Training Material - UPRC

4. Local Optimization

4.1 Structures, Models and Algorithms for Optimization

Funded by
the European Union

Educational Goals of the Training

What will the user learn?

1. Understand basic and advanced optimization techniques.
2. Learn how to apply TSP and BTSP in industrial environments.
3. Become familiar with heuristic and exact solving methods.

Contents

Section 1: Optimization

1. **Optimization**
   - **1.1 Introduction to Optimization**
   - **1.2 Why Optimization Matters in Industry?**
   - **1.3 What makes a good solution?**
   - **1.4 Components of Optimization Problems**
   - **1.5 How We Solve Optimization Problems**
   - **1.6 Further Reading in Optimization**
   - **Q&A 1: Optimization Basics**
2. The Traveling Salesman Problem (TSP)
   - 2.1 Introduction to TSP
   - 2.2 Formal Definition, Mathematical Models, and Solving Methods
   - 2.3 Why is it Challenging?
   - 2.4 TSP Example
   - 2.5 Further Reading on the TSP
   - Q&A 2: TSP Understanding
3. The Bipartite TSP (BTSP)
   - 3.1 Introduction to BTSP
   - 3.2 Formal Definition and Bipartite Graphs
   - 3.3 Example of BTSP
   - 3.4 Further Reading on the BTSP
   - Q&A 3: BTSP Understanding
4. Industrial Applications of TSP and BTSP
   - 4.1 Use Cases of TSP in Logistics and Routing
   - 4.2 Use Cases of BTSP in Robotics and Assembly Lines
   - 4.3 Case Studies and Examples
   - Q&A 4: Application Scenarios

1. Optimization

- What is Optimization?
  - Optimization is the process of finding the best possible solution to a given problem, while respecting all limitations (called constraints). In production environments, this means making decisions that lead to higher productivity, lower costs, and better use of resources.

- We want to:
  - Solve problems in an effective way → meaning we get high-quality results.
  - Solve them in an efficient way → meaning we use as little time (or CPU power) as possible.
  - In other words, smart and fast solutions are key to staying competitive in modern industry.

  - Formally, it means:

    → **Maximizing** or **minimizing** an **objective function**

    → While **satisfying constraints**

1.2 Why Optimization Matters in Industry?

- Helps businesses reduce costs, increase efficiency, and make better decisions.
- From logistics to production planning, optimization allows automated, data-driven improvements.
- Examples:
  - Reducing delivery times in logistics.
  - Assigning workers or machines efficiently.
  - Minimizing waste in manufacturing processes.

1.3 What makes a good solution?

- Every optimization method works by selecting the “best” among many options. This is done using a selection criterion, which is based on the goal:
  - Maximization problems: Increase profit, output, etc.
  - Minimization problems: Reduce cost, time, etc.
- A well-designed selection criterion ensures:
  - All constraints are respected
  - The solution is as close to the best as possible
- A weak or wrong selection rule means poor-quality decisions.

1.4 Components of Optimization Problems

- **Decision variables**: What we can control or change (e.g., route a robot takes).
- **Objective function**: What we are trying to improve (e.g., minimize time or energy).
- **Constraints**: The rules we must follow (e.g., robot can’t visit the same point twice).

1.4 Components of Optimization Problems
Decision Variables

- **What are Decision Variables?**
  - Decision variables represent the elements of the problem that we can control or choose values for.
  - They define the structure of a potential solution.
- **Example in Industry:**
  - In a robot routing problem, the decision variables may be the sequence of locations the robot visits.
- **Concept:**
  - Each possible assignment to the decision variables corresponds to a different solution.
  - Our goal is to find the assignment that gives the best value for the objective function while satisfying all constraints.

1.4 Components of Optimization Problems
Objective Function

- **What is the Objective Function?**
  - The objective function defines what we are trying to optimize.
  - It is a mathematical expression that evaluates the quality or cost of a solution.
- **Common Objectives in Industry:**
  - Minimize travel time, energy consumption, or production costs.
  - Maximize efficiency, throughput, or customer satisfaction.
- **Form:**
  - Typically written as: $\min/\max\ f(x_1, x_2, \ldots, x_n)$

  where x_i are decision variables.
- **Note:**
  - Optimization can be either **minimization** (e.g., reduce cost) or **maximization** (e.g., increase output).

1.4 Components of Optimization Problems
Constraints

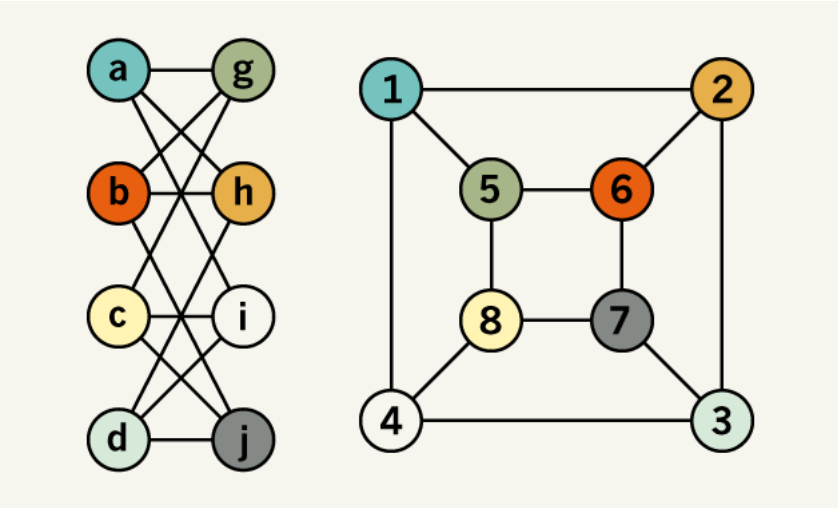
- **What are Constraints?**
  - Constraints define the rules or limits within which the solution must operate.
- **Types of Constraints:**
  - **Hard constraints**: Must be strictly followed (e.g., each city visited only once).
  - **Soft constraints**: Preferred but can be violated at a cost (e.g., preferred delivery times).
- **Examples:**
  - Robot cannot exceed battery limits.
  - Assembly must follow a specific sequence.
  - Only one robot can occupy a location at a time.
- **Purpose:**
  - Ensure the feasibility and practicality of the solution.

1.5 How We Solve Optimization Problems?

- When faced with an optimization problem (like TSP), there are different solution strategies depending on:
  - The **problem size**
  - The **desired level of accuracy**
  - The **available computational resources**
  - The **real-time requirement**

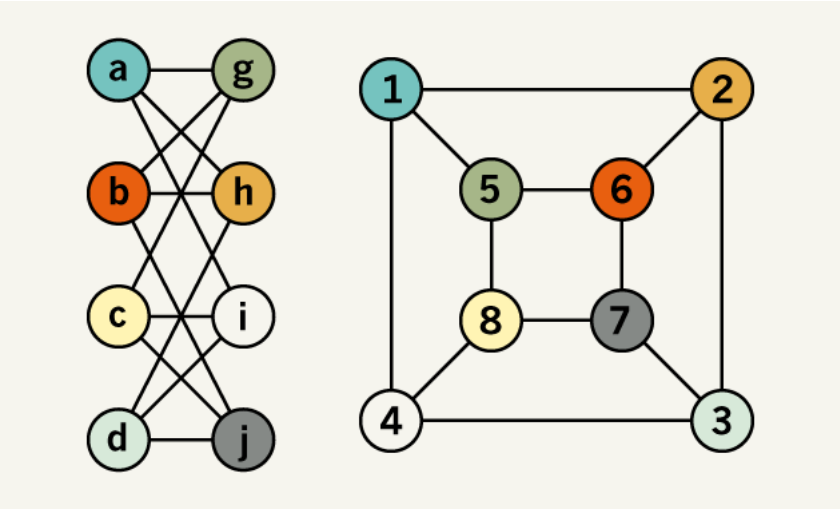
1.5 How We Solve Optimization Problems? Introduction to Graphs

- **Definition and Context**
  - In **mathematics and computer science**, a **graph** is a structure used to model pairwise relationships between entities.
- A graph is composed of:
  - **Nodes (or vertices):** The entities or locations (e.g., cities, machines, tasks).
  - **Edges (or links):** The connections or paths between them (e.g., roads, cables, task dependencies).



1.5 How We Solve Optimization Problems?
Introduction to Graphs

- **Why Graphs Matter in Optimization?**
- Graphs are **core to routing, logistics, and scheduling** problems.
- Many real-world optimization tasks are modeled using graphs:
  - TSP: Find the shortest loop visiting each node once.
  - BTSP: Alternate visits between two groups of nodes.
  - Network Design: Ensure robust and efficient connectivity.



1.5 How We Solve Optimization Problems?
Three Main Families of Methods

1. Brute-Force Methods

1. Try all possible combinations
2. Simple, but impractical for large problems
3. Good for understanding or testing small cases

2. Exact Methods

1. Solve with mathematical models
2. Guarantee optimal solutions
3. Use solvers like CBC, Gurobi, or CPLEX

3. Approximation / Heuristic / Metaheuristic Methods

1. Fast, do not guarantee optimality
2. Provide high-quality solutions for large or real-time problems
3. Examples: Nearest Neighbor, 2-opt, Genetic Algorithms

All three categories will be explored briefly in the next slides as an introduction. A more detailed analysis will follow in the dedicated parts of this training.

1.5 How We Solve Optimization Problems?
What Are Brute-Force Methods?

- **Definition:**
  Brute-force methods try **all possible solutions** and choose the best one based on a given objective.
- **Features:**
  - **Complete search:** Every combination is considered.
  - **Guaranteed optimality** – it always finds the best solution.
  - **Extremely inefficient** for large problems due to combinatorial explosion.
- **Example – TSP with 15 cities:**
  - Number of possible tours = $\frac{(n-1)!}{2}$ =43.589.145.600
  - Even with a computer evaluating 1 million tours/sec, it would take **~12 hours**.

1.5 How We Solve Optimization Problems? Advantages & Limitations of Brute-Force

- **Pros:**
  - Simple to implement.
  - Always provides the optimal solution (eventually!).
- **Cons:**
  - **Scales poorly**: Runtime grows **exponentially** with problem size.
  - Not practical for real-time or large-scale industrial problems.
- **When to use:**
  - Small-scale problems.
  - Benchmarking other methods.

1.5 How We Solve Optimization Problems?
What Are Exact Methods?

- **Definition:**
  Exact methods are algorithmic techniques that **guarantee** finding the optimal solution by **mathematically modeling the problem** and using systematic procedures to solve it.
- **Types:**
  - **Linear Programming (LP)**
  - **Integer Programming (IP)**
  - **Mixed-Integer Linear Programming (MILP)**
- **How they work:**
  - Use a **mathematical model**: objective function + constraints
  - Solve using optimization solvers (e.g., **CBC**, **Gurobi**, **CPLEX**)
  - Intelligent search: they avoid unnecessary branches, unlike brute-force

1.5 How We Solve Optimization Problems? Example of an Exact Method (IP for TSP)

- **Objective Function:**
  Minimize the total cost/time/distance of visiting all locations
- **Decision Variables:**
  $x_{ij} = 1$ if we go from node i to node j, 0 otherwise
- **Constraints:**
  - Each node is visited exactly once
  - No sub-tours are allowed (subtour elimination)
  - Flow conservation: every node has one in/out edge
- **Solvers use techniques like:**
  - Branch & Bound
  - Cutting Planes
  - Presolve reductions

1.5 How We Solve Optimization Problems?
What Are Heuristic Methods?

- **Definition:**
  Heuristic methods are problem-solving strategies that **build good solutions fast**, using **rules or logic** instead of exploring all possible options.
- **How they work:**
  - Make decisions **step-by-step**, choosing the “best local” move
  - Do **not** guarantee the best global solution
  - Aim for a **balance between speed and quality**
- **Characteristics:**
  - **Fast** and lightweight (especially on large problems)
  - **Simple to implement**
  - Can be **used in real-time systems**
- **Examples:**
  - **Nearest Neighbor (NN):** always pick the closest unvisited node
  - **Greedy Methods:** optimize a part of the solution at each step

1.5 How We Solve Optimization Problems?
What Are Metaheuristic Methods?

- **Definition:**
  Metaheuristics are **higher-level algorithms** that guide heuristics to **search smarter**, often inspired by **natural processes**, like evolution or physics.
- **How they work:**
  - Use **stochastic** or **adaptive mechanisms** to avoid poor local optima
  - Balance **exploration** (trying new areas) and **exploitation** (refining known areas)
- **Characteristics:**
  - **Flexible and problem-independent**
  - Often used when heuristics get stuck
  - Provide **good-quality solutions** even for complex, large-scale problems
- **Examples:**
  - **2-opt**: iteratively improves a route by swapping pairs
  - **Simulated Annealing:** probabilistically accepts worse moves to escape local minima
  - **Genetic Algorithms:** simulate evolution with selection, crossover, mutation

1.5 How We Solve Optimization Problems?
Heuristic Example: Nearest Neighbor

- **Scenario**:
  A delivery robot starts at **A** and must visit 4 locations (B, C, D, E), then return to A.
- **Method**:
  Always move to the **nearest unvisited location**.
- **Step-by-Step:**
  - A → B (10 km)
    B → E (17 km)
    E → D (15 km)
    D → C (30 km)
    C → A (15 km)
- **Route**: A → B → E → D → C → A

  Total Distance: **87 km**
- **Takeaway**:
  Heuristics like Nearest Neighbor are **fast** and give **good results**, even if not always optimal.

1.5 How We Solve Optimization Problems?
Optimization Example (1/2)

- **Scenario**: You're organizing a road trip with your friends.
- **Starting Point**: Athens.
- **Destinations**: 15 major cities in total (including Athens).
- **Goal**: Visit each city **exactly once**, then return to Athens.
- **Available Data**: You have access to all travel distances between cities (via Google Maps, Bing Maps, etc.).
- **Question**: What is the **best possible route** that minimizes your total travel distance?

1.5 How We Solve Optimization Problems?
Optimization Example (2/2)

- This is a classic formulation of the Traveling Salesman Problem (TSP).
- It's not just about finding a valid tour — it's about finding the optimal one.
- Discuss the combinatorial explosion of possible routes (e.g., 15! permutations).
  - $(15-1)! = 14! = 8.8 * 10^{10} = 88\ billion\ solutions\ (approx.)$
- This leads into the need for optimization techniques, not just brute-force trial and error.

1.6 Further Reading in Optimization

- **Numerical Optimization**
  Jorge Nocedal & Stephen Wright (1999)
  - A comprehensive guide to optimization algorithms.
  - Covers unconstrained and constrained optimization.
  - Excellent for understanding practical implementations and mathematical underpinnings.
  - Highly recommended for those working with **nonlinear programming** and **numerical solvers.**

- **Nonlinear Programming**
  Dimitri P. Bertsekas (1997)
  - A definitive text on **nonlinear optimization**, including theory, algorithms, and applications.
  - Explains optimality conditions, duality theory, and constrained optimization methods.
  - Widely used in operational research and industrial engineering.

- **Convex Optimization Theory**
  Dimitri Bertsekas (2009)
  - Provides rigorous treatment of **convex sets**, **functions**, and **optimization problems**.
  - Bridges the gap between theory and applications in **control**, **machine learning**, and **economics**.
  - A great resource for mastering **modern optimization techniques**.

Quick Recap Optimization: What Have We Learned?

- Optimization is about finding the best possible solution under given constraints.
- We aim for solutions that are both effective (high quality) and efficient (low cost or time).
- Computational methods and algorithms are crucial tools for modern decision-making in industry.
- A good selection criterion (objective) leads to better results.
- Real-world problems (like trip planning or production scheduling) require smart algorithms—not brute force.

Q&A 1: Optimization Basics (1/2)

1. What is the main goal of optimization?
   A. To try all possible solutions
   B. To find the best solution that meets all constraints
   C. To avoid making any decisions,
   D. All of the above
2. Which of the following is an example of a constraint in an optimization problem?
   A. Wanting to reduce travel time
   B. A machine can only work 8 hours a day
   C. Wanting to maximize output
   D. None of the above
3. Why don’t we just check all possible solutions (brute force)?
   A. Because it’s not fun
   B. Because it takes too long, even for a computer
   C. Because computers can't do it
   D. All of the above
4. What is a “selection criterion”?
   A. A way to choose the best solution
   B. A tool used for simulation
   C. A list of employees
   D. None of the above

Q&A 1: Optimization Basics (2/2)

- Answers:
  - 1. B
  - 2. B
  - 3. B
  - 4. A

Section 2: The Traveling Salesman Problem (TSP)

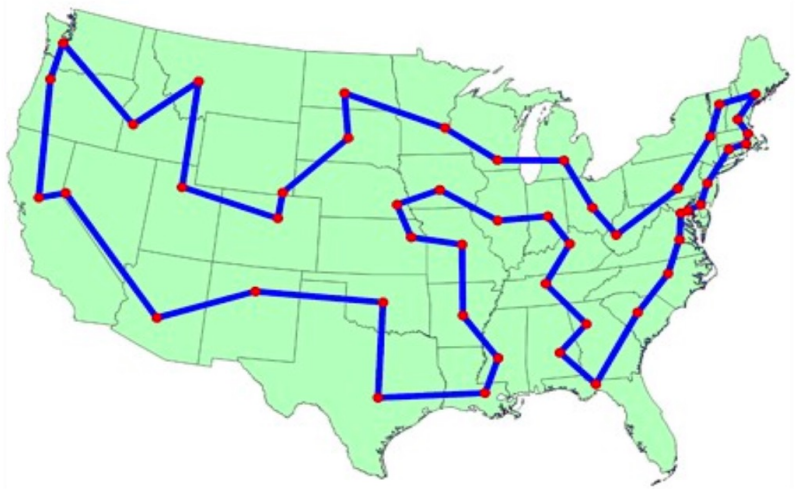
1. Optimization
   - 1.1 Introduction to Optimization
   - 1.2 Why Optimization Matters in Industry?
   - 1.3 What makes a good solution?
   - 1.4 Components of Optimization Problems
   - 1.5 How We Solve Optimization Problems
   - 1.6 Further Reading in Optimization
   - Q&A 1: Optimization Basics
2. **The Traveling Salesman Problem (TSP)**
   - **2.1 Introduction to TSP**
   - **2.2 Formal Definition, Mathematical Models, and Solving Methods**
   - **2.3 Why is it Challenging?**
   - **2.4 TSP Example**
   - **2.5 Further Reading on the TSP**
   - **Q&A 2: TSP Understanding**
3. The Bipartite TSP (BTSP)
   - 3.1 Introduction to BTSP
   - 3.2 Formal Definition and Bipartite Graphs
   - 3.3 Example of BTSP
   - 3.4 Further Reading on the BTSP
   - Q&A 3: BTSP Understanding
4. Industrial Applications of TSP and BTSP
   - 4.1 Use Cases of TSP in Logistics and Routing
   - 4.2 Use Cases of BTSP in Robotics and Assembly Lines
   - 4.3 Case Studies and Examples
   - Q&A 4: Application Scenarios

2.1 Introduction to the Traveling Salesman Problem (TSP)

- **What is the TSP?**
  - The **Traveling Salesman Problem (TSP)** is one of the most fundamental and well-studied problems in combinatorial optimization.
  - A salesman needs to visit a number of cities **exactly once** and return to the starting city.
  - The **goal** is to determine the **shortest possible route** that visits each city once and returns to the origin.
  - It's a **minimization problem** over all possible permutations of the cities.

2.1 Introduction to the Traveling Salesman Problem (TSP)

- Given the costs required for travelling between every node pair, the standard Traveling Salesman Problem (TSP) is aimed at generating the route (node sequence) that:
  - The total cost of the route is minimized
  - The route is a cycle (the route begins from the central location and
  - returns to the central station)
  - The route contains all nodes just once



2.2 Formal Definition, Mathematical Models, and Solving Methods

- The **Traveling Salesman Problem (TSP)** is defined on a **weighted complete graph** $G = (V, E)$, where:
- V is a set of **vertices** (cities/locations)
- E is a set of **edges** between all pairs of vertices
- Each edge $(i, j) \in E$ has a **non-negative weight** $\mathrm{d}(i, j)$, representing the cost (distance, time, etc.) to travel between city i and city j

2.2 Formal Definition, Mathematical Models, and Solving Methods
MTZ Mathematical Formulation

Miller–Tucker–Zemlin formulation

Label the cities with the numbers $1, \ldots, n$ and define:

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} :$$

$$\begin{aligned}
& x_{ij} \in \{0, 1\} && i, j = 1, \ldots, n; \\
& u_i \in \mathbf{Z} && i = 2, \ldots, n; \\
& \sum_{i=1, i \neq j}^{n} x_{ij} = 1 && j = 1, \ldots, n; \\
& \sum_{j=1, j \neq i}^{n} x_{ij} = 1 && i = 1, \ldots, n; \\
& u_i - u_j + n x_{ij} \leq n - 1 && 2 \leq i \neq j \leq n; \\
& 1 \leq u_i \leq n - 1 && 2 \leq i \leq n.
\end{aligned}$$

2.2 Formal Definition, Mathematical Models, and Solving Methods
DFJ Mathematical Formulation

Dantzig–Fulkerson–Johnson formulation

Label the cities with the numbers 1, …, *n* and define:

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} :$$

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n;$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n;$$

$$\sum_{i \in Q} \sum_{j \neq i, j \in Q} x_{ij} \leq |Q| - 1 \qquad \forall Q \subsetneq \{1, \ldots, n\}, |Q| \geq 2$$

2.2 Formal Definition, Mathematical Models, and Solving Methods

Objective of TSP

- Find the **shortest possible Hamiltonian cycle** (i.e., a route that visits each city **exactly once** and returns to the starting point) such that the total cost is minimized.
- Formally:
  - $Minimize$: $\sum_{i=1}^{n} d(c_i + c_{i+1}) + d(c_n, c_1)$
- Where:
  - $c_1, c_{2,} \cdots, c_n$ is a **permutation** of the cities
  - $d(c_i + c_{i+1})$ is the cost of moving from city c_i to city c_{i+1}

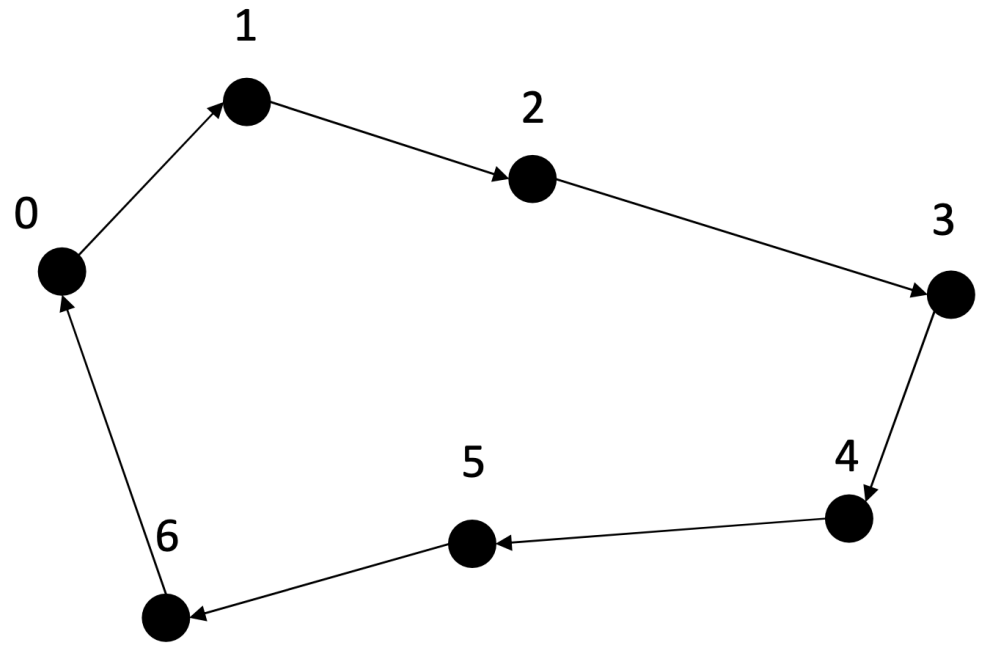
2.2 Formal Definition, Mathematical Models, and Solving Methods Solving Methods

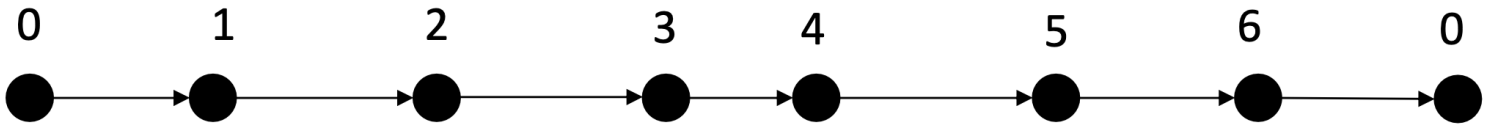
- Same 3 types as refer to Optimization part:
  - Brute-force
  - Exact
  - Heuristic
- Example:
  - **Visiting 4 Customers from a Depot**
  - How many possible tours are there?
    - (4! = 24 possible sequences)
- Brute-force would evaluate all 24.
- Heuristics like Nearest Neighbor can offer a quick good tour.
- This is the structure of a **Classical TSP problem**.

2.3 Why is it Challenging?

- The number of possible tours grows **factorially**:
  For n cities → $\frac{(n-1)!}{2}$ unique tours (due to symmetry)
- Example:
  - 4 cities → 3 possible complete tours
  - 10 cities → 181,440 possible tours
  - 15 cities → **43 billion** possible routes!
- This makes **brute-force methods** infeasible for large n — we need **optimization methods**.

2.4 TSP Example
Illustration of a TSP solution





2.4 TSP Example

Example with Cost Matrix and Nearest Neighbor Method

- We are in the central warehouse (depot) of a production site
- We must satisfy the product requests received by 4 customers
- Our company vehicle is going to be routed to start from the depot, visit the five customers and then return to the depot
- The cost for moving between every node pair is given in the following symmetric cost matrix:

From\ To	**0**	**1**	**2**	**3**	**4**
0	0	37	84	46	47
1	37	0	11	15	47
2	84	11	0	20	15
3	46	15	20	0	20
4	47	47	15	20	0

2.4 TSP Example
Solution with Nearest Neighbor – Step-by-Step (2/2)

- We start from the **Depot (Node 0)**.
- **Initial Solution:**
  Current path: **0**
  Visited: {0}
  Unvisited: {1, 2, 3, 4}
- **Solution after the 1st insertion:**
  Nearest from 0: **Node 1** (cost = 37)
  Path: **0 → 1**
  Visited: {0, 1}
  Unvisited: {2, 3, 4}
- **Solution after the 2nd insertion:**
  Nearest from 1: **Node 2** (cost = 11)
  Path: **0 → 1 → 2**
  Visited: {0, 1, 2}
  Unvisited: {3, 4}

2.4 TSP Example
Solution with Nearest Neighbor – Step-by-Step (2/2)

- **Solution after the 3rd insertion:**
  Nearest from 2: **Node 4** (cost = 15)
  Path: **0 → 1 → 2 → 4**
  Visited: {0, 1, 2, 4}
  Unvisited: {3}
- **Solution after the 4th insertion:**
  Nearest from 4: **Node 3** (cost = 20)
  Path: **0 → 1 → 2 → 4 → 3**
  Visited: {0, 1, 2, 3, 4}
- **Solution after returning to depot:**
  Return from 3 to 0: cost = 46
  Final tour: **0 → 1 → 2 → 4 → 3 → 0**
- **Total cost:**
  = 37 (0→1) + 11 (1→2) + 15 (2→4) + 20 (4→3) + 46 (3→0)
  = **129**

2.5 Further Reading on the TSP

- **The Traveling Salesman: Computational Solutions for TSP Applications**
  Gerhard Reinelt (2003)
  - Offers both theory and practical methods.
  - Includes benchmark instances used in research and competitions.
  - Great for understanding **experimental evaluation** and **implementation aspects**.
- **Traveling Salesman Problem: Theory and Applications**
  Donald Davendra (Ed.) (2010)
  - A collection of research works combining **theory**, **applications**, and **novel approaches**.
  - Discusses **metaheuristics**, **multi-objective variants**, and **real-world industrial scenarios**.
  - A good read for those interested in **diverse formulations and adaptations** of the TSP.
- **The Traveling Salesman Problem and Related Problems**
  Gavish & Graves (1978)
  - Foundational paper linking TSP with logistics and manufacturing.
- **Literature Review on Travelling Salesman Problem**
  Chetna Dahiya & Shabnam Sangwan (2018)
  - A broad **survey** of TSP variants, algorithms, and **modern trends**.
  - Useful for quick **overview of solution techniques** and applications.

Quick Recap TSP: What Have We Learned?

- **Core Idea:**
  - Visit **each city exactly once** and return to the starting city.
  - Goal: **Minimize total travel cost/distance/time**.
  - TSP is a **combinatorial optimization** problem and is **NP-hard**.
- **Learning Points:**
  - TSP applies to logistics, route planning, circuit design, etc.
  - **Exact methods** (e.g., Integer Programming) guarantee optimality but are slow for large problems.
  - **Heuristics** (e.g., Nearest Neighbor) are faster and provide near-optimal solutions.
  - **Metaheuristics** (e.g., 2-opt, Genetic Algorithms) balance performance and speed.

Q&A 2: TSP Understanding (1/2)

1. **What is the main objective of the TSP?**
   A. Visit the same city multiple times
   B. Minimize the number of routes
   C. Minimize the travel distance and return to the origin
   D. Visit only half of the cities
2. **Which algorithm is exact for solving small TSPs?**
   A. Nearest Neighbor
   B. Simulated Annealing
   C. Integer Programming
   D. None of the above
3. **What makes TSP difficult to solve for large instances?**
   A. It's a linear problem
   B. It requires multiple start points
   C. Number of possible tours grows factorially
   D. It doesn’t return to the starting city
4. **Which of the following is an application of the TSP?**
   A. Weather prediction
   B. Circuit board drilling
   C. Password encryption
   D. All of the above

Q&A 2: TSP Understanding (2/2)

- Answers:
  - 1. C
  - 2. C
  - 3. C
  - 4. B

Section 3: The Bipartite TSP (BTSP)

1. Optimization
   - 1.1 Introduction to Optimization
   - 1.2 Why Optimization Matters in Industry?
   - 1.3 What makes a good solution?
   - 1.4 Components of Optimization Problems
   - 1.5 How We Solve Optimization Problems
   - 1.6 Further Reading in Optimization
   - Q&A 1: Optimization Basics
2. The Traveling Salesman Problem (TSP)
   - 2.1 Introduction to TSP
   - 2.2 Formal Definition, Mathematical Models, and Solving Methods
   - 2.3 Why is it Challenging?
   - 2.4 TSP Example
   - 2.5 Further Reading on the TSP
   - Q&A 2: TSP Understanding
3. **The Bipartite TSP (BTSP)**
   - **3.1 Introduction to BTSP**
   - **3.2 Formal Definition and Bipartite Graphs**
   - **3.3 Example of BTSP**
   - **3.4 Further Reading on the BTSP**
   - **Q&A 3: BTSP Understanding**
4. Industrial Applications of TSP and BTSP
   - 4.1 Use Cases of TSP in Logistics and Routing
   - 4.2 Use Cases of BTSP in Robotics and Assembly Lines
   - 4.3 Case Studies and Visual Examples
   - Q&A 4: Application Scenarios

3.1 Introduction to the Bipartite Traveling Salesman Problem (BTSP)

- **What is the BTSP?**
  - The **Bipartite Traveling Salesman Problem (BTSP)** is a variation of the classical TSP.
  - In BTSP, the nodes are divided into two disjoint sets, commonly referred to as:
    - **Set A** (e.g., Gravity Racks or Pick Locations)
    - **Set B** (e.g., Kit Holders or Place Locations)
  - A valid tour **must alternate** between nodes in Set A and Set B.
- **Why is this important?**
  - BTSP models real-world industrial tasks, especially in **Pick-and-Place operations** where a robot picks items from one type of location and places them in another.
  - This alternation reflects real constraints in manufacturing, such as:
    - A robot **cannot pick two items in a row** or **place two items in a row** without violating physical/logical rules.

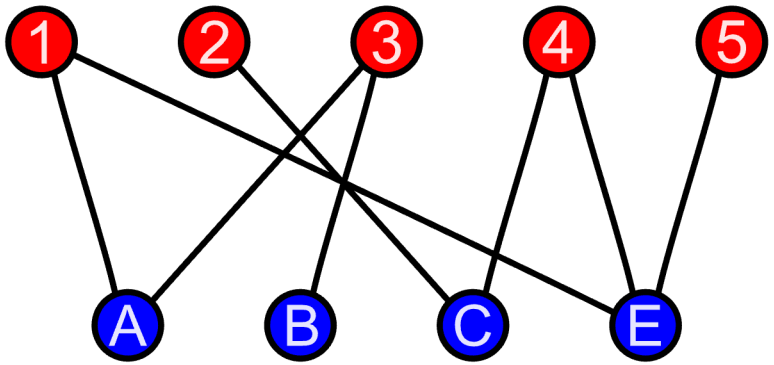
3.2 Formal Definition and Bipartite Graphs
Objective and Solution Methods of BTSP

- Find the **shortest possible alternating tour** that:
  - Starts and ends at specific nodes (e.g., depot or rest position)
  - Visits each node **once**
  - Alternates between **picking and placing**
- A valid tour must **alternate** between nodes in Set A and Set B.
- The graph is **bipartite**, meaning edges only connect nodes from A to B and vice versa, not within the same set.
- Objective: **Minimize total cost** (e.g., time or distance) while visiting each node **exactly once**, respecting the alternation rule.
- The problem is **NP-hard**, like the classical TSP.
- The solution methods are the same with the classical TSP. The solution methods will be covered in depth in the course.

3.2 Formal Definition and Bipartite Graphs

Bipartite Graphs

- **Definition**:
  A **bipartite graph** is a graph where the set of vertices V can be divided into two **disjoint sets**, V_1 and V_2, such that **every edge** connects a vertex in V_1 to one in V_2. – *Skiena (1990)*



- **Special Case – Complete Bipartite Graph**:
  If **every** vertex in V_1 is connected to **every** vertex in V_2, the graph is called a **complete bipartite graph**.

3.3 Example of BTSP
Bipartite TSP Tour with 2 Pick Locations and 2 Place Locations

- **Problem Setup**
  - A robot starts at a depot (**Node 0.0**).
  - It needs to pick components from **Set A** (Gravity Racks):
    - A1 = 1.1.1
    - A2 = 1.2.1
  - It must place them at **Set B** (Kit Holders):
    - B1 = 1.1
    - B2 = 1.2
  - The robot finishes by returning to a final depot position (**Node 0.0.0**).
- **Valid Tour Structure**
  - The robot must alternate: **Depot → A → B → A → B → Return**
  - Example tour:
    0.0 → 1.1.1 → 1.2 → 1.2.1 → 1.1 → 0.0.0

3.3 Example of BTSP
Bipartite TSP Tour with 2 Pick Locations and 2 Place Locations

- **Distance Matrix (simplified, symmetric):**

From \ To	1.1.1	1.2.1	1.1	1.2	0.0	0.0.0
1.1.1	0	10	6	7	4	9
1.2.1	10	0	8	5	6	10
1.1	6	8	0	9	7	3
1.2	7	5	9	0	5	4
0.0	4	6	7	5	0	∞
0.0.0	9	10	3	4	∞	0

3.3 Example of BTSP
Nearest Neighbor Example Solution

1. Start at **Depot (0.0)**
2. Nearest pick: **1.1.1** (distance = 4)
3. Nearest place: **1.2** (distance = 7)
4. Next pick: **1.2.1** (distance = 5)
5. Nearest place: **1.1** (distance = 8)
6. Return to **Final Depot (0.0.0)** (distance = 3)

- **Total Tour**: 0.0 → 1.1.1 → 1.2 → 1.2.1 → 1.1 → 0.0.0
- **Total Distance** = 4 + 7 + 5 + 8 + 3 = **27 units**

3.4 Further Reading on the BTSP

- **A Note on the Polytope of Bipartite TSP**
  Kovács, G., Tuza, Z., Vizvári, B., Jabbari, H. K. (2018)
  - Explores polyhedral differences between BTSP and classical TSP.
  - Proves that **comb inequalities** are not facet-defining for BTSP.
  - Essential for researchers interested in LP/IP formulations and cutting-plane techniques.
- **The Bipartite Travelling Salesman Problem: A Pyramidally Solvable Case**
  Deineko, Klinz & Woeginger (2024)
  - Identifies special cases of BTSP that are solvable in polynomial time.
  - Enhances theoretical understanding and guides practical algorithm design.
- **Solution for a Bipartite Euclidean Traveling-Salesman Problem in One Dimension**
  Caracciolo, S., Di Gioacchino, A., Gherardi, M., Malatesta, E. M. (2018)
  - Presents closed-form solution for **1D bipartite Euclidean TSP**.
  - Connects statistical physics with BTSP optimization theory.
  - Provides a theoretical baseline for evaluating algorithm performance.

Quick Recap BTSP: What Have We Learned?

- **Core Idea:**
  - Nodes are split into **two disjoint sets** (e.g., pick and place).
  - The tour **must alternate** between a node in set A and a node in set B.
  - Also returns to a **start/depot** node, just like in TSP.
- **Learning Points:**
  - Models **pick-and-place tasks** in robotics.
  - Adds additional **structure/constraints** to the standard TSP.
  - Applications in **industrial scheduling**, **assembly**, **task routing**.
  - Requires adapting classical methods to respect **bipartite rules**.

Q&A 3: BTSP Understanding (1/2)

1. **What is a unique constraint in BTSP?**
   A. You can only travel between nodes in the same set
   B. You can revisit nodes
   C. You must alternate between two sets of nodes
   D. None of the above
2. **Which classic method can be adapted to solve BTSP?**
   A. Brute Force
   B. Integer Programming
   C. Nearest Neighbor
   D. All of the above
3. **What happens if a BTSP tour does not alternate between sets?**
   A. The result is optimal
   B. The solution is invalid
   C. It improves execution time
   D. It simplifies the problem

Q&A 3: BTSP Understanding (2/2)

- Answers:
  - 1. C
  - 2. D
  - 4. B

Section 4: Industrial Applications of TSP and BTSP

1. Optimization
   - 1.1 Introduction to Optimization
   - 1.2 Why Optimization Matters in Industry?
   - 1.3 What makes a good solution?
   - 1.4 Components of Optimization Problems
   - 1.5 How We Solve Optimization Problems
   - 1.6 Further Reading in Optimization
   - Q&A 1: Optimization Basics
2. The Traveling Salesman Problem (TSP)
   - 2.1 Introduction to TSP
   - 2.2 Formal Definition, Mathematical Models, and Solving Methods
   - 2.3 Why is it Challenging?
   - 2.4 TSP Example
   - 2.5 Further Reading on the TSP
   - Q&A 2: TSP Understanding
3. The Bipartite TSP (BTSP)
   - 3.1 Introduction to BTSP
   - 3.2 Formal Definition and Bipartite Graphs
   - 3.3 Example of BTSP
   - 3.4 Further Reading on the BTSP
   - Q&A 3: BTSP Understanding
4. **Industrial Applications of TSP and BTSP**
   - **4.1 Use Cases of TSP in Logistics and Routing**
   - **4.2 Use Cases of BTSP in Robotics and Assembly Lines**
   - **4.3 Case Studies and Examples**
   - **Q&A 4: Application Scenarios**

4. Industrial Applications of TSP and BTSP

- TSP and BTSP are not just mathematical puzzles—they model real logistics, scheduling, and movement tasks in modern factories and warehouses.
- Solving these problems leads to cost savings, efficiency improvements, and enhanced customer satisfaction.

- **Why Is This Important for Industry?**
  - **Operational Efficiency:** Optimized routes and sequences lead to faster operations and reduced operational costs.
  - **Resource Optimization:** Efficient scheduling and routing minimize resource wastage, including fuel, time, and labor.
  - **Customer Satisfaction:** Timely deliveries and services enhance customer experience and loyalty.
  - **Scalability:** TSP and BTSP solutions support the scaling of operations without compromising efficiency.

4.1 Use Cases of TSP in Logistics and Routing

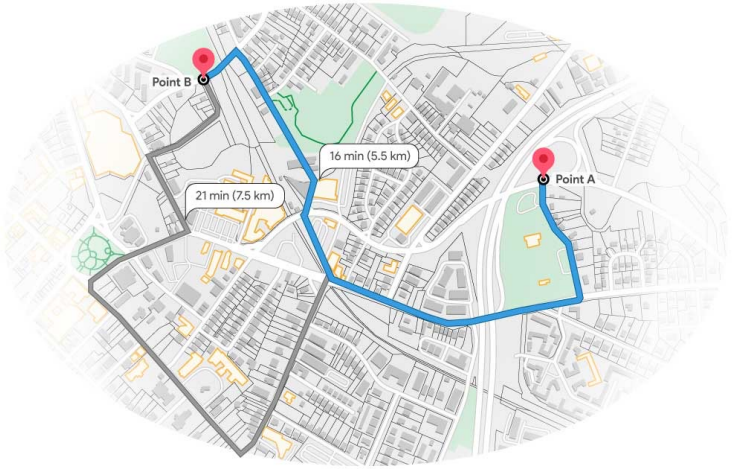
- **Vehicle Routing in Logistics**
  TSP is the core of many **route optimization tools** used in delivery, postal services, and ride-sharing.
- **Supply Chain Optimization**
  Optimize the sequence of supplier visits, warehouse replenishments, or last-mile delivery.
- **Maintenance Scheduling**
  Minimize travel time for technicians who need to service multiple locations in a shift.
- **Example**:
  A logistics company delivering parcels to 10 cities aims to find the shortest possible loop – the solution to this is a TSP tour.

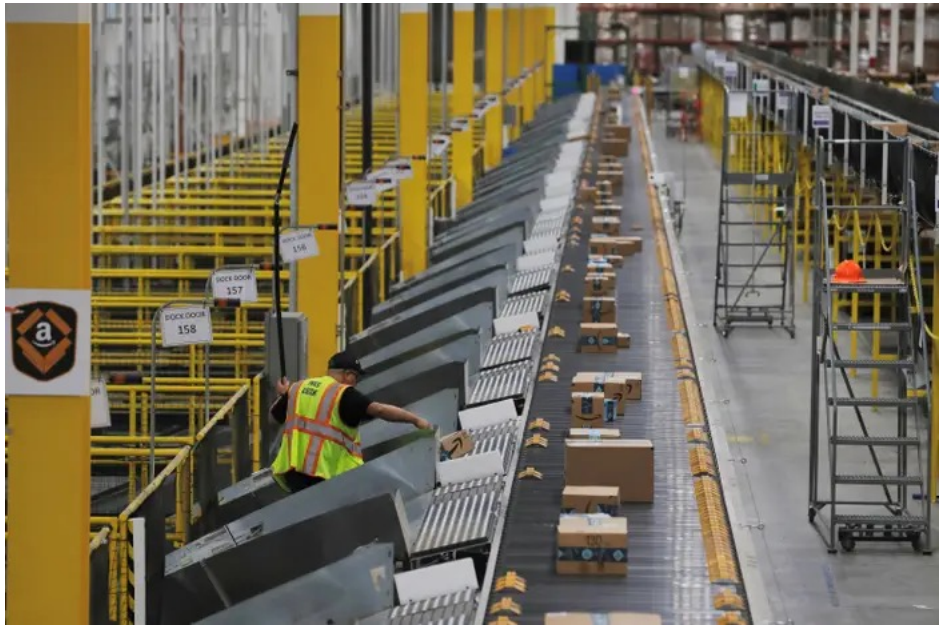
4.2 Use Cases of BTSP in Robotics and Assembly Lines

- **Pick-and-Place Robotics in Manufacturing**
  Robots pick components from **gravity racks** (Set A) and place them on **assembly points** (Set B).
- **Alternating Tasks Between Two Zones**
  Applications where a robot must alternate between two roles or zones – e.g., scanning stations and packaging zones.
- **Warehouse Automation**
  BTSP is ideal for scheduling robots that alternate between **picking and depositing** items.
- **Example**:
  In a factory, a robot arm must move between 5 component bins and 5 assembly points, alternating operations – this forms a **BTSP**.

4.3 Case Studies and Examples

- **Example – TSP in Postal Services**
- **Company**: FedEx, UPS, DHL
- **Use**: Daily route planning for parcel delivery
- **Tool**: Optimization software based on TSP





4.3 Case Studies and Examples

Case Snapshot – BTSP in MODAPTO

- **Scenario:** A robot at a factory picks parts from a gravity rack (Set A) and delivers them to fixed kit holders (Set B).
- **Challenge:** Optimize the robot's route so it alternates between pick (A) and place (B), **minimizing total movement time** and ending at the depot.
- **Solution:**
  - Modeled using **Bipartite TSP**
  - Solved via:
    - Nearest Neighbor heuristic
    - Exact Integer Programming
    - Reinforcement Learning (Q-learning)
- **Outcome:**
  - **20–40%** travel time reduction
  - Compatible with simulation tools & MODAPTO systems

- This specific case will be analyzed in depth in course Task-Specific Optimization – Robot picking sequence optimization for pick-and-place tasks (e.g., operation time, pick-and-place time)

4.3 Case Studies and Examples

Common Industrial Use Cases

1. Logistics and Transportation:

- **Delivery Route Optimization:** Companies like FedEx and UPS use TSP algorithms to determine the most efficient delivery routes, reducing fuel consumption and delivery times.
- **Freight and Passenger Transport:** TSP helps in planning routes for freight delivery and passenger transport, ensuring timely and cost-effective services.

2. Manufacturing and Production Planning:

- **Machine Maintenance Scheduling:** TSP is used to plan the shortest route for maintenance personnel to service machines, minimizing downtime.
- **PCB Drilling and CNC Machining:** TSP algorithms optimize the sequence of drilling holes or machining parts, reducing tool movement and production time.

3. Robotics and Automation:

- **Pick-and-Place Operations:** BTSP models the alternating tasks of picking components and placing them in assembly points, optimizing robotic movements in manufacturing lines.
- **Warehouse Automation:** Autonomous mobile robots (AMRs) use TSP solutions to navigate warehouses efficiently, enhancing order fulfillment processes.

4. Network Design and Optimization:

- **Telecommunications:** TSP assists in designing efficient network paths, reducing signal loss and improving performance.

5. DNA Sequencing:

- **Genome Assembly:** TSP algorithms help in determining the optimal sequence of DNA fragments, facilitating accurate genome reconstruction.

4.3 Case Studies and Examples
Real-World Use of TSP & BTSP by Leading Companies

- **Amazon (Warehouse Logistics)**
  - Uses TSP-inspired algorithms to route robots in fulfillment centers.
  - **Goal:** Minimize travel time between shelves and packing stations.
- **DHL & FedEx (Route Planning)**
  - Delivery route optimization using variants of the TSP.
  - Reduces fuel, ensures on-time delivery.
- **Tesla / BMW / KUKA (Pick-and-Place Robotics)**
  - Robotic arms in production lines use BTSP-based planning to switch between component racks and assembly lines.
  - Efficiency gains through **fewer collisions, faster cycles**.
- **IBM & Siemens (Digital Twins)**
  - Integrate optimization services into virtual replicas of factory floors.
  - Simulation + BTSP solve path planning for robotic workflows.

Q&A 4: Application Scenarios Understanding (1/2)

1. **Which of the following is NOT a typical TSP application?**
   A. Vehicle routing
   B. Maintenance planning
   C. Inventory pricing
   D. All of the above
2. **What makes BTSP different from TSP in industrial use?**
   A. It has fewer locations
   B. It does not allow revisits
   C. It alternates between two disjoint sets
   D. None of the above
3. **In which setting is BTSP most useful?**
   A. Emergency route planning
   B. Multi-stop sightseeing tours
   C. Manufacturing assembly operations
   D. All of the above

Q&A 4: Application Scenarios Understanding (2/2)

- Answers:
  - 1. C
  - 2. C
  - 3. C

CONSORTIUM

MOAPTO

meet the team

ATC
ATHENS TECHNOLOGY CENTER

Fraunhofer
IOSB

UNIVERSITÉ
DE LORRAINE

UNIVERSITY OF PIRAEUS
RESEARCH CENTER

THE E-BUSINESS RESEARCH CENTER
ELTRUN
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

EKS
InTec GmbH

BOC
www.boc-group.com

AEGIS
IT RESEARCH

LIC
LIETUVOS
INOVACIJŲ
CENTRAS

FFT

SEW
USOCOME

CRF

ILTAR
ITALBOX

Funded by
the European Union

CONTACT us

MODAPTO

modapto.eu

info@modapto.eu

MODAPTO Horizon Europe Project

@MODAPTO_eu

MODAPTO Horizon Europe Project

Funded by
the European Union