Structural Optimization Techniques

Bilal TAYFUR

Abstract. This course provides a comprehensive introduction to the optimization of structural systems. Starting from the fundamentals of optimization theory, students will learn the general difference between classical and modern optimization methods and gain the ability to apply these methods in the design and analysis of structural systems.

The codes of the relevant example can be accessed in the GitHub repository of the course via the QR codes given on the right side of the topics. This repo is updated every year. Therefore, current lecture notes and newly added applications can also be accessed through the repo and can be contributed if desired.

The following textbooks are recommended for students who wish to explore the topics in more depth:

- Cottle, R. W., & Thapa, M. N. (2017). **Linear and Nonlinear Optimization**. Springer.
- Rao, S. S. (2019). **Engineering Optimization: Theory and Practice, 5th Edition**. John Wiley & Sons.
- Arora, J. S. (2016). **Introduction to Optimum Design, 4th Edition**. Academic Press.
- Yang, X. S. (2021). **Nature-Inspired Optimization Algorithms, 2nd Edition**. Elsevier.
- Bendsøe, M. P., & Sigmund, O. (2003). **Topology Optimization: Theory, Methods, and Applications**. Springer.

Contents

1 Introduction to Optimization Theory

Optimization theory is the collection of mathematical and methodological approaches aimed at maximizing the performance of a system under certain constraints. This course will focus on the optimization of structural systems, covering fundamental concepts and application methods.

1.1 Definition of Optimization and Its Importance in Engineering

Optimization is the process of maximizing the performance of a system under certain constraints. [1] This process aims to reduce cost while increasing performance in engineering designs.

[1] The "best" decision-making processes we encounter in daily life are actually optimization problems. For example, choosing the shortest route to work or selecting the most affordable products at the supermarket.

1.2 Basic Components of Structural Optimization

Structural optimization is built on three fundamental components: objective function, design variables, and constraints. These components form the mathematical formulation of the optimization problem.

- **Objective Function:** The target to be minimized or maximized (e.g., weight, cost, stiffness)
- **Design Variables:** Parameters to be optimized (e.g., section dimensions, material properties)
- **Constraints:** Conditions that the design must satisfy (e.g., stress limits, displacement bounds)

1.3 History and Development of Optimization

The foundations of optimization theory have advanced in parallel with the development of mathematical analysis methods. Modern optimization methods have gained new dimensions with the development of computer technology. [2]

1.3.1 Important Historical Developments

- 1940s: Development of linear programming and the Simplex method
- 1950s: Dynamic programming and convex optimization theory
- 1960s: Application of finite element method to optimization
- 1970s: Development of numerical optimization algorithms
- 1980s: Emergence of metaheuristic algorithms
- 1990s: Widespread adoption of topology optimization
- 2000s: Integration of multi-objective optimization and artificial intelligence techniques

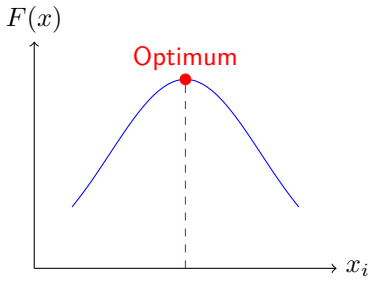

Figure 1: Illustration of an optimum point in a single-variable optimization problem

[2] The first structural optimization studies began with Michell's 1904 publication on minimum weight design of truss systems. This work forms the foundation of modern topology optimization.

1.4 General Structure of Optimization Problems

Every optimization problem is expressed as the minimization or maximization of an objective function. The problem formulation includes design variables and constraints. [3]

[3] The mathematical formulation of the optimization problem provides the structure necessary to solve the problem systematically. This formulation allows us to address different engineering problems within a common framework.

$$
\begin{aligned}
\text{minimize} \quad & f(\mathbf{x}) \\
\text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \qquad i = 1, \ldots, m \\
& h_j(\mathbf{x}) = 0, \qquad j = 1, \ldots, p \\
& x_k^L \leq x_k \leq x_k^U, \quad k = 1, \ldots, n
\end{aligned}
\tag{1}
$$

Structural Optimization Example

For the optimal design of a steel beam:

- **Objective:** Minimum weight
- **Variables:** Section height and width
- **Constraints:**
  - Maximum stress $\leq$ Yield stress
  - Maximum deflection $\leq$ Allowable deflection
  - Minimum section dimensions

1.5 Primary Application Areas of Optimization in Engineering

Optimization is widely used in various fields of engineering. Civil, mechanical, aerospace, and space engineering are the main areas of application.

- **Civil Engineering:**
  - Section optimization of steel structures
  - Reinforcement optimization for reinforced concrete elements
  - Form optimization in bridge design
- **Mechanical Engineering:**
  - Shape optimization of mechanical parts
  - Performance optimization of thermal systems
  - Vibration control and damping
- **Aerospace and Space Engineering:**
  - Wing and fuselage design
  - Composite material optimization
  - Structural weight minimization

1.6 Deterministic and Stochastic Optimization Approaches

Optimization methods are divided into two main categories based on their problem-solving approaches: deterministic [4] and stochastic [5].

1.6.1 Deterministic Approaches

- Provide the same result in each run
- Gradient-based methods fall into this category
- Risk of getting stuck in local optima
- Dependent on the starting point

[4] Deterministic methods are those that always produce the same result for the same input. In other words, they are predictable and consistent.

[5] Stochastic methods are those that produce different results for the same input due to randomness. In other words, they are unpredictable and inconsistent.

1.6.2 Stochastic Approaches

- Involve randomness
- May produce different results in each run
- Higher probability of finding the global optimum
- Methods such as genetic algorithms and simulated annealing fall into this category

These approaches offer solution strategies suitable for different types of problems. [6]

[6] The choice between deterministic and stochastic approaches depends on the structure of the problem and solution requirements. For example, stochastic methods may be more advantageous in a multi-modal problem.

1.7 Linear and Nonlinear Optimization

Optimization problems are classified as linear and nonlinear problems based on the structure of the objective function and constraints. This classification determines the solution methods to be used.

1.7.1 Linear Optimization

- Objective function and constraints are linear
- Solution space is convex
- Efficient solution methods such as the Simplex method exist
- Global optimum is guaranteed

Linear Optimization Example

A production planning problem:

$$\begin{aligned} \text{maximize} \quad & 3x_1 + 2x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \leq 100 \\ & x_1 + x_2 \leq 80 \\ & x_1, x_2 \geq 0 \end{aligned}$$

1.7.2 Nonlinear Optimization

- Objective function and/or constraints are nonlinear
- Solution space is complex
- May contain local optima
- Most structural problems fall into this category [7]

[7] The vast majority of problems encountered in structural engineering are nonlinear in character. For example, effects such as geometric nonlinearity and material nonlinearity make the problem nonlinear.

Nonlinear Optimization Example

A structural design optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(x) = x_1^2 + 2x_2^2 - 0.3x_1x_2 \\ \text{subject to} \quad & g_1(x) = x_1^2 + x_2^2 - 25 \leq 0 \\ & g_2(x) = x_1 - 2x_2 + 5 \leq 0 \\ & -10 \leq x_1, x_2 \leq 10 \end{aligned}$$

1.8 General Classification of Solution Methods

Optimization methods are classified as analytical, numerical, and metaheuristic methods based on the problem type and solution strategy. Each method group offers advantages for certain types of problems.

- **Analytical Methods**
  - Differential calculus
  - Variational methods
  - Lagrange multipliers
- **Numerical Methods**
  - Gradient-based methods
  - Linear programming
  - Nonlinear programming
- **Metaheuristic Methods**
  - Genetic algorithms
  - Particle swarm optimization
  - Simulated annealing

1.9 Global and Local Optima in Optimization Problems

In multi-modal optimization problems, there may be multiple local optimum points. This situation is frequently encountered especially in nonlinear problems.

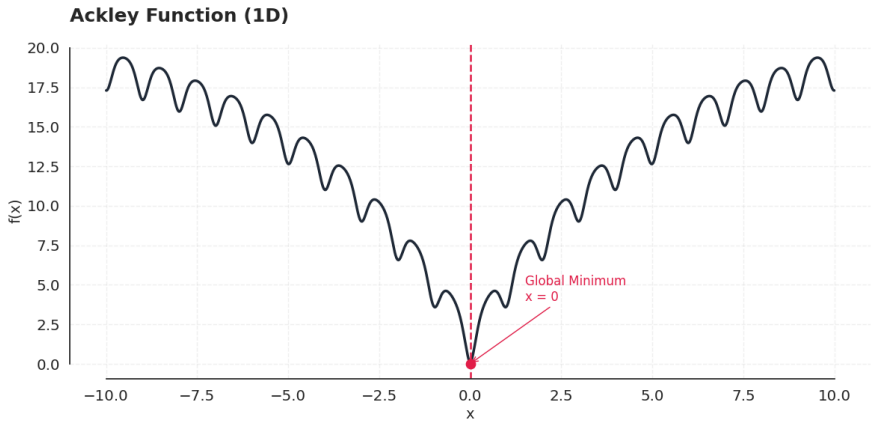

Figure 2: One-dimensional and multi-modal optimization problem

As seen in the figure above, a multi-modal function may have multiple peaks (maxima) and valleys (minima). Optimization algorithms may get stuck in a local optimum depending on the starting point and fail to find the global optimum. Therefore, metaheuristic methods may be preferred to find the global optimum, especially in complex engineering problems.

2 Basic Optimization Concepts

The fundamental concepts necessary for the mathematical formulation and solution of optimization problems will be examined in detail in this section. These concepts form the foundation for understanding more complex optimization problems.

2.1 Objective Function and Constraint Functions

In the mathematical modeling of optimization problems, there are two fundamental components: the objective function and constraint functions.

2.1.1 Objective Function

The objective function mathematically expresses the performance criterion to be optimized. This function can be:

- Minimized (e.g., cost, weight, energy consumption)
- Maximized (e.g., efficiency, strength, stiffness)

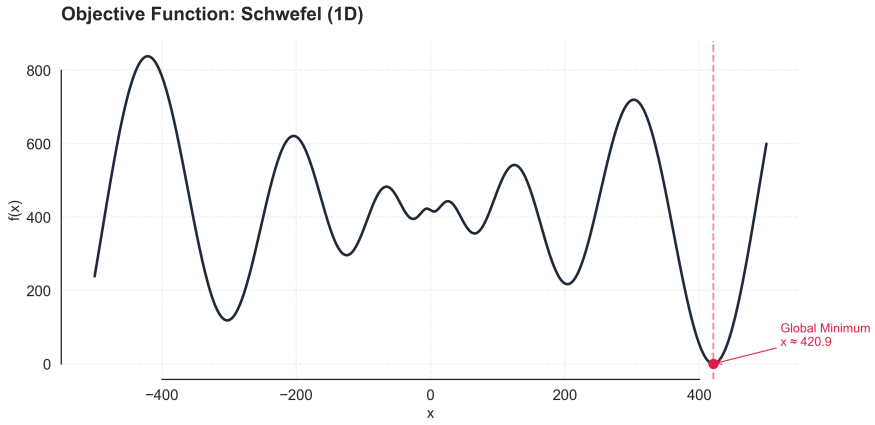

Figure 3: Objective Function

[8]

2.1.2 Constraint Functions

Constraint functions mathematically express the conditions that the design must satisfy:

- Equality constraints: $h_j(x) = 0$
- Inequality constraints: $g_i(x) \leq 0$
- Boundary constraints: $x_L \leq x \leq x_U$ [9]

Equality constraints: $h_j(x) = 0$ Equality constraints express situations where design variables must take exactly specific values. These types of constraints mathematically define conditions where certain parameters must be precisely satisfied within the optimization problem. For example, a structure's total weight being equal to a specific value or mass balance in a chemical reaction can be expressed with equality constraints. Equality constraints generally keep the optimization space in a narrower area, significantly reducing the number of possible solutions.

[8] Any maximization problem can be converted to a minimization problem by taking the negative of the objective function:

$$\max f(x) = -\min(-f(x))$$

[9] With few exceptions, structural optimization problems are generally expressed with inequality constraints. Depending on the problem, boundary constraints may also be used. However, when a statically indeterminate structure is subject to optimization, this constraint can also lead to missing the global optimum.
In fact, one of the aspects that makes structural optimization problems worth examining from an optimization perspective is that this statical indeterminacy significantly increases the unpredictability of many structural optimization problems.

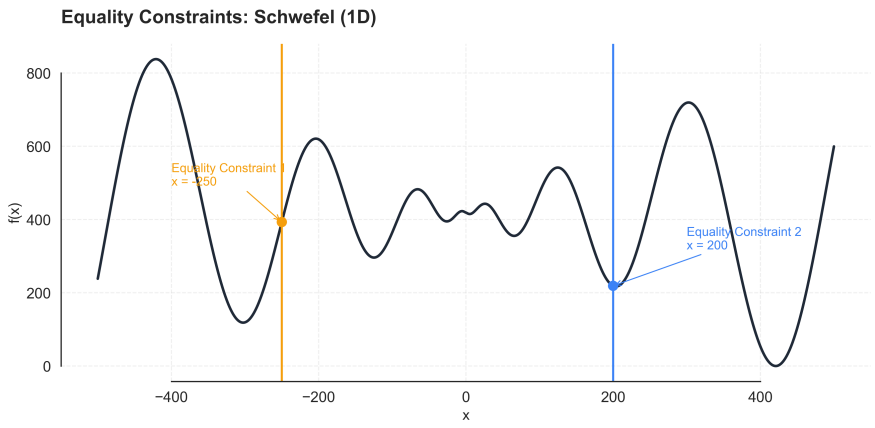

Figure 4: Equality Constraints

Inequality constraints: $g_i(x) \leq 0$ Inequality constraints express situations where design variables must not exceed certain limits. These types of constraints are used to define limitations such as the maximum stress a structure can withstand, a system's maximum energy consumption, or a material's minimum safety factor. Inequality constraints are critical for ensuring that the design is physically realizable and safe. Additionally, inequality constraints define the feasible solution area, ensuring that the optimization algorithm searches only within the valid design space.

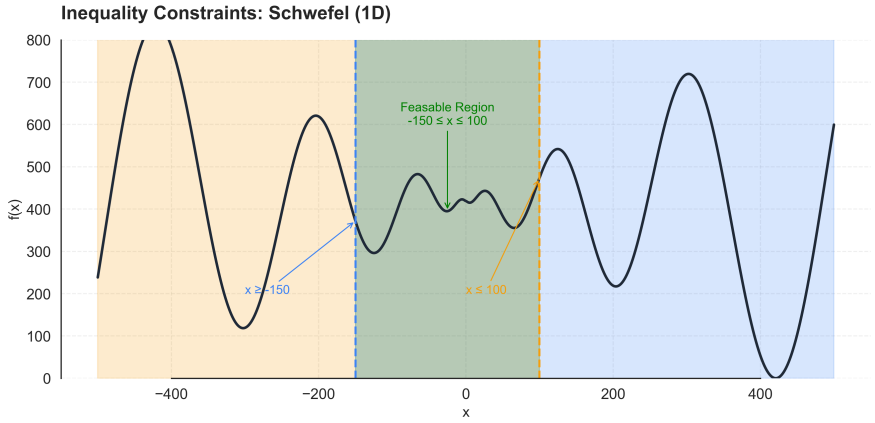

Figure 5: Inequality Constraints

Boundary constraints: $x_L \leq x \leq x_U$ Boundary constraints determine the minimum and maximum values that each design variable can take. These constraints reflect the physical limits of design variables, manufacturability conditions, or ranges determined by standards. For example, situations where a beam's thickness cannot be less than a certain value due to manufacturing constraints or cannot exceed a certain maximum value due to installation requirements are expressed with boundary constraints. Boundary constraints narrow the optimization algorithm's search space, increasing computational efficiency and ensuring the elimination of physically meaningless solutions.

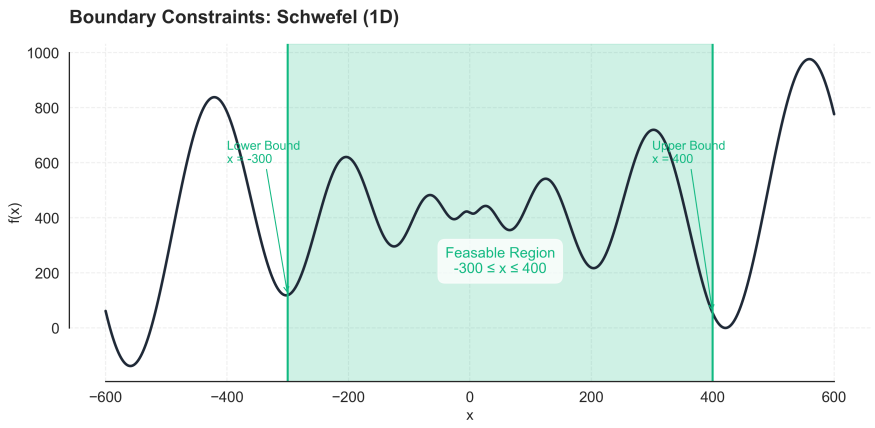

Figure 6: Boundary Constraints

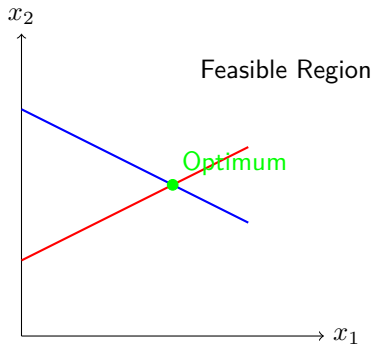

Figure 7: Feasible solution region formed by the intersection of two constraints

2.2 Decision Variables and Solution Space

2.2.1 Decision Variables

Decision variables are parameters whose values need to be determined during the optimization process:

- Continuous variables (e.g., dimensions, thicknesses)
- Discrete variables (e.g., number of elements)
- Binary variables (0-1 decisions)

Decision Variables in Structural Design

In a steel frame optimization:

- **Continuous:** Section dimensions
- **Discrete:** Number of stories
- **Binary:** Existence/non-existence of elements

2.2.2 Solution Space

The solution space is the space formed by all possible combinations of decision variables[10]:

- **Feasible Solution Region:** Set of points satisfying all constraints
- **Infeasible Region:** Points violating at least one constraint

2.3 Global and Local Minimum/Maximum Concepts

2.3.1 Local Optimum

A point is a local optimum if it has the best value within a certain neighborhood:

$$x^* \text{ local minimum } \Leftrightarrow f(x^*) \leq f(x), \forall x \in N(x^*) \tag{2}$$

2.3.2 Global Optimum

The point with the best value in the entire solution space is the global optimum:

$$x^* \text{ global minimum } \Leftrightarrow f(x^*) \leq f(x), \forall x \in S \tag{3}$$

[10] The dimension of the solution space is determined by the number of decision variables. In high-dimensional problems, the "curse of dimensionality" problem arises.

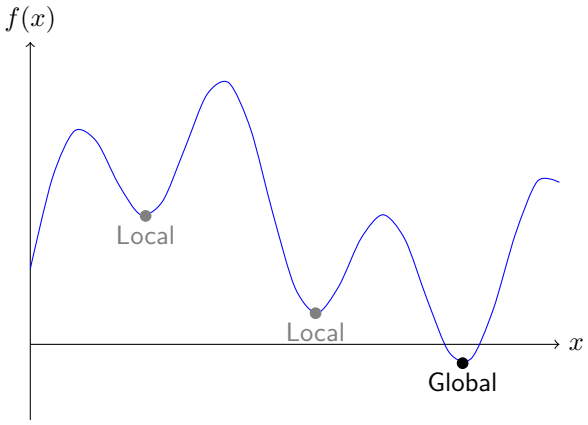

Figure 8: Local and global minimum points of a function

2.4 Physical and Mathematical Modeling

2.4.1 Physical Modeling

Modeling real-world problems based on physical principles[11]:

- Force equilibrium
- Energy conservation
- Material behavior
- Geometric relationships

[11] A good mathematical model should reflect physical reality with sufficient accuracy while avoiding unnecessary complexity. While this is not essentially the main topic of optimization, it is quite important from a structural optimization perspective.

2.4.2 Mathematical Modeling

Transforming the physical model into mathematical formulation:

- Differential equations
- Algebraic equations
- Matrix formulations
- Finite element models

2.5 Differentiability and Continuity

2.5.1 Continuity

A function being continuous means that small input changes do not cause sudden jumps in the output. Mathematically:

$$\lim_{x \to x_0} f(x) = f(x_0) \tag{4}$$

Relationship Between Continuity and Optimization

Continuity is an important property in optimization problems because:

- For continuous functions, there must be a minimum and maximum value in a closed and bounded interval (Weierstrass theorem).
- Optimization of non-continuous functions is more difficult because sudden jumps can prevent algorithms from progressing in the right direction.
- Most physical behaviors in real engineering problems are modeled with continuous functions (e.g., deformation of a beam under load).

2.5.2 Differentiability

The existence of a function's derivative means that the function's behavior can be approximately expressed with a tangent line at every point. Mathematically:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} \tag{5}$$

Relationship Between Differentiability and Optimization

Differentiability plays a critical role in the optimization process:

- **Gradient Information:** The derivative gives the direction of fastest increase/decrease of the function, so optimization algorithms know where to go.
- **Critical Points:** Points where the function's derivative is zero (critical points) are potential optimum points.
- **Second Derivative:** The second derivative helps determine whether a critical point is a minimum, maximum, or saddle point.

Real-Life Example: Mountain Climbing

Imagine you're climbing a mountain and your goal is to reach the summit. If it's a foggy day and your visibility is very limited, how should you proceed?

- **Gradient (Derivative) Information:** At each step, you can feel the slope of the ground with your feet and choose the steepest uphill direction (negative gradient).
- **Non-differentiable Points:** If there's a steep cliff (point where derivative is undefined) in your path, you can't proceed directly and need to find a different route.
- **Local Peak vs. Summit:** During your climb, you might reach a small peak (local maximum), but the real summit (global maximum) might be elsewhere.

This analogy is very similar to how gradient-based optimization algorithms work.

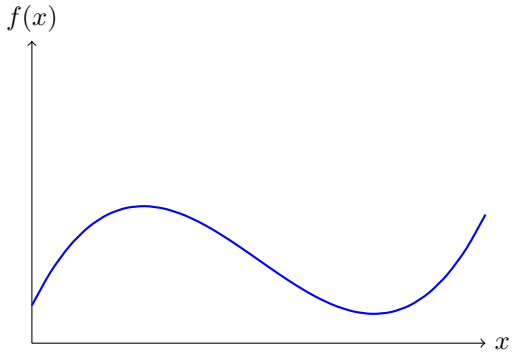

Figure 9: Differentiable Function and Gradient

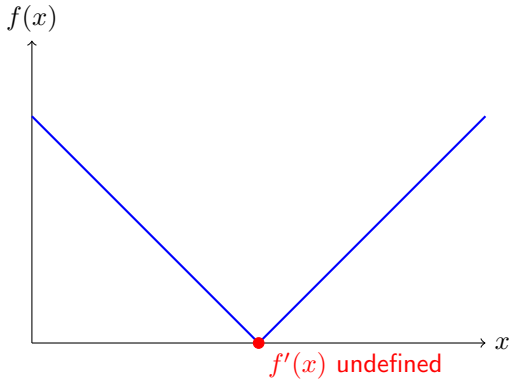

Figure 10: Non-differentiable Function (Absolute Value Function)

Optimization Methods Based on Differentiability

- **Gradient-Based Methods for Differentiable Functions:**
  - **Gradient Descent:** Proceeds in the direction of fastest decrease of the function.
  - **Newton's Method:** Uses second derivative information to achieve faster convergence.
  - **Quasi-Newton:** Approximates the second derivative (BFGS, L-BFGS, etc.).
- **Gradient-Free Methods for Non-differentiable Functions:**
  - **Simplex Search:** Determines the best direction by comparing function values (Nelder-Mead method).
  - **Genetic Algorithms:** Explores the solution space by mimicking evolutionary processes.
  - **Particle Swarm Optimization:** Approaches the solution by mimicking group behavior.
- **Hybrid Approaches for Mixed Problems:**
  - **Search + Refinement:** Search in a broad region with a gradient-free method, then refine from the found point with a gradient-based method.
  - **Subgradient Methods:** Generalized derivative approaches that allow progress even at non-differentiable points.

[12]

[12] Structural optimization problems are generally not differentiable. Therefore, finding the global optimum becomes difficult, and metaheuristic methods are preferred.

2.6 Convex and Non-convex Optimization

2.6.1 Convex Functions

A function being convex means that the line segment connecting any two points on the function's graph lies above the function graph between those two points. Mathematically, for a function $f : \mathbb{R}^n \to \mathbb{R}$:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y), \quad \forall x, y \in \mathbb{R}^n, \forall \lambda \in [0,1] \tag{6}$$

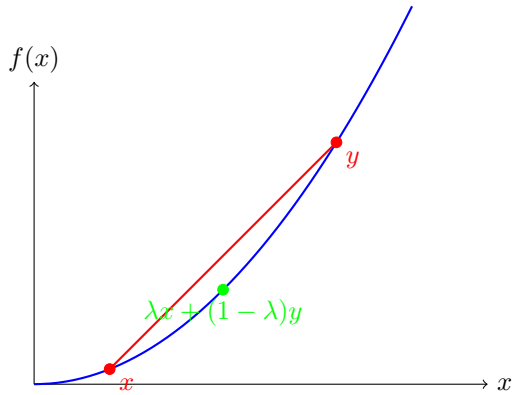

Figure 11: Example of a Convex Function

2.6.2 Convex Set

A convex set means that the line segment connecting any two points in the set lies entirely within the set:

$$\lambda x + (1-\lambda)y \in S, \quad \forall x, y \in S, \forall \lambda \in [0,1] \tag{7}$$

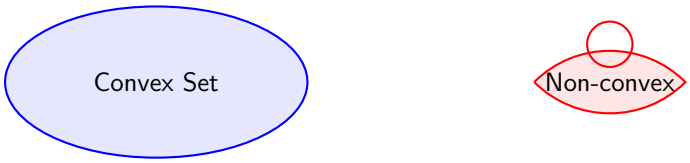

Figure 12: Examples of Convex and Non-convex Sets

2.6.3 Advantages of Convex Optimization

Convex optimization holds special importance in optimization theory[13]:

- **Local Optimum = Global Optimum:** A local minimum of a convex function is also the global minimum. This significantly simplifies the optimization process.
- **Stable Solution:** Regardless of the starting point, appropriate gradient-based algorithms converge to the same optimum point.
- **Efficient Algorithms:** Algorithms such as interior point methods and gradient descent can reach solutions in polynomial time for convex problems.

Example of Convex Optimization

Quadratic programming problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \quad & Ax \leq b \end{aligned} \tag{8}$$

If Q is a positive semi-definite matrix, this problem is convex and can be solved with efficient interior point methods.

[13] Effective algorithms and mathematical guarantees developed for convex optimization problems make solving these types of problems more reliable. The optimization problems within deep learning algorithms we frequently hear about today are generally convex optimization problems. The optimization algorithms used here are usually gradient descent methods. While they are very effective methods, they are generally inefficient in structural optimization problems.

2.6.4 Difficulties of Non-convex Optimization

Non-convex optimization problems are frequently encountered challenges in structural optimization[14]:

- **Multiple Local Optima:** The function can contain multiple local minima, making it difficult to find the global optimum.
- **Starting Point Dependency:** Gradient-based algorithms can converge to different local optima depending on the starting point.
- **Computational Cost:** Generally, more comprehensive search methods are required to guarantee finding the global optimum, which increases computational cost.

[14] Most structural optimization problems are non-convex problems that can contain multiple local optima. This creates difficulties in finding the global optimum.

Approaches to Non-convex Problems

- **Multiple Starting Points:** Multiple local optimizations are run from different starting points, and the best result is selected.
- **Metaheuristic Algorithms:** Methods such as genetic algorithms and particle swarm optimization try to approach the global optimum by exploring a wide solution space.
- **Convex Approximations:** The problem can be decomposed into convex sub-problems or solved using convex approximations (such as Sequential Convex Programming).

2.7 Constrained and Unconstrained Optimization

2.7.1 Unconstrained Optimization

Unconstrained optimization, as the name suggests, refers to problems that do not contain any constraints, only requiring the minimization or maximization of the objective function. Mathematically:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{9}$$

[15]

The main methods used to solve unconstrained optimization problems [16]:

- **Gradient Descent Method:** Aims to reach the minimum by proceeding step by step in the direction of fastest decrease of the function (negative gradient direction). This method is widely used, especially in deep learning and machine learning applications.
- **Newton's Method:** Makes more intelligent decisions about both direction and step size by also using the function's second derivative (Hessian) information. Due to its quadratic convergence property, it converges faster than gradient descent under the right conditions.
- **Quasi-Newton Methods:** Methods that estimate the Hessian matrix approximately instead of calculating it directly to reduce the computational cost of Newton's method. BFGS and L-BFGS are among the most popular Quasi-Newton algorithms.
- **Conjugate Gradient Method:** A method that is particularly effective in large-scale problems and ensures that successive search directions are "conjugate" to each other.
- **Trust Region Methods:** Methods that determine a "trust region" in each iteration where the function can be locally well-modeled and search for the optimum within this region.

[15] Although unconstrained optimization is theoretically called "unconstrained," most engineering problems in practice contain physical or mathematical constraints in some way. The term "unconstrained" here means that there are no explicit constraints in the problem formulation.

[16] The principles of these five methods can be tested with Python code through the link:

Mountain Climbing Analogy

Let's think of a mountain climbing analogy to understand unconstrained optimization methods (for a maximization problem):

- **Gradient Ascent:** You proceed in the steepest uphill direction at each step. Easy to implement but can progress slowly by zigzagging in narrow valleys.
- **Newton's Method:** You look not only at the slope of the ground (gradient) but also at the shape of the terrain (Hessian). This allows you to take large steps on flat areas and small, careful steps on steep slopes.
- **Quasi-Newton:** Instead of fully measuring the shape of the terrain, you estimate it from your previous steps. While not as effective as Newton, it requires much less effort.

2.7.2 Constrained Optimization

Constrained optimization is much more common in modeling real-world problems and includes a set of constraints in addition to the objective function. These constraints represent the conditions that the solution must satisfy. Mathematical formulation:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{10}$$

Here, $g_i(x) \leq 0$ represents inequality constraints, and $h_j(x) = 0$ represents equality constraints.
[17]

The main methods used to solve constrained optimization problems:

- **Lagrange Multipliers Method:** Defines additional variables called Lagrange multipliers for equality-constrained problems, transforming the constrained problem into an extended unconstrained problem. This method is particularly useful when constraints must be exactly satisfied.
- **Karush-Kuhn-Tucker (KKT) Conditions:** Optimality conditions valid for both equality and inequality constraints. KKT conditions are a generalization of the Lagrange multipliers method to inequality constraints.
- **Penalty Function Methods:** Transforms the constrained problem into an unconstrained problem by penalizing constraint violations. There are two main approaches:
  - **Exterior Penalty Method:** Adds penalties for constraint violations, allows infeasible solutions but penalizes them.
  - **Interior Penalty (Barrier) Method:** Adds an increasing penalty as the boundary of the feasible region is approached, ensuring the solution stays within the feasible region.
- **Active Set Methods:** In each iteration, determines which constraints are believed to be active and solves a smaller subproblem. This is particularly effective in linear and quadratic programming problems.

[17] Structural optimization problems are almost always constrained problems. When you want to minimize the weight of a bridge, there are various constraints such as the bridge being able to carry certain loads, having a certain safety factor, and being constructible. An optimization without these constraints would not find practical application.

- **Interior Point Methods:** Approaches the optimum while staying within the feasible region. Uses barrier functions but handles constraints directly. Very effective in large-scale linear and convex programming problems.
- **Sequential Quadratic Programming (SQP):** Solves constrained nonlinear problems by transforming them into a series of quadratic subproblems. Widely used in structural optimization.

Constrained Optimization Analogy: Path Finding

You can think of constrained optimization as a path-finding problem where you have to follow certain rules while trying to find the best way to your destination:

- **Your Goal (Objective Function):** To reach the mountain summit.
- **Your Constraints:** You can only walk on marked trails (equality constraints), you must stay away from certain dangerous areas (inequality constraints).
- **Lagrange Method:** Similar to continuously checking the trail map and dangerous areas while progressing.
- **Penalty Method:** If you leave the marked trail, you experience extra difficulty (getting stuck in mud, passing through thorny bushes, etc.). Despite these penalties, sometimes taking shortcuts can be advantageous.
- **Barrier Method:** You behave as if there's an invisible "repulsive force" around dangerous areas, retreating as you get closer. This way, you always stay in the safe region.

Handling Constraints

Methods to transform a constrained problem into an unconstrained problem:

- **Exterior Penalty Method:**

$$\min f(x) + c \sum \max(0, g_i(x))^2 + c \sum (h_j(x))^2 \tag{11}$$

  Here $c > 0$ is the penalty parameter and is generally increased during iterations. As constraint violations increase, the penalty also increases, thus directing the algorithm toward the feasible region over time.

- **Interior Penalty (Barrier) Method:**

$$\min f(x) - c \sum \ln(-g_i(x)) \tag{12}$$

  This method is only defined when $g_i(x) < 0$ (i.e., within the feasible region) and as the constraint boundary is approached, $\ln(-g_i(x)) \to -\infty$, ensuring the solution stays within the feasible region. The c parameter is generally decreased during iterations.

- **Augmented Lagrangian Method:**

$$\min f(x) + \sum \lambda_j h_j(x) + \frac{c}{2} \sum (h_j(x))^2 + \sum \mu_i \max(0, g_i(x)) + \frac{c}{2} \sum \max(0, g_i(x))^2 \tag{13}$$

  This method combines the advantages of Lagrange multipliers and penalty methods. Lagrange multipliers λ_j and μ_i are updated in each iteration.

2.7.3 Comparison of Constrained and Unconstrained Optimization

- **Problem Difficulty:** Constrained problems are generally more difficult and require more specialized solution methods.
- **Solution Space:** In unconstrained problems, the entire search space can be used, while in constrained problems, the search is limited to the "feasible region."
- **Realism:** Real engineering problems are almost always constrained because designs must meet certain requirements.
- **Solution Strategy:** In solving constrained problems, the goal is generally to first satisfy the constraints, then optimize the objective function, or to handle these two goals in a balanced way.

[18]

2.8 Classification of Optimization Approaches

2.8.1 By Problem Structure

Optimization problems can be categorized according to their mathematical structure, and this classification helps us determine which solution methods are appropriate:

- **Linear Programming (LP)**

[18] Structural optimization problems generally contain quite complex constraints. For example, in the design of a skyscraper, while minimizing cost, many factors such as structural strength, user comfort, seismic performance, and wind loads must be considered as constraints. The nonlinearity of these constraints and their interaction with each other requires the development of special techniques to solve the problem.

- Linear objective function: $f(x) = c^T x$
- Linear constraints: $Ax \leq b$, $Aeq \cdot x = beq$
- Basic solution methods: Simplex algorithm, Interior point methods
- Properties: Single global optimum, convex feasible region defined by constraints
- Application areas: Resource allocation, production planning, logistics optimization

- **Nonlinear Programming (NLP)**
  - Nonlinear objective function and/or constraints
  - Subcategory - Convex Programming: Convex objective function, convex constraint set
  - Subcategory - Non-convex Programming: Objective function and/or constraint set not convex
  - Solution methods: Gradient-based methods (SQP, Interior point, BFGS), metaheuristic methods
  - Application areas: Structural design, mechanical systems optimization, economic models

- **Integer Programming (IP)**
  - All variables must be integer: $x \in \mathbb{Z}^n$
  - Subcategory - Mixed Integer Programming (MIP): Some variables integer, some continuous
  - Subcategory - 0-1 Integer Programming: Variables can only take values 0 or 1
  - Solution methods: Branch-and-bound, Cutting plane, Branch-and-cut
  - Application areas: Scheduling problems, cutting/packing problems, topology optimization

- **Stochastic Programming**
  - Problems containing random variables
  - Cases where uncertainty is modeled with probability distributions
  - Solution methods: Scenario approach, sample average approximation, robust optimization
  - Application areas: Risk management, portfolio optimization, structural design under uncertainty

- **Multi-objective Programming**
  - Simultaneous optimization of multiple objective functions
  - Solution concept: Pareto-optimal solution set
  - Solution methods: Weighted sum method, epsilon-constraint method, evolutionary algorithms like NSGA-II
  - Application areas: Engineering design, environmental management, economic models

[19]

[19] A structural optimization problem generally carries characteristics of several of the above categories. For example, in a bridge design, section dimensions might be continuous variables while element types to be used might be integer variables. Additionally, material behavior might be expressed with nonlinear equa-

2.8.2 By Solution Strategy

Algorithms used to solve optimization problems can be categorized according to their search strategies:

- **Deterministic Methods**
  - **Gradient-based methods:** Progress in the direction of fastest improvement using derivative information of the function.
    - Advantages: Usually fast convergence, guaranteed reach to local optimum
    - Disadvantages: Getting stuck in local optima, derivative calculation requirement
    - Examples: Gradient descent, Newton, BFGS, Conjugate gradient
  - **Direct search methods:** Progress using function evaluations without derivative information.
    - Advantages: No derivatives required, suitable for noisy functions
    - Disadvantages: Usually slower convergence
    - Examples: Nelder-Mead simplex, Hooke-Jeeves pattern search
  - **Interior point methods:** Reach the optimum while staying within the feasible region.
    - Advantages: Effective in large-scale problems, polynomial time algorithm
    - Disadvantages: Complex implementation, starting point requirement
    - Examples: Primal-dual interior point, barrier methods
- **Stochastic/Metaheuristic Methods**
  - **Evolutionary algorithms:** Mimic natural selection and genetic mechanisms.
    - Advantages: Potential to find global optimum, no derivatives required, suitable for parallel processing
    - Disadvantages: High computational cost, sensitive parameter tuning
    - Examples: Genetic algorithms, differential evolution, evolution strategies
  - **Swarm-based algorithms:** Search for solutions by modeling collective behaviors of animal groups.
    - Advantages: Easy implementation, effective in non-convex problems
    - Disadvantages: Weak theoretical guarantees, variable solution quality
    - Examples: Particle swarm optimization, ant colony optimization, bee algorithm
  - **Physics-based algorithms:** Perform optimization by mimicking physical processes.
    - Advantages: Ability to escape local optima
    - Disadvantages: Slow convergence, parameter sensitivity
    - Examples: Simulated annealing, harmony search, big bang-big crunch

- **Hybrid Methods** [20]
  - **Deterministic + Stochastic hybrid:** Combines advantages of both approaches.
    - Advantages: Combining global search with local optimization
    - Disadvantages: Complex implementation, difficult parameter tuning
    - Examples: Memetic algorithms, multi-start gradient methods
  - **Multi-level approaches:** Handles the problem at different detail levels.
    - Advantages: Ability to solve large and complex problems
    - Disadvantages: Problem-specific design requirement
    - Examples: Coarse-fine grid methods, hierarchical optimization
  - **Adaptive strategies:** Changes algorithm parameters or strategies during the optimization process.
    - Advantages: Dynamic adaptation to problem characteristics
    - Disadvantages: Complex control mechanisms
    - Examples: Adaptive metaheuristic methods, self-adaptive evolution strategies [21]

[20] Hybrid methods combine the advantages of different approaches to provide more robust and effective solutions. For example, starting with a genetic algorithm for global search and then refining promising regions with a local gradient-based algorithm both increases the chance of finding the global optimum and ensures reaching a precise solution.

[21] In structural optimization problems, since the computational cost of finite element analysis is generally high, algorithms that perform as few function evaluations as possible are preferred. Therefore, surrogate model-based optimization methods are becoming increasingly popular. These methods speed up the optimization process by using approximate models that can be calculated faster instead of real analysis.

Optimization Algorithm Selection

Which algorithm is most suitable for an optimization problem depends on the problem's characteristics:

- **Problem size:** Interior point methods, gradient-based methods, or specially designed metaheuristic methods are suitable for large-scale problems.
- **Derivative information:** If derivative calculation is possible and economical, gradient-based methods are generally faster. If derivative calculation is difficult or impossible, direct search or metaheuristic methods are preferred.
- **Problem nature:** Deterministic methods are generally sufficient for convex problems. Metaheuristic or hybrid methods are more suitable for non-convex, multimodal problems.
- **Computational budget:** If computational resources are limited, more efficient deterministic methods may be preferred. If high computational power is available, more comprehensive global search methods can be used.
- **Importance of feasible solution:** If feasible solutions are needed in intermediate iterations, methods that stay within the feasible region (interior point, barrier methods) may be preferred.

3 Theoretical Foundations of Classical Optimization Algorithms

Traditional optimization methods not only form the fundamental principles of modern algorithms but also remain among the most reliable tools for solving engineering problems. In this section, we will provide an in-depth look at the mathematical foundations of these methods and examine how they are applied to structural optimization problems.

3.1 Analytical Foundations of Mathematical Optimization

Optimization has been one of the most fundamental problems throughout mathematical history. Even before Leibniz and Newton developed differential calculus, mathematicians were dealing with optimization problems. In the modern sense, the analytical foundations of mathematical optimization are based on three fundamental concepts: stationarity, convexity, and duality.

Historical Perspective

- **Ancient Greece:** Euclidean geometry and maximum-minimum problems
- **17th Century:** Fermat, Leibniz, and Newton's differential calculus
- **18th Century:** Lagrange and Euler's variational calculations
- **19th Century:** Hamilton and Jacobi's optimization theories
- **20th Century:** Development of linear programming, numerical optimization, and complex algorithms

3.1.1 First and Second Order Optimality Conditions

The most important analytical conditions that form the mathematical basis of optimization are:

First Order Necessary Condition (Stationarity): If x^* is a local minimum, then $\nabla f(x^*) = 0$ must hold. This means that the gradient of the function is zero, i.e., the tangent plane is horizontal.

Second Order Sufficient Condition: If $\nabla f(x^*) = 0$ and the Hessian matrix $\nabla^2 f(x^*)$ is positive definite, then x^* is a strict local minimum. Mathematically, for all $d \neq 0$, the condition $d^T \nabla^2 f(x^*) d > 0$ must be satisfied.
[22]

3.1.2 Special Case of Convex Optimization

Convex optimization is a special case of classical optimization theory and has remarkable properties:

- If the objective function and constraint set are convex, every local minimum is also a global minimum
- In convex problems, critical point conditions guarantee the global optimum
- In structural mechanics, potential energy minimization for linear elastic structures is a convex problem
- Even for non-convex problems, they can often be decomposed into convex subproblems

3.2 Gradient-Based Optimization Algorithms

Gradient-based algorithms are methods that systematically approach the optimum point using derivative information of the function. These methods are of great importance in structural optimization problems in terms of computational efficiency and convergence properties.

[22] Second order conditions help us determine whether a critical point (where the gradient is zero) is a local minimum, local maximum, or a saddle point. In structural optimization problems, the evaluation of the Hessian matrix is critical for the algorithm's progress.

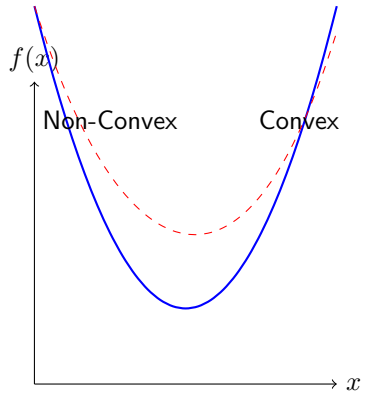

Figure 13: Comparison of convex and non-convex functions

3.2.1 Gradient Descent Method and Its Variations

The gradient descent method is the most basic and oldest gradient-based optimization method. This method aims to reach the minimum point by moving in the direction of the steepest descent of the function.

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \tag{14}$$

Here, α_k represents the step size (or learning rate) and significantly affects the algorithm's performance.

Gradient Descent Variations

- **Fixed Step Size:** The simplest approach uses a constant α for all iterations. However, it may be too low for fast convergence or too high for stability.
- **Line Search:** In each iteration, the value of α_k that minimizes $f(x_k - \alpha \nabla f(x_k))$ is found. This approach can be implemented with Armijo, Wolfe, or Goldstein conditions.
- **Momentum Gradient Descent:** Considers previous gradient values to reduce getting stuck in local minima:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) + \beta(x_k - x_{k-1})$$

- **Nesterov Accelerated Gradient:** An improved version of momentum gradient descent, it calculates the gradient not at the current position but at the point where momentum would take it.

[23]

Gradient Calculation in Structural Optimization In structural optimization problems, three methods are typically used for gradient calculation:

- **Analytical Gradient:** Obtained through direct differential calculation. Provides the most accurate result but derivative derivation can be difficult in complex systems.
- **Finite Differences:** Gradient is calculated using numerical approximation:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}$$

  Easy to compute but the choice of h is sensitive and computational cost is high for large systems.
- **Adjoint Method:** Particularly efficient for large-scale structural problems. Requires a constant number of system solutions independent of the number of design variables. Frequently used in finite element analysis.

3.2.2 Newton and Quasi-Newton Methods

The Newton method is one of the most powerful tools in classical optimization. It uses second-order derivative information (Hessian matrix) to create a local quadratic approximation of the function and moves directly toward the minimum point of this approximation.

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \tag{15}$$

[23] In structural optimization problems, the gradient descent method is typically used in the initial stages of large-scale problems. Although its convergence rate is low, its computational cost is low and it can provide a good starting point for complex problems.

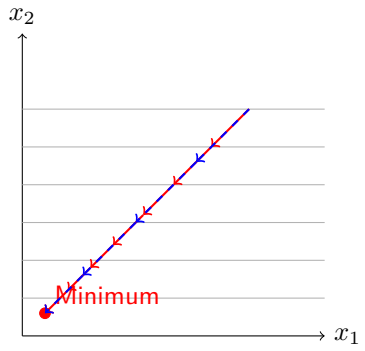

Figure 14: Comparison of gradient descent (red, solid) and momentum gradient descent (blue, dashed) methods

The greatest advantage of the Newton method is its quadratic convergence rate; that is, as it approaches the optimum point, the error decreases approximately by the square in each iteration. However, calculating and inverting the Hessian matrix is computationally expensive, especially for high-dimensional problems.

Modified Newton Methods Various modifications have been developed to reduce computational cost and achieve better performance in cases where the Newton method might be unstable:

- **Levenberg-Marquardt Algorithm:** Modifies the Hessian matrix to make it more stable:

$$x_{k+1} = x_k - [\nabla^2 f(x_k) + \lambda I]^{-1} \nabla f(x_k)$$

  Here, λ is a damping parameter that adjusts the algorithm's behavior.

- **Trust Region Newton Method:** Defines a "trust region" in each iteration where the Hessian matrix is valid and minimizes the quadratic model within this region. Particularly common in structural optimization.

Quasi-Newton Methods To eliminate the computational cost of calculating the full Hessian matrix, quasi-Newton methods iteratively update an approximation of the Hessian matrix. The most popular quasi-Newton methods are:

$$x_{k+1} = x_k - B_k^{-1} \nabla f(x_k) \tag{16}$$

Here, B_k is the approximation of the Hessian matrix at the kth iteration.

Basic Quasi-Newton Algorithms

- **BFGS (Broyden-Fletcher-Goldfarb-Shanno):** The most widely used quasi-Newton method. Maintains the positive definiteness of the Hessian approximation, which increases the algorithm's stability. Frequently preferred in structural optimization problems.
- **DFP (Davidon-Fletcher-Powell):** An older method than BFGS, but generally not as stable.
- **SR1 (Symmetric Rank-One):** Requires less computation but does not guarantee positive definiteness.
- **L-BFGS (Limited-memory BFGS):** A memory-optimized version of BFGS developed for very high-dimensional problems. Preferred in large-scale structural optimization problems, especially in topology optimization.

Newton and Quasi-Newton Applications in Structural Optimization In structural engineering, Newton and quasi-Newton methods are frequently used in the following types of problems:

- **Shape Optimization:** In optimizing the external geometry of structures, especially for aerodynamic performance or thermal behavior
- **Cross-Section Optimization:** In optimizing element cross-sections of truss and frame structures, especially in structures exhibiting nonlinear behavior

- **Parameter Calibration:** In calibrating finite element models with experimental data, determining material parameters
- **Multi-physics Optimization:** In complex engineering problems where structural, thermal, and fluid behaviors are optimized together

3.3 Constrained Optimization and Duality Theory

Engineering problems rarely appear unconstrained. In structural design, there are many constraints such as stress limits, displacement boundaries, physical constraints, and resource limitations. In this section, we will examine in depth the classical approaches to solving constrained optimization problems.

3.3.1 Lagrange Multipliers Method and Theoretical Foundations

The Lagrange multipliers method is a fundamental approach developed for solving equality-constrained optimization problems. Developed by Joseph-Louis Lagrange in the 18th century, this method is one of the cornerstones of optimization theory and variational calculus.

An equality-constrained optimization problem is expressed as:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{17}$$

The Lagrange function (or Lagrangian) is defined as:

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{j=1}^{p} \lambda_j h_j(x) \tag{18}$$

Here, the λ_j values are called Lagrange multipliers and represent the "shadow price" or marginal value of each constraint.
[24]

Lagrange Conditions The necessary conditions for a local minimum of a constrained optimization problem are:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= \nabla f(x^*) + \sum_{j=1}^{p} \lambda_j^* \nabla h_j(x^*) = 0 \\ \nabla_\lambda \mathcal{L}(x^*, \lambda^*) &= h_j(x^*) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{19}$$

These conditions express that at the optimal point, the gradient of the objective function must be equal to a linear combination of the gradients of the constraint functions. Geometrically, this means that the level curves of $f(x)$ and $h_j(x)$ must be tangent.

Second Order Conditions To determine whether a critical point is actually a minimum, the extended Hessian matrix must be examined. If this matrix is positive definite in the tangent space of the constraints, the critical point is considered a local minimum.

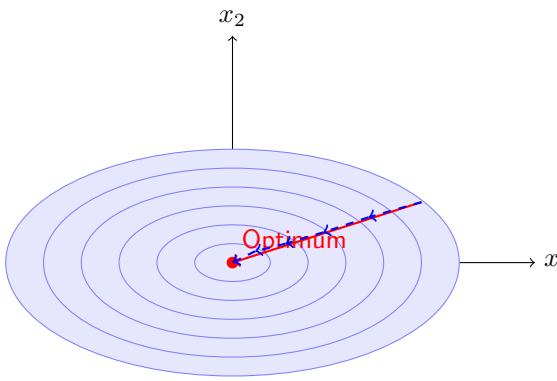

Figure 15: Comparison of Newton method (red, solid) and gradient descent method (blue, dashed). The Newton method can reach the optimum in a single step for quadratic functions.

[24] In structural engineering, Lagrange multipliers often have physical meaning. For example, in an optimization problem where we use force equilibrium as a constraint, the Lagrange multipliers typically correspond to displacements. This duality is a fundamental concept in finite element analysis and structural optimization.

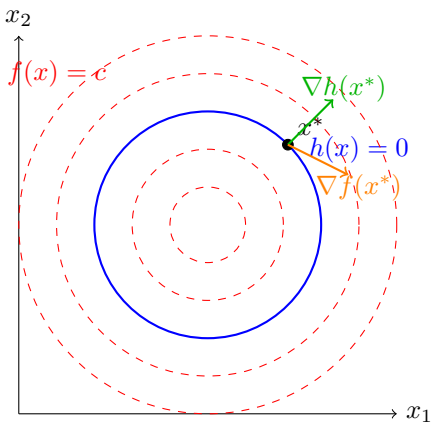

Figure 16: Geometric interpretation of the Lagrange multipliers method: At the optimum point, the level curve of the objective function is tangent to the level curve of the constraint function.

Applications of Lagrange Method in Structural Optimization

- **Steel Frame Structures:** Ensuring force equilibrium at nodes and element compatibility while minimizing weight
- **Composite Material Design:** Ensuring volume constraint and material balance conditions while maximizing stiffness
- **Truss Systems:** Ensuring isostatic equilibrium conditions while minimizing weight
- **Finite Element Analysis:** Ensuring kinematic compatibility and force equilibrium constraints in finite element formulations based on energy minimization principle

3.3.2 Karush-Kuhn-Tucker (KKT) Conditions and Inequality Constraints

The Karush-Kuhn-Tucker (KKT) conditions are a generalization of the Lagrange multipliers method for inequality-constrained problems. These conditions provide a fundamental framework for characterizing optimal solutions in the field of nonlinear programming.

An inequality-constrained optimization problem is expressed as:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{20}$$

The Lagrange function in this case is:

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) + \sum_{j=1}^{p} \lambda_j h_j(x) \tag{21}$$

KKT Conditions The necessary conditions for a local minimum of an inequality-constrained problem (KKT conditions) are:

$$\nabla f(x^*) + \sum_{i=1}^{m} \mu_i^* \nabla g_i(x^*) + \sum_{j=1}^{p} \lambda_j^* \nabla h_j(x^*) = 0$$

$$\begin{aligned} g_i(x^*) &\leq 0, \quad i = 1, \ldots, m \\ h_j(x^*) &= 0, \quad j = 1, \ldots, p \\ \mu_i^* &\geq 0, \quad i = 1, \ldots, m \\ \mu_i^* g_i(x^*) &= 0, \quad i = 1, \ldots, m \end{aligned} \tag{22}$$

The last condition is known as "complementary slackness" and expresses that a constraint must either be active (satisfied as equality) or its corresponding Lagrange multiplier must be zero.

Interpretation of KKT Conditions

- **Stationarity:** The gradient of the objective function must be expressed as a linear combination of the gradients of active constraints. This shows that at the optimum point, progress is not possible without violating at least one active constraint.
- **Primal Feasibility:** All constraints must be satisfied.
- **Dual Feasibility:** The Lagrange multipliers (Kuhn-Tucker multipliers) for inequality constraints must be non-negative. This shows that the direction of constraints is important.
- **Complementary Slackness:** If any constraint is not active (satisfied as inequality), its corresponding Lagrange multiplier must be zero. This means the constraint is "inactive."

Importance of KKT Conditions in Structural Optimization Structural optimization problems typically involve numerous inequality constraints. For example:

- Stress constraints: $\sigma_i \leq \sigma_{allow}$
- Displacement constraints: $|u_i| \leq u_{allow}$
- Buckling constraints: $P_i \leq P_{cr,i}$
- Geometric constraints: minimum thickness, width, etc.

In such problems, KKT conditions provide not only a mathematical tool but also a framework with physical meaning. For example, the Lagrange multiplier associated with a stress constraint shows the effect of a unit stress increase at the relevant point on the weight of the optimal design. This is a fundamental concept for "design sensitivity analysis."

Duality Theory and Economic Interpretation Lagrange duality examines the relationship between the primal problem and its dual problem. The dual problem involves maximizing the Lagrange function instead of minimizing it:

$$\max_{\mu \geq 0, \lambda} \min_{x} \mathcal{L}(x, \lambda, \mu) \tag{23}$$

The duality theorem states that for a convex primal problem, the optimal value of the dual problem equals the optimal value of the primal problem (strong duality). This theorem forms the basis of many optimization algorithms.

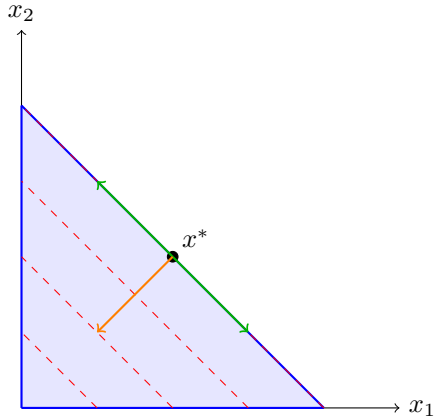

Figure 17: Geometric interpretation of KKT conditions: At the optimum point, the negative gradient of the objective function lies in the convex cone of the gradients of active constraints.

From an economic perspective, dual variables (Lagrange multipliers) represent "shadow prices." In structural optimization, this shows the effect of a marginal change in a constraint on the value of the optimal design. This interpretation allows engineers to understand which constraints have the greatest impact on the design.

Relationship Between Duality and Structural Analysis

In structural mechanics, the relationship between potential energy minimization (displacement method) and complementary energy minimization (force method) is a perfect example of the duality concept in optimization theory.
For the analysis of a structure:

- **Primal Problem (Displacement Method):** Minimizing potential energy to find the compatible displacement field
- **Dual Problem (Force Method):** Minimizing complementary energy to find the force distribution in equilibrium

This duality is also used in the design of structural optimization algorithms. Particularly, "primal-dual" methods solve both the primal and dual problems simultaneously, providing more efficient optimization.

3.4 Linear and Quadratic Programming

An important subset of classical optimization methods consists of linear and quadratic programming methods. These methods provide efficient solution algorithms specifically developed for optimization problems with certain structures.

3.4.1 Linear Programming and Structural Applications

Linear programming (LP) examines optimization problems where both the objective function and constraints are linear. Standard form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{24}$$

Linear programming made significant progress in the 1940s with George Dantzig's development of the Simplex algorithm. Today, Interior Point Methods are also widely used for large-scale linear programming problems.

Simplex Method and Geometric Interpretation The Simplex method aims to find the optimal solution by systematically examining the corner points (extreme points) of the feasible region. The fundamental theorem of LP states that the optimal solution must be at a corner point of the feasible region (if the solution is not unique).

The steps of this method:

- Find a feasible corner point for initialization
- Move to a neighboring corner point that improves the objective function value
- Continue until no better neighboring corner point can be found

LP Applications in Structural Engineering

- **Plastic Limit Analysis:** Determining the collapse load of structures, optimization of plastic hinge distribution
- **Minimum Weight Design of Truss Systems:** Optimization of element cross-sections under linear behavior and static load conditions
- **Structural Resource Allocation:** Resource distribution to maximize structural performance under limited material or budget conditions
- **Transportation Network Optimization:** Optimization of bridge and road system placement, modeling of traffic flow

Plastic Limit Analysis Example Plastic limit analysis is one of the most important applications of linear programming in structural engineering. Static theorem (lower bound) formulation:

$$\begin{aligned} \max \quad & \lambda \\ \text{s.t.} \quad & B^T q = \lambda F + F_0 \\ & |q_i| \leq q_i^p, \quad i = 1, \ldots, n \end{aligned} \tag{25}$$

Where:

- λ: load factor
- q: internal forces vector
- B: equilibrium matrix
- F: variable external load vector
- F_0: constant external load vector
- q_i^p: plastic limit capacity for element i

This formulation determines the maximum load factor that the structure can withstand without collapse.
[25]

3.4.2 Quadratic Programming

Quadratic programming (QP) deals with optimization problems where the objective function is quadratic and the constraints are linear:

$$\begin{aligned} \min \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{26}$$

Here, Q is a symmetric matrix. If Q is positive (semi) definite, the problem is convex and finding the global optimum is relatively easy.

[25] Plastic limit analysis is particularly important in the design of steel structures. By determining the maximum load that the structure can withstand using its plastic deformation capacity beyond the elastic limit, it enables more economical designs.

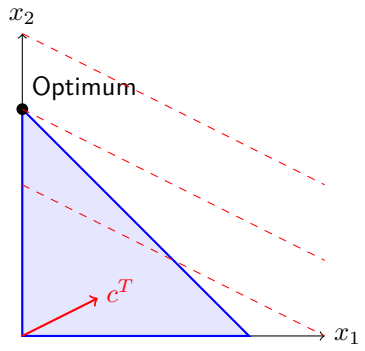

Figure 18: Feasible region and optimum point in linear programming

Solution Methods Various solution methods have been developed for quadratic programming problems:

- **Active Set Methods:** Solves unconstrained optimization subproblems by making predictions about which constraints are active
- **Interior Point Methods:** Approaches the optimum point by passing through the interior of the feasible region
- **Sequential Quadratic Programming (SQP):** Solves nonlinear optimization problems by dividing them into quadratic subproblems

QP Applications in Structural Engineering

- **Elastic Deformation Minimization:** Using the structure's stiffness matrix to minimize deformation under specific loads
- **Dynamic Behavior Optimization:** Using the structure's mass and stiffness matrices to optimize dynamic response
- **Finite Element Model Calibration:** Minimizing the squares of differences between experimental and numerical data
- **Weight and Displacement Optimization:** Multi-objective optimization involving both weight and displacement energy

Structural Compliance Minimization Example A common quadratic programming problem in structural design is the minimization of compliance (or deformation energy) under a volume constraint:

$$\begin{aligned} \min \quad & \frac{1}{2} F^T u \\ \text{s.t.} \quad & Ku = F \\ & \sum_{e=1}^{n_e} v_e \rho_e \leq V \\ & 0 \leq \rho_e \leq 1, \quad e = 1, \ldots, n_e \end{aligned} \tag{27}$$

Where:

- u: displacement vector
- F: force vector
- K: global stiffness matrix
- ρ_e: material density for element e (design variable)
- v_e: volume of element e
- V: allowed total volume

This formulation forms the basis of topology optimization and is the starting point of the SIMP (Solid Isotropic Material with Penalization) method.

3.5 Modern Classical Optimization Algorithms

Modern extensions of classical optimization methods aim to solve large and complex structural optimization problems using various approaches. In this section, we will discuss more advanced versions of classical methods.

3.5.1 Secant Methods and Structural Applications

Secant methods are approaches developed to eliminate the requirement of calculating the Hessian matrix in the Newton method. These methods create an approximate estimate of the Hessian matrix using gradient information.

The most common secant methods are:

- **Broyden Method:** Used in solving general nonlinear equation systems
- **DFP (Davidon-Fletcher-Powell) Method:** One of the first examples of quasi-Newton methods
- **BFGS (Broyden-Fletcher-Goldfarb-Shanno) Method:** Has become standard in many fields including structural optimization

In the BFGS method, the approximate value of the Hessian matrix is updated as follows:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{28}$$

Where:

- $s_k = x_{k+1} - x_k$
- $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

The BFGS method is widely used in structural optimization problems, especially when there are many design variables.

Advantages of BFGS in Structural Optimization

- **Memory Efficiency:** Applicable even to large-scale problems with the L-BFGS variant
- **Superlinear Convergence:** Convergence rate increases as it approaches the optimum point
- **Numerical Stability:** More stable than the Newton method
- **Integration with Line Search:** Can be combined with strong line search strategies

3.5.2 Trust Region and Line Search Strategies

The performance of gradient-based optimization methods depends largely on step size selection. Line search and trust region are two fundamental strategies developed for determining step size.

Line Search Strategies In line search methods, after determining the search direction d_k, a suitable step size α_k is found:

$$x_{k+1} = x_k + \alpha_k d_k \tag{29}$$

Various criteria are used to determine the optimal step size:

- **Armijo Condition:** Guarantees that the step size provides sufficient decrease in the function value
- **Wolfe Conditions:** In addition to the Armijo condition, requires that the gradient change satisfies a certain ratio
- **Goldstein Conditions:** Ensures both decrease in function value and that the step size is not too small

Trust Region Approach In trust region methods, a region where the model is reliable is defined and the model is minimized within this region:

$$\begin{aligned} \min \quad & m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \\ \text{s.t.} \quad & \|p\| \leq \Delta_k \end{aligned} \tag{30}$$

Where:

- $m_k(p)$: quadratic model of the function at the kth iteration
- $f_k = f(x_k)$, $g_k = \nabla f(x_k)$
- B_k: Hessian matrix or its approximation
- Δ_k: trust region radius

In each iteration, the trust region radius is updated by comparing the model prediction with the actual function change.

Trust Region Applications in Structural Engineering

The trust region approach is particularly preferred in structural engineering in the following cases:

- **Ill-Conditioned Problems:** When the condition number of the stiffness matrix is high
- **Nonlinear Behavior:** In structural analyses involving material or geometric nonlinearity
- **Unstable Systems:** In problems close to instability points such as buckling analysis
- **Multi-physics Analyses:** In complex multi-physics problems such as thermal-structural, fluid-structure interaction

3.5.3 Sequential Constraint Programming

Sequential Constraint Programming (SCP) is an effective approach developed for solving structural optimization problems. This method solves the nonlinear problem by transforming it into a series of linear subproblems.

The basic approach of SCP:

- Linearizes nonlinear constraints around the current iteration point
- Solves the linearized problem
- Uses the solution as a new iteration point
- Repeats until convergence

This method, with its more advanced variants such as MMA (Method of Moving Asymptotes), is widely used in structural topology optimization.

The main advantages of MMA in structural optimization:

- Efficiency in large-scale topology optimization problems
- Effective handling of nonlinear constraints
- Stable convergence by reducing oscillations
- Suitability for parallel computing

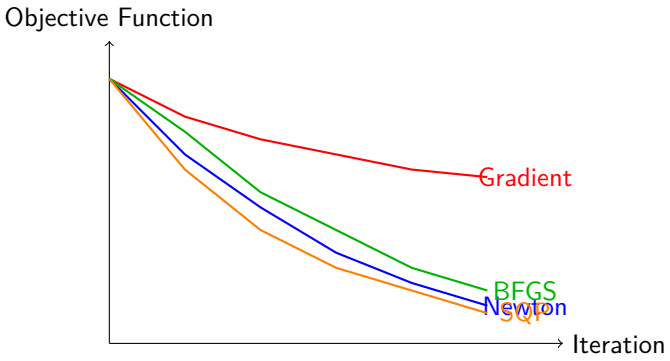

Figure 19: Comparison of convergence behaviors of different optimization algorithms

3.6 Conclusion and Transition to Modern Methods

Classical optimization methods still hold great importance in solving various problems encountered in structural engineering. These methods form the foundation of modern meta-heuristic algorithms and artificial intelligence-based approaches.

Comparison of Classical and Modern Methods

- **Classical Methods:** Mathematical robustness, exact convergence guarantees, efficiency
- **Modern Methods:** Global search for multi-modal functions, parallel computing, flexibility in handling complex constraints

Today, hybrid methods that combine the strengths of classical algorithms with the flexibility of modern approaches are receiving great interest. These hybrid approaches provide effective solutions, especially in large-scale and multi-objective structural optimization problems.

4 Benchmark Test Functions

Python codes for the test functions discussed under this heading can be accessed through the link. [26]

Standard test functions used to evaluate and compare the performance of optimization algorithms[27] appear as mathematical structures with different difficulty levels. These test functions are designed to challenge algorithms with their known global minimum points, complex local minimum structures, and various topological features.

4.1 Role of Benchmark Functions in Optimization

Standard test functions are used to evaluate the performance of optimization algorithms. These functions are important from the following perspectives:

- **Comparability:** Enables objective evaluation of different algorithms' success on the same problem.
- **Identifying Strengths and Weaknesses:** Shows which types of problems algorithms perform well on and which they struggle with.

[26]

[27] Test functions play an important role in revealing the strengths and weaknesses of algorithms. An algorithm's success in real-world problems is first evaluated on these test functions.

- **Algorithm Development:** Guides the development and improvement of new optimization algorithms.
- **Preparation for Real-World Applications:** Ensures algorithms are tested before being applied to more complex real-world problems.

An ideal benchmark function should reflect the complexity of real-world optimization problems while being mathematically understandable and analyzable. Benchmark functions are generally classified according to the following characteristics: linearity, modality (number of peaks), continuity, differentiability, boundedness, and dimensionality.

4.1.1 Common Characteristics of Benchmark Functions

Most benchmark functions have the following characteristics:

- Known global minimum value and location
- Mathematically defined structure
- Adjustable dimension (usually extendable to multidimensional spaces)
- Various challenges (e.g., local minima, flat regions, steep valleys)
- Recommended search ranges

4.2 Multimodal and Unimodal Test Functions

Test functions are divided into two categories based on the number of peaks (modes) they contain: unimodal and multimodal.

4.2.1 Unimodal Functions

Unimodal functions have only one global minimum and are generally simpler in structure. These functions are used to test the convergence speed and precision of algorithms. Examples include:

- **Sphere Function:** The mathematically simplest optimization test function, defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{31}$$

  Global minimum $f(\mathbf{x}^*) = 0$ is located at $\mathbf{x}^* = (0, 0, \ldots, 0)$.

- **Booth Function:** A two-dimensional, bowl-shaped function:

$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2 \tag{32}$$

  Global minimum is at $f(1, 3) = 0$.

4.2.2 Multimodal Functions

Multimodal functions have multiple local minima and thus test algorithms' ability to find the global minimum without getting stuck in local minima. These functions are particularly important for evaluating the performance of metaheuristic and evolutionary algorithms. Important examples include:

- **Ackley Function:** A complex test function with many local minima. Mathematically defined as:

$$f(\mathbf{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \tag{33}$$

  This function's global minimum is $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, 0, \ldots, 0)$. It is typically defined in the range $[-32.768, 32.768]$.

- **Rastrigin Function:** A challenging function with many local minima due to sinusoidal modulation:

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{34}$$

  Global minimum is $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, 0, \ldots, 0)$.

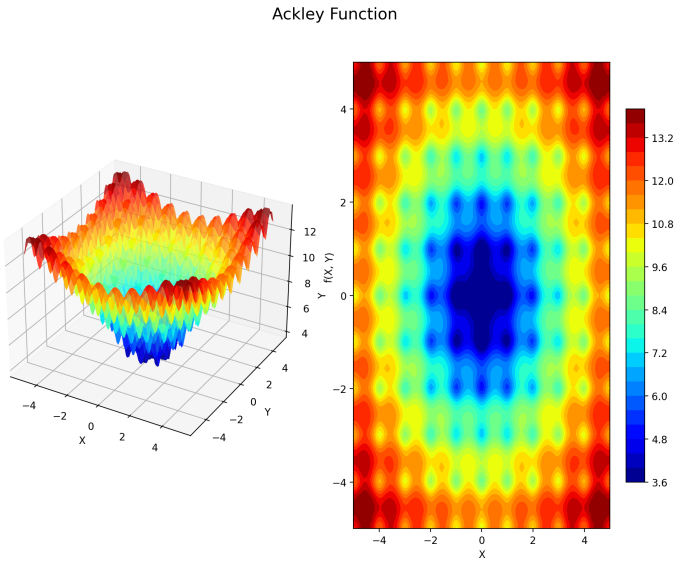

Figure 20: 3D visualization of the Ackley function (left) and contour plot (right). The function's structure with a global minimum at the center and numerous local minima around it is notable.

4.2.3 Ackley Function: In-Depth Analysis

The Ackley function, proposed by David Ackley in 1987, has become a standard test function for optimization algorithms. The function's mathematical structure reveals the following characteristics:

- **Nearly flat regions:** As we move away from the center, the function has an almost flat structure, making it difficult for gradient-based methods to determine the correct direction.
- **Periodic fluctuations:** Due to the cosine term, the function surface shows regular ups and downs, creating many local minima.
- **Deep well at the center:** There is a steep well around the global minimum, requiring the algorithm to take precise steps when close to the optimum.

The Ackley function is particularly effective in testing the following types of algorithms:

- Metaheuristic algorithms that balance local search with global search strategies
- Algorithms capable of escaping from numerous local minima
- Population-based algorithms that can explore different solution regions simultaneously

The optimization of the n-dimensional Ackley function involves the following challenges:

- The number of local minima increases exponentially with dimension
- High precision is required in regions close to the center
- Gradient information becomes insufficient in flat regions

Ackley Function Optimization Challenge

The Ackley function tests both exploration and exploitation characteristics simultaneously:

- Exploration: Finding the correct direction in wide, nearly flat regions
- Exploitation: Making precise adjustments in the steep well at the center
- Balance: Being able to converge to the global minimum while escaping local minima

4.3 High-Dimensional Optimization Problems

4.3.1 Effects of Dimension Increase

- Search space grows exponentially
- Computational cost increases
- Number of local minima increases
- Convergence becomes more difficult

High-dimensional optimization problems appear in many real-world engineering applications. The increase in dimension leads to the phenomenon known as the "curse of dimensionality." This phenomenon is characterized by the exponential growth of the search space and dramatic decrease in algorithm effectiveness as dimension increases.

4.3.2 Effects of the Curse of Dimensionality

The effects of dimension increase on the optimization process:

- **Search space width:** In a problem with n dimensions and m discrete points in each dimension, the total search space is of size m^n. For example, with 10 points in each dimension, a 2D problem has 100 points, a 10D problem has 10^{10} points, and a 100D problem has 10^{100} points.

- **Data sparsity:** In high dimensions, distances between data points increase and data becomes sparse. This makes it difficult for algorithms to determine the correct direction.
- **Sampling difficulty:** The number of points needed to sufficiently sample high-dimensional space is impractically large.

4.3.3 High-Dimensional Test Functions

Some test functions are specifically designed to evaluate algorithms' performance in high-dimensional problems:

- **Rosenbrock Function:** A function that progresses along a narrow valley and becomes more challenging as dimension increases:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{35}$$

- **Schwefel Function:** Has many wide-region local minima in high dimensions:

$$f(\mathbf{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}) \tag{36}$$

4.4 Testing Stochastic Algorithms

Stochastic algorithms are non-deterministic algorithms with random elements. These algorithms can produce different results each time they are run. Modern optimization methods such as metaheuristic algorithms, evolutionary algorithms, and artificial neural networks typically have stochastic characteristics.

4.4.1 Testing Principles for Stochastic Algorithms

- **Multiple runs:** The algorithm should be run multiple times (typically 30-50 independent runs) for each test function.
- **Statistical analysis:** The mean, standard deviation, median, best and worst values of the results should be reported.
- **Convergence analysis:** The algorithm's behavior over time should be shown with a graph of the best value against iteration/evaluation count.
- **Parameter sensitivity:** The algorithm's sensitivity to parameter changes should be tested.

4.5 Testing Principles

Some basic principles should be followed for fair and comprehensive evaluation of optimization algorithms:

- **Variety:** Test functions with different characteristics should be used.
- **Fair comparison:** The same conditions (starting points, function evaluation count, termination criteria) should be provided for all compared algorithms.
- **Sufficient repetition:** Sufficient number of independent runs should be performed for stochastic algorithms.

- **Dimension variation:** Algorithms' performance at different problem dimensions should be tested.
- **Comprehensive reporting:** Not only mean or best values but complete statistical results should be reported.

4.6 Performance Metrics

The performance of optimization algorithms can be evaluated using various metrics. These metrics can be divided into two main categories: numerical and quality metrics.

4.6.1 Numerical Metrics

- **Convergence speed:** Shows how quickly the algorithm reaches the desired solution. Usually measured as function evaluation count or iteration count.
- **Precision:** Shows how close the found solution is to the known global optimum. Usually measured as absolute or relative error.
- **Success rate:** The percentage of times the algorithm reaches an acceptable solution. Particularly important for stochastic algorithms.
- **Computational complexity:** Can be measured as running time or memory usage.

4.6.2 Quality Metrics

- **Robustness:** The algorithm's sensitivity to different problem types, initial conditions, and parameter changes.
- **Generalizability:** The algorithm's performance across different problem classes.
- **Scalability:** Changes in algorithm performance as problem dimension increases.
- **Exploration-exploitation balance:** The algorithm's ability to balance global search (exploration) and local improvement (exploitation).

[28]

No Free Lunch Theorem

No optimization algorithm can show the best performance on all problems:

- Every algorithm has strengths and weaknesses
- Choosing an algorithm suitable for the problem structure is important
- Hybrid approaches may be advantageous

[28] Performance metrics enable objective comparison of different algorithms. However, evaluating multiple metrics together rather than a single metric is more reliable.

4.6.3 Presentation of Benchmark Results

The following methods can be used for effective presentation of benchmark test results:

- **Table format:** Tables showing mean, standard deviation, best and worst values for each test function.
- **Convergence graphs:** Graphs showing the change in best value against iteration/evaluation count.
- **Box plots:** Visual graphs showing the distribution of results and statistical summaries like median and quartiles.
- **Ranking tables:** Tables showing performance rankings of algorithms for each test function.
- **Statistical significance tests:** Tests (t-test, Wilcoxon signed-rank test, Friedman test, etc.) showing whether performance differences between algorithms are statistically significant.

4.7 Categories of Benchmark Functions

Benchmark functions with different characteristics test algorithms' ability to handle various challenges. Here, basic function categories and examples are given:

4.7.1 Functions with Many Local Minima

These functions have many local minima that pose a challenge for algorithms and test global optimization ability:

- Ackley
- Rastrigin
- Griewank
- Schwefel
- Levy
- Shubert

4.7.2 Bowl-Shaped Functions

These functions have a single minimum surrounded by circular or elliptical contours and test local optimization ability:

- Sphere
- Bohachevsky
- Sum Squares
- Rotated Hyper-Ellipsoid

4.7.3 Plate-Shaped Functions

These functions have flat regions that can challenge algorithms' ability to determine the correct direction:

- Booth
- Matyas
- Zakharov
- McCormick

4.7.4 Valley-Shaped Functions

These functions have long, narrow valleys that can slow down the convergence of many algorithms:

- Rosenbrock (Banana Function)
- Six-Hump Camel
- Three-Hump Camel
- Dixon-Price

4.7.5 Functions with Steep Ridges/Drops

These functions have steep ridges or drops that can challenge gradient-based methods:

- Easom
- Michalewicz
- De Jong N.5

4.8 Conclusion and Applications

Benchmark test functions are indispensable tools for the development, testing, and comparison of optimization algorithms. These functions help evaluate algorithms' performance on different problem types and identify their weaknesses.

- **In algorithm development:** Determining strengths and weaknesses of new algorithms
- **In parameter tuning:** Optimization of algorithm parameters
- **In hybrid approaches:** Development of hybrid methods combining strengths of different algorithms
- **In education:** Understanding the behavior of optimization algorithms
- **In preparation for real-world problems:** Evaluating basic capabilities of algorithms before moving to more complex problems

5 Metaheuristic Optimization Algorithms I

Nature-inspired modern algorithms used in solving complex optimization problems will be discussed in this section. These algorithms provide effective solutions when classical methods prove insufficient.

5.1 Basic Characteristics of Metaheuristic Algorithms

Metaheuristic algorithms are nature-inspired methods used in solving complex optimization problems:

- Have stochastic characteristics
- Problem-independent structure
- Do not require gradient information
- Potential to reach global optimum

[29]

[29] The most important characteristic of metaheuristic algorithms is their potential to reach the global optimum without getting stuck in local optima in complex and multimodal problems.

5.2 Differences Between Deterministic and Stochastic Algorithms

Deterministic vs Stochastic

- **Deterministic:**
  - Same starting point → Same result
  - Gradient-based
  - Fast convergence to local optimum
- **Stochastic:**
  - Random search
  - Different result in each run
  - Global optimum potential

5.3 Classification of Metaheuristic Algorithms Based on Search Method

Metaheuristic algorithms are fundamentally divided into two categories based on search methods: population-based and single-solution search algorithms. Population-based algorithms evaluate multiple solution candidates simultaneously, while single-solution search algorithms work on a single solution. This classification provides an important framework for understanding the working principles and optimization strategies of algorithms.[30]

[30] This context can be examined through the link to better understand the working principles of algorithms.

5.3.1 Population-Based Algorithms

Population-based algorithms are methods that evaluate multiple solution candidates simultaneously during the optimization process and benefit from interactions between these solutions. These algorithms increase the probability of reaching the global optimum by exploring different regions of the search space simultaneously. Population-based approaches reduce the risk of getting stuck in local optima by maintaining the diversity of solution candidates and provide effective results in complex, multimodal problems.

The most common examples of population-based algorithms include Genetic Algorithms, Particle Swarm Optimization, and Differential Evolution. In these algorithms, each individual (solution candidate) in the population evolves according to specific rules and interacts with others. For example, Genetic Algorithms use crossover and mutation operators, while in Particle Swarm Optimization, particles move by benefiting from their own experience and the collective knowledge of the swarm. These interactions ensure that the algorithm uses both exploration and exploitation capabilities in a balanced way.

5.3.2 Single-Solution Search Based

Single-solution search based algorithms are methods that work on a single solution candidate during the optimization process and improve this solution step by step. These algorithms progress toward a better solution by evaluating potential solutions in the neighborhood of the current solution. The single-solution search approach generally offers advantages such as lower memory usage and faster iteration times, but carries the risk of getting stuck in local optima.

The most common single-solution search based metaheuristic algorithms include Simulated Annealing, Tabu Search, and Variable Neighborhood Search. Simulated Annealing, inspired by the annealing process of metals, accepts poor solutions with a certain probability at the beginning and gradually reduces this probability over time. Tabu Search prevents cyclic movements by marking recently visited solutions as "tabu" and enables exploration of wider regions of the search space.

Characteristics of Single-Solution Search Based Algorithms

- **Advantages:**
  - Low memory requirement
  - Fast iteration times
  - Simple implementation
  - Local search capabilities
- **Disadvantages:**
  - Risk of getting stuck in local optima
  - Limited exploration ability in wide search spaces
  - Dependence on initial solution

Additionally, for these types of meta-heuristic algorithms to be successful, some mechanisms to avoid local optima are generally developed. For example, the temperature parameter in the Simulated Annealing algorithm allows escaping from local optima by moving away from the current best solution. In this algorithm specifically, this probability is higher in early iterations and decreases in later iterations. Similarly, many algorithms contain different forms of mechanisms to avoid local optima.

5.4 Classification of Metaheuristic Algorithms Based on Search Strategy

5.4.1 Global Search Focused Algorithms

Global search focused algorithms are methods that focus on exploring a large portion of the solution space. These algorithms aim to reach the global optimum by systematically or randomly sampling different regions of the search space. Global search strategies are particularly important in reducing the risk of getting stuck in local optima, especially in multimodal and complex optimization problems. This approach helps identify regions where potentially better solutions might be found by exploring a wider portion of the search space.

Population-based methods such as Genetic Algorithms and Particle Swarm Optimization naturally have global search capabilities. For example, in Genetic Algorithms, high mutation rates and diversity preservation strategies increase the algorithm's exploration ability. Similarly, appropriate values of control parameters (F and CR) in the Differential Evolution algorithm strengthen the algorithm's global search ability. These algorithms tend to explore wide regions

of the search space, especially in the initial stages, and focus on more promising regions over time. Global search strategies, although computationally expensive, offer the potential to reach higher quality solutions, especially in previously unknown or complex optimization problems.

5.4.2 Local Search Focused Algorithms

Local search focused algorithms are methods that aim to reach better solutions by investigating the immediate vicinity of the current best solution in detail. These algorithms aim to find the best solution (local optimum) in a specific region by intensively researching that region. Local search generally provides faster convergence and is more computationally efficient. It is particularly effective in unimodal problems or when there is prior knowledge about the global optimum region.

Algorithms such as Hill Climbing, Simulated Annealing, and Tabu Search primarily use local search strategies. In these algorithms, potential solutions in the neighborhood of the current solution are evaluated, and the next step is selected according to specific criteria. For example, in Tabu Search, a tabu list is maintained to prevent re-evaluation of previously visited solutions; this prevents the algorithm from getting stuck in local optima and enables more effective exploration of the search space. Particularly in structural optimization problems, when gradient information can be used, local search strategies can converge quickly from a given starting point. However, the success of these algorithms largely depends on the selection of the starting point, and the risk of getting stuck in local optima is high in complex, multimodal problems.

5.4.3 Hybrid Search

Hybrid search strategies aim to increase the effectiveness of the optimization process by combining the strengths of global and local search methods. In this approach, potential solution regions are typically determined by performing global search in the initial stages of the algorithm, and then local search techniques are applied to improve solutions in these regions. Hybrid strategies provide a balanced way to both explore the wide search space (exploration) and investigate promising regions in detail (exploitation).

Memetic Algorithms are a good example of hybrid search strategies. These algorithms explore the wide search space using evolutionary methods like Genetic Algorithms, then apply local search techniques to each individual (solution candidate) to improve solutions. Similarly, hybrid versions of Particle Swarm Optimization and Gradient Descent methods have been developed. These hybrid approaches provide successful results in complex engineering problems, especially in areas like structural optimization. For example, in topology optimization, the general structure is first determined using a metaheuristic algorithm, then detailed optimization is performed using mathematical programming techniques. Hybrid search strategies provide more effective results than global or local search strategies alone by providing a better balance between computational efficiency and solution quality.

5.5 Classification of Metaheuristic Algorithms Based on Nature-Inspired Sources

5.5.1 Biological Evolution Based

Biological evolution based algorithms are optimization methods developed inspired by Darwin's theory of natural selection and evolution. These algorithms attempt to solve optimization problems by mimicking the adaptation process of living organisms to environmental conditions over generations. The most basic

evolutionary algorithm, the Genetic Algorithm, uses operators inspired by biological evolution: selection, crossover, and mutation. Through these operators, the algorithm evolves the population over generations and ensures the survival of individuals more suitable according to the objective function.

Other evolutionary algorithms such as Differential Evolution, Evolution Strategies, and Genetic Programming apply similar principles in different ways. For example, the Differential Evolution algorithm produces new solutions using vector differences between population members and thus adapts to the characteristics of the search space. Evolution Strategies, on the other hand, improve their performance by using self-adapting mutation parameters, especially in continuous optimization problems. These algorithms are particularly effective in multidimensional, discontinuous, and multimodal optimization problems. Additionally, the ease of adjusting their parameters and adaptability to different problem types enables their widespread use in engineering fields such as structural optimization, mechanical design, and materials science.

5.5.2 Swarm Intelligence Based

Swarm intelligence based algorithms are optimization methods developed inspired by the collective behaviors of swarm-living organisms in nature. In these algorithms, complex and intelligent behaviors emerge as a result of interactions between individuals with simple rules. Particle Swarm Optimization (PSO), inspired by the movements of birds and fish schools, is the most popular swarm intelligence algorithm. In PSO, each particle updates its movement using information about its own best position and the global best position of the swarm. This collective intelligence enables the swarm to quickly find the most efficient resources (optimum points).

Ant Colony Optimization simulates ants' behavior of finding the shortest path using pheromone trails and is particularly effective in combinatorial optimization problems. The Artificial Bee Colony algorithm, inspired by honey bees' nectar collection strategy, enables the exploration of different solution regions and more intensive investigation of efficient regions. The Firefly Algorithm mimics the brightness and attraction principles of fireflies, while the Bat Algorithm simulates bats' echolocation methods. These swarm intelligence based algorithms generally stand out with their ease of application, few control parameters, and global optimization capabilities. They provide successful results particularly in engineering applications such as robot trajectory planning, energy systems optimization, and sensor network positioning.

5.5.3 Physics Process Inspired

Physics process inspired algorithms are optimization methods developed based on physical phenomena and laws in nature. These algorithms apply principles from different physics fields such as thermodynamics, mechanics, electromagnetism, and quantum physics to optimization problems. Simulated Annealing, inspired by the annealing process of metals, is one of the oldest physics-based algorithms. This algorithm mimics the transition of the molecular structure to a low-energy state while the metal is heated to high temperature and slowly cooled. The algorithm, which accepts poor solutions with a high "temperature" parameter at the beginning, becomes more selective as it "cools" over time and increases the chance of converging to the global optimum.

The Gravitational Search Algorithm is based on Newton's universal law of gravitation and simulates mass interactions between solution candidates. The Harmony Search algorithm is inspired by musicians' process of creating harmonious tones during improvisation. The Big Bang-Big Crunch algorithm mimics the expansion and contraction cycles of the universe, while the Water Cycle

Algorithm adapts water's evaporation, precipitation, and flow processes to the optimization process. These physics-based algorithms tend to provide stable and reliable results because they are generally based on well-defined mathematical principles. They are widely used in solving complex and multivariable problems, particularly in engineering design, signal processing, and structural optimization.

5.5.4 Chemical, Biological, or Social Process Inspired

Chemical, biological, or social process inspired metaheuristic algorithms mimic the behaviors of various complex systems in nature and society. Chemical Reaction Optimization is based on molecules' kinetic energy, collisions, and chemical reactions. This algorithm aims to find global minima in optimization problems by mimicking molecules' tendency to reach low-energy states. Among biological process inspired algorithms, Bacterial Foraging Optimization, which simulates bacteria's food search strategies, and Artificial Immune System algorithms, which mimic the immune system's antigen-antibody responses, can be mentioned.

Social process inspired algorithms model human communities' behaviors and social interactions. Teaching-Learning-Based Optimization simulates teacher-student interaction in a classroom, while the Imperialist Competitive Algorithm mimics imperialist countries' struggle for dominance over colonies. Social Group Optimization models human groups' problem-solving methods, while the Artificial Bee Colony algorithm adapts bee colonies' nectar collection strategies to the optimization process. The common feature of these algorithms is that they produce effective solutions in multidimensional and multimodal search spaces by using emergent behaviors in complex systems. They achieve successful results particularly in data mining, neural network training, robotics, and artificial intelligence applications.

5.6 Classification of Metaheuristic Algorithms Based on Exploration and Exploitation Balance

The success of metaheuristic algorithms critically depends on establishing the right balance between exploration and exploitation. Exploration refers to the investigation of unexplored regions of the search space, while exploitation covers the more detailed examination of previously discovered and promising regions. The balance between these two processes directly affects the algorithm's performance. Exploration-heavy algorithms can scan the wide search space more comprehensively and increase the probability of reaching the global optimum, but their convergence rates are generally low. Exploitation-heavy algorithms provide fast convergence in specific regions but carry the risk of getting stuck in local optima.

Most metaheuristic algorithms dynamically adjust the balance between exploration and exploitation throughout the optimization process. For example, in Genetic Algorithms, the mutation operator increases exploration ability, while the crossover operator supports the exploitation process. In Particle Swarm Optimization, the inertia weight and acceleration coefficients control this balance. In the Simulated Annealing algorithm, the decrease in the temperature parameter over time enables the algorithm to transition from exploration-heavy behavior to exploitation-heavy behavior. Algorithms should be designed to balance these two processes according to problem characteristics and the stage of the optimization process. In engineering applications such as structural optimization, especially in complex and multimodal problems, establishing the right balance between exploration and exploitation plays a determining role in reaching the global optimum and computational efficiency.

5.7 Hyperparameter Tuning of Algorithms

Variables that directly affect the performance of metaheuristic algorithms and are determined by the user are called hyperparameters. These parameters significantly affect the algorithm's behavior, convergence speed, and solution quality. For example, population size, mutation rate, and crossover rate in Genetic Algorithms; inertia weight and learning factors in PSO; initial temperature and cooling rate in Simulated Annealing are hyperparameters. Determining the optimal values of these parameters is critical for the algorithm's success and generally varies according to problem characteristics.

5.7.1 Parameter Selection Strategies

- Experimental analysis
- Adaptive tuning
- Meta-optimization
- Statistical design

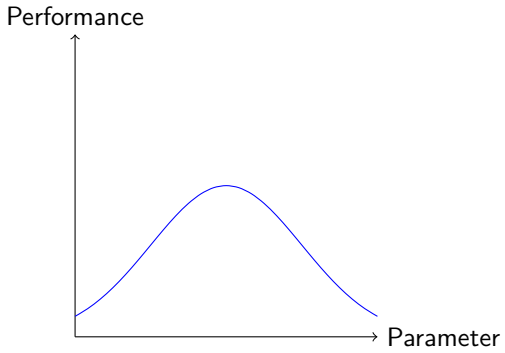

Figure 21: Relationship between parameter value and performance

5.8 Objective Comparison of Optimization Algorithms

The objective comparison of metaheuristic optimization algorithms' performances is of great importance for algorithm selection and development. This comparison is made using standard test functions, real-world problems, and statistical analysis methods. Standard test functions (Benchmark functions) provide problems at different difficulty levels and with different characteristics (multimodal, discontinuous, noisy, etc.), enabling the evaluation of algorithms' performances under various conditions. Common test functions such as Sphere, Rastrigin, Rosenbrock, and Ackley are used to test algorithms' global optimization capabilities, convergence speeds, and exploration-exploitation balances.

In the objective comparison of algorithms, a comprehensive evaluation approach including various problem types and multiple performance criteria should be adopted rather than a single test problem or performance metric. For this purpose, different performance metrics such as solution quality, convergence speed, computational cost, robustness, and scalability are evaluated together. Statistical significance tests (t-test, Wilcoxon signed-rank test, etc.) are used to determine whether performance differences between algorithms result from random variability or a real superiority. Additionally, algorithms' behaviors at different problem dimensions and constraint conditions are also an important part of the comparison.

The No Free Lunch theorem states that no optimization algorithm can be superior to others in all problem classes. Therefore, the comparison of algorithms should focus on determining which algorithm is more suitable for specific problem types or application areas. In engineering fields such as structural optimization, mechanical design, and materials science, choosing an algorithm suitable for problem characteristics is critical in terms of solution quality and computational efficiency. Objective comparison studies guide the algorithm development process by revealing the advantages and disadvantages of newly developed algorithms compared to existing methods and ensure the selection of the most appropriate algorithm for the specific application area.

5.8.1 Comparison Criteria

- Convergence speed
- Solution quality

- Computational cost
- Parameter sensitivity

6 Metaheuristic Optimization Algorithms II

Advanced applications of metaheuristic algorithms and hybrid approaches will be examined in this section. Modern techniques used in solving optimization problems and their applications will be discussed.

6.1 Simulated Annealing (SA)

Simulated Annealing (SA) is a powerful metaheuristic optimization algorithm inspired by the annealing process in metallurgy. In the annealing process of metals, the material is first heated to high temperatures, during which atoms move randomly at high energy levels. Then, the material is slowly cooled in a controlled manner, allowing atoms to settle into a low-energy, stable crystal structure. This physical process has been adapted as an effective strategy for reaching the global optimum in optimization problems. The algorithm accepts poor solutions with a certain probability at the beginning with a high "temperature" parameter, thus exploring a wide search space. As the temperature gradually decreases, the algorithm becomes more selective and performs more intensive search in promising regions. This approach reduces the risk of getting stuck in local optima and increases the probability of converging to the global optimum in complex and multimodal optimization problems.[31]

6.1.1 Algorithm Basis

An optimization algorithm inspired by the annealing process of metals:

- Random movements at high temperature
- Controlled movements as temperature decreases
- Probability of accepting poor solutions

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{kT}\right) \tag{37}$$

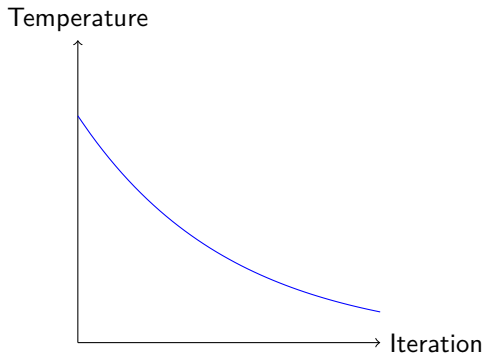

Figure 22: Temperature change in simulated annealing

[31] Simulated annealing provides successful results especially in combinatorial optimization problems. The cooling rate and initial temperature are important parameters affecting the algorithm's performance. Example Python code implementing the simulated annealing algorithm on the Ackley function:

6.1.2 Algorithm Steps

1. Determine initial temperature and solution
2. For each temperature level:
    - Generate new solution
    - Calculate energy difference
    - Accept/reject according to Metropolis criterion
3. Decrease temperature
4. Continue until stopping criterion is met

6.2 Tabu Search Algorithm

The Tabu Search Algorithm (TS), developed by Fred Glover in 1986, is a powerful metaheuristic optimization method inspired by human memory. It mimics the human intelligence's ability to benefit from past experiences and prevent repeating errors in the problem-solving process. The algorithm provides effective results especially in combinatorial optimization problems and is designed to overcome the local optima trapping problem of local search methods. Tabu Search gets its name from the tabu list, which is the algorithm's fundamental component and temporarily "forbids" recently visited solutions. This approach prevents the search process from repeatedly visiting the same solutions, enabling exploration of wider regions of the search space.

The Tabu Search Algorithm works by systematically evaluating potential solutions in the neighborhood of the current solution. In each iteration, all solutions (or a specific subset) in the neighborhood of the current solution are examined, and the best solution not in the tabu list is selected. The move corresponding to the selected solution is added to the tabu list, and the list forbids this move for a certain period (tabu tenure). However, even a move in the tabu list can be accepted if it satisfies certain conditions known as the "aspiration criterion" (for example, if it produces a better solution than the best solution found so far). The algorithm also uses "intensification" (more detailed search in promising regions) and "diversification" (moving to different regions of the search space) strategies to guide the search process. This balanced approach enables the algorithm to both effectively investigate local optima and increase the probability of converging to the global optimum.

6.2.1 Basic Concepts

- Tabu list
- Neighborhood structure
- Aspiration criterion
- Intensification and diversification

Role of Tabu List

- Stores recently visited solutions
- Prevents cyclic movements
- Explores different regions of search space
- List length is an important parameter

6.2.2 Algorithm Steps

1. Determine initial solution
2. In each iteration:
   - Generate neighbor solutions
   - Check tabu list
   - Select best suitable solution
   - Update tabu list
3. Continue until stopping criterion is met

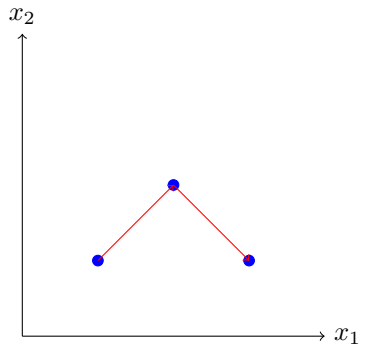

Figure 23: Tabu search algorithm movement

6.3 Genetic Algorithms (GA)

Genetic Algorithms (GA), developed by John Holland in 1975, are a meta-heuristic optimization method that mimics the natural evolution process. These algorithms work in the solution space and manage the search process using the genetic structures of solutions. GA has a structure where solutions are represented by chromosomes (genes) and these chromosomes are subjected to the evolution process under selected constraints.

The working principle of Genetic Algorithms is a process that mimics natural selection and genetic mechanisms. The algorithm begins by encoding potential solutions as chromosomes and creating a random initial population. In each iteration, the fitness values of solutions are calculated, and individuals with higher fitness have a greater chance of reproduction. Selected parents undergo crossover to share genetic information and create a new generation. Mutation is applied with a low probability to maintain diversity and explore different regions of the search space. This process is repeated until a certain stopping criterion is met (maximum number of generations, sufficient fitness value, etc.). The strength of Genetic Algorithms lies in their ability to converge to the global optimum even in complex and multidimensional problems and their easy adaptability to different problem types.

6.3.1 Basic Components

- Chromosome (solution) structure
- Population
- Selection mechanism
- Crossover operator
- Mutation operator

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)} \tag{38}$$

[32]

[32] Genetic algorithms are inspired by Darwin's theory of evolution. The principle of survival of the fittest ensures the production of better solutions in the optimization process.

6.3.2 Algorithm Steps

1. Create initial population
2. In each generation:
   - Calculate Fitness values
   - Select parents
   - Apply crossover and mutation
   - Create new generation
3. Continue until stopping criterion is met

Genetic Operators

- **Crossover:**
  - Single-point
  - Multi-point
  - Uniform
- **Mutation:**
  - Bit flip
  - Gaussian
  - Uniform

6.4 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), developed by Russell Eberhart and James Kennedy in 1995, is a metaheuristic optimization method that mimics the natural evolution process. PSO represents solutions as particles and manages the search process using the effects of other particles in the swarm.

PSO is a process where particles move by benefiting from their own experience and the collective knowledge of the swarm. Each particle tracks its own best solution (pbest) and the swarm's best solution (gbest). The velocity and position of particles are updated with the effects of other particles in the swarm, thus increasing the probability of converging to the global optimum in the search space. PSO provides successful results in multidimensional and complex optimization problems and stands out with its fast convergence ability, few parameters, and easy adaptability.

6.4.1 Basic Concepts

- Particle position and velocity
- Personal best (pbest)
- Global best (gbest)
- Inertia weight
- Learning factors

$$
\begin{aligned}
v_{i,j}^{t+1} &= w v_{i,j}^{t} + c_1 r_1 (pbest_{i,j} - x_{i,j}^{t}) + c_2 r_2 (gbest_j - x_{i,j}^{t}) \\
x_{i,j}^{t+1} &= x_{i,j}^{t} + v_{i,j}^{t+1}
\end{aligned}
\tag{39}
$$

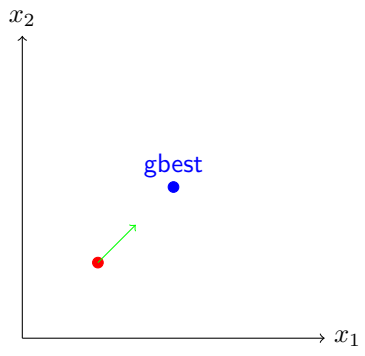

Figure 24: Particle movement in PSO

6.4.2 Algorithm Steps

1. Initialize swarm
2. In each iteration:
   - Calculate fitness values
   - Update pbest and gbest
   - Update velocities and positions
3. Continue until stopping criterion is met

[33]

[33] PSO is inspired by the behavior of bird flocks. Particles move by benefiting from both their own experience and the collective knowledge of the swarm.

6.5 Advantages and Disadvantages of Algorithms

Comparative Analysis

- **Simulated Annealing:**
  - + Simple implementation
  - + Theoretical convergence guarantee
  - - Slow convergence
  - - Sensitive parameter tuning
- **Tabu Search:**
  - + Avoids cyclic movements
  - + Effective in local search
  - - Memory requirement
  - - Dependent on neighborhood structure
- **Genetic Algorithm:**
  - + Parallel search
  - + Wide search space
  - - High computational cost
  - - Many parameters
- **PSO:**
  - + Fast convergence
  - + Few parameters
  - - Risk of premature convergence
  - - Getting stuck in local optima

6.6 Ant Colony Optimization (ACO)

An optimization algorithm inspired by ants' food search behavior. It provides effective results especially in combinatorial optimization problems.

Ant Colony Optimization (ACO) is a metaheuristic optimization algorithm that mimics ants' behavior of finding the shortest path between their nest and food source. Ants leave a chemical substance called pheromone while moving, and other ants follow these pheromone trails. Since shorter paths are completed faster, pheromone accumulation is more intense on these paths, and over time,

more ants start to prefer these paths. The algorithm is based on the principle of artificial ants wandering in the solution space leaving pheromone trails, these trails evaporating over time, and ants making path choices by combining pheromone density with heuristic information (usually distance). Through this collective intelligence mechanism, ants move toward near-optimal solutions and provide effective results especially in combinatorial optimization problems such as the traveling salesman problem, vehicle routing, and network routing.

6.6.1 Basic Principles

- Pheromone trails
- Probabilistic path selection
- Pheromone update
- Evaporation mechanism

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \tag{40}$$

[34]

[34] Ant colony optimization provides successful results especially in combinatorial optimization problems like the traveling salesman problem.

6.6.2 Algorithm Steps

1. Assign initial pheromone values
2. In each iteration:
   - Place ants
   - Create solutions
   - Update pheromone trails
   - Apply evaporation
3. Continue until stopping criterion is met

Figure 25: Pheromone trails and path selection

6.7 Differential Evolution (DE) Algorithm

A vector-based evolutionary optimization algorithm. Provides effective results in continuous optimization problems.

6.7.1 Basic Operators

- Mutation vector
- Crossover
- Selection
- Scaling factor (F)
- Crossover ratio (CR)

$$v_i = x_{r1} + F(x_{r2} - x_{r3}) \tag{41}$$

DE Strategies

- DE/rand/1/bin
- DE/best/1/bin
- DE/rand/2/bin
- DE/best/2/bin
- DE/current-to-best/1/bin

6.7.2 Algorithm Steps

1. Create initial population
2. In each generation:
   - Generate mutation vectors
   - Apply crossover
   - Perform selection
3. Continue until stopping criterion is met

[35]

[35] The differential evolution algorithm is particularly effective in continuous optimization problems. Parameter tuning is relatively easy and convergence speed is high.

6.8 Artificial Bee Colony (ABC) Optimization

An optimization algorithm inspired by honey bees' food search behavior.

6.8.1 Bee Types and Tasks

- Worker bees: Investigation of current resources
- Onlooker bees: Evaluation of promising resources
- Scout bees: Discovery of new resources

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{42}$$

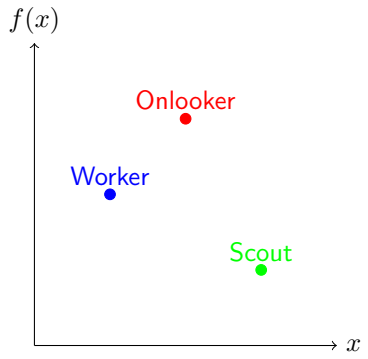

Figure 26: Role of different bee types in ABC

6.8.2 Algorithm Steps

1. Determine initial food sources
2. In each cycle:
   - Worker bee phase
   - Onlooker bee phase
   - Scout bee phase
3. Continue until stopping criterion is met

6.9 Quantum and Hybrid Optimization Algorithms

Approaches inspired by quantum mechanics principles and combining strengths of different algorithms.

6.9.1 Quantum-Inspired Algorithms

- Quantum particle swarm
- Quantum genetic algorithm
- Quantum simulated annealing

$$\psi(x,t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{43}$$

Advantages of Quantum Mechanism

- Better exploration ability
- Avoiding local optima
- Fast convergence
- Utilizing uncertainty principle

6.10 Other Metaheuristic Optimization Algorithms

Dozens of new metaheuristic optimization algorithms are added to the academic literature each year. Some of these algorithms can still present quite original approaches. However, within the basic classifications, many of them are developed by imitating certain mechanisms of previous algorithms to some extent. For an engineer looking at this topic from outside the academic perspective, understanding and being able to apply some basic algorithms is much more beneficial than constantly following new algorithms. Especially considering the development of artificial intelligence, what is really important is implementing theoretical knowledge in the right practical fields.

7 Optimization of Discrete Parameters

Solution methods for optimization problems containing discrete variables will be examined in this section. Particularly, discrete parameter optimization problems encountered in structural systems and solution strategies will be discussed.

7.1 Differences Between Discrete and Continuous Optimization

Classification of optimization problems according to their solution space structure and examination of fundamental differences.

7.1.1 Fundamental Differences

- Structure of solution space
- Usable methods
- Computational complexity
- Use of gradient information

Discrete vs Continuous Optimization

- **Discrete:**
  - Discontinuous solution space
  - Combinatorial methods
  - NP-hard problems
  - Gradient cannot be used
- **Continuous:**
  - Continuous solution space
  - Gradient-based methods
  - Differentiability
  - Use of local information

7.2 Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP)[36], a classic example of discrete optimization problems, is used in modeling many real-world problems. This problem deals with the scenario where a salesman needs to visit certain cities in the shortest distance. Each city must be visited only once, and the tour must end at the starting point. TSP is applied in areas such as logistics, production planning, PCB circuit design, and DNA sequencing. Since the solution space of the problem grows factorially as the number of cities increases (n! possible tours for n cities), finding exact solutions for large-scale problems is computationally quite challenging.

[36]

7.2.1 Problem Definition

- Finding shortest tour between N cities
- Visiting each city once
- Returning to starting point
- NP-hard problem class

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{44}$$

7.2.2 Solution Approaches

- Exact methods:
  - Branch and bound
  - Integer programming
- Heuristic methods:
  - Nearest neighbor
  - 2-opt, 3-opt
  - Lin-Kernighan
- Metaheuristic methods:

- ACO
- GA
- Tabu search

[37]

[37] TSP is the prototype example of discrete optimization problems. Many real-world problems can be reduced to TSP. However, not every optimization algorithm applied to TSP can be used for another problem. Therefore, although each optimization problem may resemble some common problems, it should be handled considering its own special structure.

7.3 Cross-Section Optimization of Steel Structures

The most common example of discrete optimization problems encountered in structural optimization is the cross-section optimization of steel structures. In this problem, an optimization problem based on the selection of standard cross-sections used in steel structures is considered.

7.3.1 Problem Definition

Cross-section optimization in steel structures is a discrete optimization problem:

- Standard cross-section tables
- Structural constraints
- Minimum weight objective
- Grouping requirements

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \rho_i L_i A_i \\ \text{s.t.} \quad & \sigma_i \leq \sigma_{allow} \\ & \delta \leq \delta_{allow} \\ & A_i \in S \end{aligned} \tag{45}$$

7.3.2 Solution Strategies

- Discrete variable optimization
- Metaheuristic methods
- Hybrid approaches
- Parallel computing

Optimization Process

1. Structural analysis
2. Cross-section selection
3. Constraint check
4. Iterative improvement

7.4 Simplifying Problem Solution with Indices

Steel structure optimization can be handled in different ways. However, if predefined steel sections are used, cross-section selection emerges as a discrete optimization problem. In this case, it is necessary to create a data structure for cross-section selection and solve the optimization problem using this data structure. At this point, section lists can be handled by indexing. However, one point that needs to be discussed is: which parameter will be used as the basis for ordering and indexing the section lists. For example, considering only the cross-sectional area as the basis for ordering does not guarantee the same ordering in terms of bending strength. However, considering bending strength as the basis also cannot guarantee the same ordering under axial load effects. Therefore, it may be more logical to handle section lists with different indexing strategies specific to the problem. For example, cross-sectional area can be used as the basis for elements under axial force, while bending strength can be used as the basis for elements under bending effects. Or we can develop a different, more effective strategy.

7.4.1 Indexing Strategy

- Section groups
- Element numbering
- Node points
- Loading conditions

$$x_i = \text{ind}(A_i), \quad i = 1, \ldots, n \tag{46}$$

[38]

[38] Indexing facilitates the solution of discrete optimization problems and increases computational efficiency.

7.4.2 Data Structures

- Section properties table
- Connection matrix
- Constraint matrix
- Index conversion table

Data Structure Example

```
sections = {
    1: {'A': 10.3, 'I': 171},
    2: {'A': 13.2, 'I': 375},
    ...
}
```

7.5 Evaluation of Optimization Results

Analysis of solution quality and performance of discrete optimization problems.

7.5.1 Performance Metrics

- Total weight
- Maximum stress ratio
- Maximum displacement
- Computation time

7.5.2 Visualization of Results

- Convergence graphs
- Stress distributions
- Displacement shapes
- Cross-section distributions

8 Optimization of Continuous Parameters

Optimization of continuous parameters is commonly encountered in structural engineering and other engineering fields. In this section, the fundamental characteristics, mathematical formulation, and solution methods of optimization problems expressed with continuous variables will be examined.

8.1 Basic Concepts of Continuous Optimization

Continuous optimization refers to optimization problems where design variables can take continuous values. In such problems, the design space consists of an infinite number of points, and variables can take any real value.

8.1.1 Continuous Design Variables

Continuous design variables are parameters that can take any value within a specific range. For example:

- Cross-sectional dimensions of a beam (width, height)
- Material properties (elasticity modulus, density)
- Geometric parameters (angles, lengths)
- Control parameters (force magnitudes, damping coefficients)

8.1.2 General Form of Continuous Optimization Problems

Continuous optimization problems are generally expressed in the following form:

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{47}$$

$$\text{s.t.} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \ldots, m \tag{48}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \ldots, p \tag{49}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \tag{50}$$

Where:

- $\mathbf{x} \in \mathbb{R}^n$: Vector of design variables
- $f(\mathbf{x})$: Objective function

- $g_i(\mathbf{x})$: Inequality constraints
- $h_j(\mathbf{x})$: Equality constraints
- $\mathbf{x}_L, \mathbf{x}_U$: Lower and upper bounds

Continuous Optimization Example

Weight minimization problem of a cantilever beam:

$$\min_{b,h} \quad \rho \cdot L \cdot b \cdot h \tag{51}$$

$$\text{s.t.} \quad \sigma_{max} = \frac{6PL}{bh^2} \leq \sigma_{allow} \tag{52}$$

$$\delta_{max} = \frac{PL^3}{3EI} \leq \delta_{allow} \tag{53}$$

$$b_{min} \leq b \leq b_{max} \tag{54}$$

$$h_{min} \leq h \leq h_{max} \tag{55}$$

Where b and h are the width and height of the beam, respectively.

[39]

[39] In continuous optimization problems, the objective function and constraints are generally continuous and differentiable functions, which allows the use of gradient-based optimization methods.

8.2 Mathematical Formulation of Continuous Optimization Problems

The mathematical formulation of continuous optimization problems is a fundamental step that determines the nature of the problem and solution methods. This formulation includes the mathematical expression of the objective function, constraints, and design variables.

8.2.1 Objective Function

The objective function mathematically expresses the engineering performance criterion to be optimized. Commonly used objective functions in structural optimization problems are:

- **Weight minimization:** $f(\mathbf{x}) = \sum_{i=1}^{n} \rho_i V_i(\mathbf{x})$
- **Flexibility minimization:** $f(\mathbf{x}) = \mathbf{F}^T \mathbf{u}(\mathbf{x})$
- **Stress minimization:** $f(\mathbf{x}) = \max_i \sigma_i(\mathbf{x})$
- **Displacement minimization:** $f(\mathbf{x}) = \max_i |u_i(\mathbf{x})|$
- **Frequency maximization:** $f(\mathbf{x}) = -\omega_1(\mathbf{x})$ (first natural frequency)
- **Cost minimization:** $f(\mathbf{x}) = \sum_{i=1}^{n} c_i x_i$

8.2.2 Constraint Functions

Constraint functions are mathematical expressions that ensure the design meets certain requirements. Constraints frequently used in structural optimization problems are:

- **Stress constraints:** $g_i(\mathbf{x}) = \sigma_i(\mathbf{x}) - \sigma_{allow} \leq 0$
- **Displacement constraints:** $g_i(\mathbf{x}) = |u_i(\mathbf{x})| - u_{allow} \leq 0$
- **Buckling constraints:** $g_i(\mathbf{x}) = P_{cr,i}(\mathbf{x}) - P_{applied} \leq 0$
- **Frequency constraints:** $g_i(\mathbf{x}) = \omega_{min} - \omega_i(\mathbf{x}) \leq 0$

- **Equilibrium constraints:** Express that the structure must satisfy static equilibrium conditions.
- **Geometric constraints:** Express that design variables must satisfy certain geometric relationships.

8.2.3 Sensitivity Analysis

Sensitivity analysis involves calculating the derivatives of the objective function and constraints with respect to design variables. These derivatives are used to determine the search direction in gradient-based optimization algorithms.

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^T \tag{56}$$

$$\nabla g_i(\mathbf{x}) = \left[\frac{\partial g_i}{\partial x_1}, \frac{\partial g_i}{\partial x_2}, \ldots, \frac{\partial g_i}{\partial x_n}\right]^T \tag{57}$$

Methods used for sensitivity analysis:

- **Analytical methods:** Direct calculation of derivatives using mathematical expressions
- **Finite difference method:** Numerical approximation of derivatives
- **Adjoint method:** Used for efficient sensitivity calculation in complex systems
- **Automatic differentiation:** Automatic derivative calculation by computer programs

8.2.4 Distinctive Characteristics of Continuous Optimization

Continuous optimization problems, unlike discrete optimization problems, are those where design variables can take continuous values. The distinctive characteristics of these problems are:

- **Continuous design space:** Design variables can take values from the set of real numbers, meaning an infinite number of possible solutions.
- **Differentiability:** The objective function and constraints are generally differentiable functions, allowing the use of gradient-based optimization methods.
- **Convexity:** Whether the problem formulation is convex determines the achievability of the global optimum. In convex problems, the local optimum is also the global optimum.
- **Continuity:** The continuity of functions ensures more stable operation of the optimization algorithm.
- **Differentiability:** The existence of higher-order derivatives allows the use of Newton-like methods.

Continuous optimization problems are commonly used in structural engineering for optimizing variables such as cross-sectional dimensions, material properties, or geometric parameters. In solving these problems, gradient-based methods (Newton's method, conjugate gradient method), gradient-free methods (Nelder-Mead simplex method), or metaheuristic algorithms (genetic algorithm, particle swarm optimization) can be used.

8.3 General Applications of Continuous Optimization in Structural Optimization

Although two common optimization problems have been exemplified at this point, many outputs calculated depending on parameters can be considered and improved as optimization problems.

8.3.1 Non-Predefined Size Optimization

Non-predefined size optimization involves considering cross-sectional dimensions of structural elements as continuous variables without being limited by standard catalog values. This approach provides designers with a wider design space and enables the potential to obtain more efficient structures. In traditional approaches, while selection is made from certain standard sections (e.g., I-profiles, box sections) for structural elements, in non-predefined optimization, section properties (area, moment of inertia, etc.) are used directly as design variables.

In such optimization problems, objectives such as minimum weight or maximum stiffness are generally pursued while considering structural constraints such as stress, displacement, and buckling. The section properties obtained as a result of optimization are then interpreted to be converted into manufacturable sections or produced as special sections. Non-predefined size optimization is commonly used in aerospace, space, and automotive industries to minimize material usage while maximizing performance.

8.3.2 Topological Optimization

Topological optimization is an advanced structural optimization method used to determine the most efficient material distribution of a structure. This method optimizes the basic form and connection structure of the structure by deciding where material should and should not be located within a specific design area. Unlike traditional optimization methods, it can change not only the dimensions or shape but also the topology of the structure.

The topological optimization process generally begins with dividing a design area into finite elements and assigning a design variable representing material density to each element. The optimization algorithm adjusts these variables to find the best material distribution under certain constraints (e.g., maximum weight or minimum flexibility). As a result, highly efficient and lightweight structures, often resembling those found in nature, emerge. This method is widely used in various fields such as designing lightweight parts in automotive and aerospace industries, developing medical implants, and creating optimized structures for 3D printing technologies.

9 Introduction to Structural Optimization

Although structural optimization has been treated as a different type of optimization, it essentially progresses with similar principles to all optimization problems. However, the definition of the problem, and consequently the selection of effective solution algorithms, depends on engineering judgment. In this section, we will try to explain how the concepts defined under classical optimization topics transform into a context in structural optimization.

9.1 Structural Optimization Terminology

9.1.1 Objective Functions

In structural optimization, the objective function mathematically expresses the engineering target to be optimized. The most commonly used objective func-

tions in structural design are:[40]

- **Weight minimization:** Aims to minimize the total weight of the structure. Particularly critical in aerospace structures.
- **Cost minimization:** Aims to minimize the production, material, and labor costs of the structure.
- **Stiffness maximization:** Aims to maximize the structure's resistance to deformation under specific loads.
- **Strength maximization:** Aims to increase the maximum load the structure can carry.
- **Energy dissipation:** Optimizes the structure's energy dissipation capacity under dynamic loads.

[40] In multi-objective optimization problems, multiple objective functions can be weighted and converted into a single function, or Pareto-optimal solutions can be sought.

9.1.2 Constraints

In structural optimization, constraints define the conditions that must be met for the design to be feasible. These constraints are the structural engineering counterparts of mathematical constraints in classical optimization problems:

- **Stress constraints:** Ensures that stresses in the structure do not exceed allowed maximum values. For example: $\sigma_i \leq \sigma_{allow}$
- **Displacement constraints:** Ensures that displacements in the structure remain within certain limits. For example: $\delta_i \leq \delta_{allow}$
- **Buckling constraints:** Ensures that the buckling loads of structural elements are greater than the applied loads by a certain safety factor.
- **Vibration constraints:** Ensures that the natural frequencies of the structure are above or below certain values.
- **Geometric constraints:** In the context of structural optimization, ensures that design variables remain within physically feasible limits. For example:
  - Minimum and maximum cross-sectional dimensions
  - Minimum wall thicknesses
  - Connection requirements between elements[41]
  - Assembly and manufacturing constraints
- **Equilibrium constraints:** Expresses that the structure must satisfy static equilibrium conditions.
- **Compatibility constraints:** Specifies that deformations must be continuous and compatible.

[41] For example, it may be desired that the cross-sectional dimensions used in the upper floors of a steel structure be larger than those in the lower floors, which is also quite logical from an application perspective.

Example of Structural Optimization Constraints

In a bridge design:

$$\sigma_{max} \leq 250 \text{ MPa} \quad \text{(Stress constraint)} \tag{58}$$
$$\delta_{mid} \leq L/400 \quad \text{(Displacement constraint)} \tag{59}$$
$$f_1 \geq 2.0 \text{ Hz} \quad \text{(Vibration constraint)} \tag{60}$$
$$t_{min} \geq 8 \text{ mm} \quad \text{(Geometric constraint)} \tag{61}$$

[42]

[42] The mathematical formulation of constraints is expressed based on the results of finite element analysis and is generally in the form of nonlinear functions.

9.1.3 Design Variables

In structural optimization, design variables represent the parameters to be optimized. These variables are parameters that can be changed by the optimization algorithm and adjusted to find the best solution. Common design variables used in structural engineering are:

- **Cross-sectional properties:**
  - Profile dimensions (width, height)
  - Wall thicknesses
  - Cross-sectional area
  - Moment of inertia
- **Material properties:**
  - Elasticity modulus
  - Density
  - Yield strength
- **Geometric parameters:**
  - Node point coordinates
  - Curvature radii
  - Angles
- **Topological parameters:**
  - Material presence/absence (0-1 variables)
  - Material density (continuous variables varying between 0-1)[43]
  - Presence of connection points

Design Variables Representation

In a typical steel frame optimization, design variables can be represented as:

$$\mathbf{x} = [A_1, A_2, \ldots, A_n, I_{y1}, I_{y2}, \ldots, I_{yn}, I_{z1}, I_{z2}, \ldots, I_{zn}]^T \quad (62)$$

Where A_i represents cross-sectional areas, and I_{yi} and I_{zi} represent moments of inertia.

[43] In many computational methods requiring optimization or regression-like coding, a normalization approach is used to handle data in a more standard way. This approach ensures that data has a value range varying between 0-1. The smallest data in the existing data becomes 0, and the largest data becomes 1. All intermediate values take a proportional value within this range.

9.2 Structural Optimization Categories

Structural optimization problems can be categorically divided into some basic headings (Shape, size, topology, etc.). However, an engineer can consider any problem with parameters producing conflicting outputs as an optimization problem.

For example, making a structure lighter often means compromising stress capacities. These conflicting outputs are the result of the same parameters.

9.2.1 Size Optimization

Size optimization is the optimization of cross-sectional dimensions of elements while keeping the general geometry of the structure constant. It is the most basic and commonly used structural optimization approach.

- **Design variables:** Cross-sectional properties such as area, thickness, width-height
- **Advantages:**
  - Relatively simpler mathematical formulation
  - Suitable for improving existing designs
  - Widespread use in industry
- **Application areas:** Steel structures, frame systems, truss systems

9.2.2 Shape Optimization

Shape optimization is performed by changing the shapes of structural elements or positions of node points. The general topology of the structure is preserved while the boundary geometry is changed.

- **Design variables:** Node point coordinates, curvature parameters, control points
- **Advantages:**
  - More design flexibility compared to size optimization
  - Effective in reducing stress concentrations
- **Challenges:**
  - Geometric changes may require remeshing of the finite element mesh
  - Complex mathematical formulation
- **Application areas:** Aerospace structures, automotive parts, bridge constructions

9.2.3 Topology Optimization

Topology optimization is performed by changing the basic structure or topology of the structure. The distribution of material within the structure is optimized, and regions where material should or should not be present are generally determined.

- **Design variables:** Material density, material presence/absence
- **Advantages:**
  - Highest design freedom
  - Ability to produce innovative and unpredictable designs
  - Significant potential for material savings
- **Challenges:**
  - Mathematically and computationally complex
  - Difficult to apply manufacturability constraints
  - Interpretation of results and conversion to feasible designs

- **Application areas:** Aerospace, automotive, medical implants, 3D printed structures

Example: Cantilever Beam Optimization

Optimization of the same cantilever beam problem with three different approaches:
Size: The variation of beam cross-section height along the length is optimized.
Shape: The shape of the beam's upper and lower surfaces is optimized.
Topology: The material distribution in the beam's internal structure is optimized, usually resulting in a truss-like structure.

9.3 Structural Optimization Formulation

A structural optimization problem can be mathematically expressed as:

$$\text{Minimize: } f(\mathbf{x}) \tag{63}$$
$$\text{Constraints: } g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \ldots, m \tag{64}$$
$$h_k(\mathbf{x}) = 0, \quad k = 1, 2, \ldots, p \tag{65}$$
$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \tag{66}$$

Where:

- $\mathbf{x}$: vector of design variables
- $f(\mathbf{x})$: objective function (for minimization problem)
- $g_j(\mathbf{x})$: inequality constraints
- $h_k(\mathbf{x})$: equality constraints
- $\mathbf{x}_L$ and $\mathbf{x}_U$: lower and upper bounds of design variables

9.3.1 Connection with Finite Element Analysis

In structural optimization problems, the objective function and constraints are generally dependent on finite element analysis (FEA) results. This connection can be expressed as:

$$\mathbf{K}(\mathbf{x})\mathbf{u} = \mathbf{F} \tag{67}$$
$$f(\mathbf{x}) = f(\mathbf{x}, \mathbf{u}(\mathbf{x})) \tag{68}$$
$$g_j(\mathbf{x}) = g_j(\mathbf{x}, \mathbf{u}(\mathbf{x})) \tag{69}$$
$$h_k(\mathbf{x}) = h_k(\mathbf{x}, \mathbf{u}(\mathbf{x})) \tag{70}$$

Where:

- $\mathbf{K}(\mathbf{x})$: stiffness matrix dependent on design variables
- $\mathbf{u}$: displacement vector
- $\mathbf{F}$: external force vector

Structural Optimization Algorithm Selection

Algorithm selection in structural optimization problems depends on the following factors:

- Problem size (number of design variables)
- Number and complexity of constraints
- Computational cost of function evaluations
- Characteristic of design space (existence of multiple local optima)
- Availability of sensitivity information

10 Topological Optimization

This section will examine topological optimization methods that aim to determine the most basic form of structural systems. The optimization of material distribution and modern topology optimization techniques will be discussed.

10.1 Foundations of Topological Optimization

Topological optimization is an advanced structural optimization method used to determine the most efficient material distribution of a structure. Unlike traditional optimization methods, topological optimization optimizes not only the dimensions or shape but also the basic form and connection structure of the structure. In this approach, decisions are made about where material should and should not be located within a specific design area.

The topological optimization process generally begins with dividing a design area into finite elements. Each element is assigned a design variable representing material density, varying between 0 and 1. The optimization algorithm adjusts these variables to find the best material distribution under certain constraints (e.g., maximum weight or minimum flexibility). As a result, highly efficient and lightweight structures, often resembling those found in nature, emerge.

This method is widely used in various fields such as designing lightweight and durable parts in automotive and aerospace industries, developing medical implants, and creating optimized structures for 3D printing technologies. Topological optimization provides engineers with the opportunity to offer innovative and efficient solutions that would be difficult to achieve with traditional design approaches.

10.1.1 Basic Concepts

- Material distribution [44]
- Structural topology [45]
- Homogenization [46]
- Design variables [47]

$$\min_{x \in [0,1]^n} \quad c(x) = F^T U(x) \tag{71}$$

[44] Distribution showing how material is placed within the design area, generally expressed with density variables.

[45] Geometric arrangement defining the basic form, connection structure, and material distribution of a structure.

[46] Method for calculating effective properties of composite materials, used to determine macro properties of microstructures.

[47] Parameters that can be changed during the optimization process, generally assigned to each finite element and representing material presence.

10.2 Relationship Between Finite Element Method and Optimization

Topological optimization is directly related to the Finite Element Method (FEM) and this method is a fundamental component of the optimization process. The finite element method allows analyzing complex geometries by dividing them into smaller and simpler elements, enabling precise calculation of structural behavior necessary for topological optimization.

In the optimization process, when the material distribution changes in each iteration, the mechanical behavior of the structure (stresses, displacements, natural frequencies, etc.) is recalculated using finite element analysis. These analysis results enable the optimization algorithm to decide where to add or remove material in the next step. Thus, FEM becomes an integral part of both the analysis and decision-making mechanism of topological optimization.

10.2.1 FEM Formulation

- Stiffness matrix
- Load vector
- Displacement field
- Element types

$$K(x)U = F \tag{72}$$

10.2.2 Creation of Finite Element Model with API

Various software offer Application Programming Interface (API) for creating and analyzing finite element models. These APIs enable topological optimization algorithms to work integrated with finite element analyses. Especially in optimization processes requiring automatic iteration, API usage provides great efficiency by eliminating manual model creation and analysis processes.

SAP2000 OAPI (Open Application Programming Interface) is a commonly used API example for structural analysis and optimization. This interface provides access to all features of SAP2000 software through programming languages such as Python, MATLAB, or C++. In the topological optimization process, the algorithm can use SAP2000 OAPI in each iteration to:

- Apply updated material properties to the model
- Run the analysis automatically
- Read analysis results (stresses, displacements, etc.)
- Calculate new material distribution based on these results

Such API integrations enable complete automation of the topological optimization process, allowing even complex structures to be optimized efficiently. Additionally, other finite element software such as ANSYS, Abaqus, and NASTRAN also offer similar APIs. More detailed examples using SAP2000 OAPI will be examined in later topics.

10.3 Density-Based Methods

The most commonly used density-based topological optimization method is SIMP (Solid Isotropic Material with Penalization). This method works by assigning a density variable varying between 0 and 1 to each finite element. Here, 0 represents material absence and 1 represents full material presence.

The basic principle of the SIMP method is to penalize intermediate density values (values between 0 and 1) to obtain a more distinct 0-1 distribution as a result of optimization. This is achieved by defining material properties (e.g., elasticity modulus) as a power function of the density variable. The penalty parameter is generally chosen as 3 or higher.

The SIMP method has been successfully used in various engineering applications such as lightweighting automotive parts, optimizing aircraft structural elements, and designing medical implants. The method has become a standard approach in industry due to its well-defined mathematical foundation and compatibility with gradient-based optimization algorithms.

10.3.1 SIMP Method

Solid Isotropic Material with Penalization:

$$E(x) = E_{min} + x^p(E_0 - E_{min}) \tag{73}$$

- Density variables: $x \in [0, 1]$
- Penalty parameter: $p > 1$
- Minimum stiffness: E_{min}
- Full material stiffness: E_0

Advantages of SIMP Method

- Simple implementation
- Fast convergence
- Penalization of intermediate densities
- Widespread use in industrial applications

10.4 ESO and BESO Methods

Evolutionary Structural Optimization (ESO) and Bi-directional Evolutionary Structural Optimization (BESO) methods are heuristic approaches used in structural topology optimization. The ESO method is based on the principle of "gradually removing inefficient material" and aims to reach the optimum design by removing elements with low stress or energy density from the structure. BESO is an improved version of ESO and includes not only material removal but also material addition to necessary regions. Although these methods do not have as solid a mathematical foundation as SIMP, they are preferred in engineering applications due to their ease of implementation and intuitive understanding.

10.5 Level-Set Method

The Level-Set method is a mathematical approach used to explicitly define structure boundaries in topology optimization. In this method, the boundaries of the structure are represented as the zero-level curve (or surface) of a level-set function. During the optimization process, this level-set function is updated using

Hamilton-Jacobi equations, thus allowing the structure boundaries to evolve smoothly. The Level-Set method has advantages such as creating sharp and clear boundaries, naturally handling topology changes, and easily incorporating manufacturability constraints. It is particularly effective in fluid-structure interaction problems and multi-material designs.

11 Size and Shape Optimization

The optimization of size and shape parameters of structural systems can also be called cross-section optimization in a more general sense. Depending on the problem, one or both of these can become parameters of the problem simultaneously.

11.1 Foundations of Size Optimization

Size optimization is an optimization method used to determine the optimal values of cross-sectional properties (width, height, thickness, etc.) of structural systems. This method aims to reach the optimum design by changing only the dimensions of elements without altering the topology of the structure.

11.1.1 Problem Parameters

Design variables used in size optimization typically include:

- **Cross-section dimensions:** Width and height of beams, thickness of plates
- **Cross-sectional areas:** Cross-sectional areas of bar elements
- **Moments of inertia:** Parameters determining bending and torsional stiffness of beams
- **Material properties:** Variables such as elasticity modulus, density
- **Reinforcement elements:** Dimensions and locations of strengthening elements

11.1.2 Problem Outputs

The outputs obtained as a result of size optimization are:

- **Optimum cross-section dimensions:** Most suitable dimensions for each structural element
- **Minimum weight/cost:** Total weight or cost of the structure obtained as a result of optimization
- **Structural performance indicators:** Stresses, displacements, natural frequencies
- **Material usage efficiency:** How efficiently the load-bearing capacity of each element is used
- **Sensitivity information:** Effects of changes in design variables on the objective function

11.1.3 Constraints and Decision Mechanism

In size optimization, various constraints limit the solution space and affect the decision mechanism:

- **Stress constraints:** $\sigma_{max} \leq \sigma_{allow}$
  - Maximum stresses in the structure must not exceed allowable values
  - Affects in the direction of increasing element dimensions
- **Displacement constraints:** $\delta_{max} \leq \delta_{allow}$
  - Ensures maximum displacements in the structure remain within certain limits
  - Generally affects in the direction of increasing structural stiffness
- **Buckling constraints:** $P_{cr} \geq P_{design}$
  - Ensures buckling loads of compression elements are greater than design load
  - Affects in the direction of increasing dimensions of slender elements
- **Frequency constraints:** $\omega_i \geq \omega_{min}$ or $\omega_i \leq \omega_{max}$
  - Ensures natural frequencies of the structure are within certain ranges
  - Important in structures subjected to dynamic loads
- **Manufacturability constraints:** $x_{min} \leq x \leq x_{max}$
  - Ensures design variables remain within practical manufacturing limits
  - Limits solution space with realistic values
- **Geometric constraints:** For example $h/b \leq \alpha$
  - Ensures cross-section ratios remain within certain limits
  - Prevents local buckling and stability issues

Size Optimization Process

1. Creation of initial design
2. Performance evaluation with structural analysis (FEM)
3. Determination of effects of design variables through sensitivity analysis
4. Updating design variables with optimization algorithm
5. Repetition of steps 2-4 until convergence is achieved

Size optimization is widely used in structural engineering applications such as cross-section optimization of steel structures, reinforcement optimization of reinforced concrete structures, and bridge and tower design. As a result of optimization, while material usage is reduced, structural performance requirements are met, thus obtaining more economical and sustainable designs.

11.2 Foundations of Shape Optimization

The optimization of external boundaries and internal voids of structural elements can be approached in different ways depending on the engineer's perspective.

- **Boundary Representation:** Geometric parameters
- **Control Points:** Shape change control
- **Smoothness:** Geometric continuity

[48]

[48] Shape optimization improves the geometry of the structure without changing the topology.

11.3 Mathematical Formulation

Mathematical expression of size and shape optimization problems:

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}, \mathbf{s}) \\ &\text{subject to} && g_i(\mathbf{x}, \mathbf{s}) \leq 0, \quad i = 1, \ldots, m \\ &&& h_j(\mathbf{x}, \mathbf{s}) = 0, \quad j = 1, \ldots, p \\ &&& \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \\ &&& \mathbf{s}^L \leq \mathbf{s} \leq \mathbf{s}^U \end{aligned} \tag{74}$$

Where:

- $\mathbf{x}$: Size parameters
- $\mathbf{s}$: Shape parameters
- f: Objective function
- g_i, h_j: Constraint functions

11.4 Parametric Modeling

Mathematical representation of design variables:

- **CAD Parameters:** Geometric dimensions
- **Spline Curves:** Boundary representation
- **Morph Box:** Shape deformation

11.5 Sensitivity Analysis

In some structural optimization problems, sensitivity analysis can be performed for design variables (parameters). However, since structural optimization problems are generally hyperstatic in nature, sensitivity analyses may not yield expected results and can even be misleading. [49]

[49] The issue of hyperstaticity is important from an optimization perspective. For example, a cross-section chosen for one parameter, despite being close to the limit strength, can completely affect the load distribution due to changes in the cross-section of another parameter of the structure, causing the stress of the first cross-section to fall well below or exceed the limit strength.

11.6 Constraint Handling

Handling of design constraints:

- **Stress Constraints:** Material strength
- **Displacement Constraints:** Deformation limits
- **Geometric Constraints:** Manufacturability

11.7 Multi-Objective Optimization

Multi-objective optimization will be discussed in much more detail later.

Multi-Objective Approaches

- **Pareto Optimization:** Trade-off analysis
- **Weighted Sum:** Single objective function
- **Goal Programming:** Ideal point approach

11.8 Mesh Adaptation

The mesh creation process, which can often turn into a nightmare in finite element models, can sometimes be the subject of optimization. However, this topic belongs to a separate field of expertise and course.

- **Mesh Quality:** Element shape control
- **Adaptive Mesh:** Automatic mesh improvement
- **Remeshing:** Mesh regeneration

11.9 Manufacturability and Practical Constraints

Practical applicability of the design:

- **Standard Sections:** Catalog selection
- **Manufacturing Method:** Production constraints
- **Cost:** Economic factors

[50]

[50] Manufacturability constraints require balancing between theoretical optimum and practical solution, and this can be considered as a completely different optimization problem.

11.10 Evaluation of Optimization Results

Analysis and interpretation of obtained results:

Evaluation Criteria

- **Performance Improvement:** Gains compared to initial state
- **Constraint Satisfaction:** Control of all design constraints
- **Manufacturability:** Analysis of practical applicability
- **Cost Analysis:** Economic evaluation

12 Multi-Objective Optimization

In the optimization of structural systems, it may sometimes be necessary to simultaneously optimize multiple conflicting objectives. This section will examine multi-objective optimization methods and applications. [51]

[51] In multi-objective optimization, instead of a single optimal solution, a set of Pareto-optimal solutions is obtained. To better understand multi-objective optimization, the example code in the link can be examined.

12.1 Foundations of Multi-Objective Optimization

Simultaneous optimization of multiple objective functions:

Basic Concepts

- **Pareto Optimality:** Dominant solutions
- **Trade-off:** Compromise between objectives
- **Decision Making:** Solution selection
- **Weighting:** Prioritization of objectives

12.2 Mathematical Formulation

General structure of multi-objective optimization problem:

$$\begin{aligned} \text{minimize} \quad & \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, && i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, && j = 1, \ldots, p \\ & \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \end{aligned} \tag{75}$$

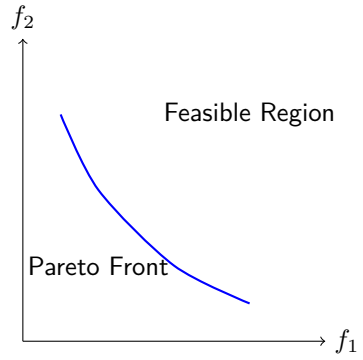

Figure 27: Example of Pareto front

12.3 Solution Approaches

The examination of basic strategies used in solving multi-objective optimization problems offers different approaches depending on how the problem is handled. That is, essentially, the decision-maker here is the engineer themselves. Whether the objective functions are superior to each other or their proportionality affects this decision and the strategy to be chosen.

12.3.1 Scalarization Methods

These are approaches that transform the multi-objective optimization problem into a single-objective problem. These methods combine multiple objectives to make the problem more easily solvable. Scalarization methods are widely used in engineering problems due to their quick and easy implementation. However, determining appropriate weights can complicate the process as it directly affects the quality of the solution.

Common Scalarization Methods

- **Weighted Sum Method:** $F(x) = \sum_{i=1}^{k} w_i f_i(x)$
- ε**-Constraint Method:** One objective is optimized while others are defined as constraints
- **Goal Programming:** $\min \sum_{i=1}^{k} w_i |f_i(x) - T_i|$ (where T_i are target values)

12.3.2 Pareto-Based Approaches

Pareto-based approaches aim to find all Pareto-optimal solutions or a good representation of them. These methods allow exploring a larger portion of the solution space and offer more alternatives to the decision-maker. Pareto-based approaches are particularly effective when evolutionary algorithms are used.

Basic components of Pareto-based approaches are:

- **Dominance Relation:** Determining whether one solution is better than another
- **Diversity Mechanisms:** Techniques ensuring homogeneous distribution of solutions along the Pareto front
- **Elite Strategies:** Mechanisms ensuring preservation of good solutions

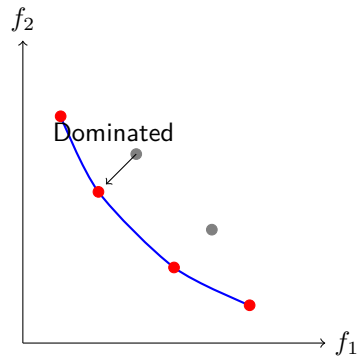

Figure 28: Pareto dominance concept

12.3.3 Interactive Methods

Interactive methods involve the decision-maker in the optimization process, narrowing the solution space according to their preferences. This approach integrates the decision-maker's knowledge and experience into the algorithm's operation. Interactive methods are valuable in complex engineering problems, particularly in terms of incorporating expert knowledge into the solution process.

Advantages of interactive approaches:

- Ability to directly reflect decision-maker's preferences into the process
- Directing computational resources to the solution region of interest
- Obtaining more meaningful and applicable results

12.4 Evolutionary Multi-Objective Optimization Algorithms

The application of evolutionary algorithms to multi-objective optimization problems provides significant advantages over classical methods. These algorithms stand out with their ability to produce multiple solutions at once, handle complex objective functions, and effectively search large solution spaces.

12.4.1 NSGA-II (Non-dominated Sorting Genetic Algorithm)

NSGA-II is one of the most widely used algorithms in the field of multi-objective evolutionary optimization. With its dominance sorting and crowding distance calculation mechanisms, it provides both convergence to Pareto-optimal solutions and diversity among solutions. NSGA-II operates quite efficiently with $O(MN^2)$ computational complexity and has been successfully applied in many engineering problems.

Basic components of NSGA-II:

- **Fast Non-dominated Sorting:** Sorts solutions in the population according to dominance relation
- **Crowding Distance:** Maintains diversity among solutions at the same dominance level
- **Binary Tournament Selection:** Selection mechanism based on dominance level and crowding distance

12.4.2 MOEA/D (Multiobjective Evolutionary Algorithm based on Decomposition)

MOEA/D is an evolutionary algorithm that solves the multi-objective optimization problem by decomposing it into a series of single-objective subproblems. Each subproblem is optimized simultaneously by sharing information with neighboring subproblems. This approach is particularly effective in problems with many objective functions and is computationally efficient.

Advantages of MOEA/D:

- Effective information sharing through neighborhood structure
- Suitability for problems with many objective functions
- Ability to use different decomposition methods (Tchebycheff, weighted sum, etc.)

12.4.3 SPEA2 (Strength Pareto Evolutionary Algorithm)

SPEA2 is an evolutionary algorithm that stores and refines Pareto-optimal solutions using a fixed-size archive. The algorithm considers both dominance relation and solution density by assigning a fitness value to each solution. SPEA2 is preferred particularly because it establishes a good balance between diversity and convergence.

Important features of SPEA2:

- **Strength Value:** Measures how many solutions a solution dominates
- **Density Estimation:** Calculated using K-nearest neighbor method
- **External Archive:** Efficiently stores and updates Pareto-optimal solutions

12.5 Decision Making Processes

Multi-objective optimization produces a set of Pareto-optimal solutions, and a choice must be made among these solutions. The decision-making process is a critical part of the optimization process and can be supported by various approaches.

12.5.1 Solution Selection Criteria

Various criteria can be used when selecting among Pareto-optimal solutions:

- **Distance Measures:** Solution closest to ideal point (e.g., Euclidean distance, Tchebycheff distance)
- **Satisfaction Level:** Solutions satisfying threshold values determined for each objective
- **Relative Improvement:** Rate of improvement of one objective relative to another (trade-off analysis)
- **Risk Analysis:** Reliability of solutions under uncertainty

Example: Weighted Tchebycheff Metric

$$d(F, F^*) = \max_{i=1,\ldots,k} \{w_i \cdot |F_i - F_i^*|\} \tag{76}$$

Where F^* is the ideal point, w_i are the weights of objectives, and F_i is the ith objective value of the current solution.

12.5.2 Weighting Strategies

Various weighting strategies can be used to determine the relative importance of objectives:

- **Direct Assignment:** Direct weight assignment by decision-maker
- **AHP (Analytic Hierarchy Process):** Weight determination through pairwise comparisons

- **Entropy-Based Methods:** Weight calculation based on data distribution
- **TOPSIS:** Weighting by similarity to ideal solution

12.5.3 Comparison Between Solutions

Comparing different Pareto-optimal solutions helps the decision-maker choose the solution suitable for their preference:

- **Visualization Techniques:** Parallel coordinate plots, star diagrams, heat maps
- **Sensitivity Analysis:** Effect of parameter changes on solution
- **Robust Evaluation:** Performance of solutions under uncertainty
- **Life Cycle Analysis:** Long-term performance and cost evaluation

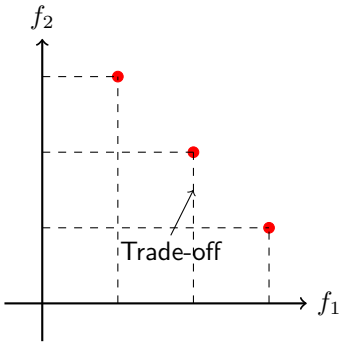

Figure 29: Trade-off analysis between Pareto solutions

12.6 Performance Metrics in Multi-Objective Optimization

12.6.1 Hypervolume

Hypervolume is one of the most common metrics used to evaluate the performance of multi-objective optimization algorithms. This metric expresses the measure of the area or volume covered by the Pareto front relative to a reference point. A higher hypervolume value indicates that the algorithm has found a better Pareto front, as this means a larger portion of the solution space is covered.

12.6.2 IGD (Inverted Generational Distance)

Inverted Generational Distance (IGD) is a measure of the distance between the true Pareto front and the solution set found by the algorithm. This metric shows how close the found solutions are to the true Pareto front and how well they represent the front. A lower IGD value indicates that the algorithm has found solutions closer to and better distributed along the true Pareto front.

12.6.3 Spread (Diversity)

The spread metric is a measure of the distance between points in the solution set and shows how homogeneously solutions are distributed along the Pareto front. This metric evaluates how well the algorithm explores the solution space and can offer various alternatives. Lower spread values are preferred for a uniformly distributed Pareto front.

12.6.4 Computation Time

Computation time measures the time spent by an algorithm to reach a solution. This metric is important for evaluating the efficiency of an algorithm and its usability in practical applications. Especially in complex engineering problems, algorithms that can provide good results in a reasonable time are preferred. Computation time can vary depending on the complexity of the algorithm, problem size, and application environment. [52]

[52] Since this parameter is quite relative, it does not always provide reliable results. Today, more modern metrics such as CPU Time are used.

13 Application I

This section will demonstrate the size optimization of a cantilever beam consisting of five different parts.
[53]

[53]

13.1 Problem Definition

In this application, an optimization problem aiming to minimize the weight of a 3-meter-long cantilever beam divided into 5 equal parts will be addressed. The details of the problem are as follows:

- A 3-meter-long cantilever beam divided into 5 equal parts
- Each part has a hollow circular cross-section with two design parameters:
  - r_{outer}: Outer radius
  - r_{inner}: Inner radius
- Material: S270 steel (E = 210 GPa, σ_y = 270 MPa)
- A force F = 500 kN is applied perpendicular to the free end of the beam
- Objective: Minimize the weight of the beam

13.1.1 Constraints

1. A maximum displacement of 2 cm is allowed at the free end of the beam
2. For each part, the outer radius must be greater than the inner radius
3. The inner radius of the previous part must be smaller than the outer radius of the next part (weldability condition)
4. The yield stress of S270 steel (270 MPa) must not be exceeded

13.2 Structural Analysis

The finite element method has been used for displacement and stress analysis of the cantilever beam:

- Each beam part is modeled as an Euler-Bernoulli beam element
- Each node has 2 degrees of freedom (displacement and rotation)
- Global stiffness matrix is created to calculate displacements
- Stresses are calculated using bending moment and section properties

13.2.1 Stiffness Matrix Formation

The stiffness matrix for the beam element is formed as follows:

$$k_e = \begin{bmatrix} \frac{12EI}{l^3} & \frac{6EI}{l^2} & -\frac{12EI}{l^3} & \frac{6EI}{l^2} \\ \frac{6EI}{l^2} & \frac{4EI}{l} & -\frac{6EI}{l^2} & \frac{2EI}{l} \\ -\frac{12EI}{l^3} & -\frac{6EI}{l^2} & \frac{12EI}{l^3} & -\frac{6EI}{l^2} \\ \frac{6EI}{l^2} & \frac{2EI}{l} & -\frac{6EI}{l^2} & \frac{4EI}{l} \end{bmatrix} \tag{77}$$

Where:

- E: Young's modulus
- I: Moment of inertia
- l: Element length

Moment of inertia for hollow circular cross-section:

$$I = \frac{\pi}{4}(r_{outer}^4 - r_{inner}^4) \tag{78}$$

13.2.2 Boundary Conditions and Solution

The left end of the cantilever beam is fixed, therefore the degrees of freedom at the first node are zero. The force applied to the right end is added to the global force vector. The displacement vector is solved using the reduced stiffness matrix and force vector:

$$\mathbf{K_{reduced}} \cdot \mathbf{u} = \mathbf{F} \tag{79}$$

13.3 Optimization Approach

The Simulated Annealing algorithm has been used for optimization:

- Random search strategy to avoid local optima
- Adaptive step size for effective exploration of solution space
- Slow cooling of process temperature to find better solutions
- Effective control of constraints to ensure physically feasible solutions

13.3.1 Objective Function

The objective function is the total weight of the beam:

$$W = \rho \sum_{i=1}^{5} A_i \cdot l_i \tag{80}$$

Where:

- ρ: Material density
- A_i: Cross-sectional area of each part ($A_i = \pi(r_{outer,i}^2 - r_{inner,i}^2)$)
- l_i: Length of each part

13.3.2 Constraint Functions

Four constraint functions are used in the optimization process:

1. Displacement constraint: $u_{max} \leq 0.02$ m
2. Radius constraint: $r_{outer,i} > r_{inner,i}$ (for each part)
3. Weldability constraint: $r_{inner,i} < r_{outer,i+1}$ (for adjacent parts)
4. Stress constraint: $\sigma_{max} \leq \sigma_{yield}$

13.4 Optimization Results

The optimization resulted in a lighter beam design compared to the initial design:

- Initial design weight: $\sim$1924 kg
- Optimized design weight: $\sim$939 kg (51% reduction)

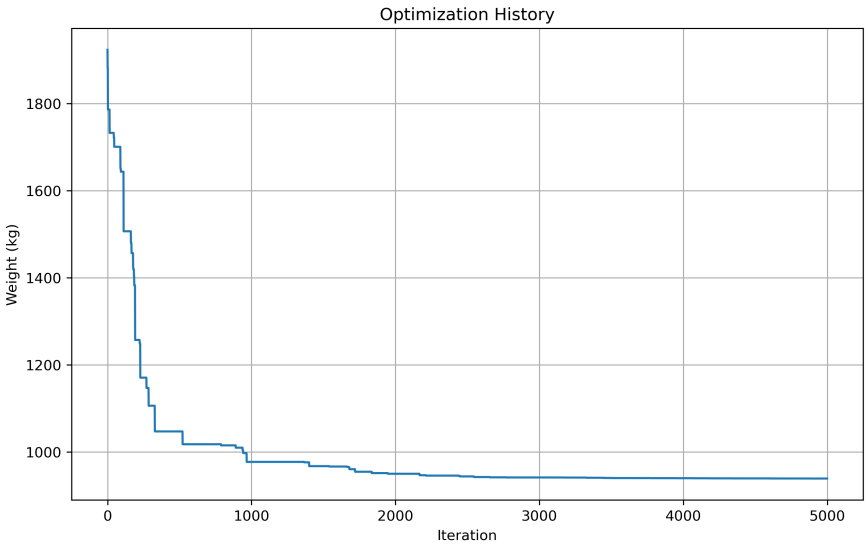

Figure 30: Weight Tracking Graph

13.4.1 Optimized Radii (cm)

Part	Outer Radius (r_{outer})	Inner Radius (r_{inner})
1	20.37	14.36
2	20.05	15.81
3	15.82	9.04
4	16.00	13.33
5	14.26	13.30

In the optimized design:

- The displacement at the end point is 0.12 cm (maximum allowed 2 cm)
- The stress constraint is active (0.00 MPa margin)
- The cross-section dimensions satisfy the weldability condition

13.4.2 Optimized Beam Design

The geometry of the optimized beam shows that the cross-section dimensions decrease from the support point (left side) towards the free end. This is consistent with the bending moment being maximum at the support and decreasing towards the free end.

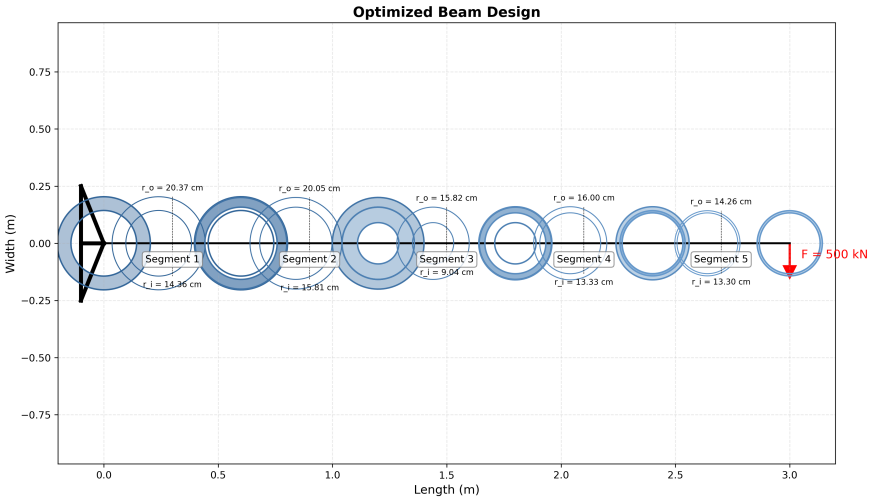

Figure 31: Optimized Beam Design

13.4.3 Deformation Shape

The deformation shape of the cantilever beam under load has a maximum displacement of 2 cm at the free end. The displacement values at each node are also shown.

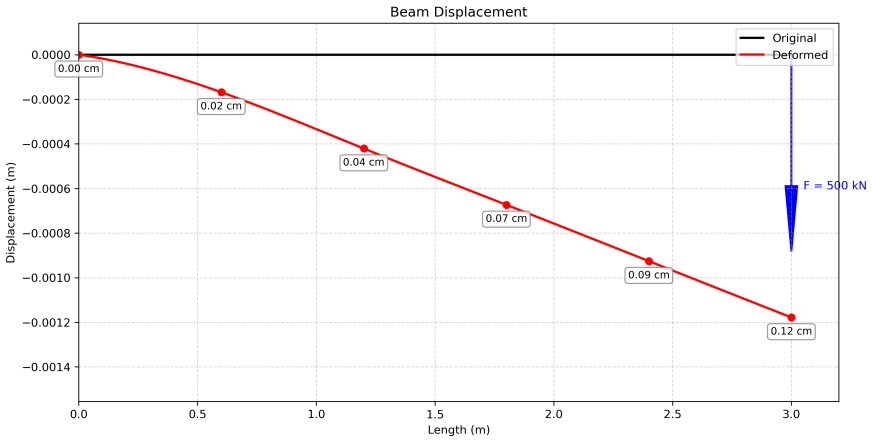

Figure 32: Deformation Shape

13.4.4 Constraint Utilization Ratios

Graphs have been created showing how much each constraint is utilized in the optimized design:

- **Displacement Constraint:** Maximum displacement limit is fully utilized (100%)
- **Stress Constraint:** Yield stress utilization ratio for each segment
- **Radius Ratio:** Ratio of inner radius to outer radius
- **Weldability:** Utilization ratio of the welding condition between adjacent segments

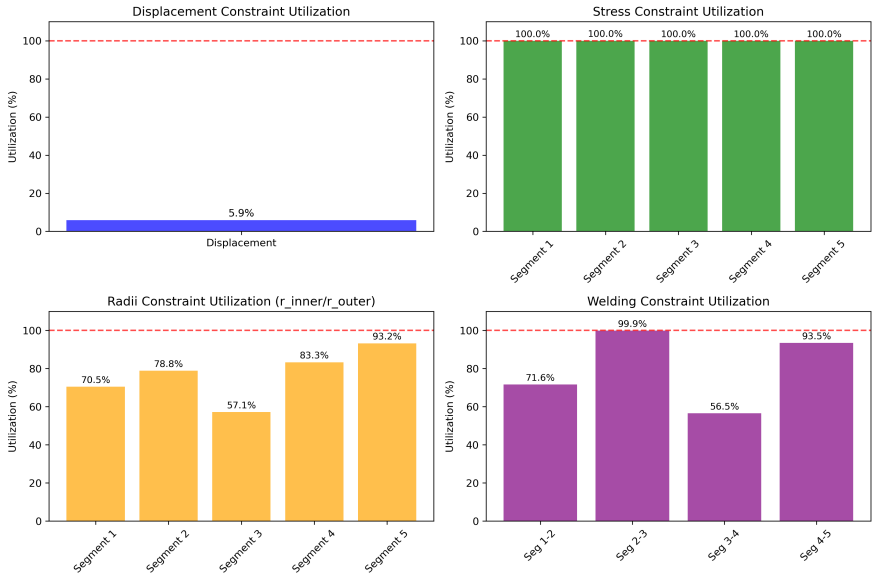

Figure 33: Constraint Capacity Utilization Ratios

As can be seen from the graphs, the displacement constraint is active in the optimal design (fully utilized). This indicates that the optimized design has reached the limit in terms of weight minimization.

13.5 Conclusion and Evaluation

In this application, the optimal design of a cantilever beam for minimum weight under constraints has been obtained using the Simulated Annealing optimization algorithm. The results show that a structure 51% lighter than the initial design has been achieved.

In the optimized design, it is observed that particularly the stress constraint is fully utilized (active). This is an expected theoretical result in weight minimization problems, as typically at least one constraint is expected to be active.

The gradual decrease in beam geometry from the support point towards the free end is also an expected structural result. Since the bending moment is maximum at the support point, larger cross-sections have formed in this region.