0.1 OP and Solvers

0.1.1 Linear Programming (LP)

Linear programming is an optimization problem where the objective function and all constraints are linear.

$$
\begin{aligned}
\min_{z} \quad & c^T x \\
\text{s.t.} \quad & Ax \leq b, \\
& A_{\text{eq}} x = b_{\text{eq}}, \\
& 0 \leq x_{\min} \leq x_t \leq x_{\max}
\end{aligned}
$$

Applications: Resource allocation, supply chain optimization, scheduling problems.

0.1.2 Quadratic Programming (QP)

Quadratic programming is an optimization problem where the objective function is a quadratic function, and the constraints are linear.

$$
\begin{aligned}
\min_{z} \quad & \frac{1}{2} x^T Q x + c^T x \\
\text{s.t.} \quad & Ax \leq b, \\
& A_{\text{eq}} x = b_{\text{eq}}, \\
& 0 \leq x_{\min} \leq x_t \leq x_{\max}
\end{aligned}
$$

Applications: Portfolio optimization, support vector machines, power grid optimization.

0.1.3 Min-Max Class (MM)

The Min-Max optimization problem aims to find the minimum of the worst-case scenario. This is often used in robust optimization and game theory.

$$
\begin{aligned}
\min_{z} \quad & \max f(x) \\
\text{s.t.} \quad & Ax \leq b, \\
& A_{\text{eq}} x = b_{\text{eq}}, \\
& 0 \leq x_{\min} \leq x_t \leq x_{\max}
\end{aligned}
$$

Applications: Adversarial learning, robust control, worst-case scenario planning.

0.1.4 Convex Programming (CP)

Convex programming is an optimization problem where the objective function and constraints are convex, ensuring a global optimum.

$$
\begin{aligned}
\min_{z} \quad & f(x) \\
\text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\
& A_{\text{eq}} x = b_{\text{eq}}, \\
& 0 \leq x_{\min} \leq x_t \leq x_{\max}
\end{aligned}
$$

Applications: Machine learning, logistics, economic modeling, finance.

0.1.5 Linear Minimum-Time (LMT)

The linear minimum-time optimization problem seeks to reach a target state in the shortest possible time while following system dynamics and constraints.

$$
\begin{aligned}
\min_{z} \quad & \tau \\
\text{s.t.} \quad & s_{t+1} = As_t + Bx_t, \\
& s_0 = s_i, \quad s_\tau = s_f, \\
& 0 \leq x_{\min} \leq x_t \leq x_{\max}, \\
& s_{\min} \leq s_t \leq s_{\max}
\end{aligned}
$$

Applications: Robotics, aerospace, motion planning, autonomous systems.

0.1.6 Geometric Programming (GP)

Geometric Programming (GP) is a type of convex optimization problem where the objective function and constraints are **posynomial functions**, and the decision variables are strictly positive.

$$\begin{aligned}
\min_{x} \quad & f_0(x) \\
\text{s.t.} \quad & f_i(x) \leq 1, \quad i = 1, 2, \dots, m, \\
& g_j(x) = 1, \quad j = 1, 2, \dots, p,
\end{aligned}$$

where:

- $x = (x_1, x_2, \dots, x_n)$ are the **decision variables**, and they must be strictly positive ($x_i > 0$).
- $f_0(x)$ and $f_i(x)$ are **posynomial functions** of x, i.e.,

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \dots x_n^{a_{nk}}, \quad c_k > 0.$$

- $g_j(x)$ are **monomial equality constraints**, i.e.,

$$g_j(x) = d_j x_1^{b_{1j}} x_2^{b_{2j}} \dots x_n^{b_{nj}}, \quad d_j > 0.$$

Applications:

- **Engineering Design:** Optimizing circuit design, mechanical structures, and material selection.
- **Wireless Communication:** Power control and resource allocation in networks.
- **Economics and Finance:** Portfolio optimization and cost minimization problems.
- **Machine Learning:** Hyperparameter tuning and optimization of neural network architectures.

0.1.7 Semi-Definite Programming (SDP)

Semi-Definite Programming (SDP) is a convex optimization problem where the objective function is linear, and the constraints involve semi-definite matrices. SDP generalizes linear programming.

$$\begin{aligned}
\min_{X} \quad & \text{Tr}(C^T X) \\
\text{s.t.} \quad & \text{Tr}(A_i^T X) \leq b_i, \quad i = 1, \dots, m, \\
& X \succeq 0
\end{aligned}$$

where:

- X is a symmetric positive semi-definite matrix ($X \succeq 0$).
- C and A_i are given symmetric matrices.
- b_i are given scalars.
- The notation $\text{Tr}(M)$ represents the trace of matrix M.

Applications:

- **Control Theory**: Stability analysis of dynamic systems.
- **Combinatorial Optimization**: Solving graph partitioning and Max-Cut problems.
- **Machine Learning**: Kernel learning and dimensionality reduction.
- **Finance**: Portfolio optimization with risk constraints.

0.1.8 Non-Smooth Optimization

Non-smooth optimization refers to optimization problems where the objective function or constraints are not differentiable at some points. Unlike smooth optimization, where gradient-based methods are effective, non-smooth problems require specialized techniques such as subgradient methods, bundle methods, or proximal algorithms.

A general non-smooth optimization problem is given by:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned}$$

where:

- $f(x)$ is a non-smooth objective function (e.g., absolute value, maximum, piecewise functions).
- $g_i(x)$ are inequality constraints, which may also be non-smooth.
- $h_j(x)$ are equality constraints.

Applications

- **Machine Learning**: Training models with non-differentiable loss functions, such as support vector machines (SVM) and L1 regularization in LASSO regression.
- **Robust Control**: Optimizing control policies under worst-case scenarios.
- **Signal Processing**: Sparse signal recovery using total variation minimization.
- **Finance**: Portfolio optimization with transaction costs and risk constraints.
- **Engineering**: Structural optimization where material properties change discontinuously.

Integer Programming (IP)

Definition

Integer Programming (IP) is an optimization problem where some or all of the decision variables are restricted to take integer values. It is a special case of linear programming with an additional integrality constraint.

Mathematical Formulation

$$\begin{aligned} \min_z \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b, \\ & A_{\text{eq}} x = b_{\text{eq}}, \\ & x_i \in \mathbb{Z}, \quad \forall i \in S, \\ & x_{\min} \leq x_t \leq x_{\max} \end{aligned}$$

where:

- x is the decision variable vector.
- $c^T x$ is the linear objective function.
- $Ax \leq b$ and $A_{\text{eq}} x = b_{\text{eq}}$ are the inequality and equality constraints.
- $x_i \in \mathbb{Z}$ imposes integer constraints on some or all variables.
- $x_{\min}$ and $x_{\max}$ define variable bounds.

Applications

Integer programming is widely used in various fields, including:

- **Supply Chain and Logistics:** Optimizing transportation routes and warehouse management.
- **Scheduling:** Workforce scheduling and job-shop scheduling problems.
- **Finance:** Portfolio selection with discrete assets.
- **Network Design:** Routing and communication network optimization.
- **Resource Allocation:** Assigning resources in constrained environments.

0.1.9 Real-Valued Programming

Real-Valued Programming (RVP) refers to an optimization problem where the decision variables take continuous real values, as opposed to integer or binary values. It is a broad category that encompasses many types of optimization problems, including linear, nonlinear, and convex programming.

$$\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\
& h_j(x) = 0, \quad j = 1, \ldots, p, \\
& x_{\min} \leq x \leq x_{\max}
\end{aligned}$$

where:

- $x \in \mathbb{R}^n$ represents the continuous decision variables.
- $f(x)$ is the objective function to be minimized.
- $g_i(x)$ are inequality constraints.
- $h_j(x)$ are equality constraints.
- $x_{\min}, x_{\max}$ define variable bounds.

Applications

- **Machine Learning:** Optimization of neural network weights and hyperparameters.
- **Finance:** Portfolio optimization, risk management, and asset allocation.
- **Engineering Design:** Optimal design of mechanical structures, circuits, and systems.
- **Logistics and Operations Research:** Resource allocation, production scheduling, and transportation optimization.
- **Robotics and Control Systems:** Motion planning, optimal control, and trajectory optimization.

0.1.10 Deterministic Programming

Deterministic programming refers to optimization problems where all parameters (objective function coefficients, constraints, and decision variables) are known with certainty and do not involve randomness. The same input data will always produce the same optimal solution.

$$\begin{aligned}
\min_{x} \quad & f(x) \\
\text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\
& h_j(x) = 0, \quad j = 1, \ldots, p, \\
& x \in X
\end{aligned}$$

where:

- $f(x)$ is the objective function to be minimized (or maximized).
- $g_i(x) \leq 0$ are inequality constraints.
- $h_j(x) = 0$ are equality constraints.
- X represents the feasible set of decision variables.

Applications

Deterministic programming is widely used in various fields, including:

- **Linear Programming (LP)**: Resource allocation, production planning.
- **Quadratic Programming (QP)**: Portfolio optimization, machine learning.
- **Network Optimization**: Transportation problems, shortest path problems.
- **Energy Systems**: Power grid optimization, scheduling.
- **Supply Chain Management**: Inventory control, logistics.

0.1.11 Stochastic Programming

Stochastic programming is an optimization framework that deals with decision-making under uncertainty. Unlike deterministic optimization, where all parameters are known beforehand, stochastic programming incorporates random variables to model uncertain elements in constraints or objective functions. It is widely used in finance, supply chain management, and energy systems where future conditions are uncertain.

$$
\begin{aligned}
\min_{x} \quad & \mathbb{E}[f(x,\xi)] \\
\text{s.t.} \quad & g(x,\xi) \le 0, \\
& x \in X
\end{aligned}
$$

where:

- x is the decision variable.
- ξ is a random variable representing uncertainty.
- $\mathbb{E}[f(x,\xi)]$ represents the expected value of the objective function over the distribution of ξ.
- $g(x,\xi) \le 0$ represents constraints that depend on uncertainty.
- X is the feasible set for decision variables.

A common approach is the **two-stage stochastic programming** formulation:

$$
\begin{aligned}
\min_{x} \quad & c^T x + \mathbb{E}[Q(x,\xi)] \\
\text{s.t.} \quad & Ax \le b, \quad x \ge 0
\end{aligned}
$$

where $Q(x,\xi)$ is the second-stage (recourse) function:

$$
\begin{aligned}
Q(x,\xi) = \min_{y} \quad & q^T y \\
\text{s.t.} \quad & Wy \le h - Tx, \quad y \ge 0
\end{aligned}
$$

Applications

- **Finance:** Portfolio optimization under uncertain returns.
- **Supply Chain Management:** Inventory control under demand uncertainty.
- **Energy Systems:** Power grid optimization considering uncertain renewable energy supply.
- **Healthcare:** Optimal resource allocation under uncertain patient arrivals.
- **Telecommunications:** Network design with uncertain traffic demands.

0.1.12 Multi-Objective Programming (MOP)

Multi-Objective Programming (MOP) is an optimization problem where multiple conflicting objective functions are optimized simultaneously. Unlike single-objective optimization, MOP does not seek a single optimal solution but instead aims to find a set of Pareto-optimal solutions where improving one objective may worsen another.

$$\begin{aligned}
\min_{x} \quad & \mathbf{F}(x) = (f_1(x), f_2(x), \ldots, f_k(x))^T \\
\text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\
& h_j(x) = 0, \quad j = 1, \ldots, p, \\
& x \in X,
\end{aligned}$$

where:

- x is the decision variable.
- $\mathbf{F}(x) = (f_1(x), f_2(x), \ldots, f_k(x))^T$ represents k conflicting objective functions.
- $g_i(x) \leq 0$ are inequality constraints.
- $h_j(x) = 0$ are equality constraints.
- X is the feasible decision space.

A solution x^* is **Pareto-optimal** if there is no other x such that:

$$f_i(x) \leq f_i(x^*), \quad \forall i \in \{1, \ldots, k\}$$

with at least one strict inequality.

Applications

Multi-objective programming is widely used in real-world problems where multiple conflicting goals must be balanced:

- **Engineering Design:** Optimizing weight, strength, and cost of materials.
- **Finance:** Portfolio optimization considering risk vs. return.
- **Supply Chain Management:** Minimizing cost while maximizing delivery speed.
- **Environmental Science:** Balancing economic growth and sustainability.
- **Machine Learning:** Hyperparameter tuning with competing metrics (e.g., accuracy vs. computation time).

0.1.13 Graph Optimization

Graph optimization refers to a class of optimization problems where the objective is to find the best way to traverse, cluster, or assign weights in a graph while satisfying given constraints. These problems arise in areas such as network design, transportation, and artificial intelligence.

A graph optimization problem is typically defined on a graph $G = (V, E)$, where:

- V is the set of vertices (nodes),
- E is the set of edges (connections between nodes),
- $w : E \to \mathbb{R}^+$ is a weight function assigning costs or distances to edges.

A general graph optimization problem can be formulated as:

$$\min_{x} \sum_{(i,j) \in E} w_{ij} x_{ij}$$

Subject to:

$$Ax = b, \quad x \in X$$

where:

- x_{ij} represents decision variables associated with edges (e.g., whether an edge is selected in a shortest path or flow problem).
- A is a constraint matrix (e.g., flow conservation in network flow problems).
- b represents boundary conditions (e.g., supply and demand constraints in flow problems).
- X is a feasible set defining domain constraints (e.g., integer constraints for combinatorial problems).

Applications:

- **Shortest Path Problems:** Finding the most efficient route in navigation (e.g., Dijkstra's algorithm, A* search).
- **Network Flow Problems:** Optimizing traffic flow, supply chains, or communication networks (e.g., Max-Flow, Min-Cost Flow).
- **Graph Clustering:** Partitioning graphs into meaningful subgroups for community detection in social networks.
- **Matching Problems:** Assigning resources optimally, such as job allocation (e.g., Bipartite Matching, Hungarian Algorithm).
- **Traveling Salesman Problem (TSP):** Finding the shortest possible route that visits all cities exactly once.

0.1.14 Network Optimization

Network optimization is a class of optimization problems where the objective is to optimize the performance of a network, such as minimizing costs, maximizing flow, or improving connectivity. These problems arise in transportation, telecommunications, supply chains, and logistics.

A general form of a network optimization problem can be written as:

1. Shortest Path Problem

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in N, \\
& x_{ij} \geq 0, \quad \forall (i,j) \in E.
\end{aligned}
$$

where: - x_{ij} is the flow along edge (i,j), - c_{ij} is the cost associated with traveling along (i,j), - s is the source node, and t is the destination node.

2. Maximum Flow Problem

$$
\begin{aligned}
\max \quad & \sum_{j\in N} x_{sj} \\
\text{s.t.} \quad & \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = \begin{cases} s, & i = s \\ -s, & i = t \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in N, \\
& 0 \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in E.
\end{aligned}
$$

where: - x_{ij} is the flow along edge (i,j), - u_{ij} is the capacity of the edge.

3. Minimum Cost Flow Problem

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = b_i, \quad \forall i \in N, \\ & 0 \le x_{ij} \le u_{ij}, \quad \forall (i,j) \in E. \end{aligned}$$

where: - b_i is the supply/demand at node i, - u_{ij} is the capacity limit of the edge.

4. Network Design Problem

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{ij} y_{ij} \\ \text{s.t.} \quad & x_{ij} \le u_{ij} y_{ij}, \quad \forall (i,j) \in E, \\ & \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = b_i, \quad \forall i \in N, \\ & y_{ij} \in \{0,1\}, \quad \forall (i,j) \in E. \end{aligned}$$

where: - y_{ij} is a binary decision variable indicating whether edge (i,j) is built.
Applications

- **Telecommunications:** Designing efficient routing protocols, bandwidth allocation.
- **Transportation:** Finding shortest paths, minimizing traffic congestion.
- **Supply Chain:** Optimizing logistics and distribution networks.
- **Power Grids:** Managing electricity flow in smart grids.
- **Internet Traffic Routing:** Load balancing and minimizing latency in data networks.

0.2 Optimization Solvers Summary

Solver	Advantages	Disadvantages	Applicable Problems
Gurobi*	Fast, supports large-scale problems, multi-threading	Expensive, requires a license	LP, QP, MIP, MIQP
CPLEX*	Industrial-grade, efficient for MILP	High cost, not open-source	LP, QP, MIP
MOSEK*	Strong for SOCP and SDP, good for convex optimization	Expensive, limited to convex problems	LP, QP, SOCP, SDP
KNITRO*	Efficient for nonlinear constrained problems, supports second-order methods	Very expensive, sensitive to initial conditions	NLP, constrained optimization
SNOPT*	Good for sparse nonlinear optimization	High cost, mainly for specific applications	NLP
BARON*	Strong for global optimization, guarantees global optima	Extremely slow for large problems, high cost	NLP (non-convex), MINLP
MATLAB fmincon*	Versatile, built into MATLAB, good for small NLPs	Limited to local optimization, cannot solve integer problems	NLP, constrained optimization
MATLAB fminunc*	Good for unconstrained nonlinear optimization	Limited to local optima, requires gradients	Unconstrained NLP
CVX (MATLAB)*	Good for convex optimization, easy to use in MATLAB	Limited to convex problems, slow for large problems	LP, QP, SDP
Ant Colony Optimization+	Good for combinatorial optimization	Very slow, requires tuning of parameters	TSP
Simulated Annealing (MATLAB)*	Can escape local optima, useful for non-convex problems	Slow convergence, no optimality guarantee	Non-convex optimization
GLPK+	Free, good for small-scale problems	Slow compared to commercial solvers, lacks advanced features	LP, MIP
CBC+	Open-source, good for integer programming	Slower than commercial solvers like Gurobi	LP, MIP
OSQP+	Open-source, efficient for quadratic programming	Only solves convex QPs, not general NLPs	QP, SOCP
SCIP+	One of the best open-source MILP solvers	Slower than commercial alternatives	LP, MIP, MINLP
IPOPT+	Open-source, efficient for large-scale NLP	Only works for convex problems, no support for MIP	NLP
SciPy Optimize+	Open-source, general-purpose optimization library	Lacks advanced features for large-scale optimization	LP, QP, NLP
NLopt+	Open-source, supports local and global optimization	No support for integer programming	NLP
CVXPY (Python)+	Python-friendly, good for convex problems	Slow, cannot handle non-convex optimization	LP, QP, SDP

Table 1: Comparison of Optimization Solvers (*: Commercial, +: Open-source)