic as the previous one.

$$\begin{array}{rcrrr} x_2 & = & 1 & +x_1 & -x_3 \\ x_4 & = & 3 & -x_1 & \\ x_5 & = & 1 & -x_1 & +x_3 \\ \hline z & = & 1 & +2x_1 & -x_3 \end{array}$$

Note: *we replaced $x_2$ with $1 + x_1 - x_3$ in the equations of $x_4$ and $x_5$ since the tableau must be written in terms of the non-basic variables.*

We notice that the new BFS $s_2$ gives us the objective value $z = 1$, meaning that it increased. Proceeding with the same method as before, we pick the next variable with non-negative coefficient. In this case, $x_1$ is the only possible variable and it can be increased by 1. Thus, we get the new BFS $s_3^T = \begin{bmatrix} 1 & 2 & 0 & 2 & 0 \end{bmatrix}$ with basis $\mathcal{B}_3 = \{1, 2, 4\}$, meaning that 5 got pivoted with 1.

Then, we write the new tableau:

$$\begin{array}{rcrrr} x_1 & = & 1 & +x_3 & -x_5 \\ x_2 & = & 2 & & -x_5 \\ x_4 & = & 2 & -x_3 & +x_5 \\ \hline z & = & 3 & +x_3 & -2x_5 \end{array}$$

telling us that the value of the objective function is now equal to $z = 3$. As before, we now increase $x_3$ by 2, giving us the BFS $s_4^T = \begin{bmatrix} 3 & 2 & 2 & 0 & 0 \end{bmatrix}$ with basis $\mathcal{B}_4 = \{1, 2, 3\}$ and the following new tableau:

$$\begin{array}{rcrrr} x_1 & = & 3 & -x_4 & \\ x_2 & = & 2 & & -x_5 \\ x_3 & = & 2 & -x_4 & +x_5 \\ \hline z & = & 5 & -x_4 & -x_5 \end{array}$$

with objective function value equal to $z = 5$. We now notice that all the coefficients in the $z$ equation are negative, implying that we can't pick a new variable to be increased. Since this system of equalities is equivalent to the original one, each feasible solution of original problem must also satisfy this tableau, implying that the value of the objective function will always be at most 5.

Thus, we conclude that $s_4$ is in fact an **optimal BFS** with objective function value equal to 5. Since $s_4$ is an optimal solution of the standard form linear program, we can ignore the values of the slack variables $x_3, x_4, x_5$ inside of $s_4$, telling us that $\begin{bmatrix}3 & 2\end{bmatrix}$ is in fact the optimal solution of the original linear program.

To give a geometrical interpretation of what we have done, consider the restriction to $x_1, x_2$ of the solutions $s_1, s_2, s_3, s_4$. By plotting the feasible region of the linear program, we notice that these restrictions correspond to the vertices of the polytope described. In particular, we notice how we moved from one vertex to the other with each **pivot step**.

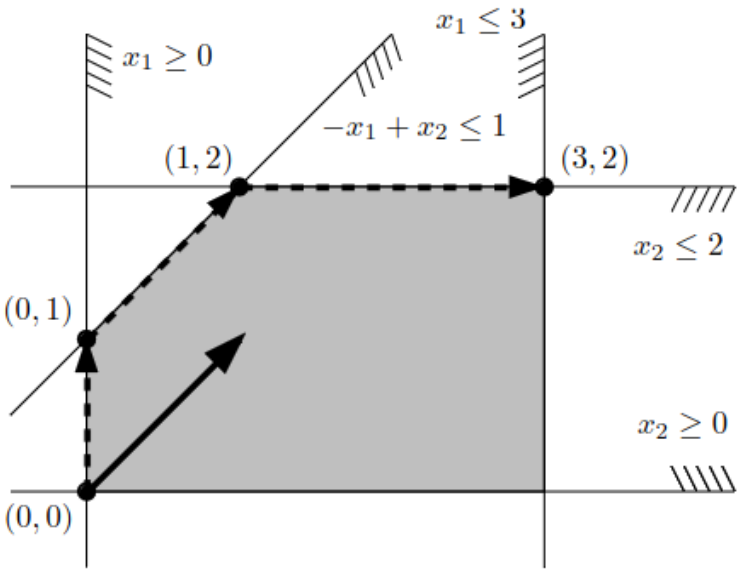



Figure 3.2: The Simplex method in action

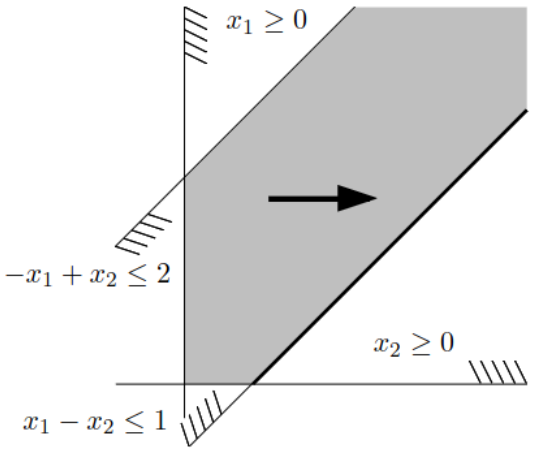
We also notice that we could've taken a "shorter route" by picking $x_1$ as the first pivot, concluding the algorithm in just two pivot steps.

## 3.2 Unboundedness

Consider the following linear program:

$$\begin{aligned} &\max x_1 \\ x_1 - x_2 &\leq 2 \\ -x_1 + x_2 &\leq 1 \\ x_1, x_2 &\geq 0 \end{aligned}$$

By plotting the constraints of this linear program, it's easy to see that the feasible region isn't a polytope, meaning that it's an unbounded convex polyhedron.




Since the problem is unbounded, we can always find a feasible solution with a greater objective function value, meaning that there is **no optimal solution**. Thus, in this case the Simplex method won't output an optimal solution. However, it will return an affine space of infinite feasible solutions with no maximum value.

First, we procede by adjusting the problem to its standard form:

$$\begin{aligned} &\max x_1 \\ x_1 - x_2 + x_3 &= 2 \\ -x_1 + x_2 + x_4 &= 1 \\ x_1, x_2 &\geq 0 \end{aligned}$$

It's easy to see that $\mathcal{B}_1 = \{3, 4\}$ is a feasible basis, giving the BFS $s_1^T = \begin{bmatrix} 0 & 0 & 2 & 1 \end{bmatrix}$ and the following simplex tableau:

$$\begin{array}{rcrrr} x_3 & = & 2 & -x_1 & +x_2 \\ x_4 & = & 1 & +x_1 & -x_2 \\ \hline z & = & & +x_1 & \end{array}$$

Now, we pivot $x_3$ with $x_1$ by setting $x_1 = 2$, giving us the basis $\mathcal{B}_2 = \{1, 4\}$ which certifies the BFS $s_2^T = \begin{bmatrix} 2 & 0 & 0 & 3 \end{bmatrix}$ and defines the following simplex tableau:

$$\begin{array}{rcrrr} x_1 & = & 2 & -x_3 & +x_2 \\ x_4 & = & 3 & -x_3 & \\ \hline z & = & 1 & -x_3 & x_2 \end{array}$$

The next step of the Simplex method requires that we pick $x_2$ as the next pivot. However, we notice that we can increase $x_2$ as much as we want since no constraint gets violated. This means that $\forall t \in \mathbb{R}^+$ by setting $x_2 = t$ we obtain a valid feasible solution with a larger and larger objective function value.

This result gives us a way to generate infinite feasible solutions: $\forall t \in \mathbb{R}^+$ it holds that $\begin{bmatrix} 1+t & t & 0 & 3 \end{bmatrix}^T$ is a feasible solution to the linear problem. In particular, this general solution defines the following affine subspace of feasible solutions:

$$X = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 3 \end{bmatrix} + t \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \;\middle|\; t \in \mathbb{R}^+ \right\}$$

*Note: this affine subspace corresponds to the black bold line in the previous image*

## 3.3 Infeasibility of the trivial basis

As previously discussed, in order to start the Simplex method we need an **initial feasible basis** through which we can define the first simplex tableau. In the examples shown until now, in the constraint inequalities defined by $Ax \leq b$ we had that $b \geq 0$, giving us a **trivial initial feasible basis** by simply choosing the slack variables as the basic variables: if the linear program is in **non-standard form**, we add the slack variables $x_{n+1}, \ldots, x_{n+m}$ automatically implying that $A^{n+1}, \ldots, A^{n+m}$ are linearly independent and that $\mathcal{B} = \{n+1, \ldots, n+m\}$ is a basis. Thus, if $b \geq 0$, the equation $A_{\mathcal{B}} x = b$ gives the trivial BFS $s = \begin{bmatrix} 0 & \ldots & 0 & b_1 & \ldots & b_m \end{bmatrix}$.

**Observation 10**

Given a linear program in non-standard form, meaning that $Ax \leq b$, if $b \geq 0$ then there exists a **trivial BFS**

However, if the condition $b \geq 0$ is not true, the basis $\mathcal{B}$ previously defined is not feasible. Thus, we need to find another method to get the initial BFS in non-trivial cases. The following propositions will give us a way to find such BFS.

**Proposition 10: Feasibility equivalent to optimality**

Checking wether a convex polyhedron $P = \{x \in \mathbb{R}^n \mid Ax = b\}$ is non-empty is **computationally equivalent** to optimizing a linear program defined as:

$$\begin{aligned} &\max c^T x \\ &x \in P \end{aligned}$$

for some $c \in \mathbb{R}^n$

*Proof.* First, we show that deciding that $P \neq \varnothing$ is reducible to deciding if the following linear program has an optimal solution:

$$\begin{aligned} &\max 0^T x \\ &x \in P \end{aligned}$$

By the own definition of the problem, if $P \neq \varnothing$ then there is a feasible solution. We notice that $0^T x \leq 0$ for all $x \in \mathbb{R}^n$ and in particular $0^T \overline{x} = 0$ for each feasible solution $\overline{x}$, implying that the objective function has an upper bound. Thus, by Theorem 6, we conclude that if $P \neq \varnothing$ then there is an optimal solution to the problem.

Moreover, again by definition of the problem, any feasible solution of the linear program must satisfy the condition $x \in P$. Thus, if there is an optimal solution (which obviously is also a feasible solution) then $P \neq \varnothing$, concluding that:

$$P \neq \varnothing \iff \begin{aligned} &\max 0^T x \\ &x \in P \end{aligned} \text{ has an opt. sol}$$

Now, we show that deciding if the following linear program has an optimal solution is reducible to deciding if $P \neq \varnothing$. Consider the following linear program:

$$\begin{aligned} &\max c^T x \\ &x \in P \end{aligned}$$

where $c \in \mathbb{R}^n$.

Suppose that we have a subroutine that checks if a polyhedron is non-empty. For all $\alpha \in \mathbb{R}$, let $P_\alpha = \{x \in P \mid c^T x \geq \alpha\}$. We notice that if $P_\alpha \neq \varnothing$ then the optimal solution has an objective function value of at least $\alpha$. Otherwise, if $P_\alpha = \varnothing$ then the optimal solution has an objective function value less than $\alpha$.

Through the subroutine, we can use binary search to find the optimal solution: if $P_\alpha \neq \varnothing$ we double the value of $\alpha$ and call again the subroutine, otherwise we halve the value of $\alpha$. Once this increment/halving can't be repeated anymore, we will have determined the value $\alpha$ corresponding to the value given by the optimal solution of $P$. Thus, for the final value $\alpha$ it holds that $P_\alpha = P$, implying that the last call of the subroutine determines if $P \neq \varnothing$. (*Note:* the last part of the proof is not very rigorous since we would have to show that the number of subroutine calls is bounded in terms of $m, n$ and $b$). □

Essentially, this proposition tells us that finding a feasible solution is as hard as finding an optimal solution. This results gives us a way to find a **non-trivial feasible basis** through the Simplex method itself: instead of finding a valid BFS by a direct method, we can solve another *longer* linear program to get such BFS, keeping the computational complexity of the algorithm unchanged.

In particular, since a BFS is a feasible solution with the maximum amount of zero variables, we can define a **bounded auxiliary linear program** that always has a trivial feasible basis and that **minimizes the number of non-zero variables**. By definition, any optimal solution to this auxiliary program will give us a valid BFS to the original program. This auxiliary program can be defined through adding *another set of slack variables* and slightly altering the constraints.

**Theorem 8: Constructing an initial BFS**

Consider the following linear program in standard form:

$$\begin{aligned} \max\ & c_1x_1 + \ldots + c_nx_n \\ & a_{1,1}x_1 + \ldots + a_{1,n}x_n = b_1 \\ & \vdots \\ & a_{m,1}x_1 + \ldots + a_{m,n}x_n = b_m \end{aligned}$$

and the following auxiliary linear program:

$$\begin{aligned} \max\ & -x_{n+1} - \ldots - x_{n+m} \\ & a_{1,1}x_1 + \ldots + a_{1,n}x_n \bullet x_{n+1} = b_1 \\ & \vdots \\ & a_{m,1}x_1 + \ldots + a_{m,n}x_n \bullet x_{n+m} = b_m \end{aligned}$$

where each $\bullet$ is a $+$ if $b_i \geq 0$ or a $-$ if $b_i < 0$.

The auxiliary program always has a trivial initial feasible basis and an optimal BFS. Moreover, the restriction to $x_1, \ldots, x_n$ of such BFS corresponds to a BFS of the original program.

*Proof.*

Let $P$ and $P'$ respectively be the convex polyhedrons defined by the original problem and the auxiliary problem. By definition of the latter, the trivial basis $\mathcal{B} = \{n+1, \ldots, n+m\}$ certifies the BFS $\overline{x} = \left[0, \ldots, 0, |b_1|, \ldots, |b_m|\right]$.

Since $\forall x \in \mathbb{R}^{n+m}$ it holds that $-x_{n+1} - \ldots - x_{n+m} \leq 0$, the objective function has an upper bound. Thus, since the objective function has an upper bound and since $\overline{x} \in P'$ implies that $P' \neq \varnothing$, by Theorem 6 we know that there is always an optimal BFS to the auxiliary program.

Let $y = \begin{bmatrix} y_1 & \dots & y_{n+m} \end{bmatrix}$ be an optimal BFS of the auxiliary linear program. Since for every optimal solution $x$ of the linear program it must hold that $-x_{n+1} - \dots - x_{n+m} = 0$, we get that $y = \begin{bmatrix} y_1 & \dots & y_n & 0 & \dots & 0 \end{bmatrix}$.

Let $\mathcal{B}'$ be a feasible basis that certifies that $y$ is a BFS. Since $y_{n+1} = \dots = y_{n+m} = 0$, in order for $y$ to be a BFS it must hold that $\mathcal{B}' \subseteq \{1, \dots, n\}$. Thus, the basis $\mathcal{B}'$ also certifies a BFS for the original linear program and, in particular, such BFS is given by $\begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$.

□

To better understand the previous theorem, we'll show an example. Consider the following linear program:

$$\begin{aligned} \max\, & x_1 + 2x_2 \\ & x_1 - 3x_2 \leq -2 \\ & x_1 - x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

and its corresponding standard form:

$$\begin{aligned} \max\, & x_1 + 2x_2 \\ & x_1 - 3x_2 + x_3 = -2 \\ & x_1 - x_2 + x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

We notice that the basis $\mathcal{B} = \{3, 4\}$ doesn't give us a trivial BFS. Thus, we define the following auxiliary linear program:

$$\begin{aligned} \max\, & -x_5 - x_6 \\ & x_1 - 3x_2 + x_3 - x_5 = -2 \\ & x_1 - x_2 + x_4 + x_6 = 1 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{aligned}$$

It's easy to see that by definition of the program itself it holds that the objective function is bounded by 0 (obtained when $x_5, x_6 = 0$). Thus, applying the Simplex method on this program always returns an optimal BFS. Moreover, this program has the trivial feasible basis $\mathcal{B}'_1 = \{5, 6\}$, giving the BFS $s'_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 1 \end{bmatrix}$ and the following tableau:

$$\begin{array}{rcrrrrr} x_5 & = & 2 & +x_1 & -3x_2 & +x_3 & \\ x_6 & = & 1 & -x_1 & +x_2 & & -x_4 \\ \hline z & = & \text{-3} & & +2x_2 & -x_3 & +x_4 \end{array}$$

*Note:* remember that the objective value $z$ is expressed in terms of the non-basic variables

Now, we pivot $x_2$ with $x_5$ by setting $x_2 = \frac{2}{3}$, obtaining the feasible basis $\mathcal{B}'_2 = \{2, 6\}$ with the BFS $s_2'^T = \begin{bmatrix} 0 & \frac{2}{3} & 0 & 0 & \frac{5}{3} & 0 \end{bmatrix}$ and the following tableau:

$$\begin{array}{rcrrrr} x_2 & = & \frac{2}{3} & +\frac{1}{3}x_1 & +\frac{1}{3}x_3 & -\frac{1}{3}x_5 \\ x_6 & = & \frac{5}{3} & -\frac{2}{3}x_1 & +\frac{1}{3}x_3 & -x_4 & -\frac{1}{3}x_5 \\ \hline z & = & -\frac{5}{3} & +\frac{2}{3}x_1 & -\frac{1}{3}x_3 & +x_4 & -\frac{2}{3}x_5 \end{array}$$

Finally, we pivot $x_1$ with $x_6$ by setting $x_1 = \frac{15}{6}$, obtaining the feasible basis $\mathcal{B}'_3 = \{1, 2\}$ with the BFS $s_3'^T = \begin{bmatrix} \frac{5}{2} & \frac{3}{2} & 0 & 0 & 0 & 0 \end{bmatrix}$ and the following tableau:

$$\begin{array}{rcrrrr} x_1 & = & \frac{5}{2} & +\frac{1}{3}x_3 & -\frac{3}{2}x_4 & -\frac{1}{2}x_5 & -\frac{3}{2}x_6 \\ x_2 & = & \frac{3}{2} & +\frac{4}{9}x_3 & -\frac{1}{2}x_4 & -\frac{11}{30}x_5 & -\frac{1}{2}x_6 \\ \hline z & = & & -\frac{1}{9}x_3 & & -\frac{11}{15}x_5 & -x_6 \end{array}$$

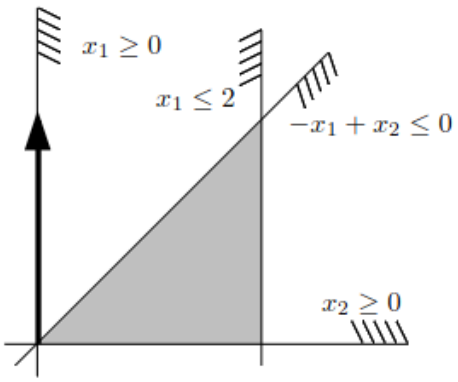
Since there are no more variables with positive coefficient, the BFS $s'_3$ must be optimal. In fact, as we said, the optimal value for this problem is 0, which is given by $s'_3$. Moreover, by own definition of the auxiliary program, restricting $s'_3$ to $x_1, x_2, x_3, x_4$ gives us a valid initial BFS $s_1 = \begin{bmatrix} \frac{5}{2} & \frac{3}{2} & 0 & 0 \end{bmatrix}$ for the original linear program. After finding such BFS, we can procede by applying the Simplex method to the original program in order to find its optimal solution.

## 3.4 Degeneracy

Consider the following linear program:

$$\begin{aligned} \max\, & x_2 \\ -x_1 + x_2 &\leq 0 \\ x_1 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

By plotting the constraints of this linear program, it's easy to see that the optimal solution is given by $s = \begin{bmatrix} 2 & 2 \end{bmatrix}$.



As usual, we start by writing the program in standard form:

$$\begin{aligned} \max\, & x_2 \\ -x_1 + x_2 + x_3 &= 0 \\ x_1 + x_4 &= 2 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned}$$

and procede by writing the first tableau using the trivial feasible basis $\mathcal{B}_1 = \{3, 4\}$:

$$\begin{array}{rcrrr} x_3 & = & & +x_1 & -x_2 \\ x_4 & = & 2 & -x_1 & \\ \hline z & = & & & +x_2 \end{array}$$

The Simplex method requires that we pivot $x_2$ with $x_1$. However, we notice that the equations of the tableau impose that $x_2$ cannot be increased since otherwise the constraint $x_3 \geq 0$ would be violated. Thus, we have to make a **degenerate pivot step**, that being a step that doesn't increment the objective function value. In fact, we get the following new tableau:

$$\begin{array}{rcrrr} x_2 & = & & +x_1 & -x_3 \\ x_4 & = & 2 & -x_1 & \\ \hline z & = & & +x_1 & -x_3 \end{array}$$

We notice that the two basis $\mathcal{B}_1 = \{3, 4\}$ and $\mathcal{B}_2 = \{2, 4\}$ **both certify the same BFS**, that being $\overline{x} = \begin{bmatrix} 0 & 0 & 0 & 2 \end{bmatrix}$. Thus, a degenerate pivot step can be interpreted as **staying on the same BFS** during a pivot step (further justifying why the objective function value doesn't increase). However, even thought we made a zero-progress step, the next pivot step where $x_4$ leaves the basis and $x_1$ enters it does still yield the optimal solution:

$$\begin{array}{rcrrr} x_1 & = & 2 & & -x_4 \\ x_2 & = & 2 & -x_3 & -x_4 \\ \hline x_2 & = & 2 & -x_3 & -x_4 \end{array}$$

Even thought in this case the Simplex method still terminated by giving the optimal solution, this isn't always the case when we are in presence of degenerate pivot steps. Intuitively, since we have multiple basis for the same BFS, the algorithm could **infinitely cycle between the basis**, meaning that the Simplex method would never terminate. Even thought this is rare, it's still a possibility. We will further discuss the topic of cycling in later sections. For now, we are just interested in formalizing the concept of degeneracy.

**Definition 25: Degenerate Pivot Step**

We say that a pivot step is **degenerate** if it doesn't increase the value of the objective function.

**Definition 26: Degenerate BFS**

Given a BFS $\overline{x}$ of a linear program, let $K = \{i \in \{1, \dots, n\} \mid x_i > 0\}$. We say that $\overline{x}$ is **degenerate** if $|K| < m$.

**Proposition 11: Non-unique certifying basis**

If a BFS is certified by **more than one basis** then such BFS is a degenerate BFS and the basic variables that aren't shared among such basis must be **set to zero**.

*Proof.* Suppose that $\overline{x}$ is certified by two basis $\mathcal{B}_1$ and $\mathcal{B}_2$. By definition of basis, we have that $\forall i \notin \mathcal{B}_1$ and $\forall j \notin \mathcal{B}_2$ it must hold that $x_i = x_j = 0$. Thus, we get that $\forall k \notin \mathcal{B}_1 \cap \mathcal{B}_2$ it must hold that $x_k = 0$.

Let $K = \{i \in \{1, \dots, n\} \mid x_i > 0\}$. It's easy to see that $K \subseteq \mathcal{B}_1 \cap \mathcal{B}_2$. Since $\mathcal{B}_1 \neq \mathcal{B}_2$, it holds that $\mathcal{B}_1 \cap \mathcal{B}_2 \neq \varnothing$. Then, since $|\mathcal{B}_1| = |\mathcal{B}_2| = m$, we conclude that $|K| \leq |\mathcal{B}_1 \cap \mathcal{B}_2| < m$ and thus that $\overline{x}$ is degenerate. □

**Observation 11**

A degenerate BFS doesn't always have more than one certifying basis

To give a **geometrical reasoning** behind degeneracy, consider again the picture shown in the previous example and the degenerate BFS $\overline{x} = \begin{bmatrix} 0 & 0 & 0 & 2 \end{bmatrix}$. We notice that the vertex $\begin{bmatrix} 0 & 0 \end{bmatrix}$ is the intersection of three constraints imposed by the linear program. We also notice that the two non-basic variables $x_1, x_2$ correspond to three of the original constraints, meaning that $x_1 \geq 0, x_2 \geq 0$ and $-x_1 + x_2 \leq 0$ are satisfied at equality. Moreover, each of these constraints define the vertex. This means that the vertex $\begin{bmatrix} 0 & 0 \end{bmatrix}$ is defined by more than one set of inequalities: the set $\{-x_1 + x_2 \leq 0, x_1 \geq 0\}$, the set $\{-x_1 + x_2 \leq 0, x_2 \geq 0\}$ and the set $\{x_1 \geq 0, x_2 \geq 0\}$.

**Proposition 12: Geometry of degeneracy**

A BFS $\overline{x}$ to the standard form $(A \mid I_m)y = b$ given by the original program $Ax \leq b$ is **degenerate** if $n$ non-basic variables correspond to the constraints in $Ax \leq b, x \geq 0$ defining the vertex of the polyhedron corresponding to $\overline{x}$.

Equivalently, the BFS $\overline{x}$ is **degenerate** if there is an $(n+1)$-st constraint of $Ax \leq b, x \geq 0$ which is tight, meaning that in $\mathbb{R}^n$ it intersects the vertex.

Both of these conditions imply that the vertex corresponding to $\overline{x}$ **isn't uniquely defined by a set of constraints** of $Ax \leq b, x \geq 0$.

**Note**: since we're talking about original and standard form together, in this case $n$ corresponds to the number of original variables and $n + m$ is the number of variables of the standard form program

## 3.5 Formalization of the Simplex method

**Definition 27: Simplex Tableau**

Given a feasible basis $\mathcal{B}$ for a linear program, a **simplex tableau** $\mathcal{T}(\mathcal{B})$ is a system of $m+1$ equations on variables $x_1, \ldots, x_m, z$ that has the same set of solutions as $Ax = b$ with $z = c^T x$ and defined as:

$$\begin{array}{rcccc} x_{\mathcal{B}} & = & p & + & Qx_{\mathcal{N}} \\ \hline z & = & z_0 & + & r^T x_{\mathcal{N}} \end{array}$$

where:

- $\mathcal{N} = \{1, \ldots, n\} - \mathcal{B}$
- $x_{\mathcal{B}} \in \mathbb{R}^m$ is the vector of basic variables
- $x_{\mathcal{N}} \in \mathbb{R}^{n-m}$ is the vector of non-basic variables
- $p \in \mathbb{R}^m$ is a vector of constants
- $Q \in \mathrm{Mat}_{m \times n-m}(\mathbb{R})$ is a coefficient matrix
- $z_0 \in \mathbb{R}$ is a constant
- $r \in \mathbb{R}^{n-m}$ is the vector of coefficients of the objective

**Lemma 7: Unique simplex tableau**

Given a feasible basis $\mathcal{B}$, there is exactly one simplex tableau $\mathcal{T}(\mathcal{B})$ such that:

- $Q = A_{\mathcal{B}}^{-1} A_{\mathcal{N}}$
- $p = A_{\mathcal{B}}^{-1} b$
- $z_0 = c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} b$
- $r^T = c_{\mathcal{N}}^T - c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} A_{\mathcal{N}}$

*Proof.*

Let $x$ be the BFS certified by $\mathcal{B}$. First, we split the matrix $Ax$ into two parts $Ax = A_{\mathcal{B}} x_{\mathcal{B}} + A_{\mathcal{N}} x_{\mathcal{N}}$. Then, since $Ax = b$ is given by the linear program, we get that $b = A_{\mathcal{B}} x_{\mathcal{B}} + A_{\mathcal{N}} x_{\mathcal{N}}$.

Since $\mathcal{B}$ is a feasible basis, the matrix $A_{\mathcal{B}}$ is non-singular, meaning that it is also always invertible. Thus, we get that

$$b = A_{\mathcal{B}} x_{\mathcal{B}} + A_{\mathcal{N}} x_{\mathcal{N}} \iff x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b - A_{\mathcal{B}}^{-1} A_{\mathcal{N}} x_{\mathcal{N}}$$

Since in a simplex tableau we must have that $x_{\mathcal{B}} = p + Qx_{\mathcal{N}}$, we can set $p := A_{\mathcal{B}}^{-1} b$ and $q := -A_{\mathcal{B}}^{-1} A_{\mathcal{N}} x_{\mathcal{N}}$ to satisfy this condition.

Now, let $z$ be the value of the objective function for $\mathcal{T}(\mathcal{B})$, implying that $z = c^T x$. Again, we can split $c^T x$ into two parts, meaning that:

$$\begin{aligned} z &= c^T x \\ &= c_{\mathcal{B}}^T x_{\mathcal{B}} + c_{\mathcal{N}}^T x_{\mathcal{N}} \\ &= c_{\mathcal{B}}^T (A_{\mathcal{B}}^{-1} b - A_{\mathcal{B}}^{-1} A_{\mathcal{N}} x_{\mathcal{N}}) + c_{\mathcal{N}}^T x_{\mathcal{N}} \\ &= c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} b - c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} A_{\mathcal{N}} x_{\mathcal{N}} + c_{\mathcal{N}}^T x_{\mathcal{N}} \\ &= c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} b + (c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} A_{\mathcal{N}} + c_{\mathcal{N}}^T) x_{\mathcal{N}} \end{aligned}$$

By setting $z_0 := c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} b$ and $r^T := c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} A_{\mathcal{N}} + c_{\mathcal{N}}^T$ we get that $z = z_0 + r^T x_{\mathcal{N}}$ as requested by the simplex tableau format.

Suppose now that there also exists $p', Q', r'^T, z_0'$ that determine a simplex tableau $\mathcal{T}'(\mathcal{B})$ for $\mathcal{B}$, implying that for each choice of $x_{\mathcal{N}}$ we have that $x_{\mathcal{B}} = p + Q x_{\mathcal{N}}$ and $x_{\mathcal{B}} = p' + Q x_{\mathcal{N}}$.

Since each $x_{\mathcal{N}}$ uniquely uniquely determines $x_{\mathcal{B}}$, for $x_{\mathcal{N}} = 0$ we get that:

$$p + Q x_{\mathcal{N}} = p' + Q' x_{\mathcal{N}} \iff p + Q \cdot 0 = p' + Q' \cdot 0 \iff p = p'$$

which also implies that $Q = Q'$. By proceeding in the same way, we can show that since $z = z_0 + r^T x_{\mathcal{N}}$ and $z = z_0' + r'^T x_{\mathcal{N}}$ is only valid if $z_0 = z_0'$ and $r^T = r'^T$, concluding that these values are unique and thus that $\mathcal{T}(\mathcal{B}) = \mathcal{T}'(\mathcal{B})$.

□

The previous lemma is needed **only to formalize the concept of simplex tableau** and to show that it is indeed unique for each feasible basis. There is no actual advantage in memorizing the way it is formulated since it is a lot easier to obtain with the procedure shown in the previous sections.

Moreover, we already observed that if $\mathcal{T}(\mathcal{B})$ is a simplex tableau such that $r \leq 0$ then the corresponding BFS is **optimal**.

**Proposition 13: Optimal solution and simplex tableaus**

If $\mathcal{T}(\mathcal{B})$ is a simplex tableau such that $r_{\mathcal{N}} \leq 0$, then the corresponding BFS given by the tableau is **optimal** and the **maximal objective function value** is $z_0$

Obviously, the previous proposition also implies that during any iteration of the Simplex method **a non-basic variable $x_e$ may enter the basis if and only if** $r_e > 0$.

Once we know which variables can enter the basis during a pivot step, we must formally define which ones can leave the basis. Given a feasible basis $\mathcal{B} \subseteq \{1, \ldots, n\}$ and $\mathcal{N} = \{1, \ldots, n\} - \mathcal{B}$, consider the following tableau $\mathcal{T}(\mathcal{B})$.

$$\begin{array}{cccccc} x_{\mathcal{B}} & = & p & + & Q x_{\mathcal{N}} \\ \hline z & = & z_0 & + & r^T x_{\mathcal{N}} \end{array}$$

Without loss of generality, let $\mathcal{B} = \{k_1, \ldots, k_m\}$ and $\mathcal{N} = \{h_1, \ldots, h_{n-m}\}$, where $k_1 < \ldots < k_m$ and $h_1 < \ldots < h_{n-m}$. Then, the $i$-th equation of the tableau is given by:

$$x_{k_i} = p_i + \sum_{j=1}^{n-m} q_{ij} x_{h_j}$$

Let $x_{h_\beta}$ be the **entering variable**, where $1 \leq \beta \leq n - m$. Since all the other non-basic variables $x_{h_j}$ such that $j \neq \beta$ should remain equal to zero, for each row $i$ the non-negativity constraint $x_{k_i} \geq 0$ limits the possible new values of $x_{h_\beta}$ by the inequality $-q_{i\beta} x_{h_\beta} \leq p_i$. In particular, if $q_{i\beta} \geq 0$ then this inequality doesn't restrict the possible value of $x_{h_\beta}$, while if $q_{i\beta} < 0$ it yields the restriction $x_{h_\beta} \leq -\frac{p_i}{q_{i\beta}}$.

Let $x_{k_\alpha}$ be the **leaving variable**, where $1 \leq \beta \leq m$. Since this variable must satisfy each possible restriction that we just described, for each row $i$ it must hold that if $q_{i\beta} < 0$ then $x_{h_\beta} \leq -\frac{p_\alpha}{q_{\alpha\beta}}$. Thus, if we want to maximize the possible value of $x_{h_\beta}$ while also keeping these restrictions valid, **the leaving variable $x_{k_\alpha}$ can be any variable such that**

$$q_{\alpha\beta} < 0 \quad \text{and} \quad -\frac{p_\alpha}{q_{\alpha\beta}} = \min\left\{-\frac{p_i}{q_{i\beta}} : q_{i\beta} < 0, 1 \leq i \leq m\right\}$$

Moreover, if there is no index $\alpha$ that satisfies these conditions, the linear program is **unbounded** since there are no restrictions on the possible value of $x_{h_\beta}$

**Lemma 8: Criteria for possible pivot steps**

Let $\mathcal{B}$ be a feasible basis. If the entering variable $x_e$ and the leaving variable $x_\ell$ have been selected according to the criteria previously described then $\mathcal{B}' = (\mathcal{B} - \{\ell\}) \cup \{e\}$ is **another feasible basis** that certifies a BFS with an objective function value greater than the previous BFS. If no $x_\ell$ satisfies the criterion for a leaving variable, the linear program is **unbounded**.

## 3.6 Pivot rules and Cycling tableaus

As discussed in the previous section, the criteria for possible pivot steps allow us to choose from different possible variables. Now, we will discuss common **pivot rules** used to choose which variables to swap from the possible ones. Each rule has a effectiveness-efficiency tradeoff.

Given the general simplex tableau:

$$\begin{array}{rcccc} x_{\mathcal{B}} & = & p & + & Qx_{\mathcal{N}} \\ \hline z & = & z_0 & + & r^T x_{\mathcal{N}} \end{array}$$

where $\mathcal{B} = \{k_1, \ldots, k_m\}$ and $\mathcal{N} = \{h_1, \ldots, h_{n-m}\}$, we can use the following rules to pick the leaving variable $x_{k_\alpha}$ and entering variable $x_{h_\beta}$ from all possible choices given by the Criteria for possible pivot steps:

1. **Largest coefficient rule**: pick $x_{k_\alpha}$ such that for all indices $k$ with $1 \leq k \leq n$ it holds that $r_{k_\alpha} \geq r_k$. Then, pick an arbitrary $x_{h_\beta}$ from the possible choices. This is the original rule suggested by Dantzig, the main researcher of the Simplex method.

2. **Largest increase rule**: pick $x_{k_\alpha}, x_{h_\beta}$ such that the increase of the objective function value is maximized, meaning that:

$$-r_\beta \cdot \frac{p_\alpha}{q_{\alpha\beta}} = \max\left\{ r_j \cdot \min\left\{ -\frac{p_i}{q_{ij}} : q_{ij} < 0, 1 \leq i \leq m \right\} : 1 \leq j \leq n, r_j > 0 \right\}$$

   or equivalently:

$$-r_\beta \cdot \frac{p_\alpha}{q_{\alpha\beta}} = \max\left\{ \min\left\{ -r_j \cdot \frac{p_i}{q_{ij}} : q_{ij} < 0, 1 \leq i \leq m \right\} : 1 \leq j \leq n, r_j > 0 \right\}$$

   This rule is computationally more expensive than the previous rule, but it locally maximizes the progress.

3. **Steepest edge rule**: pick $x_{k_\alpha}, x_{h_\beta}$ such that the pivoting basis moves the current BFS in a direction closest to the direction of the vector $c$, meaning that it maximizes the following ratio:

$$\frac{c^T(x_{\mathrm{k}_\alpha,\mathrm{h}_\beta} - x_{\mathrm{curr}})}{\left\| x_{\mathrm{k}_\alpha,\mathrm{h}_\beta} - x_{\mathrm{curr}} \right\|}$$

   where $x_{\mathrm{curr}}$ is the current BFS certified by the basis $\mathcal{B}$ and $x_{\mathrm{k}_\alpha,\mathrm{h}_\beta}$ is the BFS certified by the basis $\mathcal{B}_{\mathrm{k}_\alpha,\mathrm{h}_\beta} = (\mathcal{B} - \{k_\alpha\}) \cup \{h_\beta\}$. The idea behind this rule is to take the "shortest path" from the initial BFS to the optimal one and it's usually faster than the previous rules. In fact, this is the rule that is usually used in practical implementations of the Simplex method.

**Observation 12: Non-cycle-proof rules**

The previous pivot rules may cycle through degenerate pivots

Even thought it doesn't happen often, there is still a possibility for this observation to hold. We will first analyze what happens in the case of cycling tableaus and then proceed by formulating a rule that prevents cycling.

**Definition 28: Fickle variables**

Let $\mathcal{T}(\mathcal{B}_1) \to \ldots \to \mathcal{T}(\mathcal{B}_k) \to \mathcal{T}(\mathcal{B}_1)$ be a cycle of simplex tableaus of a linear program. We say that a variable $x_h$ is **fickle** if there are two indices $i, j$ such that $x_h \in \mathcal{B}_i$ and $x_h \notin \mathcal{B}_j$.

**Note:** this definition is equivalent to saying that a fickle variable enters and leaves the basis at least once per every cycle of tableaus

**Lemma 9: Degenerate BFS given by cycling**

Let $\mathcal{T}(\mathcal{B}_1) \to \ldots \to \mathcal{T}(\mathcal{B}_k) \to \mathcal{T}(\mathcal{B}_1)$ be a cycle of simplex tableaus of a linear program and let $F$ be the set of fickle variables. There is a unique degenerate BFS $\overline{x}$ certified by $\mathcal{B}_1, \ldots, \mathcal{B}_k$ and $\forall i \in F$ it holds that $x_i = 0$.

*Proof.*

Since each pivot step must keep the objective function value equal or increase it, it must hold that the objective function value is the same for $\mathcal{T}(\mathcal{B}_1), \ldots, \mathcal{T}(\mathcal{B}_k)$, implying that each pivot step of the cycle is degenerate.

Without loss of generality, let $x^j$ be the BFS certified by the basis $\mathcal{B}_j$ and let $\mathcal{N}_j = \{1, \ldots, n\} - \mathcal{B}_j$. The tableau $\mathcal{T}(\mathcal{B}_j)$ is given by:

$$\begin{array}{cccc} x_{\mathcal{B}_j} & = & p^j & + & Q^j x_{\mathcal{N}_j} \\ \hline z & = & z_0 & + & (r^j)^T x_{\mathcal{N}_j} \end{array}$$

Let $x_e$ be the entering variable and let $x_\ell$ be the leaving variable, meaning that $\mathcal{B}_{j+1} = (\mathcal{B}_j - \{\ell\} \cup \{e\})$.

**Claim**: for each $1 \leq i \leq n$, it holds that $x_i^j = x_i^{j+1}$

*Proof of the claim.*

Let $\mathcal{B}_j = \{k_1, \ldots, k_m\}$ and let $\mathbb{N}_j = \{h_1, \ldots, h_{n-m}\}$. We have the following four cases:

1. $i \in \mathcal{N}_j - \{e\}$, meaning that $x_i$ is a non basic variable different from the entering one
2. $i \in \mathcal{B}_j - \{\ell\}$, meaning that $x_i$ is a basic variable different from the leaving one
3. $i = e$, meaning that $x_i$ is the entering variable
4. $i = \ell$, meaning that $x_i$ is the leaving variable

Proving that the claim holds for the first case is very easy: if $i \in \mathcal{N}_j - \{e\}$ then the variable $x_i$ is non-basic for both $\mathcal{B}_j$ and $\mathcal{B}_{j+1}$. Thus, we conclude that $x_i^j, x_i^{j+1} = 0$ and so that in this case we have $x_i^j = x_i^{j+1}$.

For the other cases, let $e = h_\beta$ and let $\ell = k_\alpha$. Since $x_{k_\alpha}$ is the leaving variable, in $\mathcal{T}(\mathcal{B}_{j+1})$ it holds that

$$x_{k_\alpha}^{j+1} = 0 + \sum_{t=1}^{n-m} q_{\alpha t}^{j+1} x_{h_t^{j+1}}$$

If $i \in \mathcal{B}_j - \{\ell\}$, in the tableau $\mathcal{T}(\mathcal{B}_j)$ we have that:

$$x_i^j = p_i^j + \sum_{t=1}^{n-m} q_{it}^j x_{h_t^j}$$

Since $h_1^j, \ldots, h_t^j \in \mathcal{N}_j$ all of these coefficients must be zero in the tableau $\mathcal{T}(\mathcal{B}_j)$, meaning that $x_i^{j+1} = p^j$. In the tableau $\mathcal{T}(\mathcal{B}_{j+1})$, we can obtain the equation for $x_i^{j+1}$ by simply substituting $x_{k_\alpha}^j$ with $x_{k_\alpha}^{j+1}$.

$$\begin{aligned}
x_i^{j+1} &= p_i^j + q_{i\alpha}^j x_{k_\alpha}^j + \sum_{\substack{t=1 \\ t\neq\alpha}}^{n-m} q_{it}^j x_{h_t^j} \\
&= p_i^j + q_{i\alpha}^j \left(0 + \sum_{t=1}^{n-m} q_{\alpha t}^{j+1} x_{h_t^{j+1}}\right) + \sum_{\substack{t=1 \\ t\neq\alpha}}^{n-m} q_{it}^j x_{h_t^j} \\
&= p_i^j + q_{i\alpha}^j \sum_{t=1}^{n-m} q_{\alpha t}^{j+1} x_{h_t^{j+1}} + \sum_{\substack{t=1 \\ t\neq\alpha}}^{n-m} q_{it}^j x_{h_t^j} \\
&= p_i^j + \sum_{t=1}^{n-m} q_{it}^j x_{h_t^j}
\end{aligned}$$

Again, since $h_1^{j+1}, \ldots, h_t^{j+1} \in \mathcal{N}_{j+1}$, we get that $x_i^{j+1} = p_i$, concluding that $x_i^{j+1} = x_i^j$ holds in the second case.

Suppose now that $i = e = h_\beta$. Since $x_{h_\beta}$ is the entering variable, we know that $x_{h_\beta}^j = 0$ since it's non-basic for $\mathcal{T}(\mathcal{B}_j)$. Moreover, we know that $x_{k_\alpha}^{j+1} = 0$ since $x_{k_\alpha}$ is the leaving variable. Thus, we can just solve the equation for $x_{k_\alpha}$ in $\mathcal{T}(\mathcal{B}_j)$ for $x_{h_\beta}$, obtaining the equation of the latter in $\mathcal{T}(\mathcal{B}_{j+1})$.

Then, since the objective function value must remain the same and since $x_{h_\beta}^j = x_{k_\alpha}^{j+1}$, this implies that the equation solved for $x_{h_\beta}$ maintains the same constant term, meaning that $x_{h_\beta}^{j+1} = x_{k_\alpha}^{j+1} = 0$. Thus, we conclude that $x_i^j = x_i^{j+1}$ also holds in the third case.

Suppose that $i = \ell = k_\alpha$. Since $x_{h_\beta}$ is the entering variable, we know that $r_\beta^j > 0$ and $x_{h_\beta}^j = 0$ must hold. Then, since the pivot is degenerate, we know that the constraints impose that we can't increase the value of $x_{h_\beta}$, implying that $x_{h_\beta}^j = x_{h_\beta}^{j+1}$.

However, we also know that such increase depends on the choice of $k_\alpha$. In particular, we should increase $x_{h_\beta}$ in a way that $x_{h_\beta}^{j+1} = -\frac{p_\alpha^j}{q_{\alpha\beta}^j}$. Then, we get that $p_\alpha^j = 0$ must hold in order for this equality to hold. Finally, since $x_{k_\alpha}$ is the leaving variable for $\mathcal{T}(\mathcal{B}_{j+1})$, we already know that $p_\alpha^{j+1} = 0$, concluding that $x_{k_\alpha}^j, x_{k_\alpha}^{j+1} = 0$ and this that $x_i^j = x_i^{j+1}$ also holds in the fourth case.

□

Since the claim holds for each $1 \leq i \leq n$, this directly implies that for each index $1 \leq j \leq k$ it holds that $x^j = x^{j+1}$ and thus that the basis $\mathcal{B}_1, \ldots, \mathcal{B}_k$ all certify the same BFS $\overline{x} = x^1 = \ldots = x^k$. Obviously, in this case the set of fickle variables $F$ corresponds to the set of basic variables that aren't shared among all the basis. Thus, by Proposition 11, we conclude that $\overline{x}$ is degenerate and that $\forall i \in F$ it holds that $x_i = 0$.

□

### 3.6.1 Bland's rule

Once we have discussed cycling tableaus all share the same degenerate BFS, we are now ready to define Bland's rule, a rule that **prevents cycling**.

**Definition 29: Bland's rule**

During a pivot step, choose the entering variable $x_e$ and the leaving variable $x_\ell$ as the ones with the **lowest possible indexes**

The idea behind this rule is surprisingly simple, while also being a very inefficient way to select the next BFS since it doesn't try to optimize the increase of the objective function value, resulting in a rule slower than the previously shown ones. For this reason, Bland's rule is usually used only when a tableau cycle is detected using the other rules.

**Theorem 9: Bland's rule is cycle-proof**

The Simplex method with Bland's rule prevents tableau cycles, implying that it **always terminates**

*Proof.*

By way of contradiction, suppose that there is a tableau cycle $\mathcal{T}(\mathcal{B}_1) \to \ldots \to \mathcal{T}(\mathcal{B}_k) \to \mathcal{T}(\mathcal{B}_1)$ obtained by using Bland's rule. Let $F \subseteq \{1, \ldots n\}$ be the set of indices of the fickle variables in the cycle and let $\ell = \max(F)$.

Since $x_\ell$ is fickle, let $i$ be the index such that $\ell \in \mathcal{B}_i$ and $\ell \notin \mathcal{B}_{i+1}$ and let $j$ be the index such that $\ell \notin \mathcal{B}_j$ and $\ell \in \mathcal{B}_{j+1}$. Fix $\mathcal{B} = \mathcal{B}_i$ and fix $\mathcal{B}' = \mathcal{B}_j$.

By Lemma 9, we know that $\mathcal{B}$ and $\mathcal{B}'$ certify the same BFS $\overline{x}$, where $c^T\overline{x} = z_0$. Thus, they yield the following tableaus:

$$\begin{array}{rcccc} x_{\mathcal{B}} & = & p & + & Qx_{\mathcal{N}} \\ \hline z & = & z_0 & + & r^T x_{\mathcal{N}} \end{array} \qquad \begin{array}{rcccc} x_{\mathcal{B}'} & = & p' & + & Q'x_{\mathcal{N}'} \\ \hline z & = & z_0 & + & (r')^T x_{\mathcal{N}'} \end{array}$$

Suppose that $\mathcal{B} = \{k_1, \ldots, k_m\}$, $\mathcal{B}' = \{k'_1, \ldots, k'_m\}$ and that $\mathcal{N} = \{h_1, \ldots, h_{n-m}\}$, $\mathcal{N}' = \{h'_1, \ldots, h'_{n-m}\}$. By choice of $\ell$, we have that $\ell = k'_{\alpha'} = h_\beta$ for some $\alpha', \beta$. Since $\ell$ is the entering variable for $\mathcal{T}(\mathcal{B})$, we know that $r_{\alpha'} > 0$. Thus, since Bland's rule imposes that the entering variable $x_\ell$ is chosen with the lowest index from all possible candidates and since we picked $\ell$ to be the largest possible fickle index, we get that $\forall i \in F - \{\ell\}$ it must hold that $r_i \leq 0$.

We now look at what happens when $x_\ell = x_{k'_{\alpha'}}$ leaves $\mathcal{T}(\mathcal{B}')$. Let $x_e = x_{h'_{\beta'}}$ be the entering variable for $\mathcal{B}'$. By Lemma 8, we know that any index $k'_{d'}$ can leave the basis $\mathcal{B}'$ if:

$$q'_{d'\beta'} < 0 \quad \text{and} \quad -\frac{p_{d'}}{q_{d'\beta'}} = \min\left\{-\frac{p_{i'}}{q_{i'\beta'}} : q_{i'\beta'} < 0, 1 \leq i' \leq m\right\}$$

However, by Lemma 9 we know that for each $k'_{d'} \in B' \cap F$ in the BFS $\overline{x}$ it holds that $x_{k'_{d'}} = 0$. Therefore, the index $k'_{\alpha'}$ must be the only one satisfying $q'_{\alpha'\beta'} < 0$ among all fickle variables.

Consider the following auxiliary linear program:

$$\begin{aligned} \max\ & c^T x \\ Ax &= b \\ x_\ell &\leq 0 \\ x_{F-\{\ell\}} &\geq 0 \\ x_{\mathcal{N}-F} &= 0 \\ x_{\mathcal{B}-F} &\in \mathbb{R} \end{aligned}$$

**Claim**: The BFS $\overline{x}$ of the original program is optimal for the auxiliary linear program

*Proof of claim.*

Since $\overline{x}$ is a BFS of the original program, we know that it satisfies the constraints $Ax = b$. Moreover, we know that $\overline{x}_{\mathcal{N}} = 0$ since $\mathcal{N}$ are non-basic variables and we also know that $\overline{x}_F = 0$ by Lemma 9. Thus, since we also know that $\ell \in F$, we conclude that $\overline{x}$ is indeed a feasible solution of the auxiliary program.

For any feasible solution $y$ of the original program, the objective function value $c^T y$ can be written as $c^T y = z_0 + r^T y_{\mathcal{N}}$. Instead, for any feasible solution $y'$ of the auxiliary program we have that:

$$y'_{h_d} \begin{cases} \geq 0 & \text{if } h_d \in F - \{\ell\} \\ = 0 & \text{if } h_d \in \mathcal{N} - F \\ \leq 0 & \text{if } h_d = h_\beta = \ell \end{cases}$$

We notice that this system of inequalities implies that for each $1 \leq d \leq n-m$ it holds that $r_d y'_{h_d} \leq 0$:

- If $h_d \in \mathcal{N} - F$ we know that $y'_{h_d} = 0$, so trivially we get $r_d y' h_d \leq 0$
- If $h_d \in F - \{\ell\}$ we know that $y'_{h_d} \geq 0$. Moreover, we have previously shown that in this case $r_{h_d} \leq 0$ holds. Thus, we get that $r_d y'_{h_d} \leq 0$.
- If $h_d = h_\beta = \ell$ we know that $y'_{h_d} \geq 0$. Moreover, since $y'\ell$ is the leaving variable, we know that $r_d > 0$. Thus, we get that $r_d y'_{h_d} \leq 0$.

Then, this means that $r^T y' \leq 0$ for any feasible solution $y'$ of the auxiliary program. Thus, since the two programs share the same objective function, we get that for each feasible solution of the auxiliary program it holds that $c^T y = z_0 + r^T y' \leq z_0 = c^T \overline{x}$, concluding that $\overline{x}$ is optimal for the auxiliary program. □

**Claim 2**: The auxiliary linear program is unbounded

*Proof of claim 2.*

Since both basis share the same BFS $\overline{x}$, we now work with the basis $\mathcal{B}'$. Remember that $x_e = x_{h'_{\beta'}}$ is the variable in $\mathcal{N}'$ that enters the basis when $x_\ell$ leaves $\mathcal{B}'$.

For any $t > 0$, we define $\overline{x}(t)$ as:

$$\overline{x}(t)_{h'_d} = \begin{cases} 0 & \text{if } h'_d \in \mathcal{N}' - \{e\} \\ t & \text{if } h'_d = h'_{\beta'} = e \end{cases}$$

Since $x_\ell$ left the basis, we have that $x_\ell = 0$. Thus, by construction we have that $\overline{x}(t)_{k'_{d'}} = \overline{x}_{k'_{d'}} + t q'_{d'\beta'}$. Furthermore, it's easy to see that $\overline{x}(t)$ satisfies $Ax = b$ for each $t > 0$.

Since we have shown $k'_{\alpha'} = \ell$ is the only fickle variable such that $q'_{\alpha'\beta'} < 0$, we get that:

$$\overline{x}(t)_{k'_{d'}} = \overline{x}_{k'_{d'}} + t q'_{d'\beta'} \begin{cases} \geq 0 & \text{if } k'_{d'} \in F - \{\ell\} \\ 0 & \text{if } k'_{d'} \in \mathcal{N}' - F \\ \leq 0 & \text{if } k'_{d'} = k'_{\alpha'} = \ell \end{cases}$$

Meaning that $\overline{x}(t)$ also satisfies the remaining constraints for each $t > 0$, implying that it is a feasible solution. Moreover, since $x_e = x_{h'_{\beta'}}$ is the variable that enters the basis, we know that $(r')^T_{\beta'} > 0$ must hold. Thus, we conclude that:

$$c^T(\overline{x}(t)) = z'_0 + (r')^T \overline{x}(t)_{\mathcal{N}'} = z'_0 + t r'_{\beta'} \to +\infty$$

when $t \to +\infty$, concluding that the auxiliary program is unbounded. □

Obviously, the two claims yield a contradiction since the auxiliary program can't have an optimal solution and be unbounded at the same time. Thus, the only possibility is for the cycle to not exist. □

## 3.7 Solved exercises

**Problem 3**

Solve the following linear program using the Simplex method:

$$\begin{aligned} \max \quad & 2x_1 + x_2 - x_3 \\ & \\ & x_1 + 2x_2 + x_3 \leq 8 \\ & -x_1 + x_2 - 2x_3 \leq 4 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

*Solution:*

First, we convert the program in standard form:

$$\begin{aligned} \max \quad & 2x_1 + x_2 - x_3 \\ & \\ & x_1 + 2x_2 + x_3 + x_4 = 8 \\ & -x_1 + x_2 - 2x_3 + x_5 = 4 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

The trivial basis $\mathcal{B} = \{4, 5\}$ is feasible, so we start with the following tableau:

$$\begin{array}{rcrrrr} x_4 & = & 8 & -x_1 & -2x_2 & -x_3 \\ x_5 & = & 4 & +x_1 & -x_2 & +2x_3 \\ \hline z & = & & +2x_1 & +x_2 & -x_3 \end{array}$$

We notice that if we pivot $x_2$ with either $x_4$ or $x_5$, both of the latter become 0 (since we would set $x_2 = 4$), implying that we would get a degenerate BFS, which could lead us to a cycle. Thus, we pivot $x_1$ with $x_4$ by setting $x_1 = 8$.

$$\begin{array}{rcrrrr} x_1 & = & 8 & -2x_2 & -x_3 & -x_4 \\ x_5 & = & 12 & -3x_2 & +x_3 & -x_4 \\ \hline z & = & 16 & -3x_2 & -3x_3 & -2x_4 \end{array}$$

Since we cannot select another pivot, we conclude that $\begin{bmatrix} 8 & 0 & 0 & 0 & 12 \end{bmatrix}$ is an optimal BFS to the standard program, implying that $\begin{bmatrix} 8 & 0 & 0 \end{bmatrix}$ is the optimal solution to the original program with a value of 16.

# 4

# Duality in linear programs

## 4.1 Idea and implications

In the previous chapters we have formalized the algebraic and geometrical concepts of linear programs, which culminated in the study of the Simplex method algorithm. Now, we are ready to discuss one of the most important results regarding linear programs: the concept of **duality**.

To give an intuition, let's consider the following example:

$$\begin{aligned} \max\ & 2x_1 + 3x_2 \\ 4x_1 + 8x_2 &\leq 12 \\ 2x_1 + x_2 &\leq 3 \\ 3x_1 + 2x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

In order to avoid uselessly applying the Simplex method and eventually finding out, we want to know if there actually is an **optimal solution** to this problem. Finding an answer to this question is actually easy: we must find out of the objective function value is *bounded* or not.

By looking at the fist constraint, we notice that:

$$2x_1 + 3x_2 \leq 4x_1 + 8x_2 \leq 12$$

implying that the objective function is indeed bounded and thus that there is an optimal solution. However, this bound is obviously way to large compared to the objective function. In fact, we could halve the first constraint and still get the following bound:

$$2x_1 + 3x_2 \leq 2x_1 + 4x_2 \leq 6$$

which still gives some space, but is indeed more strict.

Following the same idea, we can sum the first and second constraint to get a perfect bound on the objective function value:

$$\begin{cases} 4x_1 + 8x_2 \leq 12 \\ 2x_1 + x_2 \leq 3 \end{cases} \implies 6x_1 + 9x_2 \leq 15 \implies 2x_1 + 3x_2 \leq 5$$

which indeed is the optimal value of the objective function (notice how the left hand side of the inequality matches the objective function).

In general, what we did was constructing a **linear combination of the constraints** in order to get a strict bound: given three non-negative coefficients $y_1, y_2, y_3$ we get that:

$$\begin{cases} 4x_1 + 8x_2 & \leq 12 \\ 2x_1 + x_2 & \leq 3 \\ 3x_1 + 2x_2 & \leq 4 \end{cases} \implies \begin{cases} y_1(4x_1 + 8x_2) & \leq 12y_1 \\ y_2(2x_1 + x_2) & \leq 3y_2 \\ y_3(3x_1 + 2x_2) & \leq 4y_3 \end{cases} \implies$$

$$y_1(4x_1 + 8x_2) + y_2(2x_1 + x_2) + y_3(3x_1 + 2x_2) \leq 12y_1 + 3y_2 + 4y_3 \implies$$

$$(4y_1 + 2y_2 + 3y_3)x_1 + (8y_1 + y_2 + 2y_3)x_2 \leq 12y_1 + 3y_2 + 4y_3$$

Now, if we impose that $4y_1 + 2y_2 + 3y_3 \geq 2$ and $8y_1 + y_2 + 2y_3 \geq 3$, we finally get that:

$$2x_1 + 3x_2 \leq (4y_1 + 2y_2 + 3y_3)x_1 + (8y_1 + y_2 + 2y_3)x_2 \leq 12y_1 + 3y_2 + 4y_3$$

effectively obtaining a bound on the original objective function.

It's easy to see now that we just obtained *a new linear program* where we want to **minimize** the value $12y_1 + 3y_2 + 4y_3$ to get an upper bound on the original objective function as strict as possible.

$$\begin{aligned} \min \; & 12y_1 + 3x_2 + 4y_3 \\ & 4y_1 + 2y_2 + 3y_3 \geq 2 \\ & 8y_1 + y_2 + 2y_3 \geq 3 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

We refer to this new linear program as the **dual program**, while the original program is referred to as the **primal program**. Moreover, by writing both programs in their matrix form, we notice that:

$$\begin{aligned} \max \; & \begin{bmatrix} 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ & \begin{bmatrix} 4 & 8 \\ 2 & 1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 12 \\ 3 \\ 4 \end{bmatrix} \\ & x_1, x_2 \geq 0 \end{aligned} \qquad \begin{aligned} \min \; & \begin{bmatrix} 12 & 3 & 4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \\ & \begin{bmatrix} 4 & 2 & 3 \\ 8 & 1 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \geq \begin{bmatrix} 2 \\ 3 \end{bmatrix} \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

showing us that the dual program is just an **"inversion" of the primal program**: the constraint vector and the objective function vector got swapped, the constraint matrix

got transposed, the inequality signs have been inverted and the objective function is now a minimization problem.

**Definition 30: Primal and Dual program**

Let $(P)$ be the following linear program:

$$\begin{aligned} \max\ & c^T x \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

We define the **dual program** $(D)$ of $(P)$ as:

$$\begin{aligned} \min\ & b^T y \\ A^T y &\geq c \\ y &\geq 0 \end{aligned}$$

Additionally, we say that $(P)$ is the **primal program** of $(D)$

As we already discussed, by construction we have that the optimal solution of the dual program gives the most strict upper bound we can get on the objective function of the linear program. Thus, we get the following theorem for free:

**Theorem 10: Weak duality theorem**

Let $(P)$ and $(D)$ be a primal-dual pair of linear programs. For each feasible solution $\overline{x}$ of $(P)$ and each feasible solution $\overline{y}$ of $(D)$, we have that:

$$c^T \overline{x} \leq b^T \overline{y}$$

In other words, each feasible solution of the dual program is a bound on the objective function of the primal program. Thus, if it exists, the optimal value of $(D)$ is the **most strict bound** on the optimal value of $(P)$.

**Observation 13**

The dual of a dual program $(D)$ is the primal program $(P)$

*Proof.* By Proposition 5, we know that $(D)$ can also be rewritten as a general maximization problem:

$$\begin{aligned} \min\ & b^T x \\ A^T y &\geq c \\ y &\geq 0 \end{aligned} \qquad \Longrightarrow \qquad \begin{aligned} \max\ & -b^T x \\ -A^T y &\leq -c \\ y &\geq 0 \end{aligned}$$

Now, the dual of $(D)$ corresponds to:

$$\begin{aligned} \min\ & -c^T z \\ -A^T z &\geq -b \\ z &\geq 0 \end{aligned}$$

Thus, by rewriting the dual of $(D)$ in the same way as before we get exactly the primal program $(P)$:

$$\begin{aligned} \min\ & -c^T z \\ -A^T z &\geq -b \\ z &\geq 0 \end{aligned} \qquad \Longrightarrow \qquad \begin{aligned} \max\ & c^T z \\ A^T z &\leq b \\ z &\geq 0 \end{aligned}$$

□

**Every linear program has a dual program**, even without having all the constraints being formulated in the canonical form $Ax \leq b$:

- If the $i$-th constraint has the sign $\leq$, we impose that $y_i \geq 0$
- If the $i$-th constraint has the sign $\geq$, we impose that $y_i \leq 0$
- If the $i$-th constraint has the sign $=$, we impose that $y_i \in \mathbb{R}$ (this constraint doesn't actually do anything)

It's easy to see that these mappings preserve the idea of the dual program. Likewise, each mapping on the constraints of the variables $x_1, \ldots, x_n$ can be "preserved" since it is actually required to be "equal". To summarize, we get the following general mappings:

| | **Primal Program** | **Dual program** |
|---|---|---|
| **Variables** | $x_1, \ldots, x_n$ | $y_m, \ldots, y_m$ |
| **Constraint matrix** | $A$ | $A^T$ |
| **Constraint vector** | $b$ | $c$ |
| **Objective function** | $\max\ c^T x$ | $\min\ b^T x$ |
| **Row constraints** | $i$-th constraint has $\leq$ | $y_i \geq 0$ |
| | $i$-th constraint has $\geq$ | $y_i \leq 0$ |
| | $i$-th constraint has $=$ | $y_i \in \mathbb{R}$ |
| **Variable constraints** | $x_j \geq 0$ | $j$-th constraint has $\geq$ |
| | $x_j \leq 0$ | $j$-th constraint has $\leq$ |
| | $x_j \in \mathbb{R}$ | $j$-th constraint has $=$ |

*Note*: to get the dual of a minimization problem, first convert it into a general maximization problem as described in Proposition 5

## 4.2 Strong duality theorem

The **weak duality theorem** that we just introduced in the previous section directly follows from the way dual programs are constructed. The term "weak" is used due to it directly implying a way stronger result: the **strong duality theorem**. To get to this stronger result, first we have to prove another theorem through the Simplex method itself.

**Theorem 11: Reciprocal boundedness**

Given a primal-dual pair of linear programs $(P)$ and $(D)$, if $(P)$ is feasible and bounded then $(D)$ is also feasible and bounded and they **share the same optimal value**

*Proof.* Let the primal problem $(P)$ be defined as:

$$\begin{aligned} \max \quad & c^T x \\ & Ax \leq b \\ & x \geq 0 \end{aligned}$$

where $A \in \text{Mat}_{n\times m}(\mathbb{R})$. We introduce the slack variables $x_{n+1}, \ldots, x_{n+m}$, obtaining the following standard program:

$$\begin{aligned} \max \quad & \overline{c}^T \overline{x} \\ & A\overline{x} = b \\ & \overline{x} \geq 0 \end{aligned}$$

where $\overline{c}^T = \begin{bmatrix} c \mid 0 \ldots 0 \end{bmatrix}$, $\overline{A} = \begin{bmatrix} A \mid I_m \end{bmatrix}$ and $\overline{x} \in \mathbb{R}^{n+m}$. Since $(P)$ is feasible and bounded, from Theorem 6 we know that there is an optimal BFS $\overline{x}^*$. Moreover, we know that the Simplex method with Bland's rule never cycles (Theorem 9), so it will always yield one of those optimal BFSs. Let $\mathcal{B}$ be the basis that certifies $\overline{x}^*$. By Lemma 7, we know that there is an unique tableau:

$$\begin{array}{rcccc} \overline{x}^*_{\mathcal{B}} & = & \overline{p} & + & \overline{Q}\,\overline{x}^*_{\mathcal{N}} \\ \hline \overline{z} & = & z_0 & + & \overline{r}^T \overline{x}^*_{\mathcal{N}} \end{array}$$

such that:

- $\overline{Q} = \overline{A}_{\mathcal{B}}^{-1}\,\overline{A}_{\mathcal{N}}$
- $\overline{p} = \overline{A}_{\mathcal{B}}^{-1} b$
- $z_0 = \overline{c}_{\mathcal{B}}^T\,\overline{A}_{\mathcal{B}}^{-1} b$
- $\overline{r}^T = \overline{c}_{\mathcal{N}}^T - \overline{c}_{\mathcal{B}}^T\,\overline{A}_{\mathcal{B}}^{-1}\,\overline{A}_{\mathcal{N}}$

Since $\mathcal{B}$ is an optimal basis, we know that $\overline{r} \leq 0$ must be true, otherwise the Simplex method would continue, contradicting the assumption of $\overline{x}^*$ being an optimal solution.

Let $x^*$ be the restriction of $\overline{x}^*$ to the first $n$ original variables, meaning that $x^*$ is an optimal solution of $(P)$. Additionally, let $y^* = (\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1})^T$.

**Claim**: The vector $y^*$ is a feasible solution to the dual problem $(D)$.

*Proof of the claim.*

To conclude that it's a feasible solution, we have to show that $A^T y^* \geq c$ and that $y^* \geq 0$. First, we notice that:

$$y^* \geq 0 \iff y^* \cdot I_m \geq 0$$

so we can combine the two necessary conditions in the following way:

$$\overline{A}^T y^* = \left[\frac{A^T}{I_m}\right] y^* \geq \overline{c} = \begin{bmatrix} c \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix}$$

*Note:* remember that $\overline{A} = [A \mid I_m]$.

Since $\overline{c}_{\mathcal{B}}^T \in \text{Mat}_{1\times m}(\mathbb{R})$ and $\overline{A}_{\mathcal{B}}^{-1} \in \text{Mat}_{m\times m}$, we get that $\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} \in \text{Mat}_{1\times m}(\mathbb{R})$. Thus, since we also have that $\overline{A} \in \text{Mat}_{m\times n+m}$, the following commutation is legal:

$$\overline{A}^T y^* = \overline{A}^T (\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1})^T = (\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} \overline{A})^T$$

Let $w = \overline{A}^T y^* = (\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} \overline{A})^T$. We notice that:

$$w = (\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} \overline{A})^T \implies w_{\mathcal{B}} = (\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} \overline{A}_{\mathcal{B}})^T = \overline{c}_{\mathcal{B}}$$

Likewise, we also notice that:

$$w = (\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} \overline{A})^T \implies w_{\mathcal{N}} = (\overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} \overline{A}_{\mathcal{N}})^T = \overline{c}_{\mathcal{N}} - \overline{r} \geq \overline{c}_{\mathcal{N}}$$

since we know that $\overline{r} = \overline{c}_{\mathcal{N}}^T - \overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} \overline{A}_{\mathcal{N}}$ and since $\overline{r} \leq 0$. Thus, since $w_{\mathcal{B}} = \overline{c}_{\mathcal{B}}$ and since $w_{\mathcal{N}} \geq \overline{c}_{\mathcal{N}}$, we conclude that $\overline{A} y^* = w \geq \overline{c}$ and thus that $y^*$ is a feasible solution for $(D)$. □

Now, we notice that

$$c^T x^* = \overline{c}^T \overline{x}^* = z = \overline{c}_{\mathcal{B}}^T \overline{A}_{\mathcal{B}}^{-1} b = y^* b = b^T y^*$$

where the last commutation was possible since $y^* \in \text{Mat}_{1\times m}(\mathbb{R})$ and $b \in \text{Mat}_{m\times 1}(\mathbb{R})$. Thus, we also conclude that $c^T x^* = b^T y^*$.

Finally, by the Weak duality theorem, since $(P)$ is the dual of $(D)$ we have that the optimal solution $x^*$ bounds any optimal solution of $(D)$, meaning that $b^T y' \leq c^T x^*$ for all optimal solutions $y'$ of $(D)$. Thus, since $b^T y^* = c^T x^*$, we conclude that $y^*$ is indeed an optimal solution of $(D)$.

□

Now, the weak duality theorem and the reciprocal boundedness theorem to directly imply the strong duality theorem simply by eliminating impossible cases.

**Theorem 12: Strong duality theorem**

Given a primal-dual pair of linear programs $(P)$ and $(D)$, exactly one of the following possibilities occur:

1. Both $(P)$ and $(D)$ are infeasible
2. $(P)$ is feasible unbounded and $(D)$ is infeasible
3. $(D)$ is feasible unbounded and $(P)$ is infeasible
4. Both $(P)$ and $(D)$ are feasible bounded and they share the same optimal value

*Proof.*

By the Weak duality theorem, any feasible solution of $(P)$ sets a bound on any solution of $(D)$, meaning that $(D)$ can't be unbounded. Moreover, since $(P)$ is the dual of $(D)$, this statement also holds if we swap $(P)$ and $(D)$. Thus, we get that:

| | $(P)$ Feas. bounded | $(P)$ Feas. unbounded | $(P)$ Infeasible |
|---|---|---|---|
| $(D)$ Feas. bounded | ? | No | ? |
| $(D)$ Feas. unbounded | No | No | ? |
| $(D)$ Infeasible | ? | ? | ? |

From Theorem 11, instead, we know that if $(P)$ is feasible bounded than $(D)$ is also feasible bounded, so it can't be infeasible. As before, the statement also works if $(P)$ and $(D)$ are swapped, concluding that:

| | $(P)$ Feas. bounded | $(P)$ Feas. unbounded | $(P)$ Infeasible |
|---|---|---|---|
| $(D)$ Feas. bounded | ? | No | No |
| $(D)$ Feas. unbounded | No | No | ? |
| $(D)$ Infeasible | No | ? | ? |

The remaining cases are all legit since the none of the two theorems get violated. Moreover, in case both $(P)$ and $(D)$ are feasible bounded we also know that the optimum is shared. Thus, we conclude that:

| | $(P)$ Feas. bounded | $(P)$ Feas. unbounded | $(P)$ Infeasible |
|---|---|---|---|
| $(D)$ Feas. bounded | Yes+ | No | No |
| $(D)$ Feas. unbounded | No | No | Yes |
| $(D)$ Infeasible | No | Yes | Yes |

□

## 4.3 Implications of the strong duality theorem

We saw in Proposition 10 that **testing feasibility** for a linear program is computationally equivalent to **optimizing** a linear program. However, we gave a *shallow* proof of how optimization can be reduced to checking feasibility. Now, we can use the strong duality theorem to give a better idea of how this can be achieved, without the use of the binary search approach shown earlier.

Suppose that we have a subroutine that checks if a program is feasible (meaning that it checks if its polyhedron is non-empty). Given a primal program $(P)$ and its dual $(D)$, if the subroutine returns true for both $(P)$ and $(D)$, meaning that both have a feasible solution, by the strong duality theorem we know that they both must be bounded and they share the same optimum, easily concluding that optimization for $(P)$ can be reduced to testing feasibility for $(P)$. However, we can obtain an **even stronger result** through the use of duality and a stronger subroutine.

**Proposition 14**

Finding a **feasible solution** is computationally equivalent to finding an **optimal solution**.

*Proof.*

Let $(P)$ be the following primal program:

$$\begin{aligned} \max\ & c^T x \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

Suppose that we have a subroutine that returns an optimal solution to a linear program. Since an optimal solution is also a feasible solution, we conclude that finding a feasible solution to $(P)$ can easily be reduced to finding an optimal solution to $(P)$.

Viceversa, suppose that we have subroutine that returns a feasible solution (if there is any). Let $(D)$ be the dual program of $(P)$:

$$\begin{aligned} \min\ & b^T y \\ A^T y &\geq c \\ y &\geq 0 \end{aligned}$$

As before, we can check if the program $(P)$ has an optimal solution by running this subroutine on both $(P)$ and $(D)$: if the subroutine returns a feasible solution for both $(P)$ and $(D)$ then we know that $(P)$ must have an optimal solution.

Once we know that there is an optimal solution, we construct the following auxiliary linear program $(A)$:

$$\begin{aligned}
\begin{bmatrix} A \mid 0_m \end{bmatrix} z &\leq \begin{bmatrix} b \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix} \\
\begin{bmatrix} 0_n \\ \hline A^T \end{bmatrix} z &\geq \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \hline c \end{bmatrix} \\
\begin{bmatrix} c^T \mid 0 \end{bmatrix} z &\geq \begin{bmatrix} 0 \mid b^T \end{bmatrix} z \\
z &\in \mathbb{R}^{n+m}
\end{aligned}$$

where $0_m$ and $0_n$ are the zero matrices of order $m$ and $n$. First, we observe that we omitted the objective function since it won't be relevant for our necessities (we could just assume that we want to maximize $0^T z$). For each $z \in \mathbb{R}^{n+m}$, let $z = \begin{bmatrix} u \mid v \end{bmatrix}$, where $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$. We notice that:

$$\begin{bmatrix} A \mid 0_m \end{bmatrix} z \leq \begin{bmatrix} b \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix} \implies Au \leq b$$

and that:

$$\begin{bmatrix} 0_n \\ \hline A^T \end{bmatrix} z \geq \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \hline c \end{bmatrix} \implies A^T v \geq c$$

so $u$ and $v$ are feasible solutions to $(P)$ and $(D)$. Moreover, we have that:

$$\begin{bmatrix} c^T \mid 0 \end{bmatrix} z \geq \begin{bmatrix} 0 \mid b^T \end{bmatrix} z \implies c^T u \geq b^T v$$

implying that the solution to $(P)$ bounds the solution to $(D)$. Since by the weak duality theorem we know that any solution to $(D)$ also bounds any solution to $(P)$, we conclude that $c^T u = b^T v$, implying that $u$ and $v$ are optimal solutions to $(P)$ and $(D)$. Thus, we get that any feasible solution to $(A)$ gives an optimal solution to $(P)$, concluding that finding an optimal solution can be reduced to finding a feasible solution.

□

## 4.4 Farkas' lemma

**Farkas' lemma** is a solvability theorem for a finite system of linear inequalities, giving a key result that underpins the linear programming duality and that has played a central role in the development of mathematical optimization. In fact, Farkas' lemma can both be derived by the strong duality theorem and derive the latter, meaning that one holds if and only if the other holds.

Since we have already proved the strong duality theorem through the use of algebraic manipulation and the Simplex method, we will give a proof of how the theorem implies the lemma to give an intuition on how the lemma also implies the theorem.

**Theorem 13: Farkas' lemma**

Given $A \in \text{Mat}_{n\times m}(\mathbb{R})$ and $b \in \mathbb{R}^m$, exactly one of the following holds:

1. $\exists x \in \mathbb{R}^n$ such that $Ax = b$ with $x \geq 0$
2. $\exists y \in \mathbb{R}^m$ such that $A^T y \geq 0$ with $b^T y < 0$

*Proof.*

Given $A \in \text{Mat}_{n\times m}(\mathbb{R})$ and $b \in \mathbb{R}^m$, let $(P)$ be the linear program defined by:

$$\begin{aligned} \max\ & 0^T x \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

and let $(D)$ be its dual:

$$\begin{aligned} \min\ & b^T y \\ & A^T y \geq 0 \\ & y \in \mathbb{R}^m \end{aligned}$$

We notice that $(D)$ is always feasible since $\overline{y} = 0$ is a valid solution. Thus, since $(D)$ is feasible, by the weak duality theorem $(P)$ must be bounded. Then, by the strong duality theorem, exactly one of the two following possibilities must hold:

- $(D)$ is feasible unbounded and $(P)$ is infeasible
- Both $(P)$ and $(D)$ are feasible bounded and they share the same optimum

If $(D)$ is feasible unbounded and $(P)$ is infeasible, from the infeasibility of the latter we automatically get that $\nexists x \in \mathbb{R}^n$ such that $Ax = b$ with $x \geq 0$, meaning that the first case statement of the lemma doesn't hold. Moreover, since $(D)$ is unbounded we get that $\exists y \in \mathbb{R}^m$ such that $A^T y \geq 0$ and $b^T y < 0$, concluding that the second case of the lemma holds.

Instead, if both $(P)$ and $(D)$ are feasible bounded and they share the same optimum, from the feasibility of $(P)$ we automatically get that $\exists x \in \mathbb{R}^n$ such that $Ax = b$ with

$x \geq 0$, meaning that the first case of the lemma holds. Furthermore, since $(P)$ and $(D)$ share the same optimum and since the objective function of $(P)$ is given by $0^T x$, we get that the shared optimal value of both $(P)$ and $(D)$ must be 0, implying that $\nexists y \in \mathbb{R}^m$ such that $A^T y \geq 0$ and $b^T y < 0$. □

**Observation 14: Variants of Farkas' lemma**

The following statements are all **equivalent to Farkas' lemma**:

- $\exists x \in \mathbb{R}^n$ such that $Ax = b$ with $x \geq 0$ if and only if $\forall y \in \mathbb{R}^m$ such that $A^T y \geq 0$ it holds that $b^T y \geq 0$
- $\exists x \in \mathbb{R}^n$ such that $Ax \leq b$ with $x \geq 0$ if and only if $\forall y \in \mathbb{R}^m$ such that $A^T y \geq 0$ and $y \geq 0$ it holds that $b^T y \geq 0$
- $\exists x \in \mathbb{R}^n$ such that $Ax \leq b$ if and only if $\forall y \in \mathbb{R}^m$ such that $A^T y = 0$ and $y \geq 0$ it holds that $b^T y \geq 0$

As we already mentioned, Farkas' lemma directly implies the strong duality theorem without using the Simplex method. Like the previous proof we gave for the Strong Duality Theorem, every statement directly follows from the weak duality theorem and Theorem 11. Thus, we just have to show that Farkas' lemma implies Theorem 11.

*Proof of Theorem 11 through Farkas' lemma.*

Let $(P)$ be the linear program defined by:

$$\begin{aligned} \max\ & 0^T x \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

and let $(D)$ be its dual:

$$\begin{aligned} \min\ & b^T y \\ & A^T y \geq 0 \\ & y \in \mathbb{R}^m \end{aligned}$$

Suppose that $(P)$ is feasible and bounded. By Theorem 6, we know that $(P)$ has an optimal solution $x^*$. Thus, let $\gamma = c^T x^*$ be the optimal value for $(P)$. Consider the following system of inequalities:

$$\begin{aligned} Ax &\leq b \\ c^T x &\geq \gamma \\ x &\geq 0 \end{aligned}$$

Obviously, $x^*$ is a feasible solution to this system.

Now, for each $\varepsilon \in \mathbb{R}$ we define the system:

$$\begin{aligned} Ax &\leq b \\ c^T x &\geq \gamma + \varepsilon \\ x &\geq 0 \end{aligned}$$

it's easy to see that if $\varepsilon > 0$, this system is infeasible since otherwise $\gamma$ wouldn't be the optimal value for $(P)$. Now, for each $\varepsilon \in \mathbb{R}$ such that $\varepsilon \geq 0$, let $\widehat{A} = \left[\frac{A}{-c^T}\right]$ and let $\widehat{b_\varepsilon} = \left[\frac{b}{-\gamma - \varepsilon}\right]$. From the previous observations we get that the system

$$\begin{aligned} \widehat{A}x &\leq \widehat{b_\varepsilon} \\ x &\geq 0 \end{aligned}$$

is feasible if $\varepsilon = 0$ (since $x^*$ is a feasible solution) and infeasible if $\varepsilon > 0$. Since when $\varepsilon > 0$ the system is infeasible, so by Farkas' lemma we have that $\exists \widehat{y} \in \mathbb{R}^{m+1}$ such that $\widehat{A}^T \widehat{y} \geq 0$ with $\widehat{b_\varepsilon}^T \widehat{y} < 0$, implying that $\left[A^T \mid -c\right] \widehat{y} \geq 0$.

Let $u \in \mathbb{R}^m$ and $z \in \mathbb{R}$ be such that $\widehat{y} = \left[\frac{u}{z}\right]$. We get that

$$\left[A^T \mid -c\right] \widehat{y} \geq 0 \implies A^T u - cz \geq 0 \implies A^T u \geq cz$$

and that

$$\widehat{b_\varepsilon}^T \widehat{y} < 0 \implies b^T u + z(-\gamma - \varepsilon) < 0 \implies b^T u < z(\gamma + \varepsilon)$$

Likewise, since when $\varepsilon = 0$ the system $\widehat{A}x \leq \widehat{b_0}, x \geq 0$ has a solution, by Farkas' lemma we have that $\forall y \in \mathbb{R}^m$ such that $\widehat{A}^T y \geq 0$ and $y \geq 0$ it must hold that $\widehat{b_0}^T y \geq 0$. Since this also holds for $\widehat{y}$, we get that

$$\widehat{b_0}^T \widehat{y} \geq 0 \implies b^T u - z\gamma \geq 0 \implies b^T u \geq z\gamma$$

Now, we notice that $z > 0$, since otherwise $\widehat{y} \geq 0$ wouldn't be satisfied. Let $w = \frac{1}{z}u$. Since $A^T u \geq cz$ and $u \geq 0$, we have that $A^T w \geq c$ and $w \geq 0$, concluding that $w$ is a feasible solution to $(D)$. Finally, since we have that:

$$z\gamma \leq b^T u < z(\gamma + \varepsilon) \implies \gamma \leq b^T w < \gamma + \varepsilon$$

we conclude that $(D)$ is also bounded. In particular, since this constraint holds for all $\varepsilon > 0$, we get that $b^T w = \gamma$ is the only possibility, concluding that that $w$ is an optimal solution to $(D)$ and that $c^T x^* = b^T w$.

□

## 4.5 Geometry behind Farkas' lemma

Once we have show how the Farkas' lemma holds if and only if the strong duality theorem holds, we can give a geometric understanding of the lemma itself. Then, we will show how the geometric version is effectively equivalent to the algebraic one that we have seen in the previous section.

**Definition 31: Ball**

Given a vector $x \in \mathbb{R}^n$ and a value $\varepsilon \in \mathbb{R}$ where $\varepsilon > 0$, the **open ball around** $x$ **of radius** $\varepsilon$, denoted with $B^n_\varepsilon(x)$, is defined as:

$$B^n_\varepsilon(x) = \{y \in \mathbb{R}^n \mid \|x - y\| < \varepsilon\}$$

Likewise, the **closed ball around** $x$ **of radius** $\varepsilon$, denoted with $B^n_\varepsilon[x]$, is defined as:

$$B^n_\varepsilon[x] = \{y \in \mathbb{R}^n \mid \|x - y\| \leq \varepsilon\}$$

Given the definition, the following observation is trivial.

**Observation 15: Ball boundedness**

A linear program defined by $n$ variables and a polyhedra $P$ is **bounded** if and only if there is a large enough ball $B^n_\varepsilon[x]$ for some value $\varepsilon > 0$ and some vector $x \in \mathbb{R}^n$ that **contains** $P$, meaning that $P \subseteq B^n_\varepsilon[x]$.

**Definition 32: Open set**

A set $X \subseteq \mathbb{R}^n$ is **open** if $\forall x \in X$ it holds that $\exists \varepsilon \in \mathbb{R}$ such that $B^n_\varepsilon(x) \subseteq X$

Even thought the definition seems counterintuitive, one has to consider the fact that $\varepsilon$ can be as small as we like.

**Examples:**

- The set $X = \{x \in \mathbb{R} \mid \|2 - x\| < 1\}$ is closed since we can always pick a small enough $\varepsilon$ such that the open ball of radius epsilon of each point in $X$ is also contained in $X$. In fact, we notice that the set $X$ corresponds to the open interval $(1, 3) \subseteq \mathbb{R}$. Moreover, we also have that $X = B_1(2)$.
- The set $X = \{x \in \mathbb{R} \mid \|2 - x\| \leq 1\}$ isn't open since for each $\varepsilon > 0$ we have that $B_\varepsilon(1) \not\subseteq X$ and $B^n_\varepsilon(1) \not\subseteq X$. In fact, we notice that the set $X$ corresponds to the closed interval $[1, 3] \subseteq \mathbb{R}$.
- The set $X = \{x \in \mathbb{R} \mid (x < 0 \land \|2 - x\| < 1) \lor (x \geq 0 \land \|2 - x\| \leq 1)\}$ isn't open since for each $\varepsilon > 0$ we have that $B^n_\varepsilon(1) \not\subseteq X$. In fact, we notice that the set $X$ corresponds to the semi-open interval $[1, 3) \subseteq \mathbb{R}$.

**Definition 33: Closed set**

A set $X \subseteq \mathbb{R}^n$ is **closed** if $\mathbb{R}^n - X$ is open

**Examples:**

- The set $X = \{x \in \mathbb{R} \mid \|2 - x\| \leq 1\}$, is closed since $\mathbb{R} - X$ is open. In fact, we have that $X$ is equivalent to the closed interval $[1, 3] \subseteq \mathbb{R}$ and $\mathbb{R} - X$ is equivalent to the union of the open sets $(-\infty, 1) \cup (2, +\infty)$
- The set $X = \{x \in \mathbb{R} \mid \|2 - x\| \leq 1\}$ isn't closed
- The set $X = \{x \in \mathbb{R} \mid (x < 0 \wedge \|2 - x\| < 1) \vee (x \geq 0 \wedge \|2 - x\| \leq 1)\}$ isn't closed
- The set $\varnothing$ is both open and closed

**Proposition 15: Union of open sets**

The union of an arbitrary amount (meaning that it can be finite or infinite) of open sets is also **open**

*Proof.*

Let $X_1, X_2, \ldots \subseteq \mathbb{R}^n$ be an arbitrary amount of open sets and let $I = \{1, 2, \ldots\}$ be the set of the indices of such sets. Additionally, let $X = \bigcup_{i \in I} X_i$. Given $x \in X$, by definition we have that $\exists i \in I$ such that $x \in X_i$. Thus, since $X_i$ is open, we have that $\exists \varepsilon > 0$ such that $B^n_\varepsilon(x) \subseteq X_i \subseteq X$, concluding that $X$ is also open.

□

**Proposition 16: Union of closed sets**

The union of a finite amount of closed sets is also **closed**

*Proof.*

Let $X_1, \ldots, X_n \subseteq \mathbb{R}^n$ a finite amount of closed sets and let $X = \bigcup_{i \in I} X_i$. By De Morgan's theorem, we have that:

$$\overline{X} = \overline{\left(\bigcup_{i=1}^{n} X_i\right)} = \bigcap_{i=1}^{n} \overline{X_i}$$

Thus, given $x \in X$ we have that for each $X_i$ it holds that $x \in X_i$, implying that $\exists \varepsilon_i > 0$ such that $B^n_{\varepsilon_i}(x) \subseteq \overline{X_i}$. Let $\varepsilon = \min\{\varepsilon_1, \ldots, \varepsilon_n\}$. For each index $i$, we have that $B^n_\varepsilon(x) \subseteq B^n_{\varepsilon_i}(x) \subseteq \overline{X_i}$ and thus $B^n_\varepsilon \subseteq \overline{X}$, implying that $\overline{X}$ is open and thus that $X$ is closed.

□

*Note:* the argument fails in case of an infinite amount of sets since we could have an infinite amount of $\varepsilon_i$, meaning that there is no minimum value (remember that $\varepsilon_1, \varepsilon_2, \ldots$ are real values, so there could always be a smaller value in the infinite set). In fact, the infinite union of all closed subsets of $\mathbb{R}^n$ corresponds to $\mathbb{R}^n$ itself, which isn't closed.

**Definition 34: Convex cone**

Given $a_1, \ldots, a_n \in \mathbb{R}^m$, the **convex cone** of $a_1, \ldots, a_n$ is the set

$$C = \{t_1 a_1 + \ldots + t_n a_n \mid t_1, \ldots, t_n \geq 0\}$$

Obviously, it can be easily shown that a convex cone is convex (proof left for the reader as an exercise).

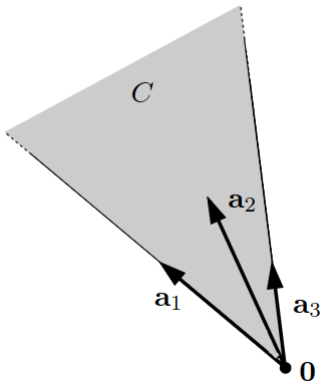



Figure 4.1: Example of a convex cone

**Definition 35: Primitive convex cone**

A convex cone $C$ defined by $a_1, \ldots, a_n \in \mathbb{R}^m$ is **primitive** if $a_1, \ldots, a_n$ are linearly independent

**Observation 16**

Every convex cone is the **union of a finite amount of primitive cones**

*Proof.*

Let $C$ be a convex cone defined by $a_1, \ldots, a_n \in \mathbb{R}^m$ and let $\mathcal{I} = \{I \subseteq \{1, \ldots, k\} \mid i_1, \ldots, i_k \in I$ such that $a_{i_1}, \ldots, a_{i_k}$ are lin. ind.$\}$. Since $\mathcal{I}$ potentially contains all subsets of $\{1, \ldots, n\}$, we have that $|\mathcal{I}| \leq 2^k$. For each $I \in \mathcal{I}$, let $C_I$ be the convex cone defined by the vectors which indexes are in $I$. By definition, we have that $\forall I \in \mathcal{I}$ it holds that $C_I$ is primitive.

Trivially, we have that $\forall I \in \mathcal{I}$ it holds that $C_I \subseteq C$ so we have that $\bigcup_{I \in \mathcal{I}} C_I \subseteq C$.

Instead, given $x \in C$, by definition we have that $x = \sum_{i=1}^{k} \alpha_i a_i$ where $\alpha_1, \ldots, \alpha_k \geq 0$.

If $v_1, \ldots, v_n$ are linearly independent then $x \in C_{\{1,\ldots,k\}} = C$ concluding that $C$ itself is the finite union of primitive cones. Otherwise, if they are linearly dependent, assume that $\alpha_1, \ldots, \alpha_k$ are picked to maximize the amount of zero scalars that make up the linear combination of $x$ (this assumption is always valid since they are linearly dependent).

Let $J = \{i \in I \mid \alpha_i > 0\}$. We notice that since $C$ is a convex cone, if $i \in I - J$ then $\alpha_i = 0$ and that $|J| < |I|$ since at least a zero scalar exists in the combination.

By way of contradiction, suppose that the set of vertices whose index is in $J$ is not linearly independent, meaning that $\exists \beta_1, \ldots, \beta_n \in \mathbb{R}$ such that $\sum_{j \in J} \beta_j a_j = 0$. Given $t \in \mathbb{R}$, consider $a_j - t\beta_j$ for some $j \in J$. Since $a_j > 0$ and $\beta_j \in \mathbb{R}$, we notice that

$$\alpha_j - t\beta_j \geq 0 \iff t \leq \frac{|\alpha_j|}{|\beta_j|}$$

Thus, we get that if $t \to +\infty$ then $\alpha_j - t\beta_j \to -\infty$, while if $t \to -\infty$ then $\alpha_j - t\beta_j \to +\infty$. Furthermore, this also implies that $\exists t' \in \mathbb{R}$ such that $\alpha_j - tb_j \geq 0$ for all $j \in J$. In particular, for the index $j$ that minimizes the value $\frac{|\alpha_j|}{|\beta_j|}$, we have that $t' = abs\alpha_j|\beta_j|$, implying that for some $j' \in J$ it holds that $\alpha_{j'} - t\beta_{j'} = 0$.

Finally, we get that:

$$\sum_{j \in J} (\alpha_j - t\beta_j) a_j = \sum_{j \in J} \alpha_j a_j - \sum_{j \in J} t\beta_j a_j$$

$$= \sum_{j \in J} \alpha_j a_j = \sum_{j \in J} \alpha_j a_j + \sum_{i \in I - J} \alpha_i a_i = \sum_{i \in I} \alpha_i a_i = x$$

concluding that $\sum_{j \in J} (\alpha_j - t\beta_j) a_j$ is a linear combination of $x$ with strictly more zero scalars than $\sum_{i \in I} \alpha_i a_i$ (since $\alpha_{j'} - t\beta_{j'} = 0$), which is a contradiction.

Thus, we get that the set of vertices whose index is in $J$ must be linearly independent, implying that $x \in C_J \subseteq \bigcup_{I \in \mathcal{I}} C_I$ and thus that $C \subseteq \bigcup_{I \in \mathcal{I}} C_I$.

□

**Proposition 17: Closed primitive convex cones**

Every primitive convex cone is **closed**

*Proof.*

Let $P \subseteq \mathbb{R}^m$ be the primitive convex cone generated by the linearly independent vectors $a_1, \ldots, a_n \in \mathbb{R}^m$, where $n \leq m$. Consider also the primitive convex cone $P_0 \subseteq \mathbb{R}^n$ generated by the linearly independent vectors $e_1, \ldots, e_n$ of the standard basis of $R^n$. Since $P_0 = \{t_1 e_1 + \ldots + t_n e_n \mid t_1, \ldots, t_k \geq 0\}$, it's easy to see that $P_0$ corresponds to the non-negative *orthant* (the $n$-dimensional generalization of a quadrant and an octant), which is obviously closed.

We define the linear mapping $f : \mathbb{R}^n \to \mathbb{R}^m : x \mapsto x_1 a_1 + \dots x_n a_n$. Since $a_1, \dots, a_n$ are linearly independent, we have that $\ker(f) = \{0\}$, implying that $f$ is injective and that $P = f(P_0)$, meaning that $P$ is equal to the image of $f$ on $P_0$. Consider now the restriction $g : \mathbb{R}^n \to \text{im}(f)$. Since $f$ is injective, we have that $g$ is also injective and that $|\mathbb{R}^n| = |\text{im}(f)|$, concluding that $g$ is an isomorphism from $\mathbb{R}^n$ to $\text{im}(f)$ and thus that there exists a inverse isomorphism $f^{-1}$.

We notice that $P = f(P_0) = (f^{-1})^{-1}(P_0)$, meaning that $P$ is the pre-image of $P_0$ for the inverse map $f^{-1}$. Moreover $\mathbb{R}^n$ and $\text{im}(f) \subseteq \mathbb{R}^m$ are euclidean spaces, so the linear maps $f$ and $f^{-1}$ are must be continuous.

Since $P = (f^{-1})^{-1}(P_0)$ and $f^{-1}$ is a continuous map, it must hold that $P$ is also closed (otherwise $f^{-1}$ wouldn't be continuous), concluding that $P$ is closed as a subset of $\text{im}(f)$. Moreover, since $\text{im}(f)$ is closed as a subset of $\mathbb{R}^m$ (due to it being a subspace), we get that $P$ must also be closed as a subset of $\mathbb{R}^m$.

□

**Corollary 6: Closed convex cones**

Every convex cone is **closed**

*(follows from Proposition 16, Observation 16 and Proposition 17)*

**Lemma 10: Closest point to a closed set**

Given a closed set $X \subseteq \mathbb{R}^n$ and a point $b \in \mathbb{R}^n$, there exists a point $x \in X$ closest to $b$, meaning that $\|x - b\|$ is minimal.

*Proof.*

Given $v \in X$, consider the closed ball $B^n_{\|v-b\|}[b]$ around $b$ of radius $\|v - b\|$. Since $X$ and $B$ are closed, it holds that $X \cap B^n_{\|v-b\|}[b]$ is also closed. Moreover, since $B^n_{\|v-b\|}[b]$ is bounded, meaning that it doesn't have an infinite ray inside, $X \cap B^n_{\|v-b\|}[b]$ is also bounded.

We define the map $f : X \cap B_{\|v-b\|}[b] \to \mathbb{R} : y \mapsto \|y - b\|$, i.e. $f(y)$ is the distance from $y \in X \cap B^n_{\|v-b\|}[b]$ to $b$. It's easy to see that $f$ is a continuous function. Thus, since we also know that $X \cap B^n_{\|v-b\|}[b]$ is closed and bounded, by the Bolzano-Weierstrass theorem we have that $f$ must have a minimum and a maximum value. Thus, we get that $\exists x \in X$ such that $f(x) = \|x - b\|$ is minimum.

□

**Theorem 14: Farkas' lemma (Geometric version)**

Given the vectors $a_1, \ldots, a_n, b \in \mathbb{R}^m$, exactly one of the following holds:

1. $b$ is **in the convex cone** $C$ defined by $a_1, \ldots, a_n$
2. There is a non-affine hyperplane $H = \{x \in \mathbb{R}^m \mid c^T x = 0\}$ for some $c \in \mathbb{R}^m$ such that $c^T b < 0$ and $\forall x \in C$ we have that $c^T x \geq 0$. In other words, the hyperplane $H$ **separates** $C$ **and** $b$.

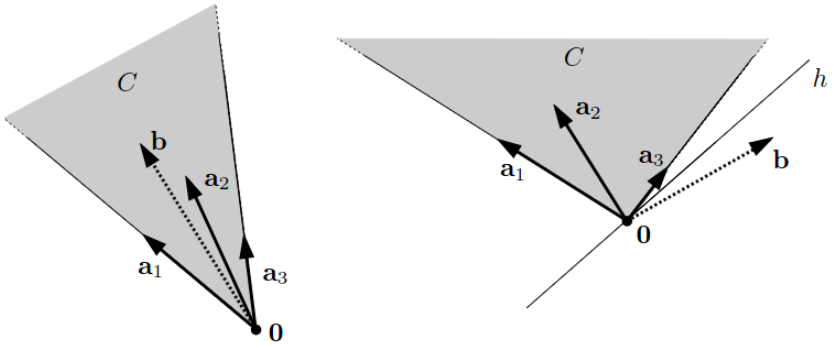



Figure 4.2: Visualization of the first and second implications of Farkas' lemma

*Proof.*

The fist implication is true if $b \in C$, so we just need to proove that when $b \notin C$ the second implication holds. Thus, suppose that $b \notin C$. By Lemma 10, we know that $\exists z \in C$ closest to $b$. Let $c = z - b$. We notice that $c$ must be orthogonal to $z$:

- If $z = 0$ then $c^T z = 0$, so they are orthogonal
- If $z \neq 0$ and they wouldn't be orthogonal, we would have that $z$ wouldn't be the point in $C$ closest to $b$

We notice that since $b \notin C$ and $0 \in C$, it holds that $c = z - b \neq 0$. Thus, since we know that $c^T z = 0$, we get that:

$$0 < \|c\| = c^T c = c^T (z - b) = c^T z - c^T b = -c^T b$$

concluding that $c^T b < 0$. Instead, given $x \in C$, we know that the angle $\theta = \angle xzb$ must be at least 90° and at most 270°. Otherwise, there would be a point in $C$ different from $z$ closer to $b$, which is a contradiction.

Thus, since $c^T(x - z) = \|c\| \cdot \|x - z\| \cdot \cos(\theta)$ and $\cos(\theta) \leq 0$, we get that $c^T(x - z) \leq 0$, implying that:

$$0 \geq c^T(x - z) = c^T x - c^T z = c^T x$$

concluding that $\forall x \in \mathbb{C}$ it holds that $c^T x \geq 0$. Thus, the non-affine hyperplane $H = \{y \in \mathbb{R}^m \mid c^T y = 0\}$ separates $C$ and $b$.

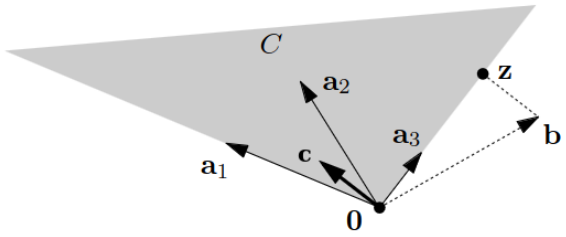



Figure 4.3: Visualization of proof's argument

□

**Observation 17**

The algebraic version and the geometrical version of Farkas' lemma are equivalent

*Proof.*

Given the vectors $a_1, \ldots, a_n, b \in \mathbb{R}^m$, let $A = \begin{bmatrix} a_1 \mid \ldots \mid a_n \end{bmatrix}$ be the $n \times m$ matrix of such vectors. Let also $C$ be the convex cone defined by $a_1, \ldots, a_n$. We notice that:

$$b \in C \iff \exists x \geq 0 \text{ such that } x_1 a_1 + \ldots + x_n a_n = b \iff Ax = b \text{ with } x \geq 0$$

concluding that the first implications of both versions are equivalent.

Viceversa, suppose that there is an hyperplane $H = \{x \in \mathbb{R}^n \mid y^T x = 0\}$ that separates $C$ from $b$. First, we notice that saying that $C$ resides on a side of $H$ is equivalent to saying that $a_1, \ldots, a_n$ reside on that side (since $a_1, \ldots, a_n$ define $C$). Thus, we get that $y^T b < 0$ and $\forall x \in C \;\; c^T y \geq 0$ if and only if $y^T b < 0$ and $y^T a_1, \ldots, y^T a_n \geq 0$. However, the latter can only if and only if $A^T y \geq 0$, concluding that the second implications of both versions are equivalent.

□

## 4.6 Complimentary slackness

**Definition 36: Slack and tight constraints**

A constraint in the form $a_1x_1 + \ldots + a_nx_n \leq b$ is said to be:

- **Slack** for a solution $x^*$ if it's a satisfied as a strict inequality, meaning that $a_1x_1^* + \ldots + a_nx_n^* < b$.
- **Tight** for a solution $x^*$ if it's satisfied as an equality, meaning that $a_1x_1^* + \ldots + a_nx_n^* = b$.

The origin of these two definitions directly comes from the concept of *slack variables*: once we convert the constraint in standard form, the original constraint was strict if and only if the added slack variable is set to a value different from 0.

Through the concept of duality, we know that each variable of the primal problem is associated to a constraint of the dual program and that every variable of the latter is associated to a constraint of the primal program. In particular, the $i$-th constraint of $(P)$ can be described as $A_ix_i \leq b_i$, where $A_i$ denotes the $i$-th row of $A$, and the $j$-th constraint of $(D)$ can be described as $A_j^Ty_j \geq c_j$. Through the concept of constraint slackness and the duality theorems, we can the derive the following relations.

**Theorem 15: Complementary slackness**

Given a primal dual pair $(P), (D)$, let $x^*$ be an optimal solution to $(P)$ and let $y^*$ be an optimal solution to $(D)$. The following relations hold:

1. If $x_i^* > 0$ then $A_i^Ty_i^* = c_i$
2. If $A_ix_i^* < b_i$ then $y_i^* = 0$
3. If $y_j^* > 0$ then $A_jx_j^* = b_i$
4. If $A_j^Ty_j^* > c_j$ then $x_j^* = 0$

*Proof.*

For any feasible solution $x$ and $y$ to $(P)$ and $(D)$, we know that $Ax \leq b$ and $A^Ty \geq c$ must hold. Thus, we get that:

$$Ax \leq b \iff y^TAx \leq y^Tb = b^Ty$$

and that:

$$A^Ty \geq c \iff y^TA \geq c^T \iff y^TAx \geq c^Tx$$

concluding that $c^Tx \leq y^TAx \leq b^Ty$ (notice that this relation is effectively a proof of the weak duality theorem). In particular, by the strong duality theorem we know that $c^Tx^* = b^Ty^*$, implying that $c^Tx^* = y^{*T}Ax^* = b^Ty^*$ holds.

Since $c^T x^* = {y^*}^T A x^*$, we get that:

$$c^T x^* = {y^*}^T A x^* \iff (c^T - {y^*}^T A)x^* = 0 \iff \sum_{i=1}^{n}(c_i^T - {y_i^*}^T A_i)x_i^* = 0$$

Now, for each index $i$, we get two cases:

- If $x_i^* > 0$ then $(c_i^T - {y_i^*}^T A_i)x_i^* = 0$ holds only if $c_i^T - {y_i^*}^T A_i = 0$, which implies that $c_i^T = {y_i^*}^T A_i = A_i^T y_i^*$, giving the first relation of the theorem
- If $A_i^T y_i^* > c_i$ then we get that $0 > c_i - A_i^T y_i^* = c_i^T - {y_i^*}^T A_i$, implying that $(c_i^T - {y_i^*}^T A_i)x_i^* = 0$ holds only if $x_i^* = 0$, giving the fourth relation of the theorem

Likewise, since $b^T y^* = {y^*}^T A x^*$, we can derive the second and third relations.

*Note:* notice that if $x_i^* = 0$ we can't derive that $A_i^T y_i^* > c_i^T$ because the zero-product property allows $c_i^T - {y_i^*}^T A_i$ to also be equal to zero. By the same reasoning, we can't conclude that $A_i^T y_i^* = c_i$ implies that $x_i^* > 0$.

□

The **complementary slackness theorem** gives us a very rapid way to find the optimal solution of the dual (or the primal) if we have an optimal solution of the primal (or the dual).

Suppose that $x^*$ is an optimal solution of $(P)$. For each index $i$ such that $x_i^* > 0$, we know that the $i$-th constraint of $(D)$ must be tight, meaning that $A_i^T y_i^* = c_i$ must hold. Moreover, for each index $j$ such that $A_j x_j^* > 0$, we know that $y_j^* = 0$ must hold. By combining these equalities, we get a system of equations that directly gives us the values of the optimal solution $y^*$.

For example, consider the following primal program (on the left) and its dual program (on the right):

$$\begin{aligned} \max\ & 2x_1 + 4x_2 + 3x_3 + x_4 \\ & \\ 3x_1 + x_2 + x_3 + 4x_4 &\leq 12 \\ x_1 - 3x_2 + 2x_3 + 3x_4 &\leq 7 \\ 2x_1 + x_2 + 3x_3 - x_4 &\leq 10 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned} \qquad \begin{aligned} \min\ & 12y_1 + 7y_2 + 10y_3 \\ & \\ 3y_1 + y_2 + 2y_3 &\geq 2 \\ y_1 - 3y_2 + y_3 &\geq 4 \\ y_1 + 2y_2 + 3y_3 &\geq 3 \\ 4y_1 + 3y_2 - y_3 &\geq 1 \\ y_1, y_2, y_3 &\geq 0 \end{aligned}$$

Consider also the optimal solution $x^* = \begin{bmatrix} 0 & 10.4 & 0 & 0.4 \end{bmatrix}$ for the primal program. Since $x_2^*, x_4^* > 0$, by the complementary slackness theorem we know that for the optimal solution $y^*$ it must hold that $y_1^* - 3y_2^* + y_3^* = 4$ and that $4y_1^* + 3y_2^* - y_3^* = 1$. Moreover, we notice that the second constraint of the primal is slack for $x^*$, meaning that $x_1^* - 3x_2^* + 2x_3^* + 3x_4^* < 7$, implying that $y_2^* = 0$ must hold.

Thus, we get the following system of equations:

$$\begin{cases} y_1^* - 3y_2^* + y_3^* = 4 \\ 4y_1^* + 3y_2^* - y_3^* = 1 \\ y_2^* = 0 \end{cases}$$

By solving the system, we get the dual optimal solution $y^* = \begin{bmatrix} 1 & 0 & 3 \end{bmatrix}$.

Complementary slackness can also be used to **check if a feasible solution is optimal** or not. For example, consider the following primal program (on the left) and its dual program (on the right):

$$\begin{aligned} \max\ & 2x_1 + 16x_2 + 12x_3 \\ & \\ 2x_1 + x_2 - x_3 &\leq 3 \\ -3x_1 + 8x_2 + 2x_3 &\leq 12 \\ x_1, x_2, x_3 &\geq 0 \end{aligned} \qquad \begin{aligned} \min\ & 3y_1 + 12y_2 \\ & \\ 2y_1 - 3y_2 &\geq 2 \\ y_1 + 8y_2 &\geq 16 \\ -y_1 + 2y_2 &\geq 12 \\ y_1, y_2 &\geq 0 \end{aligned}$$

We wan't to check if the feasible solution $x^* = \begin{bmatrix} 6 & 0 & 12 \end{bmatrix}$ is also optimal. By assuming it is primal optimal, we can apply complementary slackness theorem to find the dual optimal solution $y^*$. In this particular example, we can find two solutions:

1. Since $2x_1^* + x_2^* - x_3^* < 3$ and $-3x_1^* + 8x_2^* + 2x_3^* < 12$, we get that $y_1^*, y_2^* = 0$, giving us the dual optimal solution $y^* = \begin{bmatrix} 0 & 0 \end{bmatrix}$. However, we notice that that $y^*$ is not dual feasible, which is impossible. Thus, we get that $x^*$ cannot be an optimal solution.

2. Since $x_1^*, x_3^* > 0$, we get that $2y_1^* - 3y_2^* = 2$ and $-y_1^* + 2y_2^* = 12$, resulting in the system:

$$\begin{cases} 2y_1^* - 3y_2^* = 2 \\ -y_1^* + 2y_2^* = 12 \end{cases}$$

   By solving the system, we get the dual optimal solution $y^* = \begin{bmatrix} 40 & 26 \end{bmatrix}$. However, we notice that:

$$2x_1^* + 16x_2^* + 12x_3^* = 156 \neq 432 = 3y_1^* + 12y_2^*$$

   meaning that the two optimal solutions don't share the same value, which is impossible. Thus, we get that $x^*$ cannot be an optimal solution.

## 4.7 Solved exercises

**Problem 4: Self-duality**

Consider the following primal program:

$$\begin{aligned} \min\ & c^T x \\ Ax &\geq b \\ x &\geq 0 \end{aligned}$$

1. Form the dual program and state it as an equivalent minimization problem
2. Derive the conditions on $A, b$ and $c$ such that the dual is identical to the primal, meaning that the problem is *self-dual*
3. Give a concrete example of a primal program identical to its dual

*Solution:*

First, we notice that the given primal is a minimization problem. Thus, by Proposition 5 we convert it into a general maximization problem:

$$\begin{aligned} \min\ & c^T x \\ Ax &\geq b \\ x &\geq 0 \end{aligned} \qquad \Longrightarrow \qquad \begin{aligned} \max\ & -c^T x \\ -Ax &\leq -b \\ x &\geq 0 \end{aligned}$$

Now, its easy to see that its dual corresponds to:

$$\begin{aligned} \min\ & -b^T y \\ -A^T y &\geq -c \\ y &\geq 0 \end{aligned}$$

Moreover, we also notice that if $c = -b$ and $A = -A^T$, meaning that the matrix is *skew-symmetric*, the dual and the primal are identical. In fact, given the following primal program:

$$\begin{aligned} \min\ & x_1 + x_2 + x_3 \\ x_2 + x_3 &\geq -1 \\ -x_1 - x_3 &\geq -1 \\ -x_1 - x_2 &\geq -1 \\ x_1, x_2, x_3 &\geq 0 \end{aligned} \qquad \Longrightarrow \qquad \begin{aligned} \min\ & \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ & \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

by following the steps described above, we get that the dual corresponds to:

$$\begin{aligned} \min\ & y_1 + y_2 + y_3 \\ & \\ & y_2 + y_3 \geq -1 \\ & -y_1 - y_3 \geq -1 \\ & -y_1 - y_2 \geq -1 \\ & y_1, y_2, y_3 \geq 0 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} & \min \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \\ & \\ & \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \leq \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

# 5

# Other ways to solve a linear program

## 5.1 Runtime of the Simplex method

Once we have discussed everything regarding how the Simplex method works and how to handle infeasibility, unboundedness, degeneracy and cycling, we can finally discuss the **efficiency** of the Simplex method.

We have already discussed that a linear program has at most $\binom{n}{m}$ feasible bases. However, each pivot rule never goes through all the bases, so could there be a pivot rule that is efficient?

The **largest coefficient rule** originally suggested by Dantzig has been proven by Klee and Minty to actually be the least efficient rule. They formulated a way to construct an $n$-dimensional polyhedron in $\mathbb{R}^n$ with at most $3n$ constraints that always takes $2^n - 1$ pivot steps using Dantzig's rule.

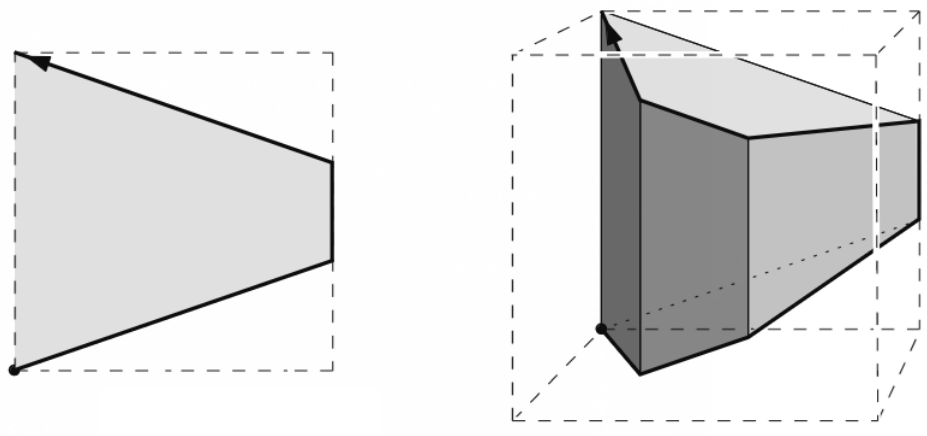


Figure 5.1: The Klee-Minty polyhedron in $\mathbb{R}^2$ and $\mathbb{R}^3$ and the respective paths of length $2^2 - 1$ and $2^3 - 1$

By *squishing* this polyhedron, which can be done by altering the coefficients of the constraints, Klee and Minty showed that the largest coefficient rule always takes these longest paths. Thus, the largest coefficient rule has a an exponential runtime, meaning that it's actually very inefficient.

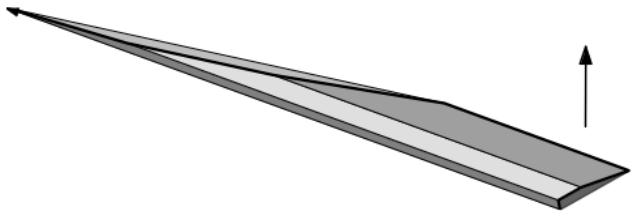


Figure 5.2: Squished version Klee-Minty polyhedron that fools Dantzig's rule

In particular, the previous squished polyhedron is given by the following linear program:

$$\max \quad 9x_1 + 3x_2 - x_3$$

$$\begin{aligned} x_1 &\leq 1 \\ 6x_1 + x_2 &\leq 9 \\ 18x_1 + 6x_2 - 2x_3 &\leq 81 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Once Dantzig's rule was fooled by Klee and Minty, researchers started analyzing the runtime of other rules. In particular, it was shown that if you randomly order the $n$ variables of the linear program and apply **Bland's rule**, the expected number of pivots is at most $e^{C\sqrt{n \ln n}}$, where $C$ is a not too large constant. Thus, Bland's rule has a super-polynomial runtime, meaning that it's worse than any polynomial but better than any exponential, which is indeed better than Dantzig's rule, but still inefficient.

In fact, the lower bound is not know even for the yet undiscovered *clairvoyant rule*, i.e. a pivot rule capable of always taking the shortest path possible. **Hirsh's conjecture** suggests that any polyhedron in $\mathbb{R}^n$ with at most $cn$ constraints has a diameter of at most $n^c$, where the diameter of a polyhedron is the maximum length of any path between two vertices.

This conjecture directly gives a polynomial lower bound for the clairvoyant rule, which implies that the Simplex method can in fact run efficiently. However, Hirsh's conjecture has been proven to be false by Leal in 2011, but this doesn't imply that the Simplex method can't run in polynomial time.

Even though in theory it is very inefficient, in practice **the Simplex method usually terminates in a reasonable amount of time**, giving the worst case only for very specifically crafted polyhedra.

### 5.1.1 Bit size of a linear program

Once we know that the Simplex method currently requires at most a super-polynomial amount of pivot steps, we can properly discuss the **computational complexity** of the Simplex method. In order to do so, we have to first define the computational measure of any algorithm that solves a linear program. This measure is based on the input **bit size $\langle P \rangle$ of the given linear program**, which is the amount of bits required to define it.

Any linear program is defined by the constraint matrix $A$ and the two vectors $b$ and $c$, so the bit size of the program corresponds to the sum of the bit sizes of these three elements, meaning that $\langle P \rangle = \langle A \rangle + \langle b \rangle + \langle c \rangle$.

Any integer $\alpha$ has a bit size equal to $\langle \alpha \rangle = \lceil \log \alpha + 1 \rceil + 1$, where $\lceil \log \alpha + 1 \rceil$ bits are used to expresses the absolute value of $\alpha$ and the last additional bit expresses the sign of $\alpha$. Moreover, even thought the program is defined on $\mathbb{R}$, any real number must be approximated as a rational number, since we can't express an infinite amount of bits. Any rational number $\frac{\beta}{\gamma}$ is represented by the integers $\beta, \gamma$, meaning that it requires a bit size of $\left\langle \frac{\beta}{\gamma} \right\rangle = \langle \beta \rangle + \langle \gamma \rangle$. Thus, the sizes of the three elements are given by:

$$\langle A \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} \langle a_{i,j} \rangle \qquad \langle b \rangle = \sum_{i=1}^{n} \langle b_i \rangle \qquad \langle c \rangle = \sum_{i=1}^{n} \langle c_i \rangle$$

where each of these numbers are an integer or a rational number (which is still represented by two integers). The discussed results of the runtime of the Simplex method are based on the number of variables that define the problem. However, such number is defined by the three elements $A, b, c$. Thus, the current runtime of the Simplex method is effectively $\mathsf{sup\text{-}poly}(\langle A \rangle + \langle b \rangle + \langle c \rangle)$.

## 5.2 The Ellipsoid method

We know that the Simplex method doesn't always give us an optimal solution in an reasonable amount of time. However, other methods can effectively give such solution in an efficient way. In particular, we have shown in Proposition 14 that finding a feasible solution is computationally equivalent to finding an optimal solution. The latter problem can always be solved in $\mathsf{poly}(\langle A\rangle + \langle b\rangle + \langle c\rangle)$ by the **Ellipsoid method**, allowing us to find an optimal solution efficiently with the method shown in Proposition 14.

Informally, an ellipsoid is the $n$ dimensional generalization of an ellipse. It's easy to see that an ellipsoid is nothing more than an **affine transformation** of a ball which *translates*, *squashes* and *stretches* the ball.

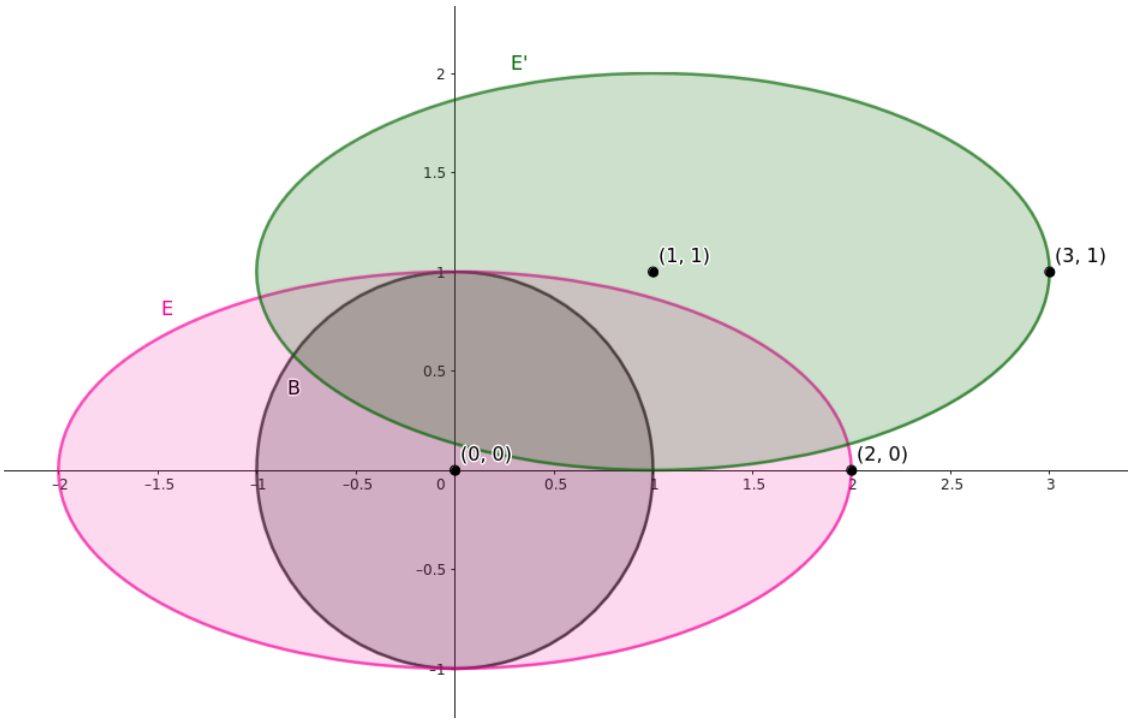
**Definition 37: Ellipsoid**

An **ellipsoid** is an affine transformation of the closed ball of radius one centered in zero, $B_1^n[0]$:

$$E = \{Mx + s \in R^n \mid x \in B_1^n[0]\}$$

where $M$ is a non-singular matrix and $s \in \mathbb{R}^n$

For example, in the following image we have $B := B_1^2[0]$, $E = \left\{ \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} x \mid x \in B \right\}$ and $E' = \left\{ \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mid x \in B \right\}$



We notice that for every ellipsoid $E$ it holds that:

$$\begin{aligned} E &= \{Mx + s \in R^n \mid x \in B_1^n[0]\} \\ &= \{y \in R^n \mid M^{-1}(y - s) \in B_1^n[0]\} \\ &= \{y \in R^n \mid \|0 - M^{-1}(y - s)\| \leq 1\} \\ &= \{y \in R^n \mid (y - s)^T (M^{-1})^T M^{-1}(y - s) \leq 1\} \\ &= \{y \in R^n \mid (y - s)^T Q^{-1}(y - s) \leq 1\} \end{aligned}$$

where $Q := MM^T$ and $Q^{-1} = (M^{-1})^T M^{-1}$. This result implies that $Q$ is a **positive definite matrix**, that being a symmetric square matrix such that $x^T Qx > 0$ for all vectors $x \in \mathbb{R}^n$ such that $x \geq 0$. Conversely, from matrix theory it is known that each positive definite matrix $Q$ can be factored as $Q = MM^T$ for some non-singular square matrix $M$. Thus, we get the following theorem and its corollary.

**Theorem 16**

$Q$ is a positive definite matrix if and only if there is a non-singular matrix $M$ such that $Q = MM^T$

**Corollary 7**

$E \subseteq \mathbb{R}^n$ is an ellipsoid with center $s \in \mathbb{R}^n$ if and only if there is a non-singular matrix $M$ such that, given $Q = MM^T$, it holds that $E = \{y \in R^n \mid (y-s)^T Q^{-1}(y-s) \leq 1\}$

Now that we have shown the main concepts involved, we can define the **Ellipsoid method**. Let $R, \varepsilon \in \mathbb{R}$ be two additional input values. Suppose that we have already verified that the polyhedron of feasible solutions $P = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$ is contained inside a ball of radius $R$, meaning that $P \subseteq B_R^n[0]$.

The algorithm proceeds by finding a series of ellipsoids $E_0, \ldots, E_t$ such that $\forall i \;\; P \subseteq E_i$ and either the center $s_t$ of $E_t$ is contained inside of $P$ or the volume of $E_t$ is less than a ball of radius $\varepsilon$ (meaning that the input value $\varepsilon$ is a threshold value). The first case happens when there is a feasible solution to the linear program, while the latter happens when such solution doesn't exist:

1. Let $E_0 = B_R^n[0]$ be the initial ellipsoid and set $k = 0$.
2. Suppose that $P \subseteq E_0, \ldots, E_k$. Let $s_k$ be the center of the ellipsoid $E_k$, i.e. the last computed ellipsoid. If $s_k \in P$, then set $t = k$ and **return** $s_k$.
3. Otherwise, let $A_i^T x \leq b_i$ be one of the constraints violated by $s_k$, where $A_i$ denotes the $i$-th row of $A$. The hyperplane given by $A_i^T x \leq A_i^T s_k$ splits the ellipsoid $E_k$ in two half-ellipsoids, where all of $P$ lies inside the half-ellipsoid $H_k = \{x \in E_k \mid A_i^T x \leq A_i^T s_k\}$, meaning that $P \subseteq H_k$.
4. Let $E_{k+1}$ be the ellipsoid with the smallest possible volume such that $H_k \subseteq E_{k+1}$. If the volume of $E_{k+1}$ is smaller than the volume of the ball $B_\varepsilon^n(0)$, return **no solution**. Otherwise, increase $k$ by 1 and repeat the process from step 2.

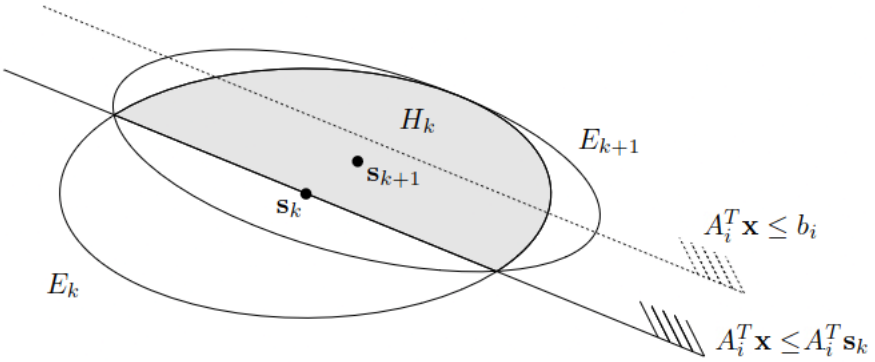



Figure 5.3: The $k$-th iteration of the Ellipsoid method

**Claim 1**: For each $k \geq 0$, the ellipsoid $E_{k+1}$ is uniquely defined and given by following the equations:

$$s_{k+1} = s_k - \frac{1}{n+1} \cdot \frac{Q_k s_k}{\sqrt{s_k^T Q_k s_k}}$$

$$Q_{k+1} = \frac{n^2}{n^2-1}(Q_k - \frac{2}{n+1} \cdot \frac{Q_k s_k s_k^T Q_k}{s_k^T Q_k s_k})$$

where $Q_0, \ldots, Q_{k+1}$ are the positive definite matrices that define the ellipsoids $E_0, \ldots, E_{k+1}$.

Moreover, it holds that:

$$\frac{\text{Vol}(E_{k+1})}{\text{Vol}(E_k)} \leq e^{-\frac{1}{2n+2}}$$

*Proof of the claim omitted.* □

In particular, we notice that $E_0 = B_R^n(0) = \{(R \cdot I_m)x \mid x \in B_1^n(0)\}$, meaning that $Q_0 = (R \cdot I_m)(R \cdot I_m)^T$. The two equations of each iteration given by the claim can be easily computed in polynomial time.

The claim also affirms that, since $e^{-\frac{1}{2n+2}} > 1$ is always true, the volume of the ellipses get **smaller and smaller by each iteration**. This ensures that, at some point, it will always hold that $\text{Vol}(E_t) < \text{Vol}(B_\varepsilon^n(0))$, meaning that the always algorithm terminates.

**Claim 2**: The Ellipsoid method terminates after $\lceil n(2n+2)\ln\left(\frac{R}{\varepsilon}\right)\rceil$ iterations.

*Proof of the claim.*

By the previous claim, we know that for each iteration $k$ it holds that:

$$\text{Vol}(E_k) \leq \text{Vol}(E_0) \cdot e^{-\frac{k}{2n+2}}$$

Since $E_0 = B_R^n(0)$, we know that $\text{Vol}(E_0) = cR^n$ for some constant $c \in \mathbb{R}$ such that $c > 1$, implying that:

$$\text{Vol}(E_k) \leq cR^n \cdot e^{-\frac{k}{2n+2}}$$

If $k \geq n(2n+2)\ln\left(\frac{R}{\varepsilon}\right)$, assuming that $\varepsilon < 1$, we get that:

$$\text{Vol}(E_k) \leq cR^n \cdot e^{-\frac{k}{2n+2}} \leq cR^n \cdot e^{-\frac{n(2n+2)\ln\left(\frac{R}{\varepsilon}\right)}{2n+2}} = \frac{cR^n}{e^{n\ln\left(\frac{R}{\varepsilon}\right)}} = c\varepsilon^n < \varepsilon$$

so the algorithm is guaranteed to terminate after $\left\lceil n(2n+2)\ln\left(\frac{R}{\varepsilon}\right)\right\rceil$ iterations.

□

Even though the idea behind the Ellipsoid method is simple and elegant, there are some **technicalities** to consider. The first technicality arises in Claim 1 since the two equations contain a square root term, which may be some value in $\mathbb{R}$. However, we are assuming to be working in $\mathbb{Q}$, so we have to introduce a *small approximation factor*, meaning that each computed ellipsoid is **slightly larger** than the smallest possible ellipsoid.

The second technicality is pretty obvious: we have to find two suitable values for $R$ and $\varepsilon$. Let $\varphi = \langle A\rangle + \langle b\rangle$ and let $K = 2^{\varphi}$.

**Claim 3**: The system $Ax \leq b$ has a feasible solution if and only if the system

$$\begin{aligned} Ax &\leq b \\ -K \leq x_1 &\leq K \\ &\vdots \\ -K \leq x_n &\leq K \end{aligned}$$

has a feasible solution.

*Proof of the claim.*

Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ and let $P' = \{x \in \mathbb{R}^n \mid Ax \leq b \text{ and } \forall i\ -K \leq x_i \leq -K\}$. If $P' \neq \varnothing$ we trivially get that $P \neq \varnothing$ since $P' \subseteq P$. Viceversa, if $P \neq \varnothing$ then there is a BFS $x^*$ which allows us to solve $Ax \leq b$ for each basic variable in terms of $A$ and $b$, implying that the amount of bits of each variable are bounded by:

$$0 \leq \langle x_i\rangle \leq \langle a_{1,1}\rangle + \ldots + \langle 1, n\rangle + \langle b_1\rangle + \ldots + \langle b_m\rangle = \langle A\rangle + \langle b\rangle = \varphi$$

Hence, for each variable we have that $0 \leq x_i \leq 2^{\varphi} = K$, implying that $x^* \in P'$ and thus $P' \neq \varnothing$.

□

This claim doesn't conclude that both problems are equal. In fact, it may not be true. However, **any solution** to the *relaxed* problem is a solution to the original problem, meaning that we can apply the Ellipsoid method to the second problem and find a valid solution to the original problem.

In particular, it's easy to see that the polyhedron $P'$ of the relaxed problem shown is contained inside an hypercube of side $2k$. The vertices of this hypercube are given by $\begin{bmatrix}\pm K & \cdots & \pm K\end{bmatrix}$. Thus, we can inscribe this hypercube (and by consequence the polyhedron) inside a $n$-dimensional ball of radius $\left\|\begin{bmatrix}\pm K & \cdots & \pm K\end{bmatrix}\right\| = K\sqrt{n}$. By setting

$R = K\sqrt{n}$, we always know that the ball $B_R^n[0]$ contains $P'$, giving us an **initial ball** to start the Ellipsoid method.

The last technicality to take care of is the choice of the value $\varepsilon$. Let $\eta = 2^{-5\varphi}$ and $\varepsilon = 2^{-6\varphi}$.

**Claim 4**: Let $z = \begin{bmatrix}\eta & \cdots & \eta\end{bmatrix}$. The system $Ax \leq b$ has a solution then $Ax \leq b + z$ has a solution. Moreover, the polyhedron of the latter system contains a ball of radius $\varepsilon$.

*Proof of the claim omitted.* □

This claim directly implies that by setting $\varepsilon = 2^{-6\varphi}$ we are guaranteed that the ball of radius $\varepsilon$ used by the Ellipsoid method works.

In summary, by setting $R = 2^{\langle A\rangle + \langle b\rangle}$ and $\varepsilon = 2^{-6(\langle A\rangle + \langle b\rangle)}$ the Simplex method is guaranteed to return a feasible solution (if there is any) after

$$\left\lceil n(2n+2)\ln\left(\frac{R}{\varepsilon}\right)\right\rceil = \lceil 7n(2n+2)(\langle A\rangle + \langle b\rangle)\rceil$$

iterations, implying that the complexity of the Ellipsoid method is $\mathsf{poly}(\langle A\rangle + \langle b\rangle)$.

However, even thought the Ellipsoid method theoretically runs faster than the Simplex method, in the average case the latter performs better. To give a "fair" justification of this, consider that in order to find an optimal solution with the Ellipsoid method we have to:

1. Compute the dual problem
2. Check with the Ellipsoid method if both the primal and the dual have a feasible solution, which involves a lot of matrix multiplications (a very slow, but polynomial, calculation)
3. Construct the auxiliary program as shown in Proposition 14, which has double the size of the input
4. Compute the auxiliary program using the Ellipsoid method, which again involves a lot of matrix multiplications

Instead, the Simplex method just requires to compute a bunch of tableaus, which is a very simple task.

## 5.3 The Interior Points methods

We have seen that, when solving a linear program, the Simplex method crawls along the boundary of the set of feasible solutions by moving from one vertex to another. The Ellipsoid method, instead, encircles the set of feasible solutions up until when, in the last step, it remains outside of it.

The **Interior Point methods** take another approach: starting from a feasible solution, they walk through the interior of the set of feasible solutions toward an optimum, carefully avoiding the boundary of the feasible region. Only at the very end, when the optimum has almost been reached, they jumps to an exact optimum by a rounding step.

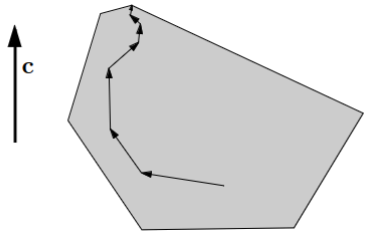



Figure 5.4: Starting from a feasible solution, the Interior Points methods follow the direction of the objective function vector until reaching an optimum

There are several rather different basic approaches in Interior Point methods, and each has many variants. In particular, we will discuss the ideas behind the most commonly used **Central Path Method**, which is also the method with better performances.

First we explain the mathematical notion of central path. As usual, we consider an arbitrary convex polyhedron $P$ in $R^n$ defined by a system $Ax \leq b$ of $m$ linear inequalities, and a linear objective function $f(x) = c^T x$. We introduce a **family of auxiliary objective functions** $f_\mu$, each depending on a parameter $\mu \in [0, +\infty)$:

$$f_\mu(x) = c^T x + \mu \cdot \sum_{i=1}^{m} \ln(b_i - A_i x)$$

where $A_i$ denotes the $i$-th row of the matrix $A$. We notice that as $x$ gets **closer to any border of** $P$, for example the one dictated by the $i$-th constraint of $A$, we get that $b_i - A_i x \to 0$, implying that $\ln(b_i - A_i x) \to -\infty$ and thus that $f_\mu(x) \to -\infty$.

We consider the auxiliary problem of maximizing $f_\mu(x)$ over $P$ , for a given $\mu > 0$. Since $f_\mu$ is undefined on the boundary of $P$, we actually maximize over the interior of $P$, denoted with $\text{int}(P)$, implying that we need to assume that $\text{int}(P) \neq= \varnothing$. By assume that, we also assume that $P$ is bounded, giving us the following claim:

**Claim**: For each $\mu \in [0, +\infty]$, the function $f_\mu(x)$ has an unique maximum on $\text{int}(P)$.

*Proof of the claim.*

In general, if $g : X \to Y$ is a continuous function and $\mathcal{U} \subseteq \text{im}(g)$ is an open set, we know that the pre-image $g^{-1}(\text{U}) \subseteq X$ is also open.

For any $z \in \text{int}(P)$, consider the open set $(-\infty, f_\mu(z))$. By the previous remark we get that $f_\mu^{-1}(-\infty, f_\mu(x)) = \{x \in P \mid f_\mu(x) < f_\mu(z)\}$ is an open set, implying by definition that $\{x \in P \mid f_\mu(x) \geq f_\mu(z)\}$ is closed.

Thus, since $f_\mu(x)$ is continuous and bounded on this set for any $z \in \text{int}(P)$, by the Bolzano-Weierstrass theorem we have that $f_\mu(x)$ also has a maximum on $\text{int}(P)$.

By way of contradiction, suppose that $\exists x, y \in P$ such that $x \neq y$ and $f_\mu(x) = f_\mu(y) = \max(f_\mu)$.

Since the ln is a strictly concave function, meaning that $\forall \alpha \in \mathbb{R}$ if $a \neq b$ then it holds that $\ln(\alpha a + (1-\alpha)b) > \alpha \ln(a) + (1-\alpha)\ln(b)$, we get that in order for $f_\mu(x) = f_\mu(y)$ to be true it must hold that $A_i x = A_i y$ forall indices $i$, implying that $Ax = Ay$.

However, this implies that $\forall \beta \in \mathbb{R}$ it holds that $x + \beta(y - x) \in P$, meaning that $P$ is unbounded since it contains a line, which is a contradiction to our assumption. Thus, it must hold that $x = y$, meaning that the maximum is unique.

□

We define the **central path** of the linear problem is defined as the set of maxima $\{x_\mu^* = \max(f_\mu) \mid \mu > 0\}$. If $\mu$ is a very large number, the influence of the logarithm term inside the function $f_\mu(x)$ is too high compared to the term $c^T x$. Thus, if we compute the maximum of $f_\mu(x)$ in this situation we get a point that is "farthest from the boundary".

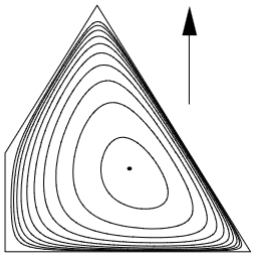


Figure 5.5: The maximum $x_\mu^*$ when $\mu$ is set to be a large number

Instead, when $\mu$ gets closer to zero the function $f_\mu(x)$ approaches the original objective function $c^T x$.

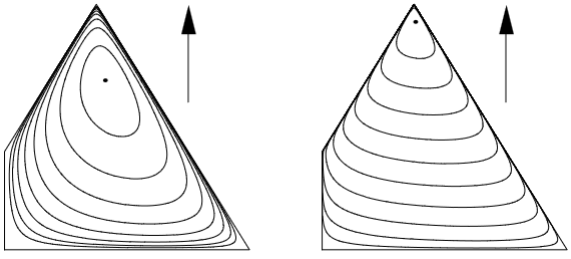


Figure 5.6: The maximum $x_\mu^*$ when $\mu$ gets closer to zero

The idea of Central Path Method is to start from $x^*(\mu)$ with a suitable large $\mu$ and then **follow the central path decreasing** $\mu$ until an optimum of $c^T x$ is reached.

Since, computing each maximum $x^*(\mu)$ requires advanced analysis operations, such as *Lagrange multipliers*, the central path gets slightly approximated. Since it would require too many topics to be discussed, we won't dive into how these calculations are done.

Tt has been proven that the Central Path method has a **polynomial complexity**, meaning that it should theoretically be faster than the Simplex method (and by extension faster than the Ellipsoid method in the general case). Moreover, published results of computational experiments suggest that, even in practice, it is much faster than the Simplex method.

Although this statements in the latter direction turned out to be somewhat exaggerated, Interior Point methods are nowadays commonly used in linear programming and *often* they beat the simplex method, especially on very large linear programs.

# 6 Integer programming

## 6.1 Intuition and examples

In the previous chapters we have discussed how linear programs are structured and how they can be solved through various methods. One main characteristic of such linear programs, up until now, is that we were assuming to be working in $\mathbb{R}$ (technically $\mathbb{Q}$ since all of the three methods we have seen work with rationals due to their nature or due to approximations). However, a large part of real world problems have require that the solution is made of **integers**. For example, if we want to optimize the number of concurrent flights of an airport subject to some constraints, there is no real meaning in having a rational result since we can't take a *slice* of an airplane. These types of linear programs are called **integer programs** and, like linear programs, they can be described with the following generalization:

$$\begin{gathered} \max\ c^T x \\ Ax \leq b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{gathered}$$

Many common optimization problems are described through this model. For example, consider the famous **0/1 Knapsack Problem**: given a knapsack that has a weight capacity $C$ and a set of $n$ items with weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$, we want to maximize the number of items in the knapsack (assuming that we have only one copy of each item). To model this problem as an integer program, we consider the variables $x_1, \ldots, x_n \in \{0, 1\}$, where $x_i = 0$ if and only if the $i$-th object is not in the knapsack. These variables are subject to the constraints given by the problem, meaning that:

$$\begin{gathered} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} w_i x_i \leq C \\ 0 \leq x \leq 1 \\ x \in \mathbb{Z}^n \end{gathered}$$

Since the 0/1 Knapsack problem is **NP-Complete** and it can be reduced to Integer programming, this shows that the latter is also NP-Complete. Thus, finding an efficient way to solve integer programs would also imply finding a way to solve all NP-Complete problems.

We will discuss how the techniques valid for a general linear programs can also be used for integer programs with a slight, but very impactful, difference.

First of all, we have to change our geometric understanding of the feasible region: we have seen how for a general linear program the feasible region is an $n$-dimensional polyhedron. For integer programs, however, this isn't true since we have some "huge gaps" between each integral feasible solution.

In fact, the set of feasible regions of an integer linear program is given by the **subset of integer solutions** inside the polyhedron $P$ defined by the **relaxed integer program**, i.e. a linear program identical to the original integer program without the constraint $x \in \mathbb{Z}^n$.

**Definition 38: Set of feasible solutions of an integer program**

Given an integer program

$$\begin{gathered} \max\ c^T x \\ Ax \leq b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{gathered}$$

Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$, that being the polyhedron of its relaxed version. The **set of feasible solutions of the integer program**, namely $P_I$, is made of all the integer solutions in $P$, meaning that $P_I = \mathbb{Z}^n \cap P$.

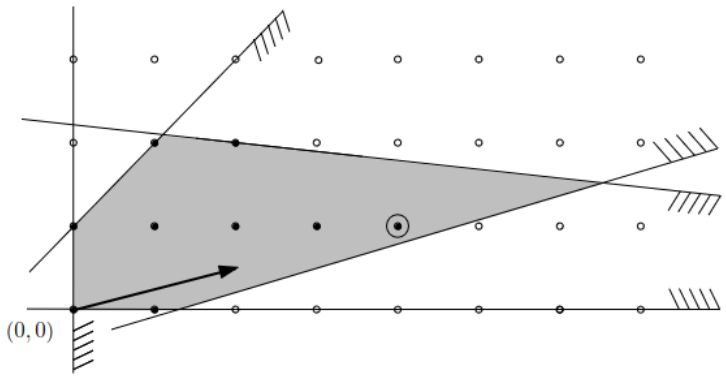



Figure 6.1: The gray area defines the polyhedron $P$, while the set of black dots defines $P_I$. The circled black dot is the optimal solution of the integer program.

**Proposition 18**

Given an integer program, let $P$ be the relaxed polyhedron and let $P_I$ be the feasible region of the integer program. Then, $P_I$ is the **integral convex hull** (and thus the set of integral convex combinations by Proposition 9) **of all the integer solutions** $u_1, \ldots, u_k \in P$, meaning that:

$$P_I = \left\{ \alpha_1 u_1 + \ldots + \alpha_k u_k \,\middle|\, \sum_{i=1}^{n} \alpha_i = 1, \alpha_1, \ldots, \alpha_k \in \mathbb{Z} \right\}$$

In other words, any integral feasible solution can be described as an integral convex combination of all the integral feasible solutions.

*(proof omitted)*

**Proposition 19: Vertices of a convex hull**

Let $C$ be a convex hull defined by $u_1, \ldots, u_k$. If $v$ is a vertex of $C$ then $v = u_i$ for some $i$.

*Proof.*

If $v$ is a vertex of $C$, by definition there is an affine hyperplane $H = \{x \in R^n \mid c^T x = c^T v\}$ such that $\{v\} = C \cap H$, meaning that $\forall x \in C - \{v\} c^T x < c^T v$. Since $v \in C$, there are $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$ such that $v = \sum_{i=1}^{k} \alpha_i u_i$ and $\sum_{i=1}^{k} \alpha_i = 1$.

By way of contradiction, suppose that $v \neq u_1, \ldots, u_k$. Then, this means that there must be an index $j$ such that $0 < \alpha_j < 1$, implying that:

$$\sum_{\substack{i=1 \\ i \neq j}}^{k} \alpha_i = 1 - \alpha_j \implies \sum_{\substack{i=1 \\ i \neq j}}^{k} \frac{\alpha_i}{1 - \alpha_j} = 1$$

Let $u' = 0 \cdot u_j + \sum_{\substack{i=1 \\ i \neq j}}^{k} \frac{\alpha_i}{1-\alpha_j} u_i$. Since $\sum_{\substack{i=1 \\ i \neq j}}^{k} \frac{\alpha_i}{1-\alpha_j} = 1$, we get that $u' \in C$. Hence, this would imply that:

$$\begin{aligned} v &= \sum_{i=1}^{k} \alpha_i u_i \\ &= \alpha_j u_j + \sum_{\substack{i=1 \\ i \neq j}}^{k} \alpha_i u_i \\ &= \alpha_j u_j + (1 - \alpha_j) \sum_{\substack{i=1 \\ i \neq j}}^{k} \frac{\alpha_i}{1 - \alpha_j} u_i \\ &= \alpha_j u_j + (1 - \alpha_j) u' \end{aligned}$$

through which we get that:

$$\begin{aligned} c^T v &= c^T(\alpha_j u_j + (1-\alpha_j)u') \\ &= \alpha_j c^T u_j + (1-\alpha_j)c^T u' \end{aligned}$$

Since $u_j, u' \in C$, we get that:

$$\begin{aligned} c^T v &= \alpha_j c^T u_j + (1-\alpha_j)c^T u' \\ &< \alpha_j c^T v + (1-\alpha_j)c^T v \\ &< c^T v \end{aligned}$$

which is clearly a contradiction. Thus, it must hold that $v = u_i$ for some index $i$.

To give a geometrical reasoning behind these algebraic steps, since $v = \alpha_j u_j + (1-\alpha_j)u'$ we get that $v$ is inside the segment from $u_j$ and $u'$. However, this means that the hyperplane $H$ must lay on such segment or pierce through it, which in both cases gives us a contradiction to $v$ being a vertex of $C$ since the condition $\{v\} = C \cap H$ is violated in both cases since $u_j, u' \in C$.

□

## 6.2 Gamory's Cutting Planes

As discussed in the previous section, the relaxed polyhedron $P$ is usually larger than the set of integral feasible solutions $P_I$. To solve an integer program, one of the main techniques is **Gamory's Cutting Planes** method: by adding constraints to the relaxed linear program, we want to *cut off* some portions of the polyhedron $P$ in order to remove non-integer optimal solutions.

To give an intuition, consider the following integer program and its relaxed form:

$$\begin{aligned} \max\ & 5x_1 + 8x_2 \\ & x_1 + x_2 \leq 6 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_1, x_2 \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned} \qquad \begin{aligned} \max\ & 5x_1 + 8x_2 \\ & x_1 + x_2 \leq 6 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_1, x_2 \geq 0 \end{aligned}$$

First, we convert the relaxed program in standard form:

$$\begin{aligned} \max\ & 5x_1 + 8x_2 \\ & x_1 + x_2 + x_3 = 6 \\ & 5x_1 + 9x_2 + x_4 = 45 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Then, we procede by applying the Simplex method in order to find an optimal solution. We start with $x_3, x_4$ as basic variables:

$$\begin{array}{rcrrr} x_3 & = & 6 & -x_1 & -x_2 \\ x_4 & = & 45 & -5x_1 & -9x_2 \\ \hline z & = & & +5x_1 & +8x_2 \end{array}$$

rhen we pivot $x_3$ with $x_4$

$$\begin{array}{rcrrr} x_1 & = & 6 & -x_2 & -x_3 \\ x_4 & = & 15 & -4x_2 & -5x_3 \\ \hline z & = & 30 & +3x_2 & -5x_3 \end{array}$$

and finally we pivot $x_4$ with $x_2$ to get the optimal BFS $\left[\frac{9}{4} \quad \frac{15}{4} \quad 0 \quad 0\right]$:

$$\begin{array}{rcrrr} x_1 & = & \frac{9}{4} & -\frac{9}{4}x_3 & -\frac{1}{4}x_4 \\ x_2 & = & \frac{15}{4} & +\frac{5}{4}x_3 & -\frac{1}{4}x_4 \\ \hline z & = & \frac{41}{4} & -\frac{5}{4}x_3 & -\frac{3}{4}x_4 \end{array}$$

Since the optimal solution we found is not integral, it isn't a valid solution to the integer program. However, the last tableau of the Simplex method tells us that the equality

$$x_2 = \frac{15}{4} + \frac{5}{4}x_3 - \frac{1}{4}x_4$$

must be satisfied by all solutions in the feasible region, including the integral ones.

In particular, we notice that:

$$x_2 = \frac{15}{4} + \frac{5}{4}x_3 - \frac{1}{4}x_4 \implies x_2 - \frac{5}{4}x_3 + \frac{1}{4}x_4 = \frac{15}{4} \implies$$

$$(1+0)\,x_2 + \left(-2 + \frac{3}{4}\right)x_3 + \left(0 + \frac{1}{4}\right)x_4 = 3 + \frac{3}{4}$$

If we assume that $x_1, x_2, x_3, x_4 \in \mathbb{Z}$ then we get that the following equality must be true for some $k \in Z$ such that $k \geq 0$:

$$0 \cdot x_2 + \frac{3}{4}x_3 + \frac{1}{4}x_4 = k + \frac{3}{4}$$

Thus, since $k \geq 0$, we also get the following inequality:

$$0 \cdot x_2 + \frac{3}{4}x_3 + \frac{1}{4}x_4 \geq \frac{3}{4} \implies 3x_3 + x_4 \geq 3$$

Thus, we get that any solution $\overline{x}$ that doesn't satisfy the last inequality cannot be an integral solution, meaning that by adding this constraint to the linear program, we can **cut off** a non-integral portion of feasible region.

In fact, the previous optimal BFS doesn't satisfy this constraint, meaning that we can run again the Simplex method on the following new linear program to try and get an integral solution:

$$\begin{aligned} \max\ & 5x_1 + 8x_2 \\ & x_1 + x_2 + x_3 = 6 \\ & 5x_1 + 9x_2 + x_4 = 45 \\ & 3x_3 + x_4 + x_5 = 3 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

*Note:* the slack variable for the added constraint has already been added

By running this process again and again, we will eventually get an **integral optimal solution**. In general, if the found optimal solution is not integral, in the final tableau we will always have at least one equality in the form:

$$x_b + (\alpha_1 + f_1)x_1 + \ldots + (\alpha_n + f_n)x_k = \alpha_0 + f_0$$

where $x_b$ is a basic variable, $x_1, \ldots, x_k$ are non-basic variables and for all $i$ it holds that $\alpha_i \in Z$ and $f_i \in Q$ where $f_i < 1$. Through this inequality, the **Gamory cut** is given by:

$$f_1 x_1 + \ldots + f_n x_k \geq f_0$$

Moreover, for each final tableau, we can apply a Gamory cut for all non-integral basic variables, meaning that we can apply **more than one Gamory cut simultaneously**. However, one Gamory cut will always be **sufficient** to remove the optimal non-integral BFS found by the Simplex method, so adding too many cuts may be an overkill since each cut enlarges the linear program.

Now that we have an intuition behind what were are doing, we are ready to formalize these concepts.

**Theorem 17: Correctness of the Gamory cut**

Let $\mathcal{B}$ be a basis that certifies an optimal BFS $\overline{x}$ of the polyhedron $P = \{x \in R^n \mid Ax = b, x \geq 0\}$. If $\overline{x} \notin \mathbb{Z}^n$ then the Gamory cut removes $\overline{x}$ from $P$ while preserving all integer points inside $P$

*Proof.*

First, we reorder the columns of $A$ in order to have $A = \begin{bmatrix} A_{\mathcal{B}} \mid A_{\mathbb{N}} \end{bmatrix}$, where $\mathcal{N} = \{1, \ldots, n\} - \mathcal{B}$. Likewise, we reorder the scalars of $\overline{x}$ in order to have $\overline{x}^T = \begin{bmatrix} \overline{x}_{\mathcal{B}} \mid 0 & \cdots & 0 \end{bmatrix}$. These two reorderings imply that $\overline{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$.

Then, we have that:

$$A\overline{x} = \begin{bmatrix} A_{\mathcal{B}} \mid A_{\mathcal{N}} \end{bmatrix} \overline{x} = b \implies A_{\mathcal{B}}^{-1} A\overline{x} = \begin{bmatrix} I_m \mid A_{\mathcal{B}}^{-1} A_{\mathcal{N}} \end{bmatrix} \overline{x} = A_{\mathcal{B}}^{-1} b$$

$$\implies \overline{x}_{\mathcal{B}} + A_{\mathcal{B}}^{-1} A_{\mathcal{N}} \overline{x}_{\mathcal{N}} = A_{\mathcal{B}}^{-1} b$$

Let $\overline{A}_{\mathbb{N}} = A_{\mathcal{B}}^{-1} A_{\mathcal{N}}$, meaning that $\overline{x}_{\mathcal{B}} + \overline{A}_{\mathbb{N}} \overline{x}_{\mathcal{N}} = A_{\mathcal{B}}^{-1} b$. We denote with $\overline{a}_{i,j}$ the entries of $\overline{A}$.

Let also $\overline{b} = A_{\mathcal{B}}^{-1} b$. Since $\overline{x} \notin \mathbb{Z}^n$, there is an index $i \in \mathcal{B}$ such that $\overline{x}_i = \overline{b_i} \notin \mathbb{Z}$. Furthermore, since $\mathcal{N}$ is the set of non-basic variables, we have that:

$$\overline{x}_i = \overline{b_i} \implies \overline{x}_i + \sum_{j \in \mathcal{N}} \overline{a}_{i,j} \, \overline{x}_j = \overline{b_i}$$

allowing us to derive the following Gamory cut:

$$\overline{x}_i + \sum_{j \in \mathcal{N}} (\lfloor \overline{a}_{i,j} \rfloor + (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor)) x_j = \lfloor \overline{b}_i \rfloor + (\overline{b_i} - \lfloor \overline{b}_i \rfloor)$$

$$\implies \sum_{j \in \mathcal{N}} (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor) x_j \geq \overline{b_i} - \lfloor \overline{b}_i \rfloor$$

*Note*: for every number $n \in \mathbb{R}$, it holds that $0 \leq n - \lfloor n \rfloor <$.

**Claim 1**: the optimal BFS $\overline{x}$ doesn't satisfy the Gamory cut

$$\sum_{j \in \mathcal{N}} (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor) x_j \geq \overline{b_i} - \lfloor \overline{b}_i \rfloor$$

*Proof of the claim.*

Since $\mathcal{N}$ is the set of non-basic variables and $\overline{b}_i \notin \mathbb{Z}$, it's easy to see that:

$$\sum_{j \in \mathcal{N}} (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor) \overline{x}_j = 0 < \overline{b_i} - \lfloor \overline{b}_i \rfloor$$

□

**Claim 2**: Every integer solution $y \in P_I$ satisfies the Gamory cut

$$\sum_{j \in \mathcal{N}} (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor) x_j \geq \overline{b_i} - \lfloor \overline{b}_i \rfloor$$

*Proof of the claim.*

Since it is a rearrangement of the row $A^i$, the equality must be valid for all points $x \in P$:

$$x_i = \overline{b_i} \implies x + \sum_{j \in \mathcal{N}} \overline{a}_{i,j} x_i = \overline{b_i}$$

Thus it also holds for every $y \in P_I \subseteq P$, implying that:

$$y_i + \sum_{j \in \mathcal{N}} (\lfloor \overline{a}_{i,j} \rfloor + (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor)) y_j = \lfloor \overline{b}_i \rfloor + (\overline{b_i} - \lfloor \overline{b}_i \rfloor) \implies$$

$$y_i + \sum_{j \in \mathcal{N}} \lfloor \overline{a}_{i,j} \rfloor \, y_j + \sum_{j \in \mathcal{N}} (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor) y_j = \lfloor \overline{b}_i \rfloor + (\overline{b_i} - \lfloor \overline{b}_i \rfloor)$$

Since $y \in \mathbb{Z}^n$ we know that $y_1, \ldots, y_n \in \mathbb{Z}$. Moreover, the constraints of $P$ imply that $y_1, \ldots, y_n > 0$. Thus, we get that the following must hold:

$$\sum_{j \in \mathcal{N}} (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor) y_j = k + (\overline{b_i} - \lfloor \overline{b}_i \rfloor)$$

for some $k \in \mathbb{Z}$ such that $k > 0$, concluding that:

$$\sum_{j \in \mathcal{N}} (\overline{a}_{i,j} - \lfloor \overline{a}_{i,j} \rfloor) y_j \geq \overline{b_i} - \lfloor \overline{b}_i \rfloor$$

□

By the two claims, we conclude the proof of the theorem. □

**Definition 39: Gamory's Cutting Planes method**

If the optimal BFS obtained through the Simplex tableau is non-integral, apply a Gamory cut on the basic variable with the **lowest possible index**

**Theorem 18**

The Simplex method with Bland's rule and Gamory's Cutting Planes method prevents tableau cycles, meaning that it **always terminates**.

*(proof omitted)*

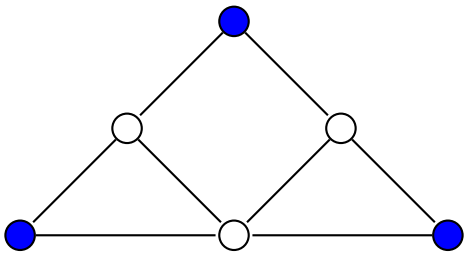
## 6.3 The Maximum-weight Independent Set problem

As discussed in the previous sections, Integer programming is NP-Complete, meaning that all NP problems can be reduced to an integer program. We will show how common NP-Complete problems can be **reduced to integer programs** and how we can find ways to apply an **optimal cut** on such problems, making it easier to find a solution for them (even thought it remains a non-efficient process).

**Definition 40: Independent set**

Let $G$ be an undirected graph. A subset $X \subseteq V(G)$ is said to be an **independent set** (or a *stable set*) if all the nodes inside it are non-adjacent to each other, meaning that:

$$\forall x, y \in X \quad (x, y), (y, x) \notin E(G)$$



*The blue nodes are the biggest possible independent set of the graph*

**Problem 5: The Maximum-weight Independent Set problem**

Given a vertex-weighted undirected graph $G$, meaning that the weights are on the vertices and not on the edges, find an independent set $X \subseteq V(G)$ such that the sum of the weights of the vertices in $X$ is maximal, meaning that:

$$X = \underset{\substack{H \text{ independent} \\ \text{set of } G}}{\arg\max} \left( \sum_{u \in H} w(u) \right)$$

This problem can be easily reduced to a 0-1 integer program, that being an integer program where every variable is either a 0 or a 1:

- For every node $v \in V(G)$, we define the variabile $x_v$ and the constant $w_v$ such that $w_v = w(v)$.
- For every edge $(i, j) \in E(G)$, we add the constraint $x_i + x_j \leq 1$. This constraint enforces that at most one between $x_i$ and $x_j$ can be set to 1.

We get that:

$$\max \sum_{i=1}^{n} w_i x_i$$

$$\begin{aligned} x_i + x_j &\leq 1 \quad \forall (i,j) \in E(G) \\ x &\in \{0,1\}^n \end{aligned}$$

*Note*: the constraint $x \in \{0,1\}$ is just a short-hand for the two constraints $0 \leq x \leq 1$ and $x \in \mathbb{Z}^n$, where $n = |V(G)|$.

By means of construction of this 0-1 integer program, **any feasible solution $\overline{x}$ corresponds to an independent set** $X$ where $\overline{x_v} = 0 \iff v \notin X$. Due to this nature, we also use the term **indicator vector** to define feasible solutions. Any **optimal indicator vector** of this program corresponds to a maximum-weight independent set.

However, the applied constraints are still to *weak* since they exclude very few points from $\mathbb{R}^n$. As shown in the previous section, we can solve this program by repeatedly applying Gamory's Cutting Planes method. Nevertheless, before applying the procedure, we can manually find some constraints to our program find which we are sure to be true.

For example, let $C \subseteq V(G)$ be a **clique** of $G$, i.e. a subset of vertices where $\forall x, y \in C \quad (x,y)(y,x) \in (G)$. By the very own definition of clique, it's easy to see that the concepts of clique and independent set are *complementary* to each other. Thus, there must be some constraint that we can apply on each clique of $G$ in order to remove them from the feasible region.

Suppose our graph has a 3-clique made of the vertices $i, j, k \in V(G)$. Since they form a clique, we know that $(i,j), (j,k), (k,i) \in E(G)$. Thus, in our linear program we must have the three constraints $x_i + x_j \leq 1$, $x_j + x_k \leq 1$ and $x_k + x_i \leq 1$. These three constraints can be summed to form the following linear combination:

$$2x_i + 2x_j + 2x_k \leq 3 \implies x_1 + x_j + x_k \leq \frac{3}{2}$$

Since it's a linear combination of the three constraints, we know that this inequality is already implied by the very same three constraints. However, since $i, j, k$ form a clique, we know that if we add *any* node between the three of them to our independent set then we cannot add any other of them. This condition can be easily described by adding the constraint $x_i + x_j + x_k \leq 1$ which can't be obtained as a linear combination of the previous three constraint, reducing the set of feasible region. In general, for each clique $C$ we can add the constraint $\sum_{i \in V(C)} x_i \leq 1$ to pick maximum one member of the clique.

In a similar way, we can enforce some constraints on the cycles of $G$: if $C'$ is an **odd cycle** of $G$ of length $2k+1$ then we can select at most $k$ nodes inside this cycle simply by skipping one node of the cycle for each selected node.

After these observations, we can restate our 0-1 integer program as:

$$\max \sum_{i=1}^{n} w_i x_i$$

$$\begin{aligned} x_i + x_j &\leq 1 \quad \forall (i,j) \in E(G) \\ \sum_{i \in V(C)} x_i &\leq 1 \quad \forall \text{ cliques } C \text{ in } G \\ \sum_{i \in V(C')} x_i &\leq k \quad \forall \text{ cycles } C' \text{ of length } 2k+1 \text{ in } G \\ x &\in \{0,1\} \end{aligned}$$

Even thought, these constraints are good enough to remove a lot of solutions from the feasible region, finding a clique of arbitrary length in a graph is also an NP-Complete problem, so finding them would be as hard as solving the linear program. However, cliques with a fixed amount of vertices and all odd cycles can be found in polynomial time, meaning that we can still preserve some of these constraints.

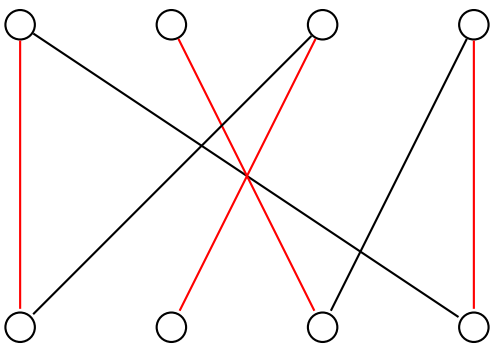
## 6.4 The Maximum-weight Perfect Matching problem

**Definition 41: Matching**

Let $G$ be an undirected graph. A subset $M \subseteq E(G)$ is said to be a **matching** if all the edges inside it have no common end-points, meaning that:

$$\forall (u,v),(x,y) \in M \quad u,v \neq x,y$$

A matching $M$ is said to be **perfect** if $V(M) = V(G)$, meaning that it covers all the vertices. Equivalently, a matching $M$ is perfect if $|M| = \frac{|V(G)|}{2}$.



*The red edges are a perfect matching of the graph*

**Problem 6: The Maximum-weight Perfect Matching problem**

Given an edge-weighted undirected graph $G$, find a perfect matching $M \subseteq E(G)$ such that the sum of the weights of the edges in $M$ is maximal, meaning that:

$$M = \underset{\substack{H \text{ perfect} \\ \text{matching of } G}}{\arg\max} \left( \sum_{e \in H} w(e) \right)$$

Just like the *Maximum-weight Independent Set problem*, we can easily formulate the Maximum-weight Perfect Matching problem as a 0-1 integer program:

- For every edge $e \in E(G)$, we define the variabile $x_e$ and the constant $w_e$ such that $w_e = w(e)$.
- For every vertex $v \in V(G)$ we add the constraint $\sum_{e \in \delta(v)} x_e = 1$, where $\delta(v)$ is the set of edges with $v$ as one of the end-points, meaning that $\delta(v) = \{(x, y) \in E(G) \mid x = v \vee y = v\}$.

We get that:

$$\max \sum_{e \in E(G)} w_e x_e$$

$$\sum_{e \in \delta(v)} x_e = 1 \qquad \forall v \in V(G)$$
$$x \in \{0, 1\}^m$$

where $x_e = 0$ if and only if $e$ is not in the matching.

**Note**: $m$ is the number of edges of $G$, i.e. $m = |E(G)|$.

We notice that the constraint added for each vertex already implies that for each edge $e$ it holds that $x_e \leq 1$.

Thus, we can actually remove this constraint from our program.

$$\max \sum_{e \in E(G)} w_e x_e$$

$$\sum_{e \in \delta(v)} x_e = 1 \qquad \forall v \in V(G)$$
$$x \geq 0$$
$$x \in \mathbb{Z}^m$$

Consider now the **LP relaxation** of this integer program, that being the program identical to the original but without the integral constraint.

$$\max \sum_{e \in E(G)} w_e x_e$$

$$\sum_{e \in \delta(v)} x_e = 1 \qquad \forall v \in V(G)$$
$$x \geq 0$$
$$x \in \mathbb{R}^m$$

First, we notice that since we said that $x_e = 0$ if and only if $e$ is not in the matching, this implies that if $0 < x_e < 1$ the edge $e$ will be considered as an element of the subset, giving us the following observation.

**Observation 18**

Given a graph $G$, if $x$ is a **non-integral feasible solution** of the above linear program then the subset $M_x \subseteq E(G)$ represented by $x$ is **not a matching of** $G$.

*Proof.*

Let $v \in V(G)$ and suppose that $x \in P - \mathbb{Z}^m$. Due to the constraints of the linear program, we have that $\sum_{e \in \delta(v)} x_e = 1$. Thus, since $x$ is non-integral, there must at least two edges adjacent to $v$ whose indicator variable is different from 0 in order to reach sum 1.

Thus, at least two edges adjacent to $v$ are in the subset $M \subseteq E(G)$ represented by $x$. However, by definition of matching, this also implies that $M$ is not a matching. □

**Lemma 11**

Given a graph $G$, any feasible solution $x$ of the linear program above is **integral if and only if** $M_x$ **is a perfect matching** of the graph $G$.

*Proof.*

If $x$ is non-integral by the previous observation we know that $M_x$ isn't a matching. Viceversa, if $x$ is integral then for each node $v \in V(G)$ only one edge adjacent to $v$ will be in $M_x$, concluding that $M_x$ is a perfect matching of $G$. □

**Definition 42: Convex hull of the perfect matchings**

Given a graph $G$, we define $P_M(G)$ as the **convex hull** of all the indicator vertices $v_1, \ldots, v_k$ representing the **perfect matchings of** $G$.

**Corollary 8**

Let $G$ be a weighted graph and let $P$ be the polyhedron given by the LP relaxation shown above. Then, it holds that $P_M(G) \subseteq P$.

*Proof.*

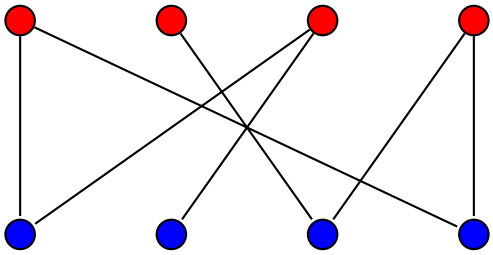
All the vertices $\overline{x_1}, \ldots, \overline{x_k}$ of $P_M(G)$ describe a perfect matching of $G$. Thus, by Lemma 11 we know that they must be integral feasible solutions of the LP relaxation, meaning that $\overline{x_1}, \ldots, \overline{x_k} \in P$. Then, since $P_M(G)$ is by definition the convex combination of $\overline{x_1}, \ldots, \overline{x_k}$, for any point $x \in P_M(G)$ it must also hold that $x \in P$. □

We will show that, if some conditions are met, the LP relaxation of this program has **only integral vertices**, meaning that all of them correspond to a maximum weight perfect matching.

### 6.4.1 The Max PM problem for bipartite graphs

**Definition 43: Bipartite graph**

An undirected graph $G$ is said to be a **bipartite graph** if it can be partitioned into two independent sets $X, Y$. Equivalently, $G$ is bipartite if there are two sets $X, Y$ such that for every edge $(x, y) \in E(G)$ it holds that $x \in X$ and $y \in Y$ (or viceversa).



*The red and blue nodes partition the graph into two independent sets*

**Proposition 20**

A graph $G$ is **bipartite** if and only if it has **no odd cycles**.

*(proof omitted)*

**Proposition 21**

If $G$ is an undirected graph where all nodes have degree at least 2, then $G$ is **cyclic**.

*(proof omitted)*

**Theorem 19: Max-weight PM in bipartite graphs**

Let $G$ be a bipartite weighted graph and let $P$ be the polyhedron defined as:

$$P = \left\{ x \in \mathbb{R}^m \,\middle|\, x \geq 0, \sum_{e \in \delta(v)} x_e = 1 \;\; \forall v \in V(G) \right\}$$

Every vertex $\overline{x} \in P$ is **integral**, meaning that $\overline{x} \in \mathbb{Z}^m$.

*Proof.*

Let $\overline{x} \in P$ be a vertex of $P$. By way of contradiction, suppose that $\overline{x} \notin \mathbb{Z}^m$ and let $\mathcal{U} = \{e \in E(G) \mid 0 < \overline{x_e} < 1\}$. Since we assumed that $\overline{x} \notin \mathbb{Z}^m$, it must hold that $\mathcal{U} \neq \varnothing$. Let $G_{\mathcal{U}}$ be the subgraph of $G$ made of the edges in $\mathcal{U}$.

**Claim**: Every vertex in $V(G_{\mathcal{U}})$ has degree at least 2

*Proof of the claim.*

For each vertex $v \in V(G_{\mathcal{U}})$, it holds that $\mathcal{U} \cap \delta(v) \neq \varnothing$. By way of contradiction, suppose that $\exists e \in \delta(v)$ such that $\overline{x_e} = 1$. Then, we get that:

$$\sum_{f \in \delta(v)} \overline{x_f} = 1 \implies \sum_{f \in \delta(v) - \{e\}} \overline{x_f} = 1 - \overline{x_e} = 0 \implies \forall f \in \delta(v) - \{e\} \;\; \overline{x_f} = 0$$

Since $x_e = 1$ and $\forall f \in \delta(v) - \{e\} \;\; \overline{x_f} = 0$, we derive that $\mathcal{U} \cap \delta(v) = \varnothing$, which is absurd. Thus, $\forall e \in \delta(v)$ it must hold that $\overline{x_e} < 1$, implying that in order for $\overline{x}$ to respect the constraints there must be one or more edges $e_1, \ldots, e_k \in (\mathcal{U} \cap \delta(v)) - \{e\}$ such that $x_e + \sum_{i=1}^{k} \overline{x_{e_i}} = 1$, meaning that $|\delta(v)| \geq 2$. □

Since $G_{\mathcal{U}}$ is undirected and all its nodes have degree at least 2, by Proposition 21 there must be a cycle $C \subseteq G_{\mathcal{U}} \subseteq G$. Moreover, since $G$ is bipartite, by Proposition 20 this cycle $C$ must be an even cycle of length $2k$ for some $k \in \mathbb{N}$.

Let $e_1, \ldots, e_{2k}$ be the edges of $C$. Let $\varepsilon = \min_{i=1}^{k}(\min(x_{e_i}, 1 - x_{e_i}))$, meaning that $\varepsilon$ is the minimal possible value such that there is a variabile between $x_{e_1}, \ldots, x_{e_{2_k}}$ for which it holds either that $x_{e_i} + \varepsilon = 1$ or that $x_{e_i} - \varepsilon = 0$.

Let $y$ and $y'$ be two indicator vectors defined as:

$$y_e = \begin{cases} \overline{x_e} - \varepsilon & \text{if } e = e_i \text{ for some } 1 \leq i \leq 2k \text{ and } i \text{ is even} \\ \overline{x_e} + \varepsilon & \text{if } e = e_i \text{ for some } 1 \leq i \leq 2k \text{ and } i \text{ is odd} \\ \overline{x_e} & \text{otherwise} \end{cases}$$

$$y'_e = \begin{cases} \overline{x_e} + \varepsilon & \text{if } e = e_i \text{ for some } 1 \leq i \leq 2k \text{ and } i \text{ is even} \\ \overline{x_e} - \varepsilon & \text{if } e = e_i \text{ for some } 1 \leq i \leq 2k \text{ and } i \text{ is odd} \\ \overline{x_e} & \text{otherwise} \end{cases}$$

In other words, $y$ is the indicator vector obtained by adding $\varepsilon$ to all even edges of the cycle $C$ and subtracting $\varepsilon$ to all the odd edges, while $y'$ is the indicator vector obtained by doing the inverse.

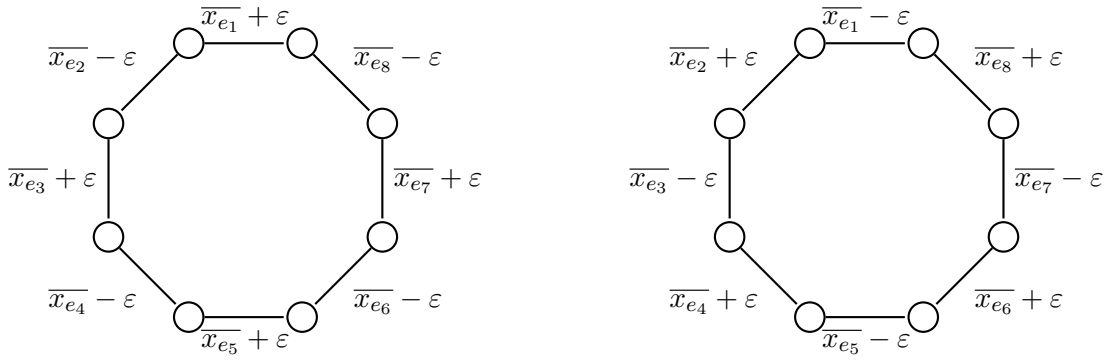



Figure 6.2: The vector $y$ is shown on the left, $y'$ on the right

Since the cycle has an even amount of edges, the value $\varepsilon$ gets added and subtracted an equal amount of times, so we get that for each node $v \in V(C)$ the constraint $\sum_{e \in \delta(v)} x_e = 1$ is still satisfied. Moreover, by choice of $\varepsilon$, we are sure that $0 \leq y, y' \leq 1$ holds, so every constraint is still satisfied by both $y$ an $y'$, meaning that they are indeed feasible solutions and thus that $y, y' \in P$.

In particular, the way $y$ and $y'$ are defined also implies that:

$$\sum_{e \in E(G)} \overline{x_e} = \sum_{e \in E(G)} x_e = \sum_{e \in E(G)} y'_e$$

meaning that $\overline{x} = \frac{1}{2}y + \frac{1}{2}y'$. Now, since $\overline{x}$ is a vertex of $P$, by definition there is an affine hyperplane $H$ such that $H = \{x \in \mathbb{R}^m \mid c^T x = c^T \overline{x}\}$, $\{x\} = P \cap H$ and $\forall z \in P - \{x\}\ \ c^T z < c^T \overline{x}$.

By linearity, we notice that:

$$\overline{x} = \frac{1}{2}y + \frac{1}{2}y' \implies c^T \overline{x} = \frac{1}{2}c^T y + \frac{1}{2}c^T y'$$

Thus, it must hold either that $c^T \overline{x} = c^T y + c^T y'$, which is a contradiction to the fact that $\{x\} = P \cap H$, or that $c^T y > c^T \overline{x} \vee c^T y' > c^T \overline{x}$, which is a contradiction to the fact that $\forall z \in P - \{x\}\ \ c^T z < c^T \overline{x}$.

In other words, $\overline{x} = \frac{1}{2}y + \frac{1}{2}y'$ implies that $\overline{x}$ is inside the segment from $y$ to $y'$, which means that this segment must either lie of the hyperplane $H$ or pierce through it, which in both cases is a contradiction to the fact that $\overline{x}$ is a vertex of $P$. Thus, the only possibility is that $\overline{x} \in \mathbb{Z}^m$ must hold.

□

**Corollary 9**

Let $G$ be a weighted graph and let $P$ be the polyhedron given by the LP relaxation shown above. If $G$ is **bipartite** then $P = P_M(G)$.

**Note**: this result is not equal to saying that $P = P_I$.

*Proof.* By Corollary 8, we know that $P_M(G) \subseteq P$. Moreover, by Theorem 19 we know that all the vertices $\overline{x_1}, \dots, \overline{x_k}$ of $P$ are integral. Thus, by Lemma 11 we know that they must describe perfect matchings of $G$, meaning that $\overline{x_1}, \dots, \overline{x_k} \in P_(G)$. Then, since $P$ is the convex combination of $\overline{x_1}, \dots, \overline{x_k}$, for any point $x \in P$ it must also hold that $x \in P_M(G)$. □

This result shows that if the graph is **bipartite** than we find an optimal integral solution simply by solving the LP relaxation, making this integer program theoretically **solvable in polynomial time** (in case of bipartite graphs). In fact, many other algorithms can solve this problem efficiently without the need of integer programming at all (this problem isn't NP-Complete, so this doesn't get us to anything).

### 6.4.2 The Max PM problem for non-bipartite graphs

What can we do if the graph isn't bipartite? With only the previous constraints, some non-bipartite graphs can be formulated in a way such that there can be non-integral vertices in the polyhedron, meaning that the previous theorem doesn't hold. Nevertheless, we can add some more constraints to the problem in order to get an **even stronger result** than the one previously shown.

First, we define an extension of the function $\delta$: if $W \subseteq V(G)$ then $\delta(W) = \{(u, v) \in E(G) \mid u \in W, v \in V(G) - W\}$, i.e. the set of edges exiting $W$. Now, we restate the 0-1 integer program with the following new constraint:

$$\max \sum_{e \in E(G)} w_e x_e$$

$$\begin{aligned} \sum_{e \in \delta(v)} x_e &= 1 \quad \forall v \in V(G) \\ \sum_{e \in \delta(\mathcal{U})} x_e &\geq 1 \quad \forall \mathcal{U} \subseteq V(G) \text{ s.t } |\mathcal{U}| = 2k+1 \\ x &\in \{0,1\}^m \end{aligned}$$

As before, the LP relaxation can be obtained by substituting the constraint $x \in \{0,1\}^m$ with the constraints $x \geq 0$ and $x \in \mathbb{R}^m$. Moreover, Lemma 11 holds even in this LP relaxation.

**Theorem 20: Edmond's theorem for general Max-weight PM**

Let $G$ be a weighted graph and let $P$ be the polyhedron given by the LP relaxation shown above. Then, it holds that $P = P_M(G)$.

*Proof.*

First, we notice that for each graph that has an odd number of nodes there cannot be a perfect matching and the polyhedron of the linear program has no feasible solution, meaning that $P_M(G) = \varnothing = P_M(G)$.

Hence, for the proof we consider only graphs with an odd number of nodes. By Corollary 8, we already know that $P_M(G') \subseteq P$ holds for all such graphs.

By way of contradiction, suppose that there are some counterexamples such that $P_M(G') \neq P$ holds. Let $G$ be the counterexample with the least amount of nodes such that $P_M(G) \neq P$. Since $P$ is bounded and $P_M(G)$ is a proper subset of $P$, there must be at least a vertex $\overline{x}$ of $P$ that is outside of $P_M(G)$.

Since by definition $P_M(G)$ is the convex hull of all the perfect matchings of $G$ and $\overline{x}$ isn't a vertex of $P_M(G)$, we know that it also isn't a perfect matching. Thus, by Lemma 11 we know that $\overline{x} \notin \mathbb{Z}^m$ and let $\mathcal{U} \subseteq V(G)$ be a subset such that $|\mathcal{U}| = 2k + 1$ for some $k \in \mathbb{N}$

Case 1. Suppose that $\sum_{e \in \delta(\mathcal{U})} x_e > 1$.

Let $\mathcal{F} = \{e \in E(G) \mid 0 < \overline{x_e} < 1\}$. Since we assumed that $\overline{x} \notin \mathbb{Z}^m$, it must hold that $\mathcal{F} \neq \varnothing$. Let $G_{\mathcal{F}}$ be the subgraph of $G$ made of the edges in $\mathcal{F}$.

**Claim 1**: Every node in $V(G_{\mathcal{F}})$ has degree at least 2

*Proof of the claim.* Under the assumption that $\sum_{e \in \delta(\mathcal{U})} x_e > 1$ holds, the proof of this claim is equivalent to the one seen for the bipartite case of this theorem (see Theorem 19). □

**Claim 2**: $G_{\mathcal{F}}$ has no even length cycle

*Proof of the claim.*

By way of contradiction, suppose there is a cycle $C$ of even length in $G_{\mathcal{F}}$ and let $e_1, \ldots, e_{2k}$ be the edges of $C$. Let $\varepsilon = \min_{\substack{X \subseteq V(G), \\ |X| \text{ odd}}} \left( \sum_{e \in \delta(X)} \frac{\overline{x_e} - 1}{|V(G)|} \right)$.

Let $y$ and $y'$ be two indicator vectors defined as:

$$y_e = \begin{cases} \overline{x_e} - \varepsilon & \text{if } e = e_i \text{ for some } 1 \leq i \leq 2k \text{ and } i \text{ is even} \\ \overline{x_e} + \varepsilon & \text{if } e = e_i \text{ for some } 1 \leq i \leq 2k \text{ and } i \text{ is odd} \\ \overline{x_e} & \text{otherwise} \end{cases}$$

$$y'_e = \begin{cases} \overline{x_e} + \varepsilon & \text{if } e = e_i \text{ for some } 1 \leq i \leq 2k \text{ and } i \text{ is even} \\ \overline{x_e} - \varepsilon & \text{if } e = e_i \text{ for some } 1 \leq i \leq 2k \text{ and } i \text{ is odd} \\ \overline{x_e} & \text{otherwise} \end{cases}$$

Just like in the bipartite case of this theorem (see Theorem 19), by choice of $\varepsilon$ we get that $y, y' \in P$ and that $\overline{x} = \frac{1}{2}y + \frac{1}{2}y'$, which is absurd since $\overline{x}$ is assumed to be a vertex of $P$. Thus, such even cycle cannot exist.

□

Since $G_{\mathcal{F}}$ has no odd cycle and all of its nodes have degree at least 2, by Proposition 21 there must be an odd cycle $C \subseteq G_{\mathcal{F}} \subseteq G$ of length $2k+1$ for some $k \in \mathbb{N}$. Let $v_1, \ldots, v_{2k+1}$ be the nodes of such cycle $C$.

**Claim 3**: For all $i \neq j$ such that $i \neq 1, j \neq 2k+1$ and $|i-j| > 1$, the nodes $v_i$ and $v_j$ are no adjacent to each other

*Proof of the claim.*

By way of contradiction, suppose the claim is false for some indices $i, j$. Then, the edge $(v_i, v_j)$ would split the odd cycle $C$ into two subcycles $C_1, C_2$ where one of them is an even cycle and the other is odd. However, we said that $G_{\mathcal{F}}$ has no even length cycle, raising a contradiction.

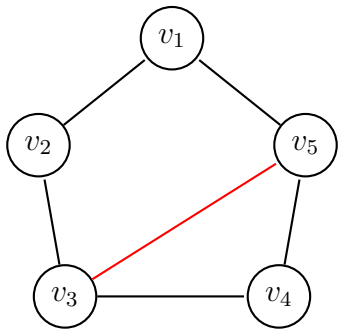



Figure 6.3: The edge $(v_3, v_5)$ would split the cycle in an odd subcycle and an even subcycle

□

**Claim 4**: There is at least a node $v_i$ in $C$ with a neighbor $u \in V(G_{\mathcal{F}})$ such that $u \notin V(C)$.

*Proof of the claim.*

Since $|C|$ is odd, by the constraints of the linear program we know that $\sum_{e \in \delta(V(C))} \overline{x_e} \geq 1$. However, we know that there is no edge $f$ in $C$ such that $x_f = 1$ and $v_i$ is one of the two ends of $f$ since this would imply that $\sum_{g \in \delta(v)} \overline{x_g} > 1$, violating the latter constraint. Thus, thanks to the previous claim, in order for the first constraint to

hold without breaking the second one, there must be an edge $e$ incident to $v_i$ that is not in $C$. □

Since we know that there is a node that touches the cycle $C$, let $w_1, \ldots, w_t$ be a walk such that:

- $w_1 = v_i$
- $w_1, \ldots, w_{t-1}$ are all distinct
- $w_2, \ldots, w_{t-1}$ are not in $C$
- $t$ is the maximal possible index

In other words, the walk $w_1, \ldots, w_t$ is the longest possible walk going outside of the cycle $C$. Since every node $G_{\mathcal{F}}$ has degree at least 2 and $t$ is chosen as maximal, it must hold that the walk either cycles back on itself or that it cycles back to $C$, i.e. it holds that $w_t = w_h$ where $1 \leq h \leq t-1$ or that $w_t = v_j$ where $i \leq j \leq 2k+1$. In both cases, we notice that the cycle formed must be of odd length since there can be no even length cycle in $G_{\mathcal{F}}$.

**Claim 5**: For all $\ell \neq j$ it holds that $w_\ell \neq v_j$

*Proof of the claim.*

By way of contradiction, suppose that there are some indices $i \neq j$ such that $w_\ell = v_j$, meaning that $v_i, w_2, \ldots, w_\ell, v_{j+1}, \ldots, v_{2k+1}, v_1, \ldots, v_i$ forms a cycle $C'$. Since $G_{\mathcal{F}}$ can only have odd length cycles.

Then, $v_1, \ldots, v_{i-1}, v_i, w_2, \ldots, w_\ell, v_{j+1}, \ldots, v_{2k}, v_1$ would make a cycle $C''$ of even length since it is the union of two odd length cycles where the common edges were excluded from both. However, such cycle can't exist in $G_{\mathcal{F}}$, raising a contradiction.

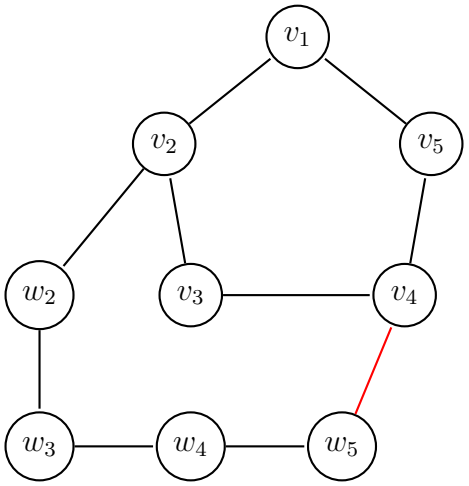



Figure 6.4: If $w_6 = v_4$ were to be true, $v_1, v_2, w_2, w_3, w_4, v_4, v_5, v_1$ would form an even cycle

□

Due to the last claim, it must hold that the walk cycles back on itself, i.e. it holds that $w_t = w_h$ where $1 \leq h \leq t-1$. Let $C'$ be the cycle $w_h, \ldots, w_{t-1}, w_h$. The current setup is given by two disjoint odd length cycles $C$ and $C'$ connected by the path $w_1, \ldots, w_h$.

Let $e_1, \ldots, e_{2k+1}$ be the edges of $C$, $f_1, \ldots, f_2 r + 1$ be the edges of $C'$ and $g_1, \ldots, g_s$ be the edges of the path $P = w_1, \ldots, w_h$

Moreover, let $\varepsilon = \min\limits_{\substack{X \subseteq V(G), \\ |X| \text{ odd}}} \left( \sum\limits_{e \in \delta(X)} \frac{\overline{x_e} - 1}{|V(G)|} \right)$.

Like in Theorem 19, we consider the following two indicator vectors $z$ and $z'$ obtained through $\overline{x}$ where the value $\varepsilon$ (or $\frac{\varepsilon}{2}$ when needed) gets added/subtracted on even index edges and subtracted/added on odd index edges of $C, C'$ and $P$. The "distribution" must equal, i.e. the total sum of the addition and subtractions equals 0 and such that $\overline{x} = \frac{1}{2}z + \frac{1}{2}z'$.

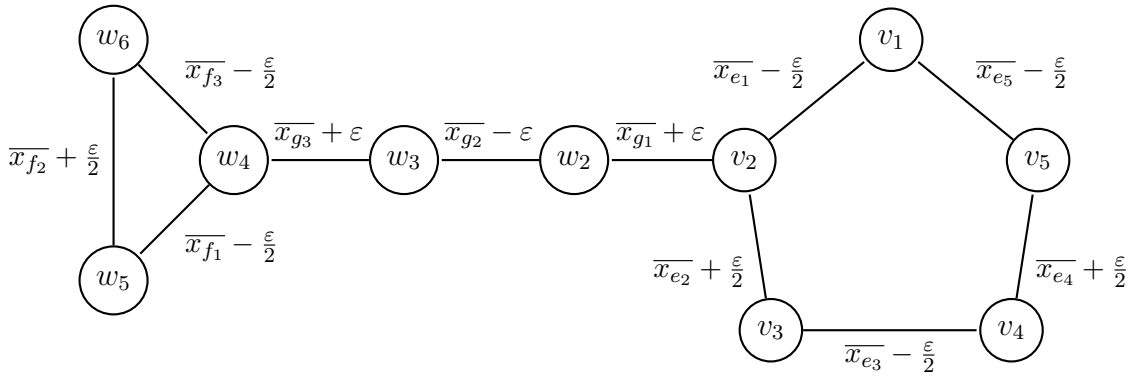



Figure 6.5: Example of equal distribution on the vector $z$. The vector $z'$ would have all increments and decrements reversed.

Since such distribution can always be found and it doesn't violate the constraints, we get that $z, z' \in P$. Moreover, the segment from $z$ and $z'$ always lies on the hyperplane of the vertex $\overline{x}$ or pierces through it, which is absurd. Finally, due to all possibilities raising a contradiction in this case, it must hold that $P_M(G) = P$ when $\sum\limits_{e \in \delta(\mathcal{U})} \overline{x_e} > 1$.

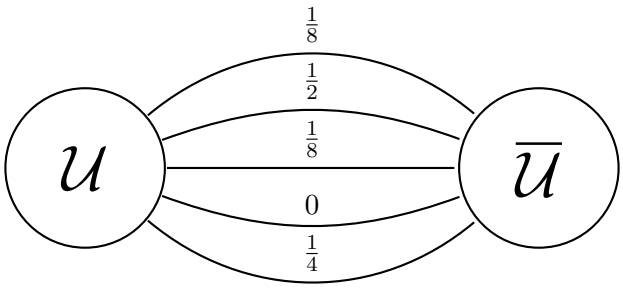
Case 2. Suppose that $\sum\limits_{e \in \delta(\mathcal{U})} \overline{x_e} = 1$.

Let $\overline{\mathcal{U}} = V(G) - \mathcal{U}$. Since we assumed that $|V(G)|$ is even and $|\mathcal{U}|$ is odd, it must also hold that $\overline{\mathcal{U}}$ is odd. Moreover, we notice that:

$$\sum_{e \in \delta(\overline{\mathcal{U}})} \overline{x_e} = \sum_{e \in \delta(\mathcal{U})} \overline{x_e} = 1$$

since $\delta(\overline{\mathcal{U}}) = \delta(\mathcal{U})$ by definition.

We notice that in order for the constraint of this case to be tight it must hold that each value $x_e$ involved is rational, meaning that $\delta(\mathcal{U}) \subseteq \mathbb{Q}$. Otherwise, if one of them were irrational, there would be a little gap which cannot be filled.




Let $G_1$ be the graph obtained by collapsing $\mathcal{U}$ in a single node $w_1$ and let $x'$ be the copy of $\overline{x}$ for $G_1$. Likewise, let $G_2$ be the graph obtained by collapsing $\overline{\mathcal{U}}$ in a single node $w_2$ and let $x''$ be the copy of $\overline{x}$ for $G_2$.

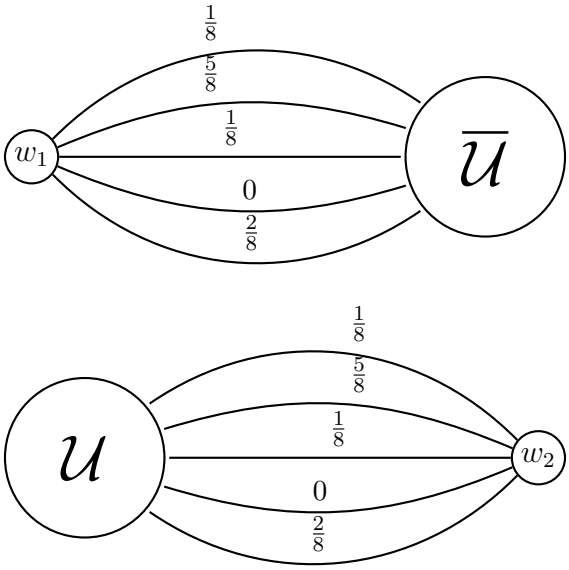



Figure 6.6: Shown above is the graph $G_1$, shown below is the graph $G_2$

Since $\sum_{e \in \delta(\mathcal{U})} x_e = 1$ holds by hypothesis, in $G_1$ it must hold that $\sum_{e \in \delta_{G_1}(w_1)} x'_e = 1$ due to the collapse of $\mathcal{U}$ into $w_1$. Consider now any subset $W \subseteq V(G_1)$ with an odd number of nodes. If $w_1 \notin W$ then $\delta_{G_1}(W) = \delta(W)$ since the collapse didn't influence it, implying that

$$\sum_{e \in \delta_{G_1}(W)} x'_e = \sum_{e \in \delta(W)} \overline{x_e} \geq 1$$

so the constraint is satisfied in $G_1$.

Otherwise, if $w_1 \in W$ then it holds that $\delta_{G_1}(W) = \delta((W - \{w_1\}) \cup \mathcal{U})$. In particular, we notice that $|(W - \{w_1\}) \cup \mathcal{U}|$ must also be odd. Hence, since all constraints are

valid for $G$, it must hold that:

$$\sum_{e \in \delta_{G_1}(W)} x'_e = \sum_{e \in \delta((W - \{w_1\}) \cup \mathcal{U})} \overline{x_e} \geq 1$$

so the constraint is satisfied in $G_1$ even in this case. Thus, we get that $x' \in P_1$, where $P_1$ is the polyhedron given by applying the LP relaxation on $G_1$. Due to $\overline{U}$ having the same properties of $\overline{U}$, proceeding in the same way we can show that $x'' \in P_2$.

Since we assumed that $G$ is the counterexample for the theorem with the least amount of nodes possible both $G_1$ and $G_2$ have fewer nodes than $G$, the theorem holds for $G_1$ and $G_2$, implying that $P_M(G_1) = P_1$ and $P_M(G_1) = P_1$.

Let $M_1, \ldots, M_k$ be the indicator vectors of the perfect matchings that define $P_M(G_1)$ and $N_1, \ldots, N_h$ the ones that define $P_M(G_2)$. Since $x' \in P_M(G_1)$ and $x'' \in P_M(G_2)$, let $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_h \in \mathbb{Q}$ such that $x' = \alpha_1 M_1 + \ldots \alpha_k M_k$ and $x'' = \beta_1 N_1 + \ldots \beta_h N_k$.

We notice that for each $i, j$ it hold that $M_i \cup N_j$ is a perfect matching of $G$ since $M_i$ and $N_j$ are respectively a perfect matching of $G_1$ and $G_2$ that share only one common edge. Hence, the idea is to match each perfect matching $M_i$ with a perfect matching $N_j$. However, since $k$ could be different from $h$, we have to rewrite $x'$ and $x''$ in a way that can be combined.

For each $i, j$, let $\alpha_i = \frac{a_i}{b_i}$ and $\beta_j = \frac{c_j}{b_j}$. Let $\Omega = \text{lcm}(b_1, \ldots, b_k, d_1, \ldots, d_h)$. Let $M'_1, \ldots, M'_{k'}$ a list of perfect matching of $G_1$ obtained through $M_1, \ldots, M_k$ by copying each $M_i$ for $t_i := \frac{\Omega}{b_i} \cdot a_i$ times. We get that:

$$\begin{aligned}
x' &= \frac{a_1}{b_1} M_1 + \ldots \frac{a_k}{b_k} M_k \\
&= \frac{\Omega \cdot a_1}{\Omega \cdot b_1} M_1 + \ldots + \frac{\Omega \cdot a_k}{\Omega \cdot b_k} M_k \\
&= \underbrace{\frac{1}{\Omega} M_1 + \ldots + \frac{1}{\Omega} M_1}_{t_1 \text{ times}} + \ldots + \underbrace{\frac{1}{\Omega} M_k + \ldots + \frac{1}{\Omega} M_k}_{t_k \text{ times}} \\
&= \frac{1}{\Omega} M'_1 + \ldots \frac{1}{\Omega} M'_{k'}
\end{aligned}$$

Likewise, let $N'_1, \ldots, N'_{h'}$ a list of perfect matching of $G_2$ obtained through $N_1, \ldots, N_h$ and rewrite $x''$ as

$$x'' = \frac{1}{\Omega} N'_1 + \ldots \frac{1}{\Omega} N'_{h'}$$

After this process, we get that $k' = h' = \Omega$, meaning that the matchings can be paired together, giving us the following convex combination of perfect matchings of $G$ for $x$:

$$\overline{x} = \frac{1}{\Omega}(M'_1 \cup N'_1) + \ldots \frac{1}{\Omega}(M'_\Omega \cup N'_\Omega)$$

concluding that $\overline{x} \in P_M(G)$, which however is a contradiction. Hence, it must hold that $P_M(G) = P$ also when $\sum_{e \in \delta(\mathcal{U})} \overline{x_e} = 1$.

$$
\begin{aligned}
x' &= \frac{1}{4}M_1 + \frac{1}{2}M_2 + \frac{1}{4}M_3 = \frac{1}{4}M_1 + \frac{1}{4}M_2 + \frac{1}{4}M_2 + \frac{1}{4}M_3 \\
x'' &= \frac{3}{4}N_1 + \frac{1}{4}N_2 = \frac{1}{4}N_1 + \frac{1}{4}N_1 + \frac{1}{4}N_1 + \frac{1}{4}N_2 \\
\implies x &= \frac{1}{4}(M_1 \cup N_1) + \frac{1}{4}(M_2 \cup N_1) + \frac{1}{4}(M_2 \cup N_1) + \frac{1}{4}(M_3 \cup N_2)
\end{aligned}
$$

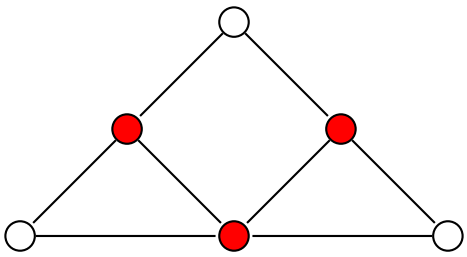
□

## 6.5 The Minimum Vertex Cover problem

**Definition 44: Vertex cover**

Let $G$ be an undirected graph. A subset $X \subseteq V(G)$ is said to be a **vertex cover** if all the edges of $G$ have at least one end-point in $X$, meaning that:

$$\forall (u, v) \in E(G) \;\; u \in X \vee v \in X$$



*The red nodes are the smallest possible vertex cover of the graph*

**Problem 7: The Minimum Vertex Cover problem**

Given an undirected graph $G$, find a vertex cover $X \subseteq V(G)$ with the fewest amount of nodes possible, meaning that:

$$X = \underset{\substack{H \text{ vertex} \\ \text{cover of } G}}{\arg\min} (|H|)$$

Like the two previous problems, we can easily formulate the Minimum Vertex Cover problem as a 0-1 integer program:

- For every node $v \in V(G)$, we define the variabile $x_v$.
- For every edge $(i, j) \in E(G)$ we add the constraint $x_i + x_j \geq 1$. This constraint enforces that at least one between $x_i$ and $x_j$ has to be set to 1.

We get that:

$$\min \sum_{i=1}^{n} x_i$$

$$\begin{aligned} x_i + x_j &\geq 1 \quad \forall (i,j) \in E(G) \\ x &\in \{0,1\}^n \end{aligned}$$

By the very own definition of the integer program, we notice that the Minimum Vertex Cover problem is actually the **complementary of the Maximum Independent Set problem**, a particular case of the Maximum-weight Independent Set problem where all the weights are set to 1. This fact is well known in graph theory and it's stated through the following more general result which directly derives from our observation:

**Theorem 21**

Given an undirected graph $G$, a subset $X \subseteq V(G)$ is a **vertex cover** if and only if $V(G) - X$ is an **independent set**.

The Minimum Vertex Cover problem is also highly related to other graph theory problems. Through the properties of linear programming we can show such results with ease. For example, consider the **dual** of the Minimum Vertex Cover problem: by transposing the constraint matrix we get constraints on the number of edges available for each node.

$$\max \sum_{i=1}^{m} y_i$$

$$\begin{aligned} \sum_{e \in \delta(v)} &\leq 1 \quad \forall v \in V(G) \\ y &\in \{0,1\}^m \end{aligned}$$

A good eye will notice that this dual program corresponds to the **Maximum Matching problem for bipartite graphs**, a particular case of the Maximum-weight Perfect Matching problem where all the weights are set to 1 and the matchings aren't required to be perfect.

Since both the primal and the dual have a feasible solution, by the **strong duality theorem** we know that they share the same optimum, giving us the following theorem "for free".

**Theorem 22: Kőnig's theorem**

Given an undirected bipartite graph $G$, the cardinality of the minimal vertex cover $X \subseteq V(G)$ is **equal** to the cardinality of the maximal matching $M \subseteq E(G)$.