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

April 12, 2024

Contents

Information and Contacts **1**

1 Flow networks **2**
1.1 Networks and flows . . . 2
1.2 Residual graphs, flow increase and *st*-cuts . . . 6
1.3 The Ford-Fulkerson algorithm . . . 12
1.3.1 The Max-flow/Min-cut theorem . . . 13
1.4 The Edmonds-Karp algorithm . . . 14
1.5 Applications of the Max-flow/Min-cut theorem . . . 18

2 Linear programming **22**
2.1 Introduction and interpretation . . . 22
2.2 Linear programs and Standard form . . . 26
2.3 Basic feasible solutions . . . 29
2.3.1 Above bounded linear programs and BFS . . . 33
2.4 Geometry of linear programs . . . 37
2.4.1 Convexity . . . 37
2.4.2 Hyperplanes, Half-Spaces and Polytopes . . . 43
2.4.3 Vertices and Basic Feasible Solutions . . . 46

3 The Simplex method **48**
3.1 Intuition and example . . . 48
3.2 Unboundedness . . . 52
3.3 Infeasibility of the trivial basis . . . 53
3.4 Degeneracy . . . 57
3.5 Formalization of the simplex method . . . 60
3.6 Pivot rules . . . 63
3.6.1 Bland's rule . . . 66

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.

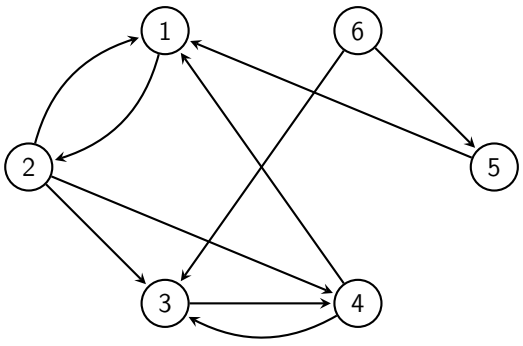
We will assume that each graph is *simple*, meaning that there are no multiple edges between the same nodes and no loops (that being an edge from a vertex to itself).

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

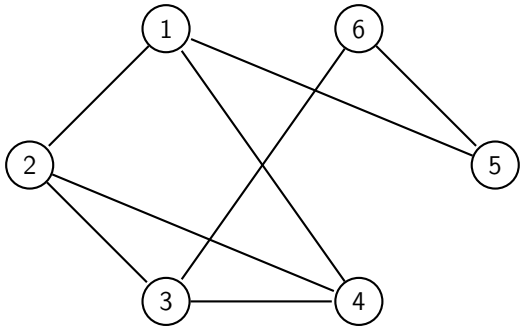
A graph can be **directed** or **undirected**. In a directed graph we consider the edges $(u, v) \in E(G)$ and $(v, u) \in E(G)$ as two different edges, while in an undirected graph they represent the same edge.

Example:

Directed graph



Undirected graph

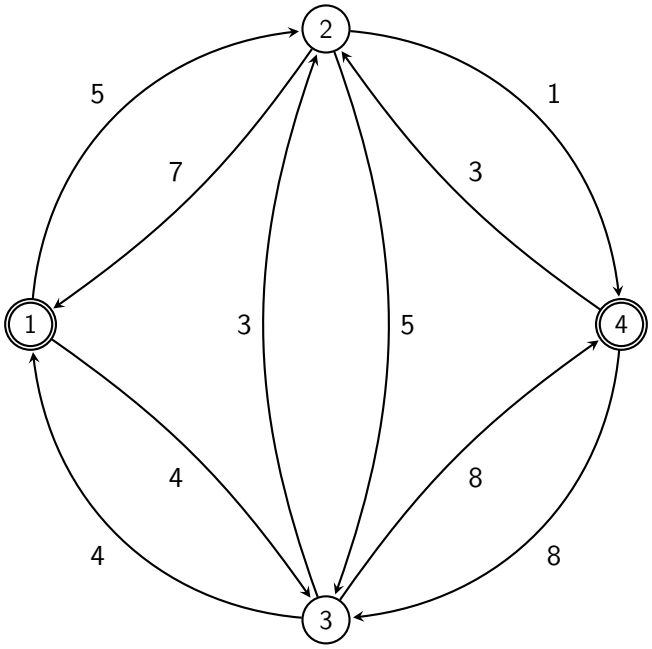


Definition 2: Network

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

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

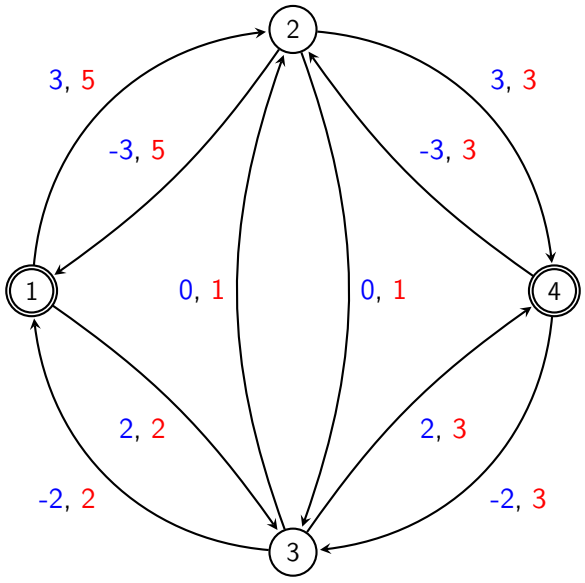
  meaning that the incoming flow in v is the same as the outgoing flow from v

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:

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

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

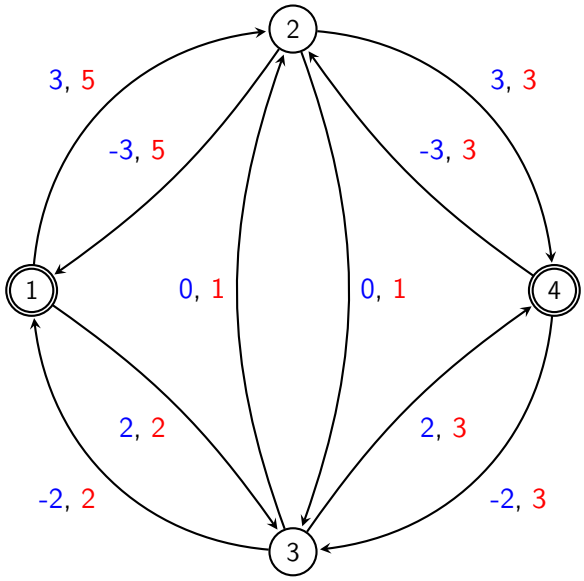
Then, by adding each edge incoming in v that has no flow we conclude that:

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

□

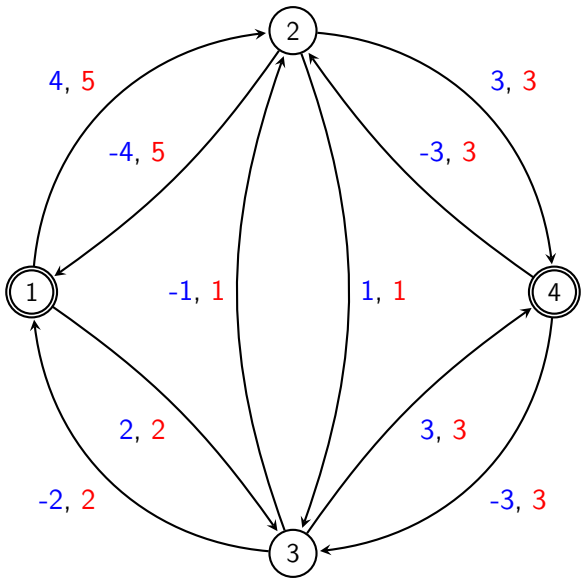
1.2 Residual graphs, flow increase and st-cuts

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:

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

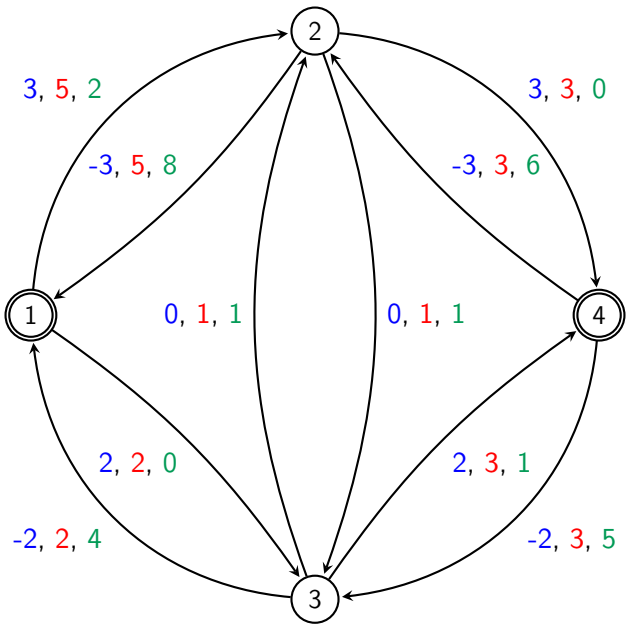
Definition 6: Residual graph

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:

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

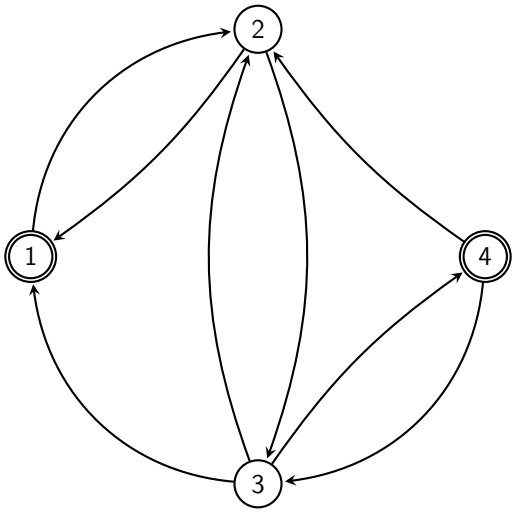
Example:

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 α 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 α 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 α 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 α 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 α_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 α_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 $x_1 + 2x_2$ while also respecting the following linear constraints:

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_1 + 6x_2 &\leq 15 \\ 4x_1 - x_2 &\leq 10 \\ -x_1 + x_2 &\leq 2 \end{aligned}$$

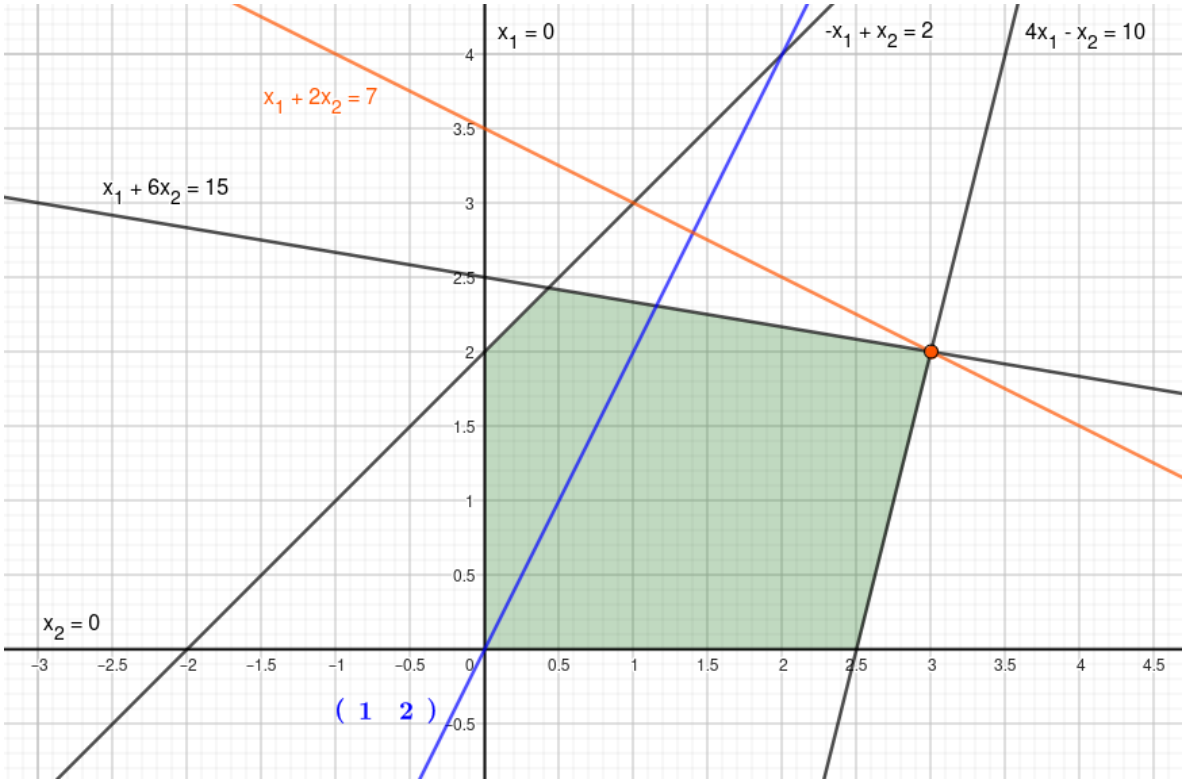
Since we have only two variables, these constraints can be interpreted as lines in a cartesian plane defined by x_1 and x_2: given a constraint $\alpha x_1 + \beta y_1 \leq \gamma$ (or $\alpha x_1 + \beta y_1 \geq \gamma$), the plane gets partitioned by the line $\alpha x_1 + \beta y_1 = \gamma$ and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

$$x_1 + 2x_2 = \begin{bmatrix} 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Considering the line described by the vector $\begin{bmatrix}1 & 2\end{bmatrix}$, 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 $\begin{bmatrix}1 & 2\end{bmatrix}$ that passes through such point is the optimal solution.



The optimal solution is given by $x_1 + 2x_2 = 7$

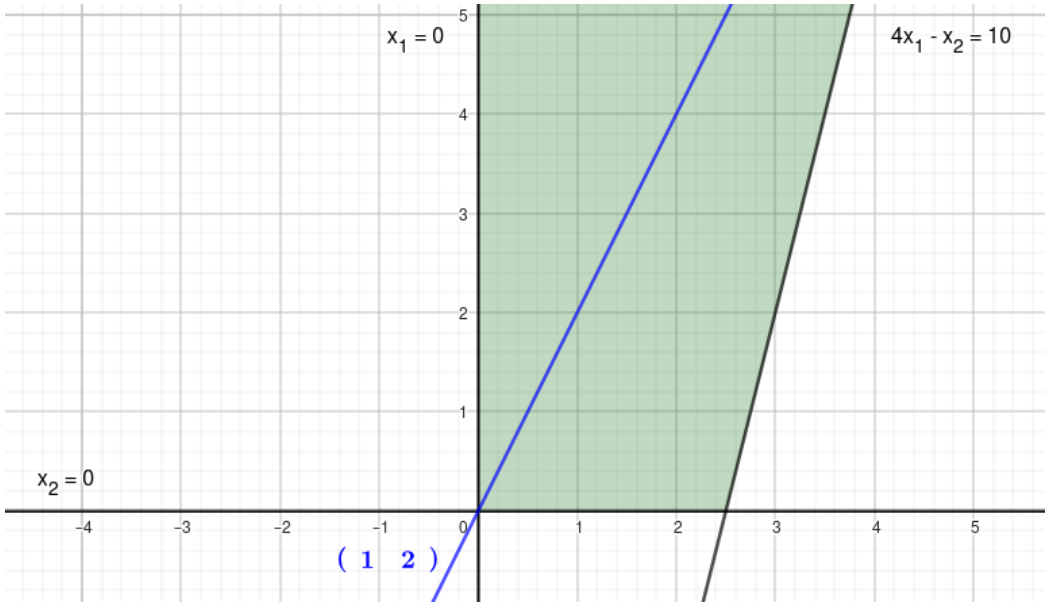
For each linear program, only one of the following cases holds:

1. There is no feasible solution
2. There are infinite feasible solutions but no optimal solution
3. There are infinite feasible solutions and only one optimal solution
4. There are infinite feasible and optimal solutions

This result is (for now) just an informal result. In later sections, we will formally define each concept and justify each result.

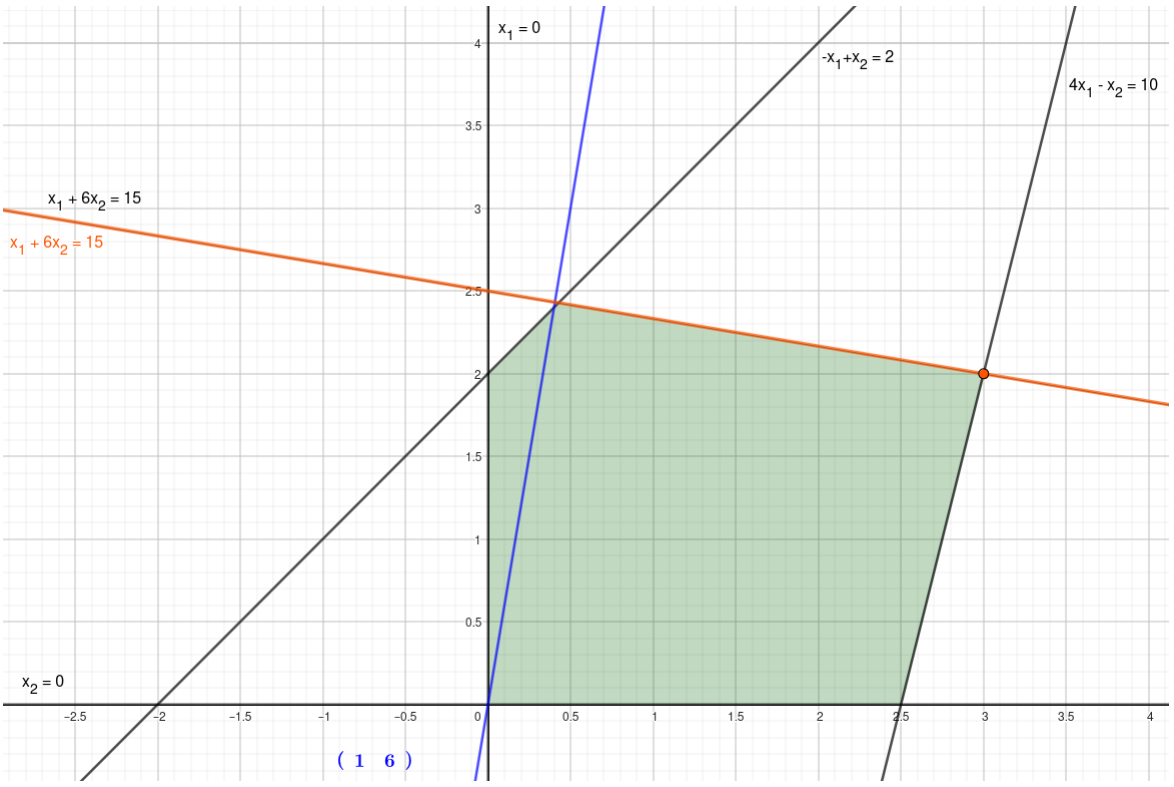
The previous problem clearly is an example of the third case. For the first case, a problem with constraints $x_1 \geq 0$ and $x_1 \leq 1$ trivially has no feasible solution since there is no possible x_1 value that respects the constraints.

The second case, instead, can be viewed as a problem with constraints such that the feasible region isn't bounded "in all directions". For example, if we remove the constraint $x_1 + 6x_2$, the feasible region isn't bounded from above, meaning that there will always be a feasible solution better than the one considered.



Example with no optimal solution

The final case can be obtained if the objective function is equal to a constraint: since the every point on the perpendicular line is an optimal solution and since in $\mathbb{R}^2$ there are infinite points on a line, we get infinite optimal solutions.



Same problem as before with objective function $x_1 + 6x_2 = 15$

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:

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

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

- **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. In particular, we will define linear programs only for maximization problems since minimizing a value $f(x)$ is equal to maximizing $-f(x)$.

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:

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

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

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:

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

Thus, we can say that $f(x) = c^T x$, where $c^T = \begin{bmatrix} c_1 & \dots & c_n \end{bmatrix}$ is the transpose of the **coefficient vector of** f.

Moreover, given an inequality $a_1 x_1 + \dots a_n x_n \geq b$, we know that:

$$a_1 x_1 + \dots + a_n x_n \geq b \iff -a_1 x_1 - \dots - a_n x_n \leq -b$$

meaning that we can substitute the first inequality on the left with the one on the right. Likewise, we want all the variables inside the solution to be non-negative. Thus, given a constraint $x_i \leq 0$, we can consider two new variables z_i, z_i' such that $x_i = z_i - z_i'$

and $z_i, z_i' \geq 0$. Thus, we can substitute the first constraint with the three constraints $z_i - z_i^i - x = 0$ and $z_i, z_i' \geq 0$.

Finally, by considering each constraint of as a row of a matrix A, we can rewrite each constraint to define the following **general formulation of linear programs**:

$$\max c^T x$$

$$Ax \leq b$$
$$x \geq 0$$

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

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

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

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:

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

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

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

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

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

$$\max c'^T x$$

$$A'x = b$$
$$x \geq 0$$

Definition 12: Standard form

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

$$\max c^T x$$

$$Ax = b$$
$$x \geq 0$$

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 given the equation $Ax = b$ of a linear program in standard form, the set of feasible solutions is equal to $\ker(A)$, implying that it 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**.

Hence, 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 $m \leq n$. Thus, since $\mathrm{rowrank}(A) = m$ and $m \leq n$, by basic linear algebra results we also get that $\mathrm{colrank}(A) = m$.

Observation 4: Assumptions on standard form LPs

Given a linear program in standard form defined by the equation $Ax = b$, 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.

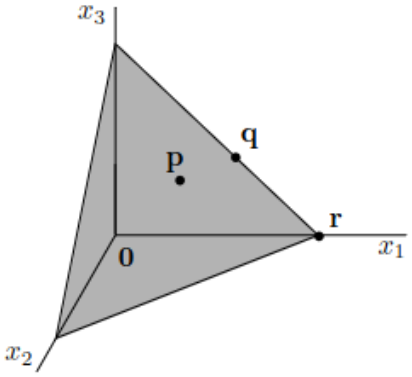


Figure 2.1: A set of feasible solutions

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:

$$\max c^T x$$

$$Ax = b$$
$$x \geq 0$$

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 4

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 5

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 is bounded above 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 is bounded above. 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. 2.1, 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 is bounded above, 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 is bounded above. 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 of linear programs

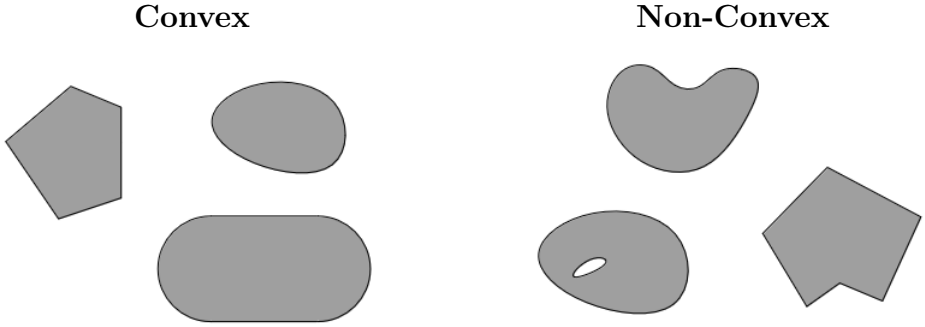
2.4.1 Convexity

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 6

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 β 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 $x_1, \ldots, x_n$.

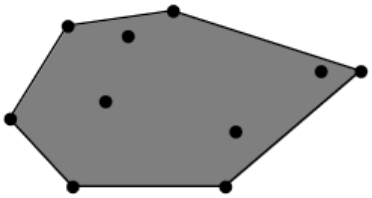


Figure 2.2: Example of convex hull

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:

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

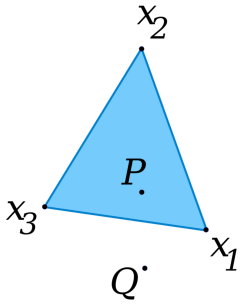


Figure 2.3: The point P is a convex combination of x_1, x_2, x_3, while Q is not

Proposition 7

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 $\tilde{C} \subseteq C$.

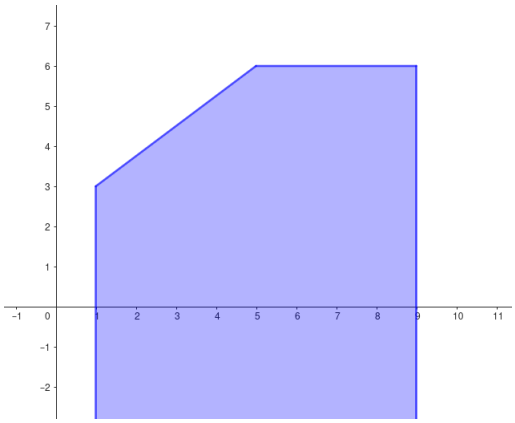
□

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:

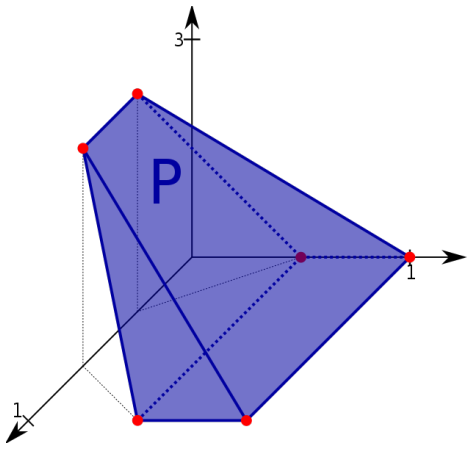
- If $n = 2$ then the set of convex combinations corresponds to a segment between x_1 and x_2
- If $n = 3$ then the set of convex combinations corresponds to a convex bounded polygon with vertices x_1, x_2 and x_3
- If $n = 4$ then the set of convex combinations corresponds to a convex bounded polyhedron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

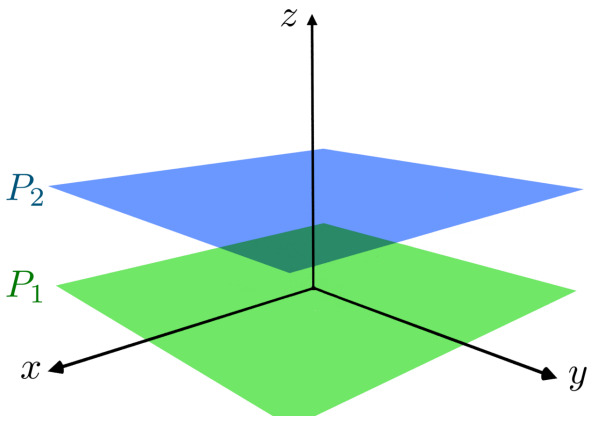


Figure 2.4: P_2 is an affine subspace with P_1 as its direction

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:

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

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

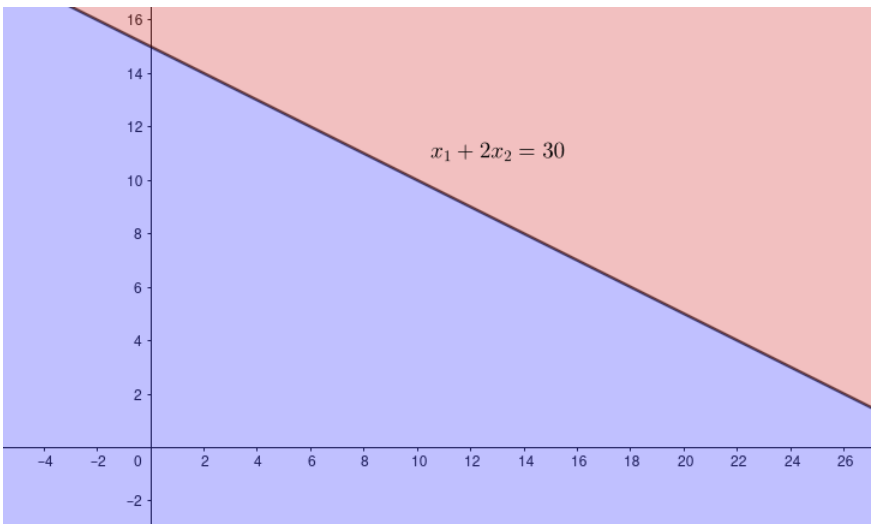


Figure 2.5: The (affine) plane $x_1 + 2x_2 = 30$ splits the plane into two half spaces

Observation 9

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

Proof.

The proof is the same as the proof of Proposition 6, 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.

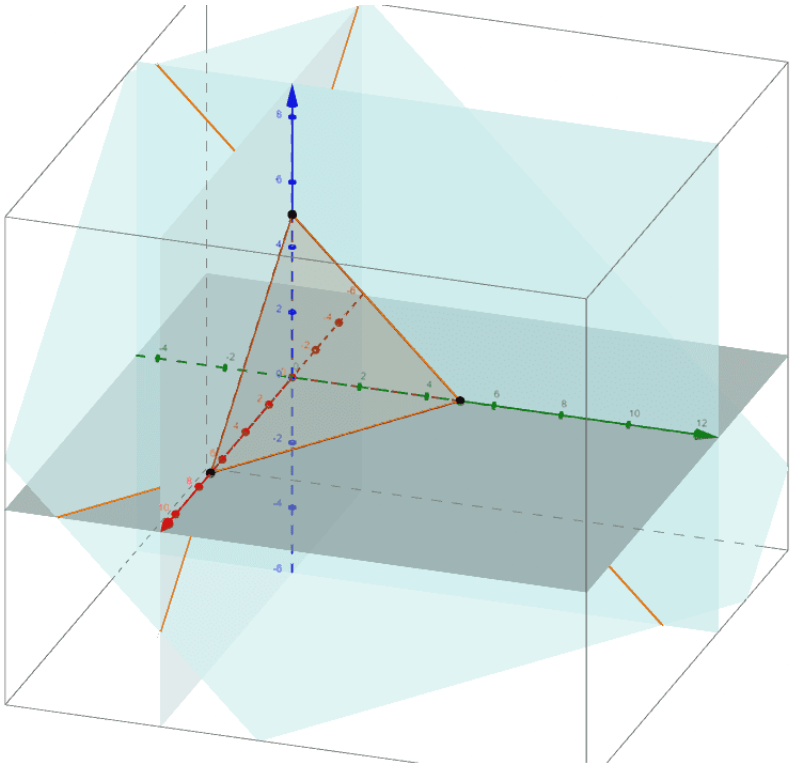


Figure 2.6: The feasible region (red) of a linear program as a polytope generated by the constraint matrix's hyperplanes (light blue)

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.

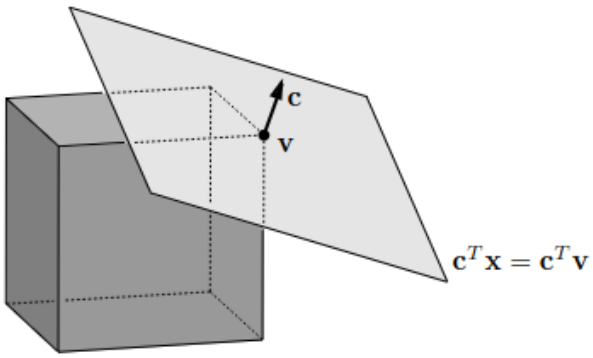
Definition 24: Face

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 $H = \{x \in \mathbb{R}^n \mid c^T x = b\}$ for which:

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

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 $c^T v = b$, 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 $c^T y < b$.

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

3

The Simplex method

3.1 Intuition and example

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).

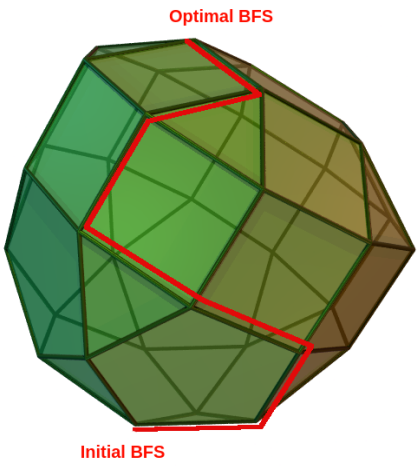


Figure 3.1: Example of execution of the simplex algorithm

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 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**.

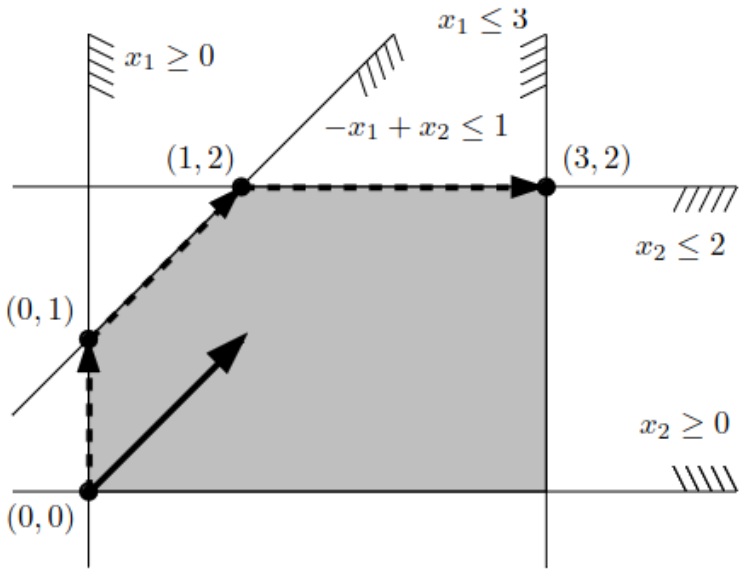


Figure 3.2: The simplex method in action

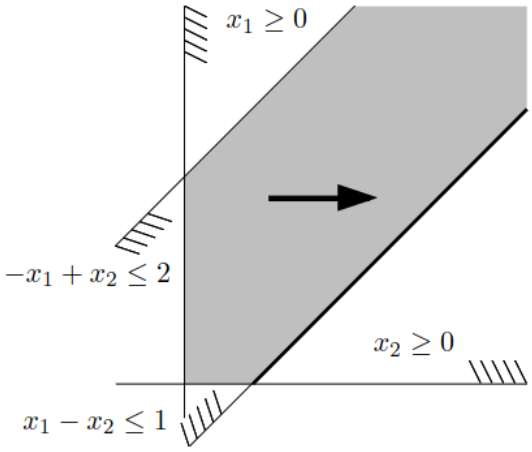
We also notice that we could've taken a "shorter route" by picking x_1 as the first pivot, concluding the algorithm in just two pivot steps.

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 8: 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 is bounded above. 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 $P \neq \varnothing$:

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

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 α. Otherwise, if $P_\alpha = \varnothing$ then the optimal solution has an objective function value less than α.

Through the subroutine, we can use binary search to find the optimal solution: if $P_\alpha \neq \varnothing$ we double the value of α and call again the subroutine, otherwise we halve the value of α. Once this increment/halving can't be repeated anymore, we will have determined the value α corresponding to the value given by the optimal solution of P. Thus, for the final value α 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 is bounded above. Thus, since the objective function is bounded above 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:

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

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

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

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:

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

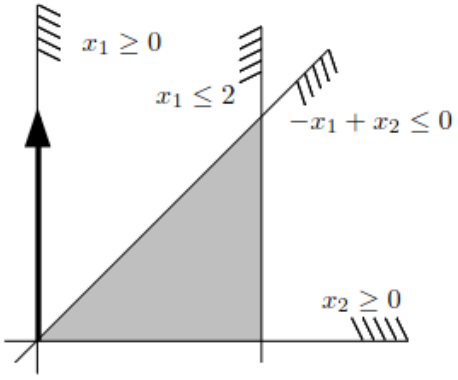
Since there are no more variables with positive coefficient, the BFS s'_3 must be optimal. In fact, as we said, the optimal value for this problem is 0, which is given by s'_3. Moreover, by own definition of the auxiliary program, restricting s'_3 to x_1, x_2, x_3, x_4 gives us a valid initial BFS $s_1 = \begin{bmatrix} \frac{5}{2} & \frac{3}{2} & 0 & 0 \end{bmatrix}$ 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 $\mathcal{B}_1 = \{3, 4\}$:

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

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:

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

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:

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

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 9: 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 10: 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 section.

Moreover, in the previous sections 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 11: 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_β} 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_β} 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_β}, while if $q_{i\beta} < 0$ it yields the restriction $x_{h_\beta} \leq -\frac{p_i}{q_{i\beta}}$.

Let x_{k_α} 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_β} while also keeping these restrictions valid, **the leaving variable x_{k_α} can be any variable such that**

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

and $q_{\alpha\beta} < 0$. Moreover, if there is no index α that satisfies these conditions, the linear program is **unbounded** since there are no restrictions on the possible value of x_{h_β}

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_ℓ 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_ℓ satisfies the criterion for a leaving variable, the linear program is **unbounded**.

3.6 Pivot rules

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_α} and entering variable x_{h_β} from all possible choices given by the Criteria for possible pivot steps:

1. **Largest coefficient rule**: pick x_{k_α} 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_β} 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_{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 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_ℓ 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_α} 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_β} 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_α} is the leaving variable. Thus, we can just solve the equation for x_{k_α} in $\mathcal{T}(\mathcal{B}_j)$ for x_{h_β}, 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_β} 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_β} 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_β}, implying that $x_{h_\beta}^j = x_{h_\beta}^{j+1}$.

However, we also know that such increase depends on the choice of k_α. In particular, we should increase x_{h_β} 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_α} 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 9, 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_ℓ 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

Proof. Todo □