Fundamentals of optimization
Part I

M. Grazia Speranza
University of Brescia
Operations research

Management science

Analytics

Optimization

Simulation
Operations research

Management science

Analytics

Optimization

Simulation

What if analysis
Operations research

Management science

Analytics

Optimization

Simulation
Optimization

What is it?

Why do we care?

When do we use it?
Optimization

\[ \min c(x) \]
\[ x \in F \]

Decision variables

One objective (objective function)

Constraints (feasible region)
Optimization

One objective only?
Optimization

One objective only?

Safety  ‘Low’ cost

Small traveling time

Fair traffic assignment  Environmentally friendly
Optimization

Multiple objectives

a) Combination of objectives into a single one

\[ \min f(x) \quad \max g(x) \]

\[ \max g(x) - p f(x) \]

Method not always appropriate
Optimization

Multiple objectives

b) One objective chosen as objective with the others into constraints with thresholds

\[
\begin{align*}
\min f(x) \\
\max g(x)
\end{align*}
\]

\[
\begin{align*}
\min f(x) \\
g(x) \geq k
\end{align*}
\]
Optimization

a) \[ \max g(x) - p f(x) \]

Choice of p

b) \[ \min f(x) \]
\[ g(x) \geq k \]

Choice of k

Multiple optimization problems, each with one objective

Efficient frontier
Optimization problem

\[ \min c(x) \]
\[ x \in F \]
Types of objectives

\[ \min \sum_i c_i(x) \quad \text{Average} \]

\[ \min \max c_i(x) \quad \text{Worst} \]

\[ \min \sum_i c_i(x) \quad \text{sum over a given percentage of the worst i} \quad \text{CVaR-like} \quad \text{(can be used on scenarios)} \]
Local and global optima

Local optima (max)

Global optimum (max)
What do we need?

• We know how to analytically find local optima for ‘simple’ optimization problems

• We know how to analytically find the global optimum for some ‘very simple’ optimization problems

• We do not know for all the others – **algorithms needed**
Convex problem

\[
\min_{x \in F} c(x)
\]

- \(F\): Convex set
- \(c(x)\): Convex function on \(F\)

Convex problem
Convex problem

**Property:**
For a convex problem, any local optimum is a global optimum

**Effective and efficient algorithms**
Convex problem

Theorem:

If $\varphi(x)$ is convex, $g_i(x)$ are concave, $h_j(x)$ are linear, the optimization problem is a convex problem.

\[
\begin{align*}
\min & \quad \varphi(x) \\
g_i(x) & \geq 0 \quad (i = 1, \ldots, q) \\
h_j(x) & = 0 \quad (j = 1, \ldots, p)
\end{align*}
\]
Linear programming

\[
\begin{align*}
\text{min } c'x \\
Ax &= b \\
x &\geq 0
\end{align*}
\]

Special case of convex problem
Model - algorithm

Mathematical programming formulation

Algorithm
(for the formulation and not for a specific problem)
Linear programming

\[ \min c'x \]
\[ Ax = b \]
\[ x \geq 0 \]

LP problems can be solved with efficient algorithms:
- Simplex method
- Interior point methods (Khachian, Karmarkar)
Computational complexity

Computational complexity or simply complexity of an algorithm is the amount of resources required to run it.

Resources: time and memory
Computational complexity

Worst-case complexity of an algorithm

\[ n \rightarrow f(n) \]

\( n \): size of the input

\[ f(n) = \max f(I) \text{ over all instances } I \text{ of size } n \]

Problem: Sorting
Instance I: 6, 35, 7, 15, 27, 12, 18
Algorithm: Bubblesort
Size: 7
\[ f(7) = \max f(I) \text{ over all instances } I \text{ of size 7} \]
Computational complexity

\[ f(n) \text{ very difficult to obtain} \]

**Asymptotic behavior** (when \( n \) tends to the infinity)

The complexity is expressed by using big O notation

Complexity of Bubblesort: \( O(n^2) \)
Computational complexity

The complexity of a problem is the infimum of the complexities of the algorithms that solve the problem, including unknown algorithms.

Complexity of problem Sorting: $O(n \log n)$
(thanks to Heapsort)
Computational complexity

Why do we care about computational complexity?

- $O(n)$
- $O(n \log n)$
- $O(n^2)$
- $O(2^n)$

<table>
<thead>
<tr>
<th></th>
<th>$n = 10$</th>
<th>$n = 20$</th>
<th>$n = 30$</th>
<th>$n = 40$</th>
<th>$n = 50$</th>
<th>$n = 60$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>$10^{-5''}$</td>
<td>$2 \cdot 10^{-5''}$</td>
<td>$3 \cdot 10^{-5''}$</td>
<td>$4 \cdot 10^{-5''}$</td>
<td>$5 \cdot 10^{-5''}$</td>
<td>$6 \cdot 10^{-5''}$</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>$10^{-4''}$</td>
<td>$4 \cdot 10^{-4''}$</td>
<td>$9 \cdot 10^{-4''}$</td>
<td>$16 \cdot 10^{-4''}$</td>
<td>$25 \cdot 10^{-4''}$</td>
<td>$36 \cdot 10^{-4''}$</td>
</tr>
<tr>
<td>$O(2^n)$</td>
<td>$10^{-3''}$</td>
<td>$1''$</td>
<td>$17.9' '$</td>
<td>$12.7 \text{ days}$</td>
<td>$35.7 \text{ years}$</td>
<td>$366 \text{ centuries}$</td>
</tr>
</tbody>
</table>

CPU speed *1000  \[ O(2^n) \rightarrow \] size solved increases by 10
Computational complexity: classes

- **P Problems**
- **NP Problems**
- **NP Complete**

**P=NP?**  Conjecture: no
Computational complexity: LP

George Dantzig (1914-2005) in 1947 invented the **simplex method**

Worst-case complexity: $O\left(2^{\frac{m}{2}}\right)$

In practice: Rarely time required is greater than $O(m \log n)$

Complexity of LP unknown until 1979

Leonid Khachiyan (1952-2005) in 1979 invented the **ellipsoid method**

Worst-case complexity: $O(n^6)$

In practice: $O(n^6)$
Computational complexity: LP
Solution of LP

More and more powerful software available

(CPLEX, Gurobi)
(GLPK, LP-SOLVE)
Planning problems
Planning problems

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

s.t. \( \sum_{j=1}^{n} x_{ij} = s_i \quad (i = 1, 2, \ldots, m) \) (Supply constraints)

\( \sum_{i=1}^{m} x_{ij} = d_j \quad (j = 1, 2, \ldots, n) \) (Demand constraints)

\( x_{ij} \geq 0 \quad (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n) \)
Optimization problems

- Mixed integer linear programming
- Non-linear programming
- Global optimization
- Non-convex optimization
- Stochastic programming
- Robust optimization
Mixed integer linear programming

Fixed costs

Location

\[
\begin{align*}
\min \ c'x \\
Ax &= b \\
x &\geq 0
\end{align*}
\]

Selection

Scheduling

Integer

Routing
Mixed integer linear programming

NP Problems

P Problems

NP Complete

MILP
Mixed integer linear programming

Branch-and-bound

Branch-and-cut

Branch-and-price

\[ \min c'x \]
\[ Ax = b \]
\[ x \geq 0 \text{ binary} \]

Exponential number of variables
Branch-and-price

- Original Problem Formulation
  - Master Problem
    - Restricted Master Problem
      - Solve Relaxation of RMP
        - Solve subproblem to find column with negative reduced cost. Column found?
          - Yes
            - Add such column to RMP
              - Yes
                - Solution Integral?
                  - Yes
                    - Done
                  - No
                    - No
                      - Branch
          - No
MILP

More and more powerful software
Mixed integer linear programming

- Fixed costs
- Location
- Specialized algorithms (exact and heuristic)
- Selection
- Scheduling
- Routing