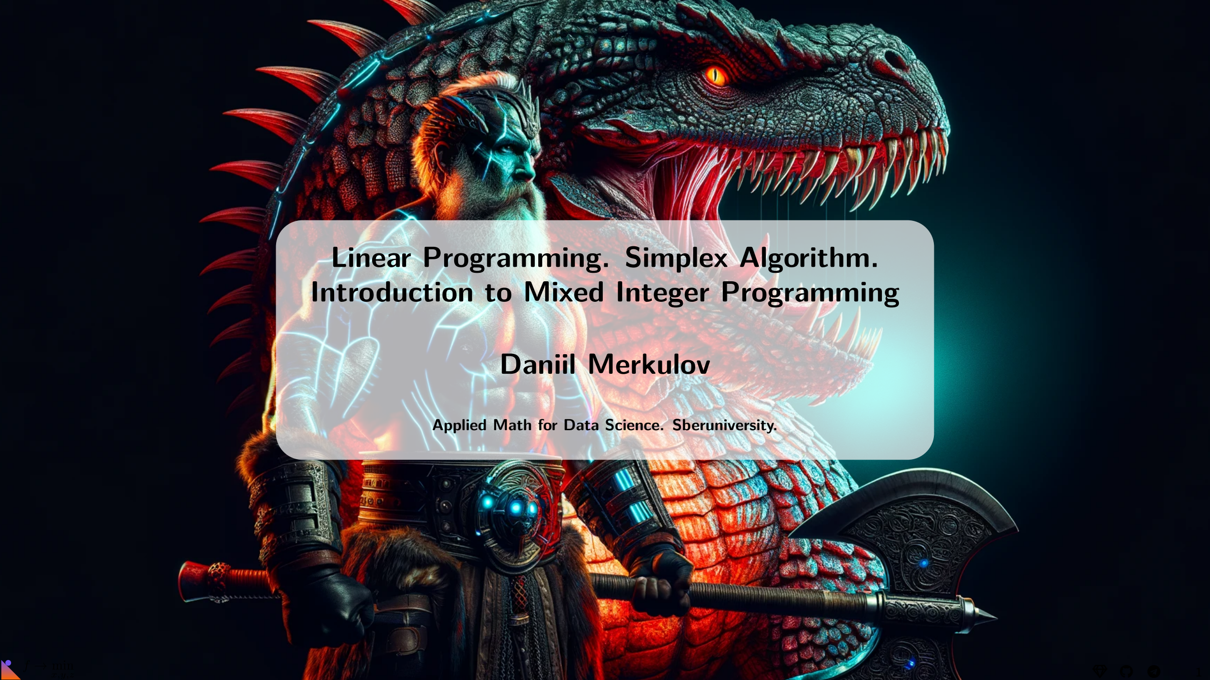


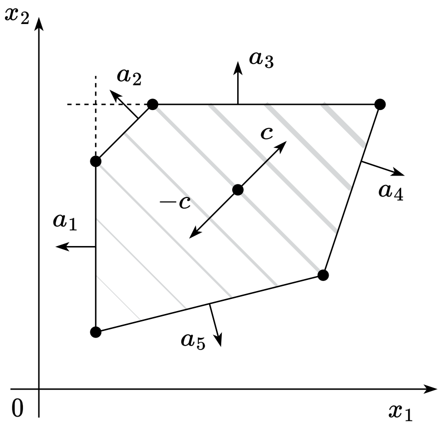
Linear Programming. Simplex Algorithm. Introduction to Mixed Integer Programming

Daniil Merkulov

Applied Math for Data Science. Sberuniversity.

Linear Programming

What is Linear Programming?

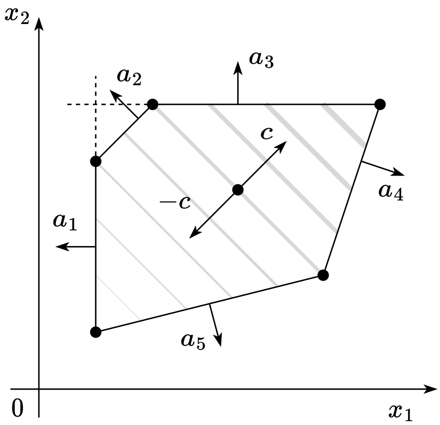


Generally speaking, all problems with linear objective and linear equalities/inequalities constraints could be considered as Linear Programming. However, there are some formulations.

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
&\text{s.t. } Ax \leq b
\end{aligned}
\tag{LP.Basic}
$$

for some vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and matrix $A \in \mathbb{R}^{m \times n}$. Where the inequalities are interpreted component-wise.

What is Linear Programming?



Generally speaking, all problems with linear objective and linear equalities/inequalities constraints could be considered as Linear Programming. However, there are some formulations.

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax \leq b
\end{aligned}
\tag{LP.Basic}
$$

for some vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and matrix $A \in \mathbb{R}^{m \times n}$. Where the inequalities are interpreted component-wise.

Standard form. This form seems to be the most intuitive and geometric in terms of visualization. Let us have vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and matrix $A \in \mathbb{R}^{m \times n}$.

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax = b \\
& x_i \geq 0, \; i = 1, \dots, n
\end{aligned}
\tag{LP.Standard}
$$

Example: Diet problem

🍌 🍰 🍗 🥚 🐟

$$
\begin{array}{c}
\text{Proteins} \\ \text{Carbs} \\ \text{Fats} \\ \text{Calories} \\ \text{Vitamin D}
\end{array}
\begin{bmatrix}
\\ \\ \text{Amount per 100g} \\ W \in \mathbb{R}^{n \times p} \\ \\ \\
\end{bmatrix}
$$

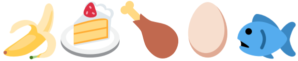
$c \in \mathbb{R}^p$, price per 100g

$r \in \mathbb{R}^n$, nutrient requirements

$x \in \mathbb{R}^p$, amount of products, 100g

$$
\begin{aligned}
\min_{x \in \mathbb{R}^p} \ & c^T x \\
Wx & \succeq r \\
x & \succeq 0
\end{aligned}
$$

Example: Diet problem



$$
\begin{array}{c}
\text{Proteins} \\[2em]
\text{Carbs} \\[2em]
\text{Fats} \\[2em]
\text{Calories} \\[2em]
\text{Vitamin D}
\end{array}
\left[
\begin{array}{c}
\\[2em]
\\[2em]
\text{Amount per 100g} \\
W \in \mathbb{R}^{n \times p} \\[2em]
\\[2em]
\end{array}
\right]
$$

$c \in \mathbb{R}^p$, price per 100g

$r \in \mathbb{R}^n$, nutrient requirements

$x \in \mathbb{R}^p$, amount of products, 100g

$$
\begin{split}
&\min_{x \in \mathbb{R}^p} c^T x \\
&Wx \succeq r \\
&x \succeq 0
\end{split}
$$

Imagine, that you have to construct a diet plan from some set of products: bananas, cakes, chicken, eggs, fish. Each of the products has its vector of nutrients. Thus, all the food information could be processed through the matrix W. Let us also assume, that we have the vector of requirements for each of nutrients $r \in \mathbb{R}^n$. We need to find the cheapest configuration of the diet, which meets all the requirements:

$$
\begin{split}
&\min_{x \in \mathbb{R}^p} c^\top x \\
\text{s.t. } & Wx \succeq r \\
& x_i \geq 0, \; i = 1, \dots, n
\end{split}
$$

Open In Colab

Basic transformations

- Max-min

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
&\text{s.t. } Ax \leq b
\end{aligned}
\quad \leftrightarrow \quad
\begin{aligned}
&\max_{x \in \mathbb{R}^n} -c^\top x \\
&\text{s.t. } Ax \leq b
\end{aligned}
$$

Basic transformations

- Max-min

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax \leq b
\end{aligned}
\quad \leftrightarrow \quad
\begin{aligned}
&\max_{x \in \mathbb{R}^n} -c^\top x \\
\text{s.t. } & Ax \leq b
\end{aligned}
$$

- Equality to inequality

$$
Ax = b \leftrightarrow \begin{cases} Ax \leq b \\ Ax \geq b \end{cases}
$$

Basic transformations

- Max-min

$$\begin{aligned} \min_{x \in \mathbb{R}^n} c^\top x \\ \text{s.t. } Ax \leq b \end{aligned} \quad \leftrightarrow \quad \begin{aligned} \max_{x \in \mathbb{R}^n} -c^\top x \\ \text{s.t. } Ax \leq b \end{aligned}$$

- Equality to inequality

$$Ax = b \leftrightarrow \begin{cases} Ax \leq b \\ Ax \geq b \end{cases}$$

- Inequality to equality by increasing the dimension of the problem by m.

$$Ax \leq b \leftrightarrow \begin{cases} Ax + z = b \\ z \geq 0 \end{cases}$$

Basic transformations

- Max-min

$$
\begin{aligned}
\min_{x \in \mathbb{R}^n} \; & c^\top x \\
\text{s.t. } & Ax \leq b
\end{aligned}
\quad \leftrightarrow \quad
\begin{aligned}
\max_{x \in \mathbb{R}^n} \; & -c^\top x \\
\text{s.t. } & Ax \leq b
\end{aligned}
$$

- Equality to inequality

$$
Ax = b \leftrightarrow \begin{cases} Ax \leq b \\ Ax \geq b \end{cases}
$$

- Inequality to equality by increasing the dimension of the problem by m.

$$
Ax \leq b \leftrightarrow \begin{cases} Ax + z = b \\ z \geq 0 \end{cases}
$$

- Unsigned variables to nonnegative variables.

$$
x \leftrightarrow \begin{cases} x = x_+ - x_- \\ x_+ \geq 0 \\ x_- \geq 0 \end{cases}
$$

Example: Chebyshev approximation problem

$$
\min_{x \in \mathbb{R}^n} \|Ax - b\|_\infty \leftrightarrow \min_{x \in \mathbb{R}^n} \max_i |a_i^T x - b_i|
$$

Could be equivalently written as an LP with the replacement of the maximum coordinate of a vector:

Example: Chebyshev approximation problem

$$
\min_{x \in \mathbb{R}^n} \|Ax - b\|_\infty \leftrightarrow \min_{x \in \mathbb{R}^n} \max_{i} |a_i^T x - b_i|
$$

Could be equivalently written as an LP with the replacement of the maximum coordinate of a vector:

$$
\begin{split}
&\min_{t \in \mathbb{R}, x \in \mathbb{R}^n} t \\
\text{s.t. } & a_i^T x - b_i \leq t, \; i = 1, \dots, n \\
& - a_i^T x + b_i \leq t, \; i = 1, \dots, n
\end{split}
$$

ℓ_1 approximation problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_1 \leftrightarrow \min_{x \in \mathbb{R}^n} \sum_{i=1}^n |a_i^T x - b_i|$$

Could be equivalently written as an LP with the replacement of the sum of coordinates of a vector:

ℓ_1 approximation problem

$$
\min_{x \in \mathbb{R}^n} \|Ax - b\|_1 \leftrightarrow \min_{x \in \mathbb{R}^n} \sum_{i=1}^n |a_i^T x - b_i|
$$

Could be equivalently written as an LP with the replacement of the sum of coordinates of a vector:

$$
\begin{aligned}
&\min_{t \in \mathbb{R}^n, x \in \mathbb{R}^n} \mathbf{1}^T t \\
\text{s.t. } & a_i^T x - b_i \leq t_i, \; i = 1, \dots, n \\
& -a_i^T x + b_i \leq t_i, \; i = 1, \dots, n
\end{aligned}
$$

Duality in Linear Programming

Duality

Primal problem:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax = b \\
& x_i \geq 0, \; i = 1, \dots, n
\end{aligned}
\tag{1}
$$

Duality

Primal problem:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax = b \\
& x_i \geq 0, \; i = 1, \dots, n
\end{aligned}
\tag{1}
$$

KKT for optimal x^*, ν^*, λ^*:

$$
\begin{aligned}
&L(x, \nu, \lambda) = c^T x + \nu^T (Ax - b) - \lambda^T x \\
&-A^T \nu^* + \lambda^* = c \\
&Ax^* = b \\
&x^* \succeq 0 \\
&\lambda^* \succeq 0 \\
&\lambda_i^* x_i^* = 0
\end{aligned}
$$

Duality

Primal problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & c^\top x \\ \text{s.t. } & Ax = b \\ & x_i \geq 0, \; i = 1, \dots, n \end{aligned} \tag{1}$$

KKT for optimal x^*, ν^*, λ^*:

$$\begin{aligned} & L(x, \nu, \lambda) = c^T x + \nu^T (Ax - b) - \lambda^T x \\ & -A^T \nu^* + \lambda^* = c \\ & Ax^* = b \\ & x^* \succeq 0 \\ & \lambda^* \succeq 0 \\ & \lambda_i^* x_i^* = 0 \end{aligned}$$

Has the following dual:

$$\begin{aligned} \max_{\nu \in \mathbb{R}^m} \; & -b^\top \nu \\ \text{s.t. } & -A^T \nu \preceq c \end{aligned} \tag{2}$$

Find the dual problem to the problem above (it should be the original LP). Also, write down KKT for the dual problem, to ensure, they are identical to the primal KKT.

Strong duality in linear programming

(i) If either problem Equation 1 or Equation 2 has a (finite) solution, then so does the other, and the objective values are equal.

Strong duality in linear programming

(i) If either problem Equation 1 or Equation 2 has a (finite) solution, then so does the other, and the objective values are equal.

(ii) If either problem Equation 1 or Equation 2 is unbounded, then the other problem is infeasible.

Strong duality in linear programming

(i) If either problem Equation 1 or Equation 2 has a (finite) solution, then so does the other, and the objective values are equal.

(ii) If either problem Equation 1 or Equation 2 is unbounded, then the other problem is infeasible.

Strong duality in linear programming

(i) If either problem Equation 1 or Equation 2 has a (finite) solution, then so does the other, and the objective values are equal.

(ii) If either problem Equation 1 or Equation 2 is unbounded, then the other problem is infeasible.

PROOF. For (i), suppose that Equation 1 has a finite optimal solution x^*. It follows from KKT that there are optimal vectors λ^* and ν^* such that (x^*, ν^*, λ^*) satisfies KKT. We noted above that KKT for Equation 1 and Equation 2 are equivalent. Moreover, $c^T x^* = (-A^T \nu^* + \lambda^*)^T x^* = -(\nu^*)^T A x^* = -b^T \nu^*$, as claimed.

A symmetric argument holds if we start by assuming that the dual problem Equation 2 has a solution.

Strong duality in linear programming

(i) If either problem Equation 1 or Equation 2 has a (finite) solution, then so does the other, and the objective values are equal.

(ii) If either problem Equation 1 or Equation 2 is unbounded, then the other problem is infeasible.

PROOF. For (i), suppose that Equation 1 has a finite optimal solution x^*. It follows from KKT that there are optimal vectors λ^* and ν^* such that (x^*, ν^*, λ^*) satisfies KKT. We noted above that KKT for Equation 1 and Equation 2 are equivalent. Moreover, $c^T x^* = (-A^T \nu^* + \lambda^*)^T x^* = -(\nu^*)^T A x^* = -b^T \nu^*$, as claimed.

A symmetric argument holds if we start by assuming that the dual problem Equation 2 has a solution.

To prove (ii), suppose that the primal is unbounded, that is, there is a sequence of points x_k, $k = 1, 2, 3, \ldots$ such that

$$
c^T x_k \downarrow -\infty, \quad A x_k = b, \quad x_k \geq 0.
$$

Strong duality in linear programming

(i) If either problem Equation 1 or Equation 2 has a (finite) solution, then so does the other, and the objective values are equal.

(ii) If either problem Equation 1 or Equation 2 is unbounded, then the other problem is infeasible.

PROOF. For (i), suppose that Equation 1 has a finite optimal solution x^*. It follows from KKT that there are optimal vectors λ^* and ν^* such that (x^*, ν^*, λ^*) satisfies KKT. We noted above that KKT for Equation 1 and Equation 2 are equivalent. Moreover, $c^T x^* = (-A^T \nu^* + \lambda^*)^T x^* = -(\nu^*)^T A x^* = -b^T \nu^*$, as claimed.

A symmetric argument holds if we start by assuming that the dual problem Equation 2 has a solution.

To prove (ii), suppose that the primal is unbounded, that is, there is a sequence of points x_k, $k = 1, 2, 3, \ldots$ such that

$$c^T x_k \downarrow -\infty, \quad Ax_k = b, \quad x_k \geq 0.$$

Suppose too that the dual Equation 2 is feasible, that is, there exists a vector $\bar{\nu}$ such that $-A^T \bar{\nu} \leq c$. From the latter inequality together with $x_k \geq 0$, we have that $-\bar{\nu}^T A x_k \leq c^T x_k$, and therefore

$$-\bar{\nu}^T b = -\bar{\nu}^T A x_k \leq c^T x_k \downarrow -\infty,$$

yielding a contradiction. Hence, the dual must be infeasible. A similar argument can be used to show that the unboundedness of the dual implies the infeasibility of the primal.

Example: Transportation problem

The prototypical transportation problem deals with the distribution of a commodity from a set of sources to a set of destinations. The object is to minimize total transportation costs while satisfying constraints on the supplies available at each of the sources, and satisfying demand requirements at each of the destinations.

Figure 1: Western Europe Map. Open In Colab

Example: Transportation problem

Customer / Source	Arnhem [€/ton]	Gouda [€/ton]	Demand [tons]
London	n/a	2.5	125
Berlin	2.5	n/a	175
Maastricht	1.6	2.0	225
Amsterdam	1.4	1.0	250
Utrecht	0.8	1.0	225
The Hague	1.4	0.8	200
Supply [tons]	550 tons	700 tons	

$$\text{minimize:} \quad \text{Cost} = \sum_{c \in \text{Customers}} \sum_{s \in \text{Sources}} T[c, s] x[c, s]$$

Example: Transportation problem

Customer / Source	Arnhem [€/ton]	Gouda [€/ton]	Demand [tons]
London	n/a	2.5	125
Berlin	2.5	n/a	175
Maastricht	1.6	2.0	225
Amsterdam	1.4	1.0	250
Utrecht	0.8	1.0	225
The Hague	1.4	0.8	200
Supply [tons]	550 tons	700 tons	

$$\text{minimize:} \quad \text{Cost} = \sum_{c \in \text{Customers}} \sum_{s \in \text{Sources}} T[c,s]x[c,s]$$

$$\sum_{c \in \text{Customers}} x[c,s] \le \text{Supply}[s] \qquad \forall s \in \text{Sources}$$

Example: Transportation problem

Customer / Source	Arnhem [€/ton]	Gouda [€/ton]	Demand [tons]
London	n/a	2.5	125
Berlin	2.5	n/a	175
Maastricht	1.6	2.0	225
Amsterdam	1.4	1.0	250
Utrecht	0.8	1.0	225
The Hague	1.4	0.8	200
Supply [tons]	550 tons	700 tons	

$$\text{minimize:} \quad \text{Cost} = \sum_{c \in \text{Customers}} \sum_{s \in \text{Sources}} T[c,s]x[c,s]$$

$$\sum_{c \in \text{Customers}} x[c,s] \leq \text{Supply}[s] \qquad \forall s \in \text{Sources}$$

$$\sum_{s \in \text{Sources}} x[c,s] = \text{Demand}[c] \qquad \forall c \in \text{Customers}$$

This can be represented in the following graph:

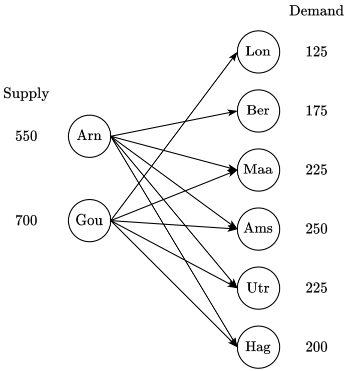
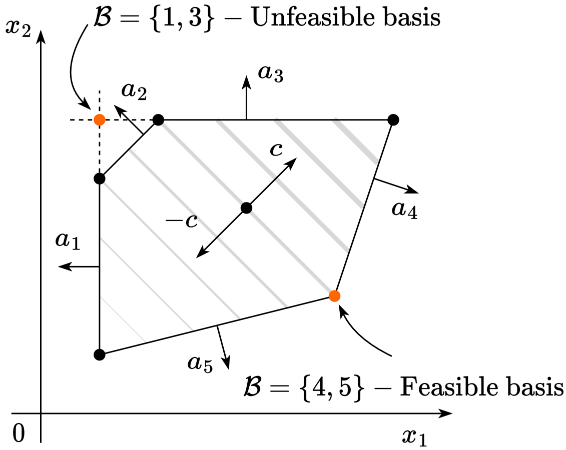


Figure 2: Graph associated with the problem

Simplex Algorithm

Geometry of simplex algorithm

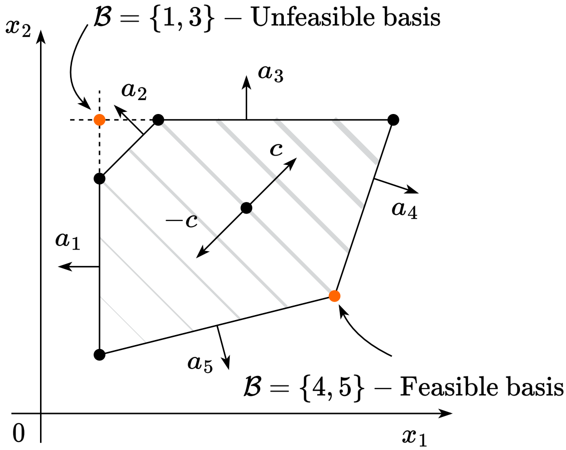


We will consider the following simple formulation of LP, which is, in fact, dual to the Standard form:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} c^\top x \\ \text{s.t. } &Ax \leq b \end{aligned} \tag{LP.Inequality}$$

- Definition: a **basis** $\mathcal{B}$ is a subset of n (integer) numbers between 1 and m, so that $\text{rank} A_{\mathcal{B}} = n$.

Geometry of simplex algorithm

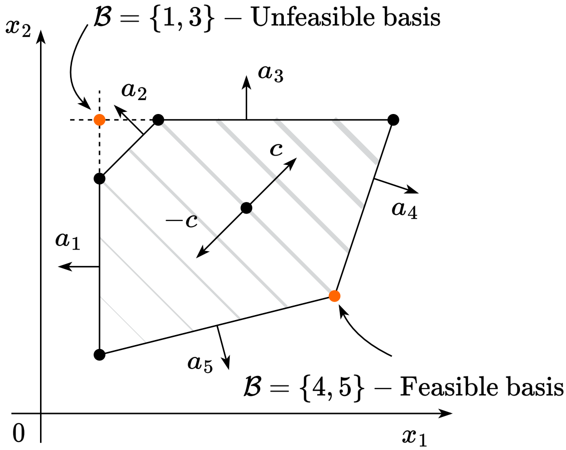


We will consider the following simple formulation of LP, which is, in fact, dual to the Standard form:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
&\text{s.t. } Ax \leq b
\end{aligned}
\tag{LP.Inequality}
$$

- Definition: a **basis** $\mathcal{B}$ is a subset of n (integer) numbers between 1 and m, so that $\text{rank} A_{\mathcal{B}} = n$.
- Note, that we can associate submatrix $A_{\mathcal{B}}$ and corresponding right-hand side $b_{\mathcal{B}}$ with the basis $\mathcal{B}$.

Geometry of simplex algorithm

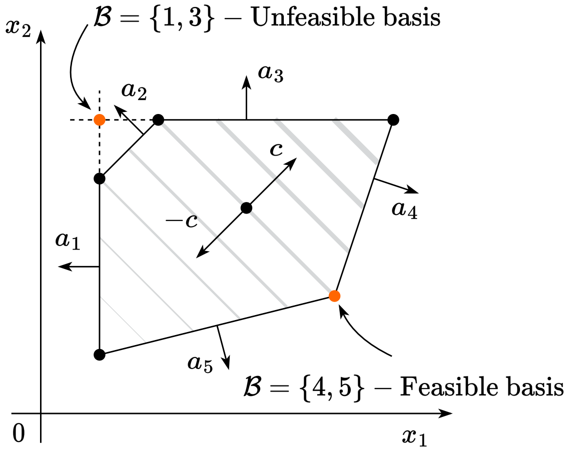


We will consider the following simple formulation of LP, which is, in fact, dual to the Standard form:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax \leq b
\end{aligned}
\tag{LP.Inequality}
$$

- Definition: a **basis** $\mathcal{B}$ is a subset of n (integer) numbers between 1 and m, so that $\text{rank} A_{\mathcal{B}} = n$.
- Note, that we can associate submatrix $A_{\mathcal{B}}$ and corresponding right-hand side $b_{\mathcal{B}}$ with the basis $\mathcal{B}$.
- Also, we can derive a point of intersection of all these hyperplanes from the basis: $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b_{\mathcal{B}}$.

Geometry of simplex algorithm

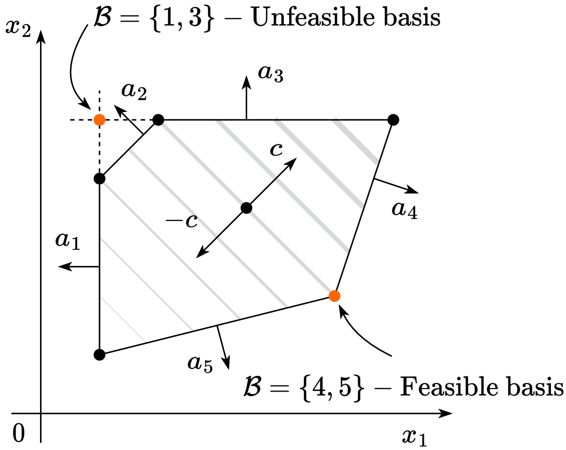


We will consider the following simple formulation of LP, which is, in fact, dual to the Standard form:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax \leq b
\end{aligned}
\tag{LP.Inequality}
$$

- Definition: a **basis** $\mathcal{B}$ is a subset of n (integer) numbers between 1 and m, so that $\operatorname{rank} A_{\mathcal{B}} = n$.
- Note, that we can associate submatrix $A_{\mathcal{B}}$ and corresponding right-hand side $b_{\mathcal{B}}$ with the basis $\mathcal{B}$.
- Also, we can derive a point of intersection of all these hyperplanes from the basis: $x_{\mathcal{B}} = A^{-1}_{\mathcal{B}} b_{\mathcal{B}}$.
- If $Ax_{\mathcal{B}} \leq b$, then basis $\mathcal{B}$ is **feasible**.

Geometry of simplex algorithm

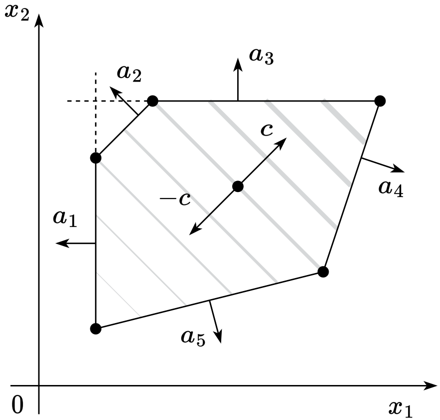


We will consider the following simple formulation of LP, which is, in fact, dual to the Standard form:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } &Ax \leq b
\end{aligned}
\tag{LP.Inequality}
$$

- Definition: a **basis** $\mathcal{B}$ is a subset of n (integer) numbers between 1 and m, so that $\text{rank} A_{\mathcal{B}} = n$.
- Note, that we can associate submatrix $A_{\mathcal{B}}$ and corresponding right-hand side $b_{\mathcal{B}}$ with the basis $\mathcal{B}$.
- Also, we can derive a point of intersection of all these hyperplanes from the basis: $x_{\mathcal{B}} = A^{-1}_{\mathcal{B}} b_{\mathcal{B}}$.
- If $Ax_{\mathcal{B}} \leq b$, then basis $\mathcal{B}$ is **feasible**.
- A basis $\mathcal{B}$ is optimal if $x_{\mathcal{B}}$ is an optimum of the LP.Inequality.

The solution of LP if exists lies in the corner

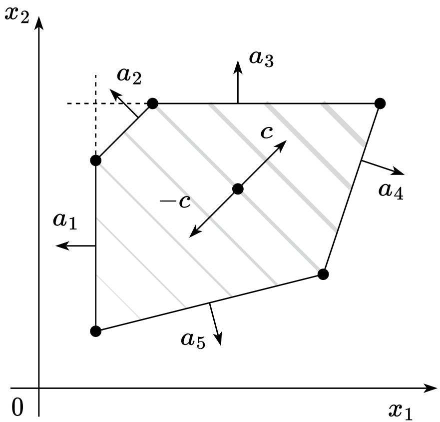


Theorem

1. If Standard LP has a nonempty feasible region, then there is at least one basic feasible point

The high-level idea of the simplex method is following:

The solution of LP if exists lies in the corner

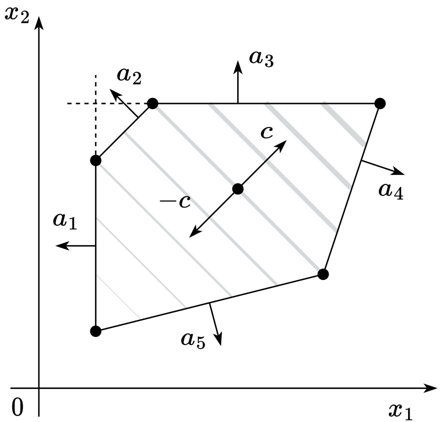


Theorem

1. If Standard LP has a nonempty feasible region, then there is at least one basic feasible point
2. If Standard LP has solutions, then at least one such solution is a basic optimal point.

The high-level idea of the simplex method is following:

The solution of LP if exists lies in the corner

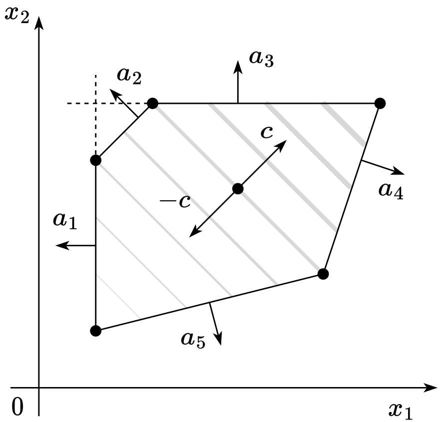


Theorem

1. If Standard LP has a nonempty feasible region, then there is at least one basic feasible point
2. If Standard LP has solutions, then at least one such solution is a basic optimal point.
3. If Standard LP is feasible and bounded, then it has an optimal solution.

The high-level idea of the simplex method is following:

The solution of LP if exists lies in the corner

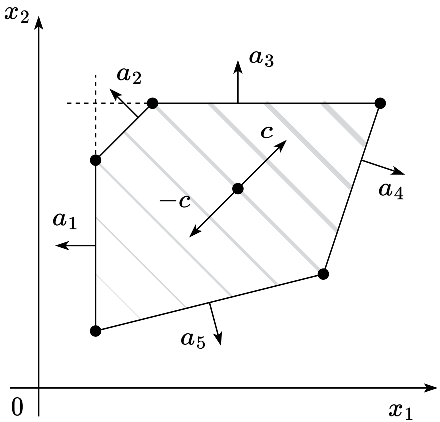


Theorem

1. If Standard LP has a nonempty feasible region, then there is at least one basic feasible point
2. If Standard LP has solutions, then at least one such solution is a basic optimal point.
3. If Standard LP is feasible and bounded, then it has an optimal solution.

The high-level idea of the simplex method is following:

The solution of LP if exists lies in the corner



Theorem

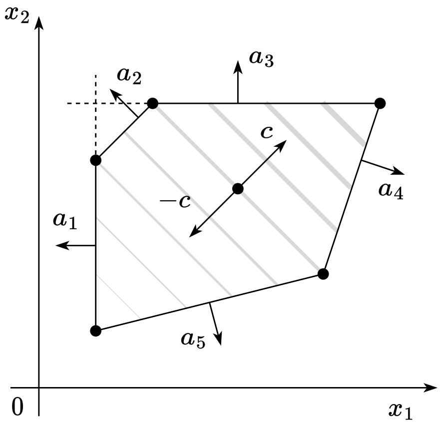
1. If Standard LP has a nonempty feasible region, then there is at least one basic feasible point
2. If Standard LP has solutions, then at least one such solution is a basic optimal point.
3. If Standard LP is feasible and bounded, then it has an optimal solution.

For proof see Numerical Optimization by Jorge Nocedal and Stephen J. Wright theorem 13.2

The high-level idea of the simplex method is following:

- Ensure, that you are in the corner.

The solution of LP if exists lies in the corner



Theorem

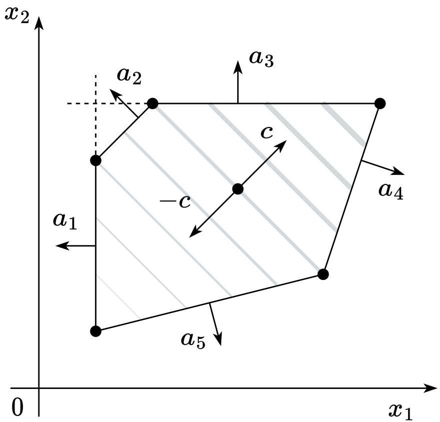
1. If Standard LP has a nonempty feasible region, then there is at least one basic feasible point
2. If Standard LP has solutions, then at least one such solution is a basic optimal point.
3. If Standard LP is feasible and bounded, then it has an optimal solution.

For proof see Numerical Optimization by Jorge Nocedal and Stephen J. Wright theorem 13.2

The high-level idea of the simplex method is following:

- Ensure, that you are in the corner.
- Check optimality.

The solution of LP if exists lies in the corner



Theorem

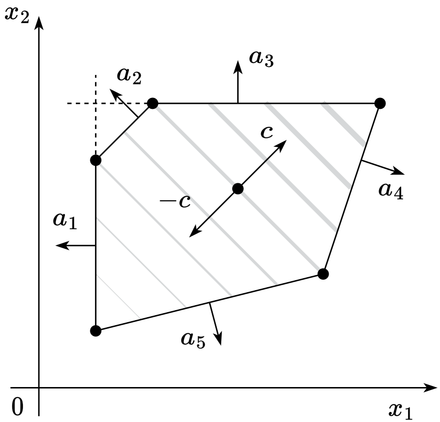
1. If Standard LP has a nonempty feasible region, then there is at least one basic feasible point
2. If Standard LP has solutions, then at least one such solution is a basic optimal point.
3. If Standard LP is feasible and bounded, then it has an optimal solution.

For proof see Numerical Optimization by Jorge Nocedal and Stephen J. Wright theorem 13.2

The high-level idea of the simplex method is following:

- Ensure, that you are in the corner.
- Check optimality.
- If necessary, switch the corner (change the basis).

The solution of LP if exists lies in the corner



Theorem

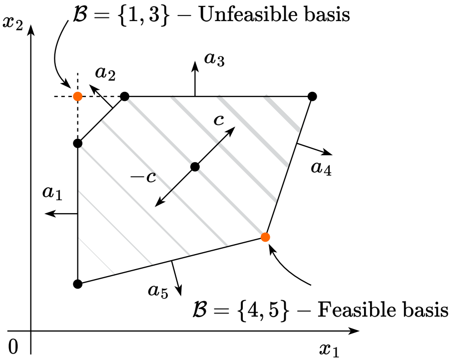
1. If Standard LP has a nonempty feasible region, then there is at least one basic feasible point
2. If Standard LP has solutions, then at least one such solution is a basic optimal point.
3. If Standard LP is feasible and bounded, then it has an optimal solution.

For proof see Numerical Optimization by Jorge Nocedal and Stephen J. Wright theorem 13.2

The high-level idea of the simplex method is following:

- Ensure, that you are in the corner.
- Check optimality.
- If necessary, switch the corner (change the basis).
- Repeat until converge.

Optimal basis



Since we have a basis, we can decompose our objective vector c in this basis and find the scalar coefficients $\lambda_{\mathcal{B}}$:

$$\lambda_{\mathcal{B}}^T A_{\mathcal{B}} = c^T \leftrightarrow \lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$$

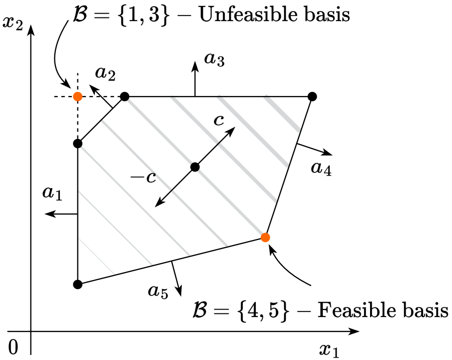
Theorem

If all components of $\lambda_{\mathcal{B}}$ are non-positive and $\mathcal{B}$ is feasible, then $\mathcal{B}$ is optimal.

Proof

$$\exists x^* : Ax^* \leq b, c^T x^* < c^T x_{\mathcal{B}}$$

Optimal basis



Since we have a basis, we can decompose our objective vector c in this basis and find the scalar coefficients $\lambda_{\mathcal{B}}$:

$$\lambda_{\mathcal{B}}^T A_{\mathcal{B}} = c^T \leftrightarrow \lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$$

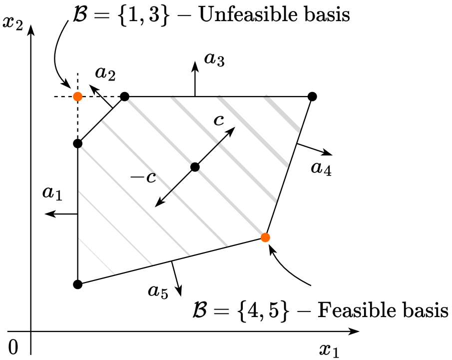
Theorem

If all components of $\lambda_{\mathcal{B}}$ are non-positive and $\mathcal{B}$ is feasible, then $\mathcal{B}$ is optimal.

Proof

$$\exists x^* : Ax^* \leq b, c^T x^* < c^T x_{\mathcal{B}}$$
$$A_{\mathcal{B}} x^* \leq b_{\mathcal{B}}$$

Optimal basis



Since we have a basis, we can decompose our objective vector c in this basis and find the scalar coefficients $\lambda_{\mathcal{B}}$:

$$\lambda^T_{\mathcal{B}} A_{\mathcal{B}} = c^T \leftrightarrow \lambda^T_{\mathcal{B}} = c^T A_{\mathcal{B}}^{-1}$$

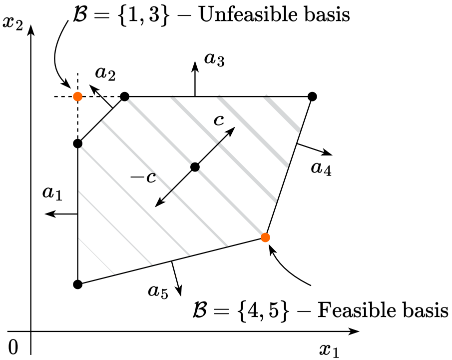
Theorem

If all components of $\lambda_{\mathcal{B}}$ are non-positive and $\mathcal{B}$ is feasible, then $\mathcal{B}$ is optimal.

Proof

$$\exists x^* : Ax^* \leq b, c^T x^* < c^T x_{\mathcal{B}}$$
$$A_{\mathcal{B}} x^* \leq b_{\mathcal{B}}$$
$$\lambda^T_{\mathcal{B}} A_{\mathcal{B}} x^* \geq \lambda^T_{\mathcal{B}} b_{\mathcal{B}}$$

Optimal basis



Since we have a basis, we can decompose our objective vector c in this basis and find the scalar coefficients $\lambda_{\mathcal{B}}$:

$$\lambda_{\mathcal{B}}^T A_{\mathcal{B}} = c^T \leftrightarrow \lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$$

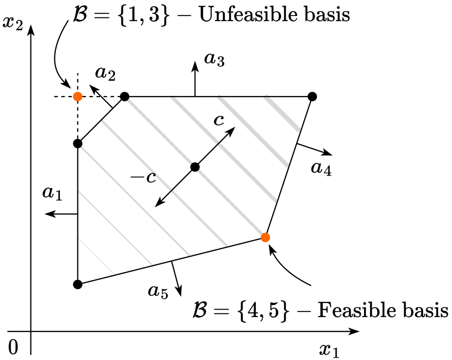
Theorem

If all components of $\lambda_{\mathcal{B}}$ are non-positive and $\mathcal{B}$ is feasible, then $\mathcal{B}$ is optimal.

Proof

$$\begin{aligned} \exists x^* : Ax^* \leq b, c^T x^* < c^T x_{\mathcal{B}} \\ A_{\mathcal{B}} x^* \leq b_{\mathcal{B}} \\ \lambda_{\mathcal{B}}^T A_{\mathcal{B}} x^* \geq \lambda_{\mathcal{B}}^T b_{\mathcal{B}} \\ c^T x^* \geq \lambda_{\mathcal{B}}^T A_{\mathcal{B}} x_{\mathcal{B}} \end{aligned}$$

Optimal basis



Since we have a basis, we can decompose our objective vector c in this basis and find the scalar coefficients $\lambda_\mathcal{B}$:

$$\lambda_\mathcal{B}^T A_\mathcal{B} = c^T \leftrightarrow \lambda_\mathcal{B}^T = c^T A_\mathcal{B}^{-1}$$

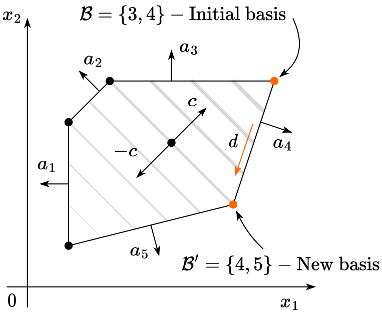
Theorem

If all components of $\lambda_\mathcal{B}$ are non-positive and $\mathcal{B}$ is feasible, then $\mathcal{B}$ is optimal.

Proof

$$\begin{aligned}
\exists x^* : Ax^* &\leq b, c^T x^* < c^T x_\mathcal{B} \\
A_\mathcal{B} x^* &\leq b_\mathcal{B} \\
\lambda_\mathcal{B}^T A_\mathcal{B} x^* &\geq \lambda_\mathcal{B}^T b_\mathcal{B} \\
c^T x^* &\geq \lambda_\mathcal{B}^T A_\mathcal{B} x_\mathcal{B} \\
c^T x^* &\geq c^T x_\mathcal{B}
\end{aligned}$$

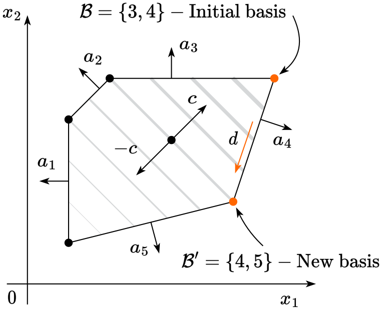
Changing basis



Suppose, some of the coefficients of $\lambda_{\mathcal{B}}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$

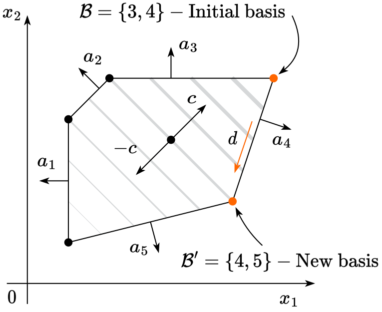
Changing basis



Suppose, some of the coefficients of $\lambda_{\mathcal{B}}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda^T_{\mathcal{B}} = c^T A_{\mathcal{B}}^{-1}$
- Let's assume, that $\lambda^k_{\mathcal{B}} > 0$. We'd like to drop k from the basis and form a new one:

Changing basis

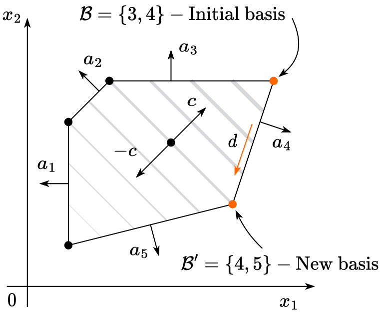


Suppose, some of the coefficients of $\lambda_{\mathcal{B}}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$
- Let's assume, that $\lambda_{\mathcal{B}}^k > 0$. We'd like to drop k from the basis and form a new one:

$$\begin{cases} A_{\mathcal{B} \setminus \{k\}} d = 0 \\ a_k^T d = -1 \end{cases}$$

Changing basis

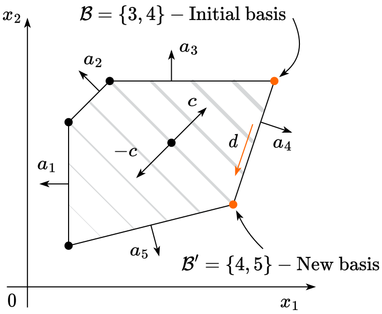


Suppose, some of the coefficients of $\lambda_{\mathcal{B}}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$
- Let's assume, that $\lambda_{\mathcal{B}}^k > 0$. We'd like to drop k from the basis and form a new one:

$$\begin{cases} A_{\mathcal{B}\setminus\{k\}} d = 0 \\ a_k^T d = -1 \end{cases} \qquad c^T d$$

Changing basis

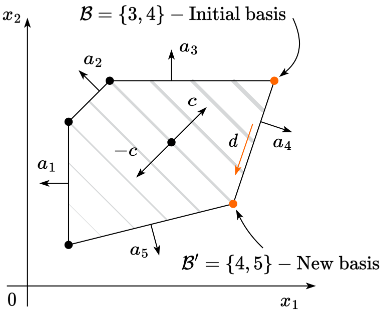


Suppose, some of the coefficients of $\lambda_{\mathcal{B}}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$
- Let's assume, that $\lambda_{\mathcal{B}}^k > 0$. We'd like to drop k from the basis and form a new one:

$$\begin{cases} A_{\mathcal{B}\setminus\{k\}} d = 0 \\ a_k^T d = -1 \end{cases} \qquad c^T d = \lambda_{\mathcal{B}}^T A_{\mathcal{B}} d$$

Changing basis

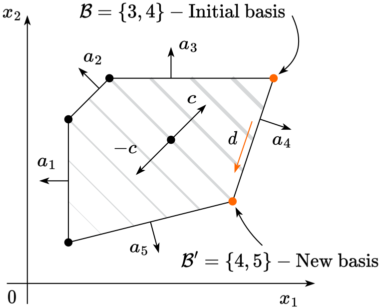


Suppose, some of the coefficients of $\lambda_{\mathcal{B}}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda^T_{\mathcal{B}} = c^T A_{\mathcal{B}}^{-1}$
- Let's assume, that $\lambda^k_{\mathcal{B}} > 0$. We'd like to drop k from the basis and form a new one:

$$\begin{cases} A_{\mathcal{B} \setminus \{k\}} d = 0 \\ a_k^T d = -1 \end{cases} \qquad c^T d = \lambda^T_{\mathcal{B}} A_{\mathcal{B}} d = \sum_{i=1}^{n} \lambda^i_{\mathcal{B}} (A_{\mathcal{B}} d)^i$$

Changing basis

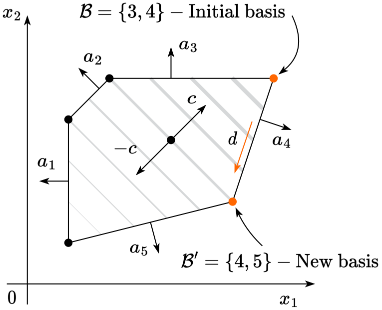


Suppose, some of the coefficients of $\lambda_\mathcal{B}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda_\mathcal{B}^T = c^T A_\mathcal{B}^{-1}$
- Let's assume, that $\lambda_\mathcal{B}^k > 0$. We'd like to drop k from the basis and form a new one:

$$\begin{cases} A_{\mathcal{B} \setminus \{k\}} d = 0 \\ a_k^T d = -1 \end{cases} \qquad c^T d = \lambda_\mathcal{B}^T A_\mathcal{B} d = \sum_{i=1}^n \lambda_\mathcal{B}^i (A_\mathcal{B} d)^i = -\lambda_\mathcal{B}^k < 0$$

Changing basis



Suppose, some of the coefficients of $\lambda_\mathcal{B}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

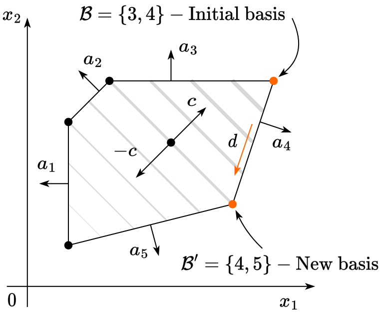
- Suppose, we have a basis $\mathcal{B}$: $\lambda^T_\mathcal{B} = c^T A_\mathcal{B}^{-1}$
- Let's assume, that $\lambda^k_\mathcal{B} > 0$. We'd like to drop k from the basis and form a new one:

$$\begin{cases} A_{\mathcal{B} \setminus \{k\}} d = 0 \\ a_k^T d = -1 \end{cases} \qquad c^T d = \lambda^T_\mathcal{B} A_\mathcal{B} d = \sum_{i=1}^n \lambda^i_\mathcal{B} (A_\mathcal{B} d)^i = -\lambda^k_\mathcal{B} < 0$$

- For all $j \notin \mathcal{B}$ calculate the projection stepsize:

$$\mu_j = \frac{b_j - a_j^T x_\mathcal{B}}{a_j^T d}$$

Changing basis



Suppose, some of the coefficients of $\lambda_{\mathcal{B}}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$
- Let's assume, that $\lambda_{\mathcal{B}}^k > 0$. We'd like to drop k from the basis and form a new one:

$$\begin{cases} A_{\mathcal{B}\setminus\{k\}} d = 0 \\ a_k^T d = -1 \end{cases} \qquad c^T d = \lambda_{\mathcal{B}}^T A_{\mathcal{B}} d = \sum_{i=1}^{n} \lambda_{\mathcal{B}}^i (A_{\mathcal{B}} d)^i = -\lambda_{\mathcal{B}}^k < 0$$

- For all $j \notin \mathcal{B}$ calculate the projection stepsize:

$$\mu_j = \frac{b_j - a_j^T x_{\mathcal{B}}}{a_j^T d}$$

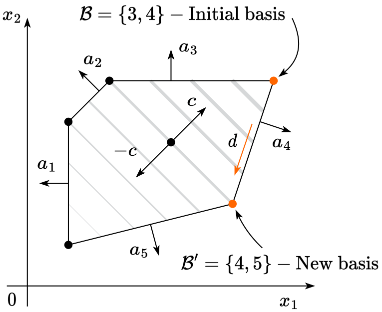
- Define the new vertex, that you will add to the new basis:

$$t = \text{arg}\min_j \{\mu_j \mid \mu_j > 0\}$$

$$\mathcal{B}' = \mathcal{B}\setminus\{k\} \cup \{t\}$$

$$x_{\mathcal{B}'} = x_{\mathcal{B}} + \mu_t d = A_{\mathcal{B}'}^{-1} b_{\mathcal{B}'}$$

Changing basis



Suppose, some of the coefficients of $\lambda_{\mathcal{B}}$ are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

- Suppose, we have a basis $\mathcal{B}$: $\lambda_{\mathcal{B}}^T = c^T A_{\mathcal{B}}^{-1}$
- Let's assume, that $\lambda_{\mathcal{B}}^k > 0$. We'd like to drop k from the basis and form a new one:

$$\begin{cases} A_{\mathcal{B} \setminus \{k\}} d = 0 \\ a_k^T d = -1 \end{cases} \qquad c^T d = \lambda_{\mathcal{B}}^T A_{\mathcal{B}} d = \sum_{i=1}^n \lambda_{\mathcal{B}}^i (A_{\mathcal{B}} d)^i = -\lambda_{\mathcal{B}}^k < 0$$

- For all $j \notin \mathcal{B}$ calculate the projection stepsize:

$$\mu_j = \frac{b_j - a_j^T x_{\mathcal{B}}}{a_j^T d}$$

- Define the new vertex, that you will add to the new basis:

$$t = \text{arg}\min_j \{\mu_j \mid \mu_j > 0\}$$

$$\mathcal{B}' = \mathcal{B} \setminus \{k\} \cup \{t\}$$

$$x_{\mathcal{B}'} = x_{\mathcal{B}} + \mu_t d = A_{\mathcal{B}'}^{-1} b_{\mathcal{B}'}$$

- Note, that changing basis implies objective function decreasing

$$c^T x_{\mathcal{B}'} = c^T (x_{\mathcal{B}} + \mu_t d) = c^T x_{\mathcal{B}} + \mu_t c^T d$$

Finding an initial basic feasible solution

We aim to solve the following problem:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
&\text{s.t. } Ax \leq b
\end{aligned}
\tag{3}
$$

The proposed algorithm requires an initial basic feasible solution and corresponding basis.

Finding an initial basic feasible solution

We aim to solve the following problem:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax \leq b
\end{aligned}
\tag{3}
$$

The proposed algorithm requires an initial basic feasible solution and corresponding basis.

We start by reformulating the problem:

$$
\begin{aligned}
&\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} c^\top (y - z) \\
\text{s.t. } & Ay - Az \leq b \\
& y \geq 0, z \geq 0
\end{aligned}
\tag{4}
$$

Finding an initial basic feasible solution

We aim to solve the following problem:

$$\begin{aligned}
&\min_{x \in \mathbb{R}^n} c^\top x \\
&\text{s.t. } Ax \leq b
\end{aligned} \tag{3}$$

The proposed algorithm requires an initial basic feasible solution and corresponding basis.

We start by reformulating the problem:

$$\begin{aligned}
&\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} c^\top (y - z) \\
&\text{s.t. } Ay - Az \leq b \\
&y \geq 0, z \geq 0
\end{aligned} \tag{4}$$

Given the solution of Problem 4 the solution of Problem 3 can be recovered and vice versa

$$x = y - z \qquad \Leftrightarrow \qquad y_i = \max(x_i, 0), \quad z_i = \max(-x_i, 0)$$

Now we will try to formulate new LP problem, which solution will be basic feasible point for Problem 4. Which means, that we firstly run Simplex algorithm for Phase-1 problem and run Phase-2 problem with known starting point. Note, that basic feasible solution for Phase-1 should be somehow easily established.

Finding an initial basic feasible solution

$$
\begin{aligned}
\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} \; & c^\top (y - z) \\
\text{s.t.} \; & Ay - Az \leq b \\
& y \geq 0, z \geq 0
\end{aligned}
\qquad \text{(Phase-2 (Main LP))}
$$

Finding an initial basic feasible solution

$$
\begin{aligned}
\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} \; & c^\top (y - z) \\
\text{s.t. } & Ay - Az \leq b \\
& y \geq 0, z \geq 0
\end{aligned}
\tag{Phase-2 (Main LP)}
$$

$$
\begin{aligned}
\min_{\xi \in \mathbb{R}^m, y \in \mathbb{R}^n, z \in \mathbb{R}^n} \; & \sum_{i=1}^m \xi_i \\
\text{s.t. } & Ay - Az \leq b + \xi \\
& y \geq 0, z \geq 0, \xi \geq 0
\end{aligned}
\tag{Phase-1}
$$

Finding an initial basic feasible solution

$$
\begin{aligned}
\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} \; & c^\top (y - z) \\
\text{s.t. } & Ay - Az \leq b \\
& y \geq 0, z \geq 0
\end{aligned}
\tag{Phase-2 (Main LP)}
$$

$$
\begin{aligned}
\min_{\xi \in \mathbb{R}^m, y \in \mathbb{R}^n, z \in \mathbb{R}^n} \; & \sum_{i=1}^m \xi_i \\
\text{s.t. } & Ay - Az \leq b + \xi \\
& y \geq 0, z \geq 0, \xi \geq 0
\end{aligned}
\tag{Phase-1}
$$

- If Phase-2 (Main LP) problem has a feasible solution, then Phase-1 optimum is zero (i.e. all slacks ξ_i are zero).
  Proof: trivial check.

Finding an initial basic feasible solution

$$
\begin{aligned}
&\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} c^\top (y - z) \\
\text{s.t. } & Ay - Az \leq b \\
& y \geq 0, z \geq 0
\end{aligned}
\tag{Phase-2 (Main LP)}
$$

$$
\begin{aligned}
&\min_{\xi \in \mathbb{R}^m, y \in \mathbb{R}^n, z \in \mathbb{R}^n} \sum_{i=1}^m \xi_i \\
\text{s.t. } & Ay - Az \leq b + \xi \\
& y \geq 0, z \geq 0, \xi \geq 0
\end{aligned}
\tag{Phase-1}
$$

- If Phase-2 (Main LP) problem has a feasible solution, then Phase-1 optimum is zero (i.e. all slacks ξ_i are zero).
  Proof: trivial check.
- If Phase-1 optimum is zero (i.e. all slacks ξ_i are zero), then we get a feasible basis for Phase-2.
  Proof: trivial check.

Finding an initial basic feasible solution

$$
\begin{aligned}
&\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} c^\top (y - z) \\
\text{s.t. } & Ay - Az \leq b \\
& y \geq 0, z \geq 0
\end{aligned}
\tag{Phase-2 (Main LP)}
$$

$$
\begin{aligned}
&\min_{\xi \in \mathbb{R}^m, y \in \mathbb{R}^n, z \in \mathbb{R}^n} \sum_{i=1}^{m} \xi_i \\
\text{s.t. } & Ay - Az \leq b + \xi \\
& y \geq 0, z \geq 0, \xi \geq 0
\end{aligned}
\tag{Phase-1}
$$

- If Phase-2 (Main LP) problem has a feasible solution, then Phase-1 optimum is zero (i.e. all slacks ξ_i are zero).
  Proof: trivial check.
- If Phase-1 optimum is zero (i.e. all slacks ξ_i are zero), then we get a feasible basis for Phase-2.
  Proof: trivial check.

Finding an initial basic feasible solution

$$
\begin{aligned}
&\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} c^\top (y - z) \\
\text{s.t. } & Ay - Az \leq b \\
& y \geq 0, z \geq 0
\end{aligned}
\tag{Phase-2 (Main LP)}
$$

$$
\begin{aligned}
&\min_{\xi \in \mathbb{R}^m, y \in \mathbb{R}^n, z \in \mathbb{R}^n} \sum_{i=1}^{m} \xi_i \\
\text{s.t. } & Ay - Az \leq b + \xi \\
& y \geq 0, z \geq 0, \xi \geq 0
\end{aligned}
\tag{Phase-1}
$$

- If Phase-2 (Main LP) problem has a feasible solution, then Phase-1 optimum is zero (i.e. all slacks ξ_i are zero).
  Proof: trivial check.
- If Phase-1 optimum is zero (i.e. all slacks ξ_i are zero), then we get a feasible basis for Phase-2.
  Proof: trivial check.

- Now we know, that if we can solve a Phase-1 problem then we will either find a starting point for the simplex method in the original method (if slacks are zero) or verify that the original problem was infeasible (if slacks are non-zero).

Finding an initial basic feasible solution

$$
\begin{aligned}
&\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^n} c^\top (y - z) \\
\text{s.t. } & Ay - Az \leq b \\
& y \geq 0, z \geq 0
\end{aligned}
\tag{Phase-2 (Main LP)}
$$

$$
\begin{aligned}
&\min_{\xi \in \mathbb{R}^m, y \in \mathbb{R}^n, z \in \mathbb{R}^n} \sum_{i=1}^m \xi_i \\
\text{s.t. } & Ay - Az \leq b + \xi \\
& y \geq 0, z \geq 0, \xi \geq 0
\end{aligned}
\tag{Phase-1}
$$

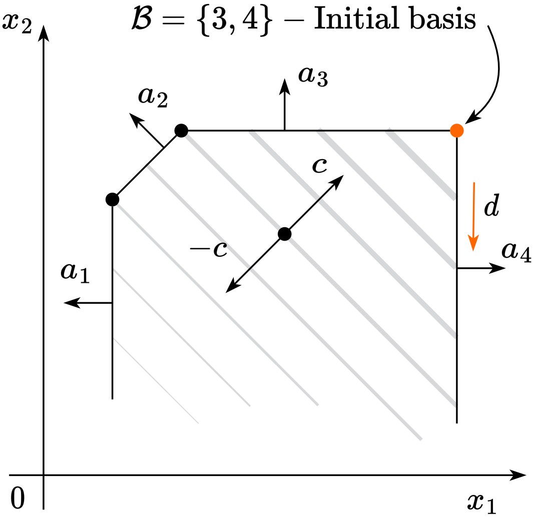
- If Phase-2 (Main LP) problem has a feasible solution, then Phase-1 optimum is zero (i.e. all slacks ξ_i are zero).
  Proof: trivial check.
- If Phase-1 optimum is zero (i.e. all slacks ξ_i are zero), then we get a feasible basis for Phase-2.
  Proof: trivial check.

- Now we know, that if we can solve a Phase-1 problem then we will either find a starting point for the simplex method in the original method (if slacks are zero) or verify that the original problem was infeasible (if slacks are non-zero).
- But how to solve Phase-1? It has basic feasible solution (the problem has $2n + m$ variables and the point below ensures $2n + m$ inequalities are satisfied as equalities (active).)

$$
z = 0 \quad y = 0 \quad \xi_i = \max(0, -b_i)
$$

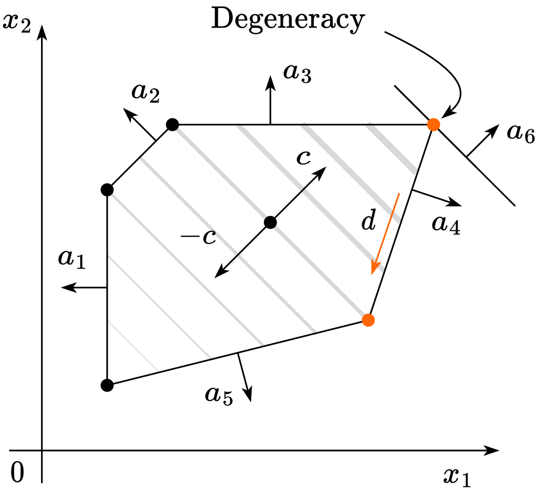
Convergence of the Simplex Algorithm

Unbounded budget set



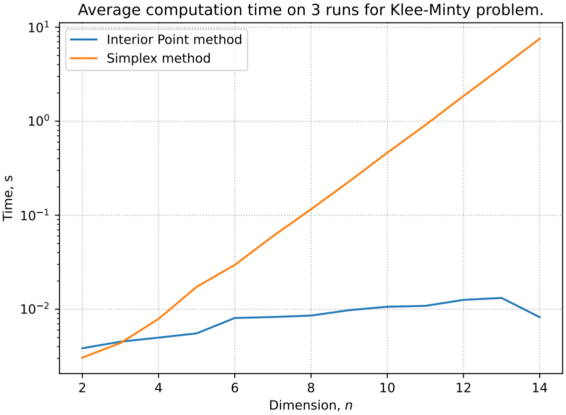
In this case, all μ_j will be negative.

Degeneracy



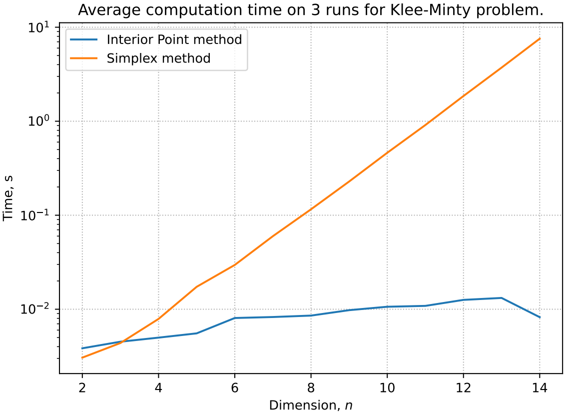
One needs to handle degenerate corners carefully. If no degeneracy exists, one can guarantee a monotonic decrease of the objective function on each iteration.

Exponential convergence



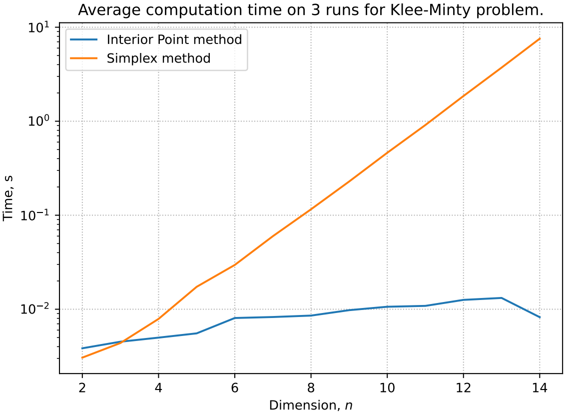
- A wide variety of applications could be formulated as linear programming.

Exponential convergence



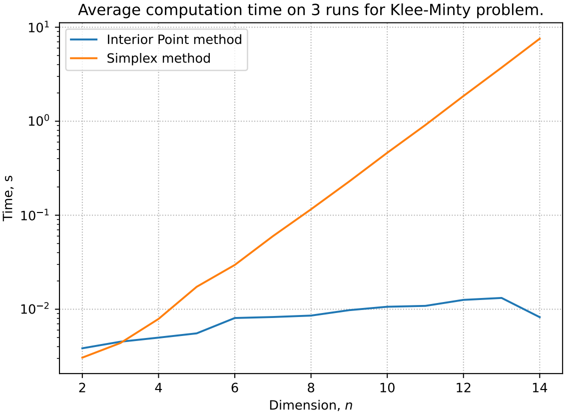
- A wide variety of applications could be formulated as linear programming.
- Simplex algorithm is simple but could work exponentially long.

Exponential convergence



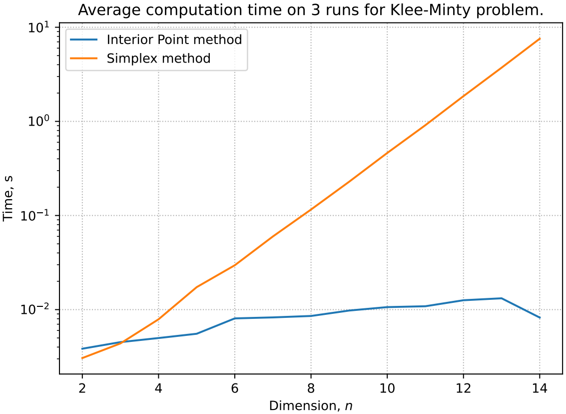
- A wide variety of applications could be formulated as linear programming.
- Simplex algorithm is simple but could work exponentially long.
- Khachiyan's ellipsoid method (1979) is the first to be proven to run at polynomial complexity for LPs. However, it is usually slower than simplex in real problems.

Exponential convergence



- A wide variety of applications could be formulated as linear programming.
- Simplex algorithm is simple but could work exponentially long.
- Khachiyan's ellipsoid method (1979) is the first to be proven to run at polynomial complexity for LPs. However, it is usually slower than simplex in real problems.
- Major breakthrough - Narendra Karmarkar's method for solving LP (1984) using interior point method.

Exponential convergence



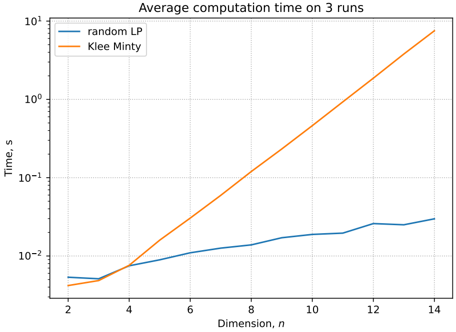
- A wide variety of applications could be formulated as linear programming.
- Simplex algorithm is simple but could work exponentially long.
- Khachiyan's ellipsoid method (1979) is the first to be proven to run at polynomial complexity for LPs. However, it is usually slower than simplex in real problems.
- Major breakthrough - Narendra Karmarkar's method for solving LP (1984) using interior point method.
- Interior point methods are the last word in this area. However, good implementations of simplex-based methods and interior point methods are similar for routine applications of linear programming.

Klee Minty example

Since the number of edge points is finite, the algorithm should converge (except for some degenerate cases, which are not covered here). However, the convergence could be exponentially slow, due to the high number of edges. There is the following iconic example when the simplex algorithm should perform exactly all vertexes.

In the following problem, the simplex algorithm needs to check $2^n - 1$ vertexes with $x_0 = 0$.

$$
\begin{aligned}
&\max_{x \in \mathbb{R}^n} 2^{n-1}x_1 + 2^{n-2}x_2 + \dots + 2x_{n-1} + x_n \\
\text{s.t. } & x_1 \leq 5 \\
& 4x_1 + x_2 \leq 25 \\
& 8x_1 + 4x_2 + x_3 \leq 125 \\
& \ldots \\
& 2^n x_1 + 2^{n-1}x_2 + 2^{n-2}x_3 + \ldots + x_n \leq 5^n \\
& x \geq 0
\end{aligned}
$$



Other

Minimization of convex function as LP

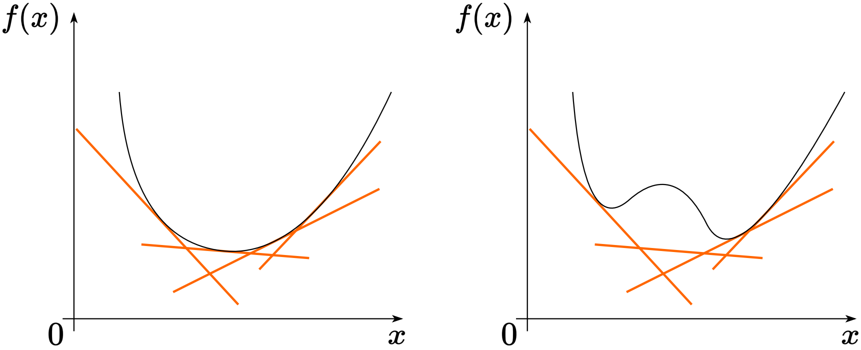


Figure 3: How LP can help with general convex problem

- The function is convex iff it can be represented as a pointwise maximum of linear functions.

Minimization of convex function as LP

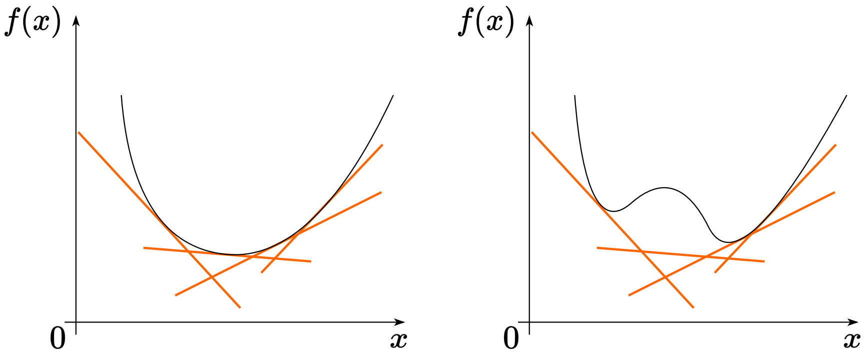


Figure 3: How LP can help with general convex problem

- The function is convex iff it can be represented as a pointwise maximum of linear functions.
- In high dimensions, the approximation may require too many functions.

Minimization of convex function as LP

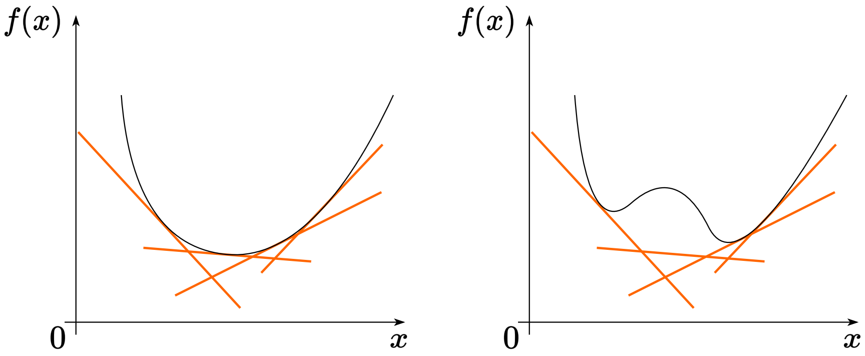


Figure 3: How LP can help with general convex problem

- The function is convex iff it can be represented as a pointwise maximum of linear functions.
- In high dimensions, the approximation may require too many functions.
- More efficient convex optimizers (not reducing to LP) exist.

Mixed Integer Programming

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 &\to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 &\leq 14 \\
x_i \in \{0,1\} \quad &\forall i
\end{aligned}
\tag{5}
$$

Complexity of MIP

Consider the following Mixed Integer Programming (MIP): Relax it to:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1, x_2, x_3, x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in \{0, 1\} \quad \forall i
\end{aligned}
\tag{5}
$$

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1, x_2, x_3, x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in [0, 1] \quad \forall i
\end{aligned}
\tag{6}
$$

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in \{0,1\} \quad \forall i
\end{aligned} \tag{5}
$$

Optimal solution

$$
x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ and } z = 21.
$$

Relax it to:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in [0,1] \quad \forall i
\end{aligned} \tag{6}
$$

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1, x_2, x_3, x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in \{0, 1\} \quad \forall i
\end{aligned}
\tag{5}
$$

Optimal solution

$$
x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ and } z = 21.
$$

Relax it to:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1, x_2, x_3, x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in [0, 1] \quad \forall i
\end{aligned}
\tag{6}
$$

Optimal solution

$$
x_1 = x_2 = 1, x_3 = 0.5, x_4 = 0, \text{ and } z = 22.
$$

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1, x_2, x_3, x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in \{0, 1\} \quad \forall i
\end{aligned}
\tag{5}
$$

Optimal solution

$$
x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ and } z = 21.
$$

Relax it to:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1, x_2, x_3, x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in [0, 1] \quad \forall i
\end{aligned}
\tag{6}
$$

Optimal solution

$$
x_1 = x_2 = 1, x_3 = 0.5, x_4 = 0, \text{ and } z = 22.
$$

- Rounding $x_3 = 0$: gives $z = 19$.

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
& z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1,x_2,x_3,x_4} \\
& \text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
& x_i \in \{0,1\} \quad \forall i
\end{aligned} \tag{5}
$$

Optimal solution

$$
x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ and } z = 21.
$$

Relax it to:

$$
\begin{aligned}
& z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1,x_2,x_3,x_4} \\
& \text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
& x_i \in [0,1] \quad \forall i
\end{aligned} \tag{6}
$$

Optimal solution

$$
x_1 = x_2 = 1, x_3 = 0.5, x_4 = 0, \text{ and } z = 22.
$$

- Rounding $x_3 = 0$: gives $z = 19$.
- Rounding $x_3 = 1$: Infeasible.

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 &\to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 &\leq 14 \\
x_i \in \{0,1\} \quad &\forall i
\end{aligned}
\tag{5}
$$

Optimal solution

$$
x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ and } z = 21.
$$

Relax it to:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 &\to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 &\leq 14 \\
x_i \in [0,1] \quad &\forall i
\end{aligned}
\tag{6}
$$

Optimal solution

$$
x_1 = x_2 = 1, x_3 = 0.5, x_4 = 0, \text{ and } z = 22.
$$

- Rounding $x_3 = 0$: gives $z = 19$.
- Rounding $x_3 = 1$: Infeasible.

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1, x_2, x_3, x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in \{0, 1\} \quad \forall i
\end{aligned} \tag{5}
$$

Optimal solution

$$
x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ and } z = 21.
$$

Relax it to:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1, x_2, x_3, x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in [0, 1] \quad \forall i
\end{aligned} \tag{6}
$$

Optimal solution

$$
x_1 = x_2 = 1, x_3 = 0.5, x_4 = 0, \text{ and } z = 22.
$$

- Rounding $x_3 = 0$: gives $z = 19$.
- Rounding $x_3 = 1$: Infeasible.

! MIP is much harder, than LP

- Naive rounding of LP relaxation of the initial MIP problem might lead to infeasible or suboptimal solution.

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in \{0,1\} \quad \forall i
\end{aligned}
\tag{5}
$$

Optimal solution

$$
x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ and } z = 21.
$$

Relax it to:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 \to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i \in [0,1] \quad \forall i
\end{aligned}
\tag{6}
$$

Optimal solution

$$
x_1 = x_2 = 1, x_3 = 0.5, x_4 = 0, \text{ and } z = 22.
$$

- Rounding $x_3 = 0$: gives $z = 19$.
- Rounding $x_3 = 1$: Infeasible.

! MIP is much harder, than LP

- Naive rounding of LP relaxation of the initial MIP problem might lead to infeasible or suboptimal solution.
- General MIP is NP-hard.

Complexity of MIP

Consider the following Mixed Integer Programming (MIP):

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 &\to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 &\leq 14 \\
x_i \in \{0,1\} \quad &\forall i
\end{aligned} \tag{5}
$$

Optimal solution

$$x_1 = 0, x_2 = x_3 = x_4 = 1, \text{ and } z = 21.$$

Relax it to:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 &\to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 &\leq 14 \\
x_i \in [0,1] \quad &\forall i
\end{aligned} \tag{6}
$$

Optimal solution

$$x_1 = x_2 = 1, x_3 = 0.5, x_4 = 0, \text{ and } z = 22.$$

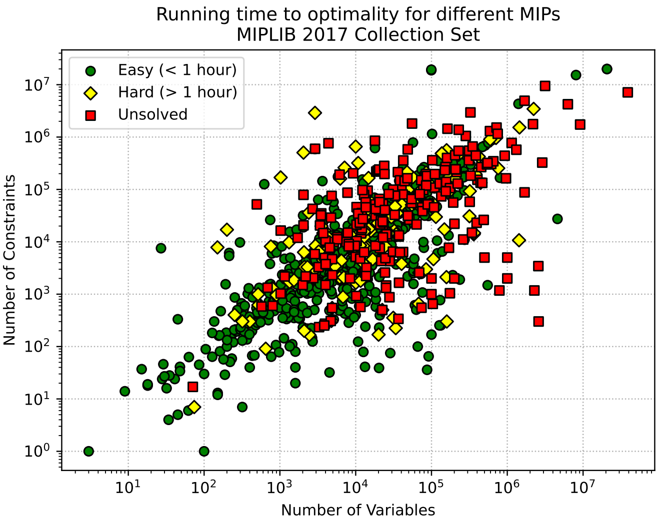
- Rounding $x_3 = 0$: gives $z = 19$.
- Rounding $x_3 = 1$: Infeasible.

! MIP is much harder, than LP

- Naive rounding of LP relaxation of the initial MIP problem might lead to infeasible or suboptimal solution.
- General MIP is NP-hard.
- However, if the coefficient matrix of an MIP is a *totally unimodular matrix*, then it can be solved in polynomial time.

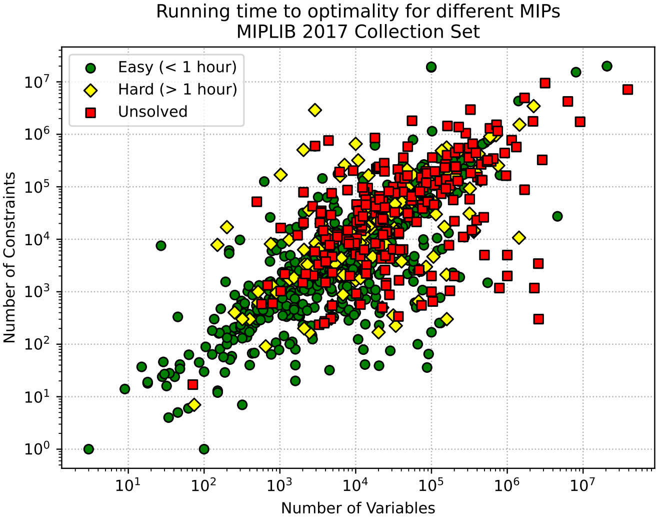
Unpredictable complexity of MIP

- It is hard to predict what will be solved quickly and what will take a long time



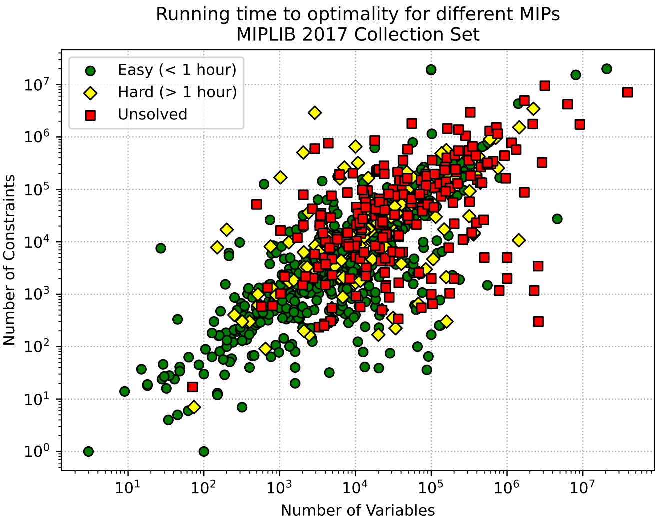
Unpredictable complexity of MIP

- It is hard to predict what will be solved quickly and what will take a long time
- 🔗Dataset



Unpredictable complexity of MIP

- It is hard to predict what will be solved quickly and what will take a long time
- Dataset
- Source code



Hardware progress vs Software progress

What would you choose, assuming, that the question posed correctly (you can compile software for any hardware and the problem is the same for both options)? We will consider the time period from 1992 to 2023.

Hardware

Solving MIP with an old software on the modern hardware

Software

Solving MIP with a modern software on the old hardware

Hardware progress vs Software progress

What would you choose, assuming, that the question posed correctly (you can compile software for any hardware and the problem is the same for both options)? We will consider the time period from 1992 to 2023.

Hardware

Solving MIP with an old software on the modern hardware

$\approx 1.664.510$ x speedup

Moore's law states, that computational power doubles every 18 monthes.

Software

Solving MIP with a modern software on the old hardware

$\approx 2.349.000$ x speedup

R. Bixby conducted an intensive experiment with benchmarking all CPLEX software version starting from 1992 to 2007 and measured overall software progress (29000 times), later (in 2009) he was a cofounder of Gurobi optimization software, which gives additional ≈ 81 speedup on MILP.

[1] R. Bixby report Recent study

Hardware progress vs Software progress

What would you choose, assuming, that the question posed correctly (you can compile software for any hardware and the problem is the same for both options)? We will consider the time period from 1992 to 2023.

Hardware

Solving MIP with an old software on the modern hardware

$\approx 1.664.510$ x speedup

Moore's law states, that computational power doubles every 18 monthes.

Software

Solving MIP with a modern software on the old hardware

$\approx 2.349.000$ x speedup

R. Bixby conducted an intensive experiment with benchmarking all CPLEX software version starting from 1992 to 2007 and measured overall software progress $(29000$ times), later (in 2009) he was a cofounder of Gurobi optimization software, which gives additional ≈ 81 speedup on MILP.

It turns out that if you need to solve a MILP, it is better to use an old computer and modern methods than vice versa, the newest computer and methods of the early 1990s![1]

1 R. Bixby report Recent study

Idea of Branch and Bound method

1. **Initial Relaxation**:

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.
     - In the second subproblem, add the constraint $x_i = 1$.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.
     - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:

Idea of Branch and Bound method

1. **Initial Relaxation**:
    - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
    - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
    - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
    - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
        - In the first subproblem, add the constraint $x_i = 0$.
        - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
    - Solve the LP relaxation of each subproblem.

Idea of Branch and Bound method

1. **Initial Relaxation**:
    - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
    - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
    - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
    - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
        - In the first subproblem, add the constraint $x_i = 0$.
        - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
    - Solve the LP relaxation of each subproblem.
    - Calculate the objective value (upper bound) for each subproblem.

Idea of Branch and Bound method

1. **Initial Relaxation**:
    - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
    - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
    - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
    - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
        - In the first subproblem, add the constraint $x_i = 0$.
        - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
    - Solve the LP relaxation of each subproblem.
    - Calculate the objective value (upper bound) for each subproblem.
    - If the upper bound of a subproblem is less than the best known integer feasible solution, prune that subproblem (do not explore it further).

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.
     - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
   - Solve the LP relaxation of each subproblem.
   - Calculate the objective value (upper bound) for each subproblem.
   - If the upper bound of a subproblem is less than the best known integer feasible solution, prune that subproblem (do not explore it further).
4. **Fathoming**:

Idea of Branch and Bound method

1. **Initial Relaxation**:
    - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
    - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
    - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
    - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
        - In the first subproblem, add the constraint $x_i = 0$.
        - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
    - Solve the LP relaxation of each subproblem.
    - Calculate the objective value (upper bound) for each subproblem.
    - If the upper bound of a subproblem is less than the best known integer feasible solution, prune that subproblem (do not explore it further).
4. **Fathoming**:
    - If a subproblem yields an integer solution that is better than the best known solution, update the best known solution.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.
     - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
   - Solve the LP relaxation of each subproblem.
   - Calculate the objective value (upper bound) for each subproblem.
   - If the upper bound of a subproblem is less than the best known integer feasible solution, prune that subproblem (do not explore it further).
4. **Fathoming**:
   - If a subproblem yields an integer solution that is better than the best known solution, update the best known solution.
   - If a subproblem's upper bound is less than or equal to the best known solution, prune it.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.
     - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
   - Solve the LP relaxation of each subproblem.
   - Calculate the objective value (upper bound) for each subproblem.
   - If the upper bound of a subproblem is less than the best known integer feasible solution, prune that subproblem (do not explore it further).
4. **Fathoming**:
   - If a subproblem yields an integer solution that is better than the best known solution, update the best known solution.
   - If a subproblem's upper bound is less than or equal to the best known solution, prune it.
   - If a subproblem is infeasible, prune it.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.
     - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
   - Solve the LP relaxation of each subproblem.
   - Calculate the objective value (upper bound) for each subproblem.
   - If the upper bound of a subproblem is less than the best known integer feasible solution, prune that subproblem (do not explore it further).
4. **Fathoming**:
   - If a subproblem yields an integer solution that is better than the best known solution, update the best known solution.
   - If a subproblem's upper bound is less than or equal to the best known solution, prune it.
   - If a subproblem is infeasible, prune it.
5. **Iteration**:

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.
     - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
   - Solve the LP relaxation of each subproblem.
   - Calculate the objective value (upper bound) for each subproblem.
   - If the upper bound of a subproblem is less than the best known integer feasible solution, prune that subproblem (do not explore it further).
4. **Fathoming**:
   - If a subproblem yields an integer solution that is better than the best known solution, update the best known solution.
   - If a subproblem's upper bound is less than or equal to the best known solution, prune it.
   - If a subproblem is infeasible, prune it.
5. **Iteration**:
   - Repeat the branching, bounding, and fathoming steps until all subproblems are either pruned or solved to integer optimality.

Idea of Branch and Bound method

1. **Initial Relaxation**:
   - Solve the linear programming (LP) relaxation of the original MIP by relaxing the integer constraints $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$.
   - This gives an upper bound on the optimal value of the MIP since the feasible region of the LP relaxation contains the feasible region of the original MIP.
2. **Branching**:
   - If the solution to the LP relaxation is integer (i.e., $x_i \in \{0, 1\}$ for all i), then it is the optimal solution to the MIP.
   - If the solution is not integer, select a variable x_i that is fractional (e.g., $x_i = 0.5$) and create two subproblems (branches):
     - In the first subproblem, add the constraint $x_i = 0$.
     - In the second subproblem, add the constraint $x_i = 1$.
3. **Bounding**:
   - Solve the LP relaxation of each subproblem.
   - Calculate the objective value (upper bound) for each subproblem.
   - If the upper bound of a subproblem is less than the best known integer feasible solution, prune that subproblem (do not explore it further).
4. **Fathoming**:
   - If a subproblem yields an integer solution that is better than the best known solution, update the best known solution.
   - If a subproblem's upper bound is less than or equal to the best known solution, prune it.
   - If a subproblem is infeasible, prune it.
5. **Iteration**:
   - Repeat the branching, bounding, and fathoming steps until all subproblems are either pruned or solved to integer optimality.
   - The best known integer solution at the end of the process is the optimal solution to the original MIP.

MIP Example

Consider the following MIP:

$$
\begin{aligned}
z = 8x_1 + 11x_2 + 6x_3 + 4x_4 &\to \max_{x_1,x_2,x_3,x_4} \\
\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 &\leq 14 \\
x_i \in \{0,1\} \quad &\forall i
\end{aligned}
$$