Unit 1

1 Optimization problems

An optimization problem consists in maximizing or minimizing some **objective function** relative to some **feasible set**, representing a range of choices for the variables available in a certain situation. The objective function allows comparison of the different variables choices for determining which might be best.

1.1 Structure of an optimization problem

The **general form** of an optimization problem is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_j(x) \le b_j, \quad j = 1, \dots, m \\ & x \in \Omega, \end{aligned}$$

where

- $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be minimized over the n-variable vector x,
- $g_j(x) \le b_j$ are called inequality constraints, with $g_j : \mathbb{R}^n \to \mathbb{R}$, $b_j \in \mathbb{R}$, and $m \ge 0$,
- Ω is the feasible space for x, for example it can be equal to $\mathbb{R}^n$ or $\mathbb{Z}^n$.

The feasible set of the optimization problem is

$$\mathcal{S} \triangleq \{x \in \Omega \,:\, g_j(x) \le b_j,\, j = 1, \dots, m\}.$$

1.2 Feasibility and optimality

- $\overline{x}$ is said to be an **infeasible point** if $\overline{x} \notin \mathcal{S}$, that is, a $\overline{j} \in \{1, \dots, m\}$ exists such that $g_{\overline{j}}(\overline{x}) > b_j$ or $\overline{x} \notin \Omega$.
- $\overline{x}$ is said to be a **feasible point** if $\overline{x} \in \mathcal{S}$, that is, $g_j(\overline{x}) \le b_j$, for all $j = 1, \dots, m$, and $\overline{x} \in \Omega$.
- $\overline{x}$ is said to be a (global) **optimal point** if $\overline{x} \in \mathcal{S}$ (it is feasible) and

$$f(\overline{x}) \le f(x), \quad \forall\, x \in \mathcal{S}.$$

- If $\mathcal{S} = \emptyset$ (it is empty), then we have an **infeasible problem**, that is a problem which admits neither feasible nor optimal points.

- If for any $M > 0$ a point $\overline{x} \in \mathcal{S}$ exists such that $f(\overline{x}) \leq -M$, then we have an **unbounded problem**; such a problem does not admit optimal points.
- Let the problem admit an optimal point $\overline{x}$; we say that $\overline{f} = f(\overline{x})$ is the **optimal value** of the problem (it is unique).

1.3 Constraints

Given a point $\overline{x} \in \Omega$, the jth inequality constraint is:

- a **violated constraint** at $\overline{x}$ if $g_j(\overline{x}) > b_j$;
- a **satisfied constraint** at $\overline{x}$ if $g_j(\overline{x}) \leq b_j$;
- an **active constraint** at $\overline{x}$ if $g_j(\overline{x}) = b_j$;
- a **redundant constraint** at $\overline{x}$ if $g_j(\overline{x}) < b_j$.

If the jth inequality constraint is redundant at any feasible point then it is a redundant constraint and it can be removed from the problem.

1.4 Recast a problem in its general form

Let us consider the problem

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & r(x) \\ \text{subject to} \quad & s_1(x) \leq t_1 \\ & s_2(x) \geq t_2 \\ & s_3(x) = t_3 \\ & x \in \Omega. \end{aligned}$$

It has the same optimal points of the following problem in general form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & -r(x) \\ \text{subject to} \quad & s_1(x) \leq t_1 \\ & -s_2(x) \leq -t_2 \\ & s_3(x) \leq t_3 \\ & -s_3(x) \leq -t_3 \\ & x \in \Omega. \end{aligned}$$

Practical rules:

- maximizing r is equivalent to minimizing $f = -r$;
- the constraint $s_j(x) \geq t_j$ is equivalent to $g_j(x) \leq b_j$ with $g_j = -s_j$ and $b_j = -t_j$;
- the constraint $s_j(x) = t_j$ is equivalent to

$$\begin{cases} g_j(x) \leq b_j \\ g_{j+1}(x) \leq b_{j+1} \end{cases}$$

  with $g_j = s_j$, $b_j = t_j$, $g_{j+1} = -s_j$, and $b_{j+1} = -t_j$.

1.5 Linear functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is a **linear function** if it is defined as

$$f(x) = c^T x = \sum_{i=1}^{n} c_i x_i.$$

Proposition 1.1 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is a linear function if and only if*

(i) $f(x+y) = f(x) + f(y)$ for all $x, y \in \mathbb{R}^n$;

(ii) $f(\lambda x) = \lambda f(x)$ for all $x \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$.

Proof. $(\Rightarrow)$ Let f be linear. For all $x, y \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ we get

(i) $f(x+y) = \displaystyle\sum_{i=1}^{n} c_i(x_i + y_i) = \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} c_i y_i = f(x) + f(y)$;

(ii) $f(\lambda x) = \displaystyle\sum_{i=1}^{n} c_i(\lambda x_i) = \lambda \sum_{i=1}^{n} c_i x_i = \lambda f(x)$.

Therefore properties (i) and (ii) hold.

$(\Leftarrow)$ Let properties (i) and (ii) hold. Let $\{e^1, \ldots, e^n\}$ be the canonic base of $\mathbb{R}^n$. We get

$$f(x) = f\left(\sum_{i=1}^{n} x_i e^i\right) \stackrel{(i)}{=} \sum_{i=1}^{n} f\left(x_i e^i\right) \stackrel{(ii)}{=} \sum_{i=1}^{n} x_i f\left(e^i\right) = \sum_{i=1}^{n} c_i x_i,$$

where $c_i = f\left(e^i\right)$, $i = 1, \ldots, n$. Therefore f is linear. $\square$

1.6 Classify an optimization problem

An optimization problem is a

- **Linear Program** (LP) if f and all g_j are linear functions and $\Omega = \mathbb{R}^n$;
- **Integer Linear Program** (ILP) if f and all g_j are linear functions and $\Omega = \mathbb{Z}^n$;
- **Mixed-Integer Linear Program** (MILP) if f and all g_j are linear functions and $\Omega = \{x \in \mathbb{R}^n : x_i \in \mathbb{Z}, 0 \leq i \leq \overline{n} \leq n\}$;
- **NonLinear Program** (NLP) if $\Omega = \mathbb{R}^n$;
- **Mixed-Integer NonLinear Program** (MINLP) if $\Omega = \{x \in \mathbb{R}^n : x_i \in \mathbb{Z}, 0 \leq i \leq \overline{n} \leq n\}$.

2 Product mix with concurrent resources

The product mix problem occurs where it is possible to manufacture a variety of products ($P1, \ldots, Pn$). Any product Pi has a certain margin of utility per unit p_i (e.g. a profit per unit), and uses a common pool of limited **concurrent resources** ($R1, \ldots, Rm$). In this case the decision problem consists in identifying for each product Pi the quantity q_i which will maximize the utility subject to the availability of limited resource constraints.

Suppose for example that the products require processing in m types of machines. In this case the resource Rj is the available jth machine hours per day b_j. While the hours required on the jth machine to produce one unit of the product Pi is a_{ij}.

Finally any quantity q_i must be in the **box** $[l_i, u_i]$, with $l_i \geq 0$.

This decision problem can be modeled as an LP:

$$\begin{aligned}
\underset{q}{\text{maximize}} \quad & \sum_{i=1}^{n} p_i q_i \\
\text{subject to} \quad & \sum_{i=1}^{n} a_{ij} q_i \leq b_j, \quad j = 1, \ldots, m \\
& l_i \leq q_i \leq u_i, \quad i = 1, \ldots, n \\
& q \in \mathbb{R}^n.
\end{aligned}$$

Let us consider a particular instance of the product mix problem:

	p	l	u	$R1$	$R2$	$R3$	
b				480	480	300	
$P1$	1000	0	100	20	31	16	
$P2$	1500	0	100	30	42	81	
$P3$	2200	0	100	62	51	10	
							a

The resulting model is the following:

$$\begin{aligned}
\underset{q}{\text{maximize}} \quad & 1000q_1 + 1500q_2 + 2200q_3 \\
\text{subject to} \quad & 20q_1 + 30q_2 + 62q_3 \leq 8 \\
& 31q_1 + 42q_2 + 51q_3 \leq 8 \\
& 16q_1 + 81q_2 + 10q_3 \leq 5 \\
& 0 \leq q_i \leq 100, \quad i = 1, \ldots, n.
\end{aligned}$$

We can compute an optimal point by employing the GNU Linear Programming Kit (GLPK). To do this, we need to translate the problem in GNU MathProg. Specifically, the model can be defined in a **mod file** named `product_mix1.mod`:

```
set P;
set R;

param p{P};
param a{P,R};
param b{R};
param l{P};
param u{P};

var q{i in P} >= l[i], <= u[i];

maximize utility:
  sum{i in P} p[i]*q[i];

subject to concurrent_resources {j in R}:
  sum{i in P} a[i,j]*q[i] <= b[j];

end;
```

And the data can be specified in a **dat file** named `product_mix1.dat`:

```
set P := P1 P2 P3;
set R := R1 R2 R3;

param: p l u :=
P1 1000 0 100
P2 1500 0 100
P3 2200 0 100;
```

```
param a: R1 R2 R3 :=
P1 20 31 16
P2 30 42 81
P3 62 51 10;

param: b :=
R1 480
R2 480
R3 300;
```

Now, we run `glpsol` in our console:

```
glpsol -m product_mix1.mod -d product_mix1.dat -o output.txt
```

We get the output file named `output.txt`:

```
Problem:    product_mix1
Rows:       4
Columns:    3
Non-zeros:  12
Status:     OPTIMAL
Objective:  utility = 18949.4707 (MAXimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 utility      B        18949.5
     2 concurrent_resources[R1]
                    NU           480                         480       19.8952
     3 concurrent_resources[R2]
                    NU           480                         480        18.662
     4 concurrent_resources[R3]
                    NU           300                         300       1.47332

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 q[P1]        B        2.96628             0           100
     2 q[P2]        B        2.42497             0           100
     3 q[P3]        B         5.6117             0           100

Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err = 1.14e-13 on row 2
        max.rel.err = 1.18e-16 on row 2
        High quality

KKT.PB: max.abs.err = 0.00e+00 on row 0
```

```
        max.rel.err = 0.00e+00 on row 0
        High quality

KKT.DE: max.abs.err = 4.55e-13 on column 3
        max.rel.err = 1.03e-16 on column 3
        High quality

KKT.DB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

End of output
```

The computed optimal point is $q = (2.96628\ 2.42497\ 5.6117)^T$ whose optimal value is 18949.4707.

Exercise: Modify the model by adding some **percentage constraints**. Specifically, add these constraints:

- the quantity of $P1$ cannot be less than 40% of the total produced quantity
- the quantity of $P3$ cannot be more than 20% of the total produced quantity.

Solve it with GLPK.
[Produce the model file `product_mix2.mod` and the data file `product_mix2.dat`.]

3 GNU Linear Programming Kit

The GNU Linear Programming Kit (GLPK) is a software package intended for solving large-scale linear programming (LP), mixed-integer linear programming (MILP), and other related problems. GLPK is written in ANSI C and organized as a callable library. GLPK package is part of the GNU Project and is released under the GNU General Public License (GPL).

GLPK has a default client: the glpsol program that interfaces with the GLPK API. The name "glpsol" comes from GNU Linear Program Solver. Indeed, usually a program like glpsol is called a solver rather than a client, so we shall use this nomenclature from here forward.

To use glpsol we issue on a console the command

```
glpsol -m inputfile.mod -d inputfile.dat -o outputfile.txt
```

The options `-m inputfile.mod -d inputfile.dat` tells the glpsol solver that the problem to be solved is described in the files inputfile.mod (model) and inputfile.dat (data), and the problem is described in the GNU MathProg language. The option `-o outputfile.sol` tells the glpsol solver to print the results (the solution with some sensitivity information) to the file outputfile.sol.

GLPK, like all GNU software, is open source. It is available to all operating systems and platforms you may ever use. This is the reason we use GLPK in this course.