Abstract

This thesis explores the application of Particle Swarm Optimization (PSO) to solve the Capacitated Vehicle Routing Problem (CVRP). The proposed PSO-based approach demonstrates superior performance in terms of solution quality and computational efficiency compared to existing methods. The findings highlight the potential of PSO as an effective tool for addressing complex combinatorial optimization problems like the CVRP, with implications for real-world transportation and logistics applications.

Key Words

Capacitated Vehicle Routing Problem, Particle Swarm Optimization, Best Known Position

Contents

ListofTables

Acknowledgement

I thank Almighty God for blessing us with good health and the determination to embark on and complete this thesis.

I would like to express my heartfelt gratitude to my thesis advisor, **Mr. Ramdani Zoubir**, for his guidance, direction, support, and valuable advice throughout the process.

I extend my sincere thanks to all the professors, jury members, guest speakers, and everyone who, through their words, writings, guidance, and constructive criticism, have influenced my thinking and willingly met with me to answer my queries during my research.

I am grateful to my dear parents who have always been there for me, and I appreciate the encouragement I have received from my sisters and brother.

To all these individuals, I offer my thanks, respect, and profound gratitude.

General Introduction

Operational research (OR) has long been a subject of debate, as scholars have sought to define its scope and purpose. According to the Journal of the Operational Research Society (U.K.)[18], OR is defined as follows:

> "Operational research is the application of scientific methods to address complex problems arising in the management and direction of large systems encompassing people, machinery, materials, and financial resources. It entails developing scientific models of these systems, incorporating measurements of factors such as chance and risk, to predict and compare the outcomes of alternative decisions, strategies, or controls. The ultimate aim is to provide management with scientifically grounded policies and actions."

At the heart of OR lies the concept of optimization. In the context of operational research, optimization refers to the act of improving or perfecting a solution. In practice, optimization involves formulating and analytically or statistically solving problems that entail either minimizing or maximizing a specific objective function.

Optimization methods can be broadly categorized into two approaches. The first is single-objective optimization, which seeks to minimize or maximize a single objective function, aiming to identify the optimal solution—the best possible outcome within the given constraints. The second approach is multi-objective optimization, which concurrently optimizes multiple objective functions, considering the inherent trade-offs between them.

Various methods exist for solving optimization problems. Exact methods provide optimal solutions for small-scale problems but become computationally infeasible for larger ones. On the other hand, metaheuristic methods offer efficient ways to tackle large-scale optimization problems, even though they do not guarantee optimality. Instead, they provide high-quality solutions within a reasonable computational time. Prominent metaheuristic methods include Constraint Logic Programming, Genetic Algorithms, Particle Swarm Optimization Algorithms, and Ant Colony Optimization Algorithms, among others.

In this thesis, we focus on addressing the Capacitated Vehicle Routing Problem (CVRP), a critical challenge in logistics and transportation management. CVRP involves determining optimal vehicle routes while considering capacity constraints. Specifically, every customer must be visited by a single vehicle with limited carrying capacity, originating and concluding its route at a central depot.

To tackle this problem, we propose utilizing the Particle Swarm Optimization (PSO) algo-

rithm. PSO, a metaheuristic algorithm inspired by the collective behavior of bird flocking or fish schooling, has shown promise in addressing complex optimization problems. By leveraging the principles of PSO, we aim to find effective solutions for the CVRP, optimizing the routes of multiple vehicles while adhering to their capacity limitations. In the first chapter, i will explore

the fundamental concepts and definitions pertaining to solving the Capacitated Vehicle Routing Problem (CVRP) using Particle Swarm Optimization (PSO). In the second chapter, I will present a general idea of the vehicle routing problem and its origin and some insights into the mathematical concept of the problem.

In the third chapter, I will talk about meta-heuristics algorithms, and provide a brief definition and concepts of the most responsive algorithms, which are Ant Colony Optimization (ACO), genetic Algorithms (GA), and our study subject Particle Swarm Optimization (PSO).

In the fourth chapter, I will apply the Particle Swarm Optimization Algorithm to the Capacitated Vehicle Routing Problem (CVRP), I will show and analyze the results of the study.

Chapter 1

Combinatorial Optimization

Introduction

In this chapter, we will explore the fundamental concepts and definitions pertaining to solving the Capacitated Vehicle Routing Problem (CVRP) using Particle Swarm Optimization (PSO).

1.1 Optimization and optimization problems

1.1.1 Definition of optimization

Optimization is the process of finding the best possible solution or set of solutions to a given problem. It involves systematically exploring and evaluating different choices or configurations to optimize a specific objective or criteria. The objective can be either maximizing a desired outcome or minimizing an undesired outcome, often subject to constraints. Optimization encompasses a wide range of mathematical, computational, and algorithmic techniques aimed at improving efficiency, effectiveness, or performance in various domains and disciplines.

1.1.2 The primary objectives of optimization

Maximization

Maximization in optimization refers to the process of finding the best possible solution that maximizes a specific objective or criterion. This objective is often associated with achieving the highest level of performance, output, profit, utility, or any other desirable measure. For example:

- Maximizing the profit of a company by optimizing production, pricing, and resource allocation.
- Maximizing the accuracy of a machine learning model by optimizing its parameters and training process.
- Maximizing the efficiency of a manufacturing process by optimizing the use of resources and minimizing waste.

Minimization

Minimization in optimization involves finding the best possible solution that minimizes a particular objective or criterion. This objective is typically associated with reducing costs, errors, risks, energy consumption, or any other undesirable measure. For example:

- Minimizing the total travel distance for a delivery fleet to reduce fuel consumption and transportation costs.
- Minimizing the error or loss function in regression or classification models to improve their predictive accuracy.
- Minimizing the time needed to complete a project or task to improve efficiency and meet deadlines.

1.2 Overview of Optimization Problems

Optimization problems encompass a wide range of mathematical or computational challenges that involve finding the best possible solution(s) to optimize an objective function. These problems arise in various fields, including engineering, economics, operations research, computer science, and many others, At the core of an optimization problem is the objective function which is defined as:

1.2.1 Objective function

An objective function is a mathematical function used in optimization problems to measure the quality or performance of a solution. It quantifies the goal to be maximized or minimized and guides the search for the best solution.

1.2.2 Types of solutions in optimization

In the field of optimization, various types of solutions are considered when solving problems. Understanding these solution types is crucial for evaluating and comparing different approaches. In this section, we provide brief definitions for key solution types, including the best-known solution, feasible solution, optimal solution, sub-optimal solution, infeasible solution, Pareto optimal solution, and local optimum. These definitions will help establish a foundation for analyzing and interpreting solutions in optimization problems.

- **Best-known solution**: The highest-quality solution currently known for a given problem. It serves as a reference point for evaluating the performance of other solutions.
- **Feasible solution**: A solution that satisfies all the specified constraints and is valid within the problem's constraints and limitations.
- **Optimal solution**: The solution that achieves the best possible objective value among all feasible solutions. It represents the highest level of optimality and meets all the problem requirements.
- **Suboptimal solution**: A solution that is not the best possible but still provides an acceptable level of quality or performance, falling short of the optimal solution.
- **Infeasible solution**: A solution that violates one or more constraints of the problem, rendering it invalid or impractical.
- **Pareto optimal solution**: In multi-objective optimization, a solution that cannot be improved in any objective without sacrificing the performance of at least one other objective. It represents a trade-off between conflicting objectives.
- **Local optimum**: A solution that is optimal within a specific region of the solution space but may not be the global optimum. It may be trapped in a local optimum due to the search algorithm's limitations.

1.2.3 types of optimization problems

Various types of optimization problems include:

Continuous optimization

- **Portfolio Optimization:** Determining the optimal allocation of investment funds across a range of assets to maximize return while minimizing risk.
- **Production Planning:** Optimizing the allocation of resources, such as labor and machinery, to maximize production output and minimize costs.

1.2.4 Discrete optimization

- **raveling Salesman Problem:** Finding the shortest possible route for a salesperson to visit a set of cities exactly once and return to the starting point.
- **Bin Packing Problem:** Efficiently packing a set of items into a minimal number of bins, considering their size and capacity constraints.

Mixed-integer optimization

- **Facility Location Problem:** Deciding the optimal locations for setting up new facilities, considering both the continuous variables of site coordinates and the discrete decision of which facilities to open.
- **Vehicle Routing Problem with Time Windows:** Optimizing the routes and schedules of a fleet of vehicles to serve a set of customers within specific time windows, involving both continuous vehicle movement and discrete customer visits.

1.3 combinatorial optimization

Within the realm of optimization, one specific class of problems that poses unique challenges is combinatorial optimization. Unlike continuous optimization problems, which deal with continuous decision variables, combinatorial optimization focuses on discrete decision variables and seeks to find the best arrangement or combination of these variables.

1.3.1 Definition of combinatorial optimization

Combinatorial optimization is a field of study within optimization that involves solving problems with discrete decision variables. It seeks to find the best arrangement or combination of these variables, subject to constraints, in order to optimize an objective function.

1.3.2 The main characteristics of combinatorial optimization

Certainly! Here's the rewritten version in LaTeX:

- **Discrete Decision Variables**: Combinatorial optimization problems involve discrete decision variables, where the variables can only take on specific values or choices from a finite set.
- **Combinatorial Nature**: The number of possible solutions in combinatorial optimization problems grows exponentially with the problem size, posing significant computational challenges.

- **Constraints**: Combinatorial optimization problems typically have constraints that limit the feasible solutions, defining the allowable combinations or arrangements of the decision variables.
- **Objective Function**: The goal of combinatorial optimization is to optimize an objective function that quantifies the measure of optimality, involving maximizing or minimizing specific criteria.
- **NP-Hardness**: Many combinatorial optimization problems are classified as NP-hard, indicating their computational complexity and lack of efficient algorithms for finding optimal solutions in polynomial time.
- **Search and Exploration**: Solving combinatorial optimization problems often requires systematic search and exploration of the solution space using various algorithms and heuristics.
- **Application Diversity**: Combinatorial optimization finds applications in diverse domains such as logistics, scheduling, network design, and resource allocation, addressing real-world challenges with discrete decision-making and resource allocation.

1.3.3 Examples of combinatorial Optimization

These examples demonstrate the wide-ranging applications of combinatorial optimization, showcasing their importance in solving complex decision problems across industries and sectors.

- **Traveling Salesman Problem (TSP)**: Finding the shortest possible route for a salesperson to visit a set of cities exactly once and return to the starting point. TSP is essential in logistics, transportation planning, and network optimization.
- **Job Scheduling**: Optimizing the assignment of tasks or jobs to resources, such as machines or workers, to minimize completion time or maximize resource utilization. Job scheduling is crucial in production planning, project management, and workforce optimization.
- **Knapsack Problem**: Selecting a subset of items with limited capacity, maximizing their total value while respecting weight or volume constraints. The knapsack problem has applications in resource allocation, portfolio optimization, and resource management.
- **Graph Coloring**: Assigning colors to vertices of a graph such that no adjacent vertices share the same color. Graph coloring is vital in scheduling, register allocation in compilers, frequency assignment in wireless communication, and timetabling.
- **Facility Location**: Determining optimal locations for establishing facilities to minimize costs or maximize service coverage. Facility location problems are relevant in supply chain management, network design, and facility layout planning.
- **Cutting Stock Problem**: Finding the most efficient way to cut raw materials into smaller pieces, minimizing waste and optimizing resource utilization. Cutting stock problems are significant in manufacturing, paper cutting, and material optimization.
- **Vehicle Routing Problem (VRP)**: Optimizing the routes and schedules of a fleet of vehicles to serve a set of customers, considering capacity constraints and minimizing total distance or time. VRP has applications in logistics, delivery services, and transportation planning.

1.4 Conclusion

For a better understanding of the following three chapters, we presented this chapter that gives an introduction to the vast optimization field, especially the combinatorial optimization, in which we chose our subject of study, which is Particle Swarm Optimization using Capacitated Vehicle Routing Problem.

Chapter 2

The Vehicle Routing Problem

Introduction

The Vehicle Routing Problem (VRP) is a well-known optimization problem in the field of logistics and transportation management. It involves determining optimal routes for a fleet of vehicles to serve a set of customers, taking into consideration various constraints and objectives[20].

The VRP is closely related to the Traveling Salesman Problem (TSP), another well-studied combinatorial optimization problem. In the TSP, the objective is to find the shortest possible route that visits a given set of cities exactly once and returns to the starting city. The VRP extends the TSP by adding capacity constraints and multiple vehicles to the problem[6].

In this chapter, we will first provide a brief overview of the Traveling Salesman Problem (TSP) and its significance in the field of optimization. We will then delve into the mathematical formulation of the VRP, discussing its key principles and characteristics.

2.1 The Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) is one of the most well-known combinatorial optimization problems. Given a set of cities and the distances between them, the objective is to find the shortest possible route that visits each city exactly once and returns to the starting city.

The TSP has been extensively studied due to its theoretical implications and its practical applications in various fields such as logistics, transportation, and manufacturing. It serves as a fundamental benchmark problem for testing optimization algorithms and techniques[20].

In the TSP, the objective function is to minimize the total distance or cost of the tour. The problem assumes that the distance between any two cities is symmetric, meaning that the distance from city A to city B is the same as the distance from city B to city A. The objective is to find the optimal permutation of cities that minimizes the total distance traveled. Mathematically, let's define the TSP as follows:

$$\text{Minimize: } z = \sum_{i=1}^{n} \sum_{j=1}^{n} d(i,j) \cdot x(i,j)$$

subject to:

$$\sum_{j=1}^{n} x(i,j) = 1, \text{ for all } i \in \{1,2,\ldots,n\}$$

$$\sum_{i=1}^{n} x(i,j) = 1, \text{ for all } j \in \{1,2,\ldots,n\}$$

$$\sum_{i \in S} \sum_{j \in S} x(i,j) \leq |S| - 1, \text{ for all non-empty proper subsets } S \subseteq \{1,2,\ldots,n\}$$

The objective function seeks to minimize the total distance traveled, while the constraints ensure that each city is visited and left exactly once and prevent the existence of sub-tours, which would violate the requirement of visiting each city only once.

Solving the TSP optimally for large instances is computationally challenging, as the problem is known to be NP-hard. Various algorithms and heuristics, such as branch and bound, dynamic programming, and evolutionary algorithms, have been developed to approximate the optimal solution efficiently[6].

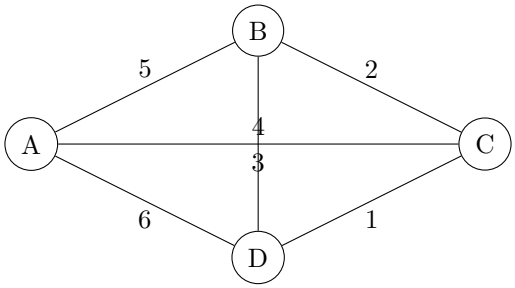


Figure 2.1: Example graph representing cities in the TSP

2.2 Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is a combinatorial optimization problem that involves determining the optimal routes and schedules for a fleet of vehicles to serve a set of customers or locations, while minimizing the total cost or distance traveled. In VRP, the objective is to find the most efficient way to allocate vehicles, considering factors such as vehicle capacity constraints, time windows, customer demands, and operational constraints[18].

The VRP can be formulated as a mathematical optimization problem, where the goal is to find the optimal allocation of customers to vehicles, the sequence of visits for each vehicle, and the overall routing plan that minimizes the total distance traveled, time spent, or other cost-related objectives.

The VRP is characterized by the following key elements:

1. Customers or Locations: There is a set of customers or locations that need to be serviced by the vehicles. Each customer/location has specific demands, such as the quantity of goods to be delivered or the service required.

2. Vehicles: There is a fleet of vehicles available for serving the customers/locations. Each vehicle has a certain capacity, representing the maximum amount it can carry or the number of customers it can serve in a single trip.

3. Constraints: Various constraints can be involved, such as vehicle capacity constraints, time windows specifying when customers/locations can be served, maximum route duration or working hours for vehicles, and any operational limitations or regulations.

4. Objective Function: The objective is to minimize a specific criterion, such as the total distance traveled, the total time spent, the number of vehicles used, or a combination of these factors. The aim is to find the optimal solution that optimizes the objective function while satisfying all the constraints.

The VRP has a wide range of applications in transportation, logistics, and distribution, including delivery services, fleet management, waste collection, public transportation, and more. Solving the VRP optimally for large-scale instances is computationally challenging, and various heuristic and metaheuristic algorithms have been developed to approximate near-optimal solutions efficiently.

By addressing the Vehicle Routing Problem, organizations can improve their operational efficiency, reduce transportation costs, enhance customer service, and effectively utilize their resources and vehicles.

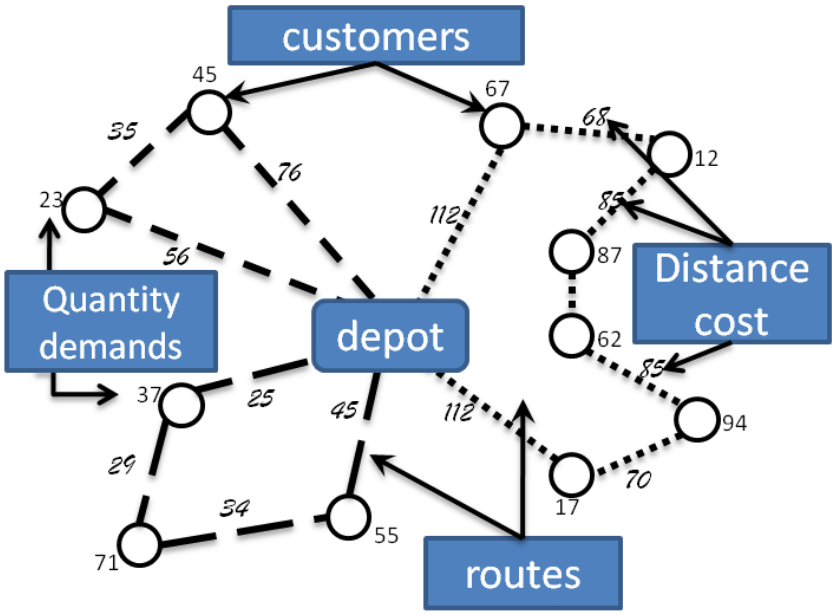


Figure 2.2: An example of a Vehicle Routing Problem

2.2.1 Mathematical Modeling of Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) can be formulated as a mathematical optimization problem. Let's define the VRP using the following notations:

- N: Set of customer locations, indexed by $i = 1, 2, \ldots, n$.
- V: Set of vehicles available for serving the customers, indexed by $j = 1, 2, \ldots, m$.
- Q_j: Maximum capacity of vehicle j.
- d_{ij}: Distance or cost of traveling from location i to location j.
- q_i: Demand of customer location i.
- x_{ij}: Binary decision variable indicating if vehicle j travels from location i to location j. Takes the value of 1 if the vehicle travels directly from i to j, and 0 otherwise.
- u_i: Cumulative demand up to customer location i.

The VRP can be mathematically formulated as follows:

$$\text{Minimize:} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} \cdot x_{ij}$$

$$\text{subject to:} \quad \sum_{j=1}^{m} x_{ij} = 1, \quad \forall i \in N \quad \text{(Each customer is visited exactly once)}$$

$$\sum_{i=1}^{n} q_i \cdot x_{ij} \leq Q_j, \quad \forall j \in V \quad \text{(Vehicle capacity constraint)}$$

$$u_i + q_i \leq u_j + M \cdot (1 - x_{ij}), \quad \forall i \in N, j \in N \setminus \{i\} \quad \text{(Subtour elimination constraint)}$$

$$u_i \geq q_i, \quad \forall i \in N \quad \text{(Demand fulfillment constraint)}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in N, j \in V \quad \text{(Binary decision variable)}$$

In the objective function, we seek to minimize the total distance or cost of the routes taken by the vehicles. The first constraint ensures that each customer is visited exactly once. The second constraint guarantees

- Minimize: $\sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} \cdot x_{ij}$

  This objective function seeks to minimize the total distance or cost of the routes taken by the vehicles. It is the sum of the distances (or costs) multiplied by the binary decision variables indicating whether a vehicle travels from location i to location j.

- Subject to: $\sum_{j=1}^{m} x_{ij} = 1, \forall i \in N$

  This constraint ensures that each customer is visited exactly once. It states that the sum of the binary decision variables for each customer i should be equal to 1, indicating that exactly one vehicle visits that customer.

- Subject to: $\sum_{i=1}^{n} q_i \cdot x_{ij} \leq Q_j, \forall j \in V$

  This constraint represents the vehicle capacity constraint. It states that the sum of the demands of the assigned customers (multiplied by the binary decision variables) for each vehicle j should be less than or equal to the maximum capacity Q_j of that vehicle.

- Subject to: $u_i + q_i \leq u_j + M \cdot (1 - x_{ij}), \forall i \in N, j \in N \setminus \{i\}$

  This constraint is the subtour elimination constraint. It ensures that there are no subtours, which would violate the requirement of visiting each customer only once. It states that the cumulative demand at customer i, plus the demand of customer i, should be less than or equal to the cumulative demand at customer j, plus a large constant M multiplied by $(1 - x_{ij})$. This ensures that if x_{ij} is 0, indicating that there is no direct connection between customers i and j, the constraint is not binding[12].

- Subject to: $u_i \geq q_i, \forall i \in N$

  This constraint represents the demand fulfillment constraint. It ensures that the cumulative demand at each customer i is greater than or equal to its demand q_i. This ensures that the assigned vehicles can meet the demands of the customers.

- Subject to: $x_{ij} \in \{0, 1\}, \forall i \in N, j \in V$

  This constraint defines the binary decision variables x_{ij} as either 0 or 1, indicating whether vehicle j travels from location i to location j or not.

2.3 Different Variants of VRP

There are several variants of the Vehicle Routing Problem (VRP), each addressing specific characteristics and requirements. In this section, we will discuss 10 different variants of VRP, along with brief definitions of each variant.

2.3.1 VRP with Time Windows (VRPTW)

VRPTW extends the VRP by incorporating time windows, which specify the time intervals within which customers must be serviced. The objective is to find routes that satisfy the time constraints while minimizing the total distance or time traveled.

Example

- Customers: The company has a list of customers, each with a specific location, a time window during which they can receive the delivery, and a package to be delivered.
- Vehicles: The company has a fleet of vehicles with limited capacity that can be used for making the deliveries. Each vehicle has its own starting location.
- Time Windows: Each customer has a time window during which they can receive the delivery. The vehicles must arrive at the customer's location within this time window.
- Capacity Constraints: Each vehicle has a maximum capacity limit. The total volume or weight of packages assigned to a vehicle should not exceed this limit.

2.3.2 VRP with Pickup and Delivery (VRPPD)

VRPPD involves pickup and delivery tasks, where goods are collected from pickup locations and delivered to corresponding delivery locations. The objective is to optimize the routes and schedules for pickup and delivery while considering vehicle capacity and time constraints[4].

Example

- Customers: The company provides delivery and pickup services for customers. Each customer has specific pickup and delivery locations.
- Packages: Customers request package deliveries and returns. Each package has its own characteristics such as size, weight, and priority.
- Pickup and Delivery Points: The company has designated locations where packages are picked up from or delivered to customers.

2.3.3 VRP with Split Deliveries (VRPSD)

VRPSD allows the splitting of customer demands across multiple vehicles. The objective is to minimize the number of vehicles used and the total distance traveled while ensuring that the split deliveries satisfy the vehicle capacity constraints.[4]

Example

- Customers: The company provides delivery services to multiple customers. Each customer has a specific delivery location.
- Orders: Customers place orders for products or goods that need to be delivered.
- Split Deliveries: Some orders may require splitting into multiple deliveries due to factors such as different delivery time preferences or product availability.
- Delivery Points: The company has designated delivery points where goods are dropped off to customers.

2.3.4 VRP with Backhauls (VRPB)

VRPB considers the transportation of goods from customers back to a central depot or other designated locations, known as backhauls. The objective is to optimize the routes for both deliveries and backhauls while considering vehicle capacity and cost constraints[17].

Example

- Customers: The company provides delivery services to multiple customers who require goods to be delivered to their locations.
- Pickup Points: In addition to deliveries, the company also needs to pick up goods from certain locations, such as suppliers or warehouses.
- Vehicles: The company has a fleet of vehicles with limited capacity that can be used for both deliveries and pickups.

2.3.5 VRP with Multiple Depots (VRPMD)

VRPMD involves multiple depots from which vehicles start and return. The objective is to assign customers to different depots and determine the routes for each vehicle while minimizing the total distance or cost[17].

Example

- Depots: The company operates multiple depots or distribution centers from which goods are dispatched for delivery.
- Customers: The company serves a diverse set of customers located in different regions or areas.
- Vehicles: The company has a fleet of vehicles with limited capacity to transport goods from the depots to the customers.
- Vehicle Depots: Each vehicle is assigned to a specific depot and starts and ends its routes at the same depot.

2.3.6 VRP with Heterogeneous Fleet (VRPHF)

VRPHF considers a fleet of vehicles with different capacities, speeds, or cost rates. The objective is to assign customers to appropriate vehicles and optimize the routes considering the heterogeneous characteristics of the fleet[7].

2.3.7 VRP with Time-Varying Demand (VRPTD)

VRPTD deals with customer demands that change over time. The objective is to dynamically assign vehicles and optimize the routes to handle the time-varying demand while minimizing the total distance or cost.[8]

2.3.8 VRP with Stochastic Demand (VRPSD)

VRPSD takes into account customer demands that are uncertain or stochastic. The objective is to determine robust routes that can handle different demand scenarios with minimal cost or distance[8].

2.3.9 Green VRP (GVRP)

GVRP focuses on reducing the environmental impact of the VRP by considering factors such as fuel consumption, emissions, and vehicle routing efficiency. The objective is to optimize the routes while minimizing carbon footprint and promoting sustainability[1].

2.3.10 Capacitated VRP (CVRP)

Capacitated VRP (CVRP) is a variant of the Vehicle Routing Problem that incorporates vehicle capacity constraints in addition to customer demands. In CVRP, the objective is to determine optimal routes for a fleet of vehicles, considering both the capacity limitations of each vehicle and the total distance or cost traveled[18].

The objective function of CVRP is to minimize the total distance or cost of the routes taken by the vehicles. It can be mathematically represented as:

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij} \cdot x_{ij}$$

where:

- d_{ij} represents the distance or cost of traveling from location i to location j,
- x_{ij} is a binary decision variable indicating if vehicle j travels from location i to location j. It takes the value of 1 if the vehicle travels directly from i to j, and 0 otherwise.

The capacity constraints in CVRP ensure that the total demand of the assigned customers does not exceed the maximum capacity of each vehicle. These constraints can be formulated as:

$$\sum_{i=1}^{n} q_i \cdot x_{ij} \leq Q_j, \quad \forall j \in V$$

where:

- q_i represents the demand of customer location i,
- Q_j is the maximum capacity of vehicle j.

By optimizing the routes and satisfying the capacity constraints, CVRP aims to efficiently allocate vehicles and serve customers while minimizing transportation costs. Solving CVRP efficiently contributes to improved operational efficiency, reduced transportation expenses, and better resource utilization.

2.4 Conclusion

In this chapter, we have introduced the Vehicle Routing Problem (VRP) and presented its mathematical formulation. Our focus of interest lies specifically in the Capacitated Vehicle Routing Problem (CVRP), a variant of VRP that considers vehicle capacity constraints in addition to customer demands.

In the next chapter, we will delve into various methods that can be employed to find solutions for the CVRP. Specifically, we will explore three different approaches: Ant Colony Optimization, Genetic Algorithms, and Particle Swarm Optimization. These methods offer innovative and efficient ways to tackle the CVRP and provide near-optimal solutions.

By studying and implementing these algorithms, we aim to address the challenges posed by the CVRP and contribute to the field of optimization and logistics. These techniques have shown promising results in solving complex routing problems and optimizing the allocation of resources.

Chapter 3

Metaheuristics For Solving VRP

Introduction

In this chapter, we shall introduce a set of methods called metaheuristics, which are a series of optimization algorithms designed to solve difficult optimization problems for which no classical methods are known to be more effective. They are often used as general methods to optimize a wide range of different problems without fundamentally modifying the employed algorithms. most importantly we will mention three of the most successful metaheuristics which are **Ant Colony Optimization (ACO)**, **Genetic Algorithms (GA)**, and our subject of study, **Particle Swarm Optimization (PSO)**.

3.1 Overview of metaheuristics

3.1.1 definition of meraheuristics

Metaheuristics are high-level problem-solving methodologies used in optimization that provide flexible and adaptive approaches for finding good or near-optimal solutions to complex problems. These algorithms employ iterative and stochastic techniques, often inspired by natural processes, to explore large solution spaces and handle constraints. While not guaranteeing optimal solutions, metaheuristics aim to efficiently search for satisfactory solutions within a reasonable amount of time.[16] Examples of popular metaheuristic algorithms include Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Tabu Search (TS), Ant Colony Optimization (ACO), and Differential Evolution (DE). These algorithms have been successfully applied to various operational research problems, including vehicle routing, scheduling, resource allocation, and facility location, among others. Metaheuristics offers a powerful and versatile approach to solving complex optimization problems in operational research, providing a balance between solution quality and computational efficiency. They are particularly useful when traditional exact methods are not feasible due to the problem's size, complexity, or time constraints.

3.1.2 When to Apply Metaheuristics

Complex Optimization Problems

When the problem is complex and lacks a well-defined mathematical formulation, metaheuristics provide a flexible and adaptable approach to finding near-optimal solutions.

Combinatorial Optimization Problems

Metaheuristics are effective for solving combinatorial optimization problems where the search space is large and discrete, such as the Traveling Salesman Problem or the Vehicle Routing Problem.

Lack of Exact Methods

When traditional exact methods, such as mathematical programming or exhaustive search algorithms, are not feasible due to the problem's size, complexity, or computational requirements, metaheuristics offer a practical alternative.[17]

Noisy or Uncertain Environments

Metaheuristics can handle problems with uncertain or stochastic components, where the objective is to find robust solutions that perform well across different scenarios or with noisy input data.[19]

Multi-objective Optimization

Metaheuristics excel in solving multi-objective optimization problems, where multiple conflicting objectives need to be optimized simultaneously, and a trade-off between them must be found.[3]

Lack of Domain-specific Knowledge

When specific domain knowledge or problem-specific algorithms are not available or difficult to apply, metaheuristics provide a general-purpose approach that can be applied to a wide range of problems without requiring specialized expertise.

3.1.3 Classifications of Metaheuristics

These classifications provide an overview of different dimensions along which metaheuristics can be categorized. It is important to note that many metaheuristics exhibit characteristics that span multiple classifications, as researchers continue to develop and adapt these algorithms for various problem domains.[9]

Population-Based vs. Single-Solution

Metaheuristics can be classified based on whether they operate on a population of solutions or a single solution at a time. Population-based metaheuristics, such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), maintain a group of candidate solutions and perform operations like reproduction, crossover, and mutation to explore the solution space. Single-solution metaheuristics, such as Simulated Annealing (SA) and Tabu Search (TS), iteratively improve a single solution by exploring neighboring solutions.

Stochastic vs. Deterministic

Metaheuristics can be categorized based on the level of randomness or determinism in their search process. Stochastic metaheuristics, such as Genetic Algorithms and Ant Colony Optimization (ACO), employ randomization in their solution generation, search operators, or decision-making processes. Deterministic metaheuristics, such as Simulated Annealing and Hill Climbing, follow a deterministic search path but may use randomization for diversification or exploration.

Trajectory-Based vs. Population-Based

Metaheuristics can be divided into trajectory-based and population-based approaches. Trajectory-based metaheuristics, such as Hill Climbing and Simulated Annealing, follow a single trajectory in the solution space, exploring and improving upon a single solution[11]. Population-based metaheuristics, such as Genetic Algorithms and Particle Swarm Optimization, maintain a population of solutions that interact and evolve over time.

Memory-Based vs. Memoryless

Metaheuristics can be classified based on their usage of memory. Memory-based metaheuristics, such as Tabu Search and Scatter Search, utilize memory structures to store information about past solutions or search patterns. This memory aids in avoiding revisiting the same solutions or guiding the search towards promising areas. Memoryless metaheuristics, such as Genetic Algorithms and Simulated Annealing, do not explicitly use memory and make decisions based solely on the current solution or search state.[5]

Exploration vs. Exploitation

Metaheuristics can be characterized by their balance between exploration and exploitation. Exploration refers to the process of searching diverse regions of the solution space to discover new and potentially better solutions. Exploitation involves intensifying the search around promising solutions to improve their quality. Some metaheuristics, like Genetic Algorithms, emphasize exploration by maintaining population diversity, while others, like Hill Climbing, focus more on exploitation by intensifying the search around the current solution.

3.2 principals of metaheuristics

3.2.1 Ant colony optimization (ACO)

Definition

Ant Colony Optimization (ACO) is a metaheuristic algorithm inspired by the foraging behavior of ants. It aims to solve optimization problems by simulating the collective intelligence of ant colonies[10]. ACO algorithms utilize artificial ants that construct solutions by depositing and following pheromone trails. The algorithm iteratively updates the pheromone levels based on the quality of the solutions found, facilitating the discovery of good or near-optimal solutions for complex problems.

Main Variants

There are several variants and extensions of the ACO algorithm that have been proposed to enhance its performance and address specific problem characteristics. Some notable variants include:

- Ant System (AS)
- Max-Min Ant System (MMAS)
- Rank-based Ant System (RAS)
- Ant Colony System (ACS)
- Elitist Ant System (EAS)

These variants introduce various modifications such as pheromone bounds, ranking mechanisms, local search, and elitism to improve solution quality and exploration capabilities.

3.2.2 Ant colony algorithm

Algorithm 1 Ant Colony Optimization (ACO)

Require: Problem instance
Ensure: Best solution
1: Initialize pheromones and solutions
2: Initialize parameters (number of iterations, ant colony size, etc.)
3: **while** termination condition is not met **do**
4: **for** each ant **do**
5: Construct the solution by following probabilistic selection rules
6: Update the pheromone on edges based on the quality of the solution constructed by the ant
7: **end for**
8: Perform global update to track the best solution found
9: Evaporate pheromone on all edges
10: Perform local search to intensify the search in promising regions
11: Perform diversification mechanism to explore new areas of the search space
12: Update the exploration-exploitation balance based on problem-specific factors
13: **end while**
14: **return** Best solution

the ACO algorithm for the CVRP utilizes artificial ants that construct solutions by iteratively selecting customers to visit, taking into account pheromone levels and heuristic information. The pheromone levels are updated based on the quality of the solutions, and the algorithm balances exploration and exploitation to guide the ants in finding optimal or near-optimal solutions. The process is repeated for a certain number of iterations, and the best solution found is returned as the output.[13]

1. Initialization: Initialize a population of artificial ants and set initial pheromone levels on the edges of the graph representing the problem.

2. Ant Movement: Each ant constructs a solution by iteratively selecting a customer to visit based on pheromone levels and heuristic information, considering the capacity constraints of the vehicles.

3. Pheromone Update: After all ants have constructed their solutions, update the pheromone levels on the edges based on the quality of the solutions.

4. Local Search: Optionally, apply local search techniques to improve the quality of the solutions by making small modifications to the routes.

5. Best Solution Update: Track the best solution found so far and update it if a new better solution is discovered.

6. Exploration-Exploitation Balance: Adjust the exploration and exploitation trade-off by updating the pheromone levels to guide the ants in finding good-quality solutions while exploring new routes.

7. Iteration: Repeat steps 2-6 for a specified number of iterations or until a termination condition is met.

8. Return Best Solution: Return the best solution found during the iterations as the output, representing the optimized vehicle routes for the CVRP.

3.2.3 Advantages of Ant Colony Optimization

- Flexibility: ACO can be applied to various optimization problems, making it versatile.
- Global Search: ACO explores the entire search space, avoiding local optima.
- Adaptability: ACO adjusts well to dynamic problem environments.
- Robustness: ACO handles uncertainty and robustly makes decisions.
- Exploration-Exploitation Balance: ACO effectively balances exploration and exploitation for better solutions.
- Parallelization: ACO can be parallelized, enabling faster convergence and efficient resource usage.

3.3 Genetic Algorithms

3.3.1 definition

Genetic Algorithm (GA) is a metaheuristic algorithm inspired by the process of natural selection and evolution. It is used to solve optimization problems by mimicking the principles of genetics and evolution. In a GA, a population of candidate solutions, represented as individuals, undergoes a process of selection, crossover, and mutation. Through successive generations, individuals with better fitness, determined by an objective function, are more likely to be selected and contribute to the next generation. This iterative process allows the algorithm to explore and exploit the search space, gradually converging toward an optimal or near-optimal solution. The GA's ability to combine exploration and exploitation makes it effective for solving complex optimization problems with large solution spaces and non-linear fitness landscapes.[14]

3.3.2 Concepts of Genetic Algorithm for CVRP

the main variables of GA

- Population: It represents a group of individuals or candidate solutions. The population size determines the number of solutions considered in each generation.
- Individual: It refers to a single candidate solution within the population. Each individual is encoded using a set of chromosomes.
- Chromosomes: They are structures that contain the genetic information of an individual. Chromosomes are often represented as strings of genes or binary values.
- Genes: They are the basic units of information within a chromosome. Genes represent specific variables or characteristics of a solution. In binary-encoded GAs, genes are typically represented as bits.
- Bits: In binary-encoded GAs, bits are the smallest units of information that make up a gene. Each bit can have a value of 0 or 1, representing different alleles or choices for that particular gene.

The Algorithm of GA

Algorithm 2 Genetic Algorithm (GA)

Require: Problem instance
Ensure: Best solution
1: Initialize population with random solutions
2: Evaluate the fitness of each solution
3: **repeat**
4: Select parent solutions based on their fitness
5: Perform crossover to create offspring solutions
6: Apply mutation to introduce small random changes to the offspring solutions
7: Evaluate the fitness of the new offspring solutions
8: Select the best solutions from the current population and the offspring solutions to form the next generation population
9: **until** termination condition is met
10: **return** Best solution found in the final population

the concept of Genetic algorithms for CVRP

1. Initialization: Generate an initial population of candidate solutions, where each solution represents a set of routes for the vehicles.

2. Evaluation: Evaluate the fitness of each solution in the population based on objective criteria, such as total distance traveled or total cost.

3. Selection: Select a subset of solutions from the population to serve as parents for the next generation. The selection process is typically based on fitness, favoring solutions with higher fitness values.

4. Crossover: Create new offspring solutions by combining genetic information from the selected parents. Crossover operators like one-point crossover or uniform crossover are used to exchange genetic material between parents.

5. Mutation: Introduce random changes in the genetic information of offspring solutions to promote exploration of the search space. Mutation operators modify a small portion of the solution, such as swapping or randomly altering genes.

6. Replacement: Replace some of the existing solutions in the population with the offspring solutions. The replacement process is often based on a combination of elitism (keeping the best solutions) and diversity preservation.

7. Termination: Check if a termination condition is met, such as reaching a maximum number of generations or finding a satisfactory solution. If not, go back to step 2.

Advantages of Genetic algorithms

- Effective exploration: The GA can explore a large search space efficiently, allowing it to discover diverse and potentially better solutions to the CVRP.

- Adaptability: The GA can adapt and evolve its solutions over generations, allowing it to handle dynamic or changing CVRP instances.
- Robustness: The GA is robust against local optima, as it maintains population diversity and can escape suboptimal solutions.
- Flexibility: The GA can handle various problem formulations and constraints of the CVRP, making it suitable for different real-world scenarios.
- Scalability: The GA can handle large-scale CVRP instances by parallelizing evaluations and utilizing evolutionary operators efficiently[19].
- Solution quality: The GA tends to find good-quality solutions, if not optimal, for the CVRP within reasonable computational time.

3.4 Particle Swarm Optimization

3.4.1 Introduction

PSO (Particle Swarm Optimization) is a metaheuristic optimization algorithm inspired by the collective behavior of organisms in nature, such as bird flocking or fish schooling. It was first proposed by Kennedy and Eberhart in 1995. PSO is widely used to solve various optimization problems by simulating the behavior of a swarm of particles in a search space.

3.4.2 definition

PSO is a population-based optimization algorithm that uses a swarm of particles to explore and exploit the search space. Each particle represents a potential solution to the optimization problem, and the swarm collectively moves through the search space to find the optimal solution. The movement of particles is guided by their own experience and the knowledge shared within the swarm.

3.4.3 The main characters of PSO

For a better understanding of Particle swarm Optimization, these are the main variables.

- Swarm: PSO operates with a population of particles forming a swarm.
- Particle: Each particle represents a potential solution in the search space.
- Position: Each particle has a position vector that represents its current solution.
- Velocity: Each particle has a velocity vector that determines its movement in the search space.
- Fitness: The fitness value evaluates the quality of a particle's solution.
- Personal Best: Each particle maintains its best position encountered so far.
- Global Best: The swarm tracks the best position found among all particles.
- Social Interaction: Particles communicate by adjusting their velocities based on personal and global information.

- Exploration: PSO promotes exploration by allowing particles to explore different areas of the search space.
- Exploitation: PSO encourages exploitation by leveraging the best solutions found so far.
- Iterations: The algorithm progresses through a series of iterations, updating positions and velocities.
- Termination Criterion: PSO continues until a termination condition is met, such as reaching a maximum number of iterations or finding a satisfactory solution.

3.4.4 PSO Algorithm

Require: Problem instance
Ensure : Best solution
Initialize particles with random positions and velocities Initialize parameters (number of iterations, swarm size, inertia weight, etc.) Initialize the global best position and fitness **while** *termination condition is not met* **do**
  for *each particle* **do**
    Update the velocity using the inertia weight and acceleration terms Update the position based on the new velocity Evaluate the fitness of the current position **if** *the fitness is better than the personal best fitness* **then**
      Update the personal best position and fitness
    end
    if *the fitness is better than the global best fitness* **then**
      Update the global best position and fitness
    end
  end
end
return *Best solution*

The algorithm simulates the behavior of a swarm of particles searching for the optimal solution in a given problem space. By adjusting their velocities and positions based on personal and global information, the particles explore the search space and converge towards promising regions, ultimately finding a near-optimal solution.[14]

- Initialize particles with random positions and velocities.

  Each particle represents a potential solution to the optimization problem. The particles are randomly initialized within the search space.

- Initialize parameters such as the number of iterations, swarm size, inertia weight, etc.

  These parameters control the behavior and convergence of the algorithm.

- Initialize the global best position and fitness.

  The global best position and fitness are initially set to the values of one of the particles.

- While the termination condition is not met, do the following steps iteratively:
  - This loop continues until a certain stopping criterion is satisfied (e.g., a maximum number of iterations).

- For each particle, do the following steps:
  - Update the velocity of the particle based on the inertia weight and acceleration terms.
    The velocity determines the direction and magnitude of the particle's movement.
  - Update the position of the particle based on the new velocity.
    The position represents a potential solution in the search space.
  - Evaluate the fitness of the current position.
    The fitness function quantifies the quality of the solution.
  - If the fitness of the current position is better than the personal best fitness of the particle, update the personal best position and fitness accordingly.
    The personal best position and fitness represent the best solution found by the particle.
  - If the fitness of the current position is better than the global best fitness, update the global best position and fitness accordingly.
    The global best position and fitness represent the best solution found by any particle in the swarm.

- Return the best solution found (global best position and fitness).

3.4.5 Mathematical Presentation of Particle Swarm Optimization (PSO)

- Particle Position: Each particle i in the swarm has a position represented by a vector $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{in})$, where n is the dimensionality of the problem.

- Particle Velocity: Each particle i also has a velocity represented by a vector $\mathbf{v}_i = (v_{i1}, v_{i2}, ..., v_{in})$, which determines the direction and speed of movement in the search space.

- Particle Fitness: The fitness of each particle is evaluated based on its position in the search space. It represents the quality of the solution associated with that position.

- Personal Best: Each particle maintains its personal best position, denoted as $\mathbf{p}_i = (p_{i1}, p_{i2}, ..., p_{in})$, which corresponds to the best position it has encountered so far with the highest fitness.

- Global Best: The swarm keeps track of the global best position, denoted as $\mathbf{g} = (g_1, g_2, ..., g_n)$, which represents the position with the highest fitness among all particles in the swarm.

- Velocity Update Equation: The velocity of each particle is updated using the following equation:

$$\mathbf{v}_{ij}(t+1) = w \cdot \mathbf{v}_{ij}(t) + c_1 \cdot \text{rand}_1 \cdot (\mathbf{p}_{ij} - \mathbf{x}_{ij}(t)) + c_2 \cdot \text{rand}_2 \cdot (\mathbf{g}_j - \mathbf{x}_{ij}(t))$$

  where $\mathbf{v}_{ij}(t+1)$ is the velocity of particle i in dimension j at time $t+1$, w is the inertia weight, c_1 and c_2 are the acceleration coefficients, rand_1 and rand_2 are random numbers between 0 and 1, $\mathbf{p}_{ij}$ is the personal best position of particle i in dimension j, and $\mathbf{g}_j$ is the global best position in dimension j.

- Position Update Equation: The position of each particle is updated using the following equation:

$$\mathbf{x}_{ij}(t+1) = \mathbf{x}_{ij}(t) + \mathbf{v}_{ij}(t+1)$$

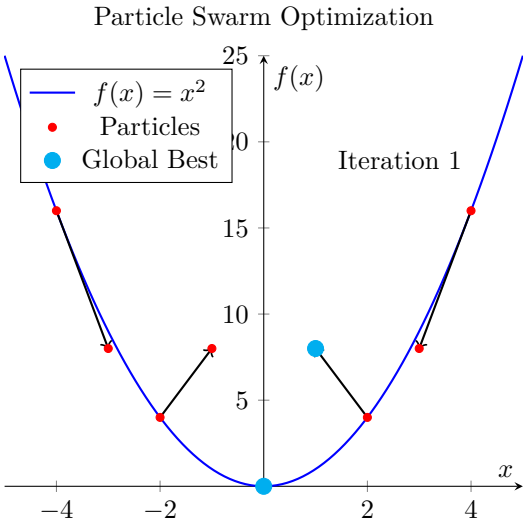
  where $\mathbf{x}_{ij}(t+1)$ is the updated position of particle i in dimension j at time $t+1$.

- Termination Criterion: PSO continues to update the velocities and positions of particles until a termination criterion is met, such as reaching a maximum number of iterations or finding a satisfactory solution.

  These mathematical equations govern the movement and interaction of particles in the PSO algorithm, allowing them to explore the search space, update their positions and velocities, and converge towards optimal solutions[2].

3.4.6 Example

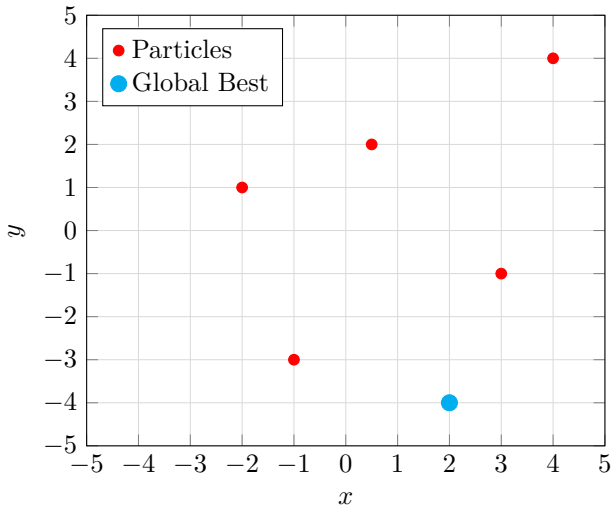
we are using Particle Swarm Optimization (PSO) to find the minimum of a quadratic function $f(x) = x^2$. PSO is a population-based metaheuristic algorithm that imitates the social behavior of bird flocking or fish schooling to solve optimization problems.



The graph illustrates the optimization process of PSO over multiple iterations. Here's a step-by-step explanation of what is happening in the graph:

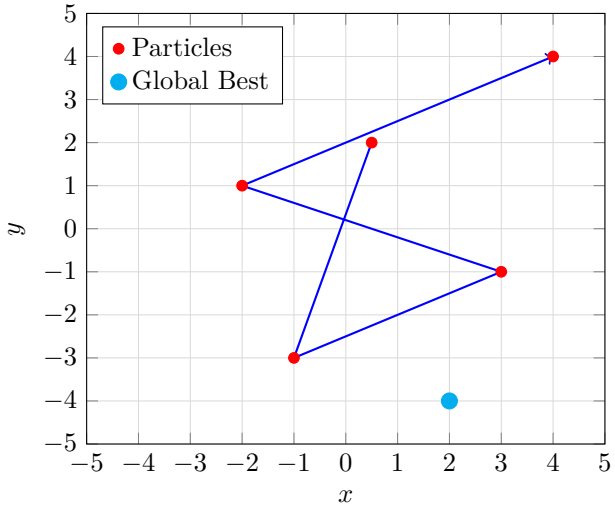
Explanation of the Example

Step 1: Initialization



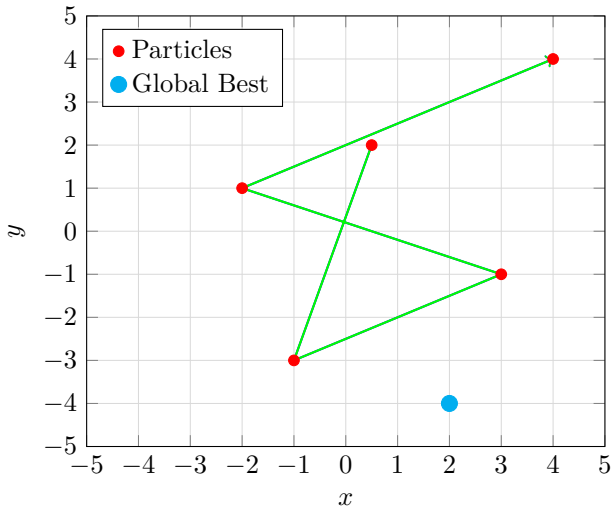
The red dots represent the initial positions of particles in the search space. The cyan dot represents the global best position found so far.

Step 2: Movement towards local best



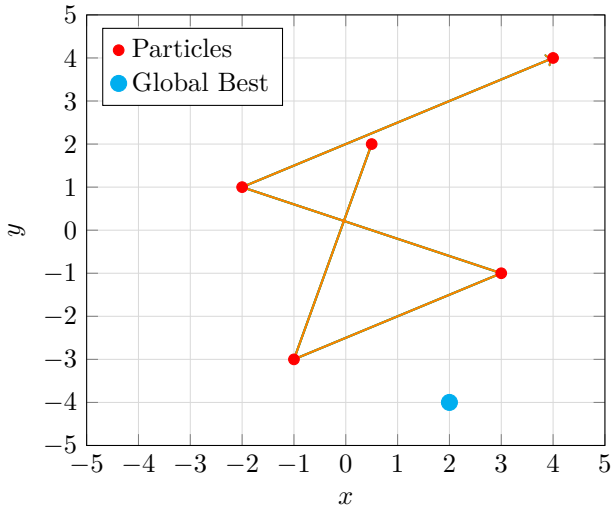
In each iteration, particles move towards their personal best positions (indicated by the blue arrows). The personal best is the best position a particle has found for itself. The length and direction of each arrow represent the movement of particles towards their personal best.

Step 3: Movement towards global best



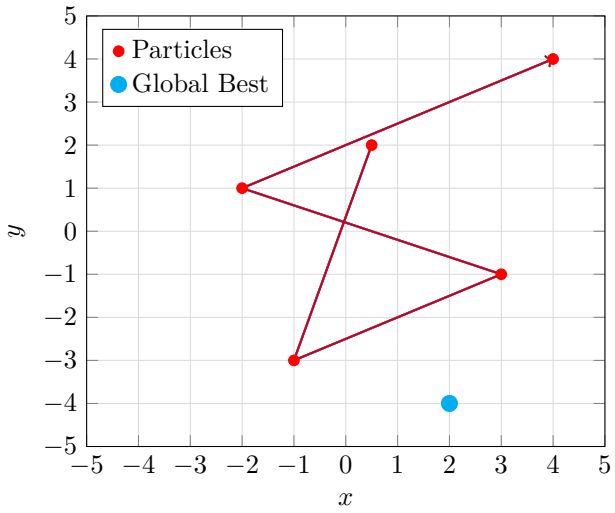
Additionally, particles are attracted towards the global best position found by any particle in the swarm (indicated by the green arrows). The global best is the best position found by any particle in the entire swarm. The length and direction of each green arrow represent the movement of particles towards the global best.

Step 4: Iterative improvement



As the iterations progress, particles continue to adjust their positions based on personal best, global best, and their current velocities. In this example, particles move towards improved positions by incorporating velocity adjustments (indicated by the orange arrows). The length and direction of each orange arrow represent the movement of particles based on their current velocities.

Step 5: Convergence



As the iterations continue, the particles gradually converge to a common solution. In this example, which is at $x = 0$. the particles move closer to a region that represents the optimal solution to the problem. The convergence is indicated by the particles clustering around a specific area.

Example Conclusion

In summary, Particle Swarm Optimization (PSO) is an optimization algorithm that iteratively adjusts the positions of particles to find near-optimal solutions. The particles explore the search space by moving towards their personal best and the global best positions. Through iterative improvement, the particles gradually converge to a solution. The convergence can be observed in the graph where the particles cluster around the optimal solution region.

3.5 Adaptation of PSO for CVRP

3.6 Problem Formulation and PSO-based Approach

The mathematical approach for adapting the Particle Swarm Optimization (PSO) algorithm to solve the Capacitated Vehicle Routing Problem (CVRP) involves formulating the objective function, defining the encoding scheme, and specifying the velocity and position update equations. Let's explore the mathematical aspects of this adaptation:

3.6.1 Objective Function

The objective function aims to minimize the total distance or cost required to serve all customers while respecting the vehicle capacity constraints and other problem-specific requirements[15]. It can be represented mathematically as:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \sum_{k=1}^{m} x_{ijk}$$

subject to:

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{ijk} = 1 \quad \forall i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ijk} = 1 \quad \forall j = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} \leq Q \quad \forall k = 1, 2, \ldots, m$$

$$\sum_{i \in S}\sum_{j \in S, j \neq i} x_{ijk} \leq |S| - 1 \quad \forall S \subseteq \{1, 2, \ldots, n\}, |S| \geq 2$$

where:

- n is the total number of customers
- m is the number of vehicles
- d_{ij} represents the distance or cost between customer i and customer j
- x_{ijk} is a binary decision variable indicating whether vehicle k visits customer i before customer j
- Q is the capacity of each vehicle

3.6.2 Encoding Scheme

The encoding scheme represents a candidate solution as a particle, where each particle's position corresponds to a vehicle route. The position of a particle consists of a sequence of customers to be visited, including the depot as the starting and ending point.

3.6.3 Velocity and Position Update Equations

The velocity and position update equations govern the movement of particles during the optimization process. They are responsible for balancing exploration and exploitation to guide the search towards optimal solutions. The velocity update equation for particle p at iteration $t + 1$ is given by:

$$v_p(t+1) = \omega \cdot v_p(t) + c_1 \cdot r_1 \cdot (p_{\text{best}} - x_p(t)) + c_2 \cdot r_2 \cdot (g_{\text{best}} - x_p(t))$$

where:

- $v_p(t)$ is the velocity of particle p at iteration t
- ω is the inertia weight controlling the impact of the previous velocity
- c_1 and c_2 are the acceleration coefficients controlling the impact of the particle's best position (p_{best}) and the swarm's best position (g_{best}), respectively
- r_1 and r_2 are random numbers between 0 and 1
- $x_p(t)$ represents the position (vehicle route) of particle p at iteration t

The position update equation is then given by:

$$x_p(t+1) = x_p(t) + v_p(t+1)$$

After each position update, the particle's position is adjusted if necessary to ensure that it satisfies the problem-specific constraints, such as the vehicle capacity constraints.

By formulating the objective function, defining the encoding scheme, and specifying the velocity and position update equations, we establish the mathematical foundation for the adaptation of the PSO algorithm to solve the CVRP. These mathematical aspects guide the implementation and execution of the PSO-based approach to efficiently explore the solution space and find optimal or near-optimal solutions to the CVRP.

3.7 Conclusion

In this chapter, we presented some of the metaheuristic methods in a general manner, but we focused on our subject of study which is Particle Swarm Optimization, and included its mathematical adaptation to solve the Capacitated Vehicle Routing Problem.

Chapter 4

The Application of the PSO Algorithm For CVRP

Introduction

In this chapter, we are interested in solving the Capacitated Vehicle Routing Problem. The objective is to minimize the cost of transportation with respecting the capacity constraint of the vehicles in the problem, the approach to this objective is done with Particle Swarm Optimization.

4.1 Problem description

The problem we are discussing is a timeless one-way delivery, meaning the capacitated vehicle routing problem in this study does not consider the time requirements for delivering goods to customers when constructing the delivery routes.

Delivery from the Depot to multiple customers using multiple delivery vehicles, where the location of each customer and the order is already given and confirmed. Each delivery vehicle has a certain load capacity. The coming conditions are necessary to improve the objective function.

- The total demand of each client cannot exceed the capacity of the distribution.
- The need of each client must be satisfied, and the delivery must be done with just one vehicle.

The problem that we will deal with in this study is delivering cosmetic products to 10 customers with 5 vehicles.

4.2 The approach Method

To solve this problem we will use the Particle Swarm Optimization and its algorithm.

Data

The data used in this study are presented bellow: The vehicle capacity used is 50

	Depot	1	2	3	4	5	6	7	8	9
Depot	0	42	81	73	52	96	37	28	61	10
1	42	∞	88	25	79	35	89	65	53	86
2	81	88	∞	29	96	84	57	33	47	62
3	73	25	29	∞	57	44	92	80	15	50
4	52	79	96	57	∞	25	60	45	69	81
5	96	35	84	44	25	∞	38	76	95	23
6	37	89	57	92	60	38	∞	61	15	76
7	28	65	33	80	45	76	61	∞	66	90
8	61	53	47	15	69	95	15	66	∞	73
9	10	86	62	50	81	23	76	90	73	∞

Table 4.1: The Distance Matrix.

Customer	Demand
1	10
2	8
3	12
4	6
5	18
6	4
7	15
8	22
9	2
10	17

Table 4.2: Customer Demands

4.2.1 Swarm Initialization

Initializing the swarm:

Creating a population of particles, where each particle represents a potential solution.

Each particle consists of a set of routes, and each route represents a sequence of customers to be visited.

Randomly initializing the positions (solution) and velocities for each particle.

Mathematical formulation:

Let N be the number of particles in the swarm.

Let M be the number of routes per particle.

Let K be the number of customers.

Each particle i has a position vector $\mathbf{X}_i = [X_{i1}, X_{i2}, ..., X_{iM}]$ representing the routes.

Each route j in particle i has a sequence of customers, $R_{ij} = [R_{ij1}, R_{ij2}, ..., R_{ijL}]$.

Each customer k has a demand D_k.

Initialization process:

Initialize the positions (routes) for each particle randomly:

For each particle i from 1 to N:

- For each route j from 1 to M:
  - Randomly assign a sequence of customers to route R_{ij}.

Initialize the velocities for each particle randomly:
For each particle i from 1 to N:

- For each route j from 1 to M:
  - Randomly assign a velocity V_{ij} to each customer in route R_{ij}.

Given Data:

- Number of particles (N) = 10
- Number of routes per particle (M) = 1
- Number of customers (K) = 10
- Customer demands:
  - Customer 1: 10
  - Customer 2: 8
  - Customer 3: 12
  - Customer 4: 6
  - Customer 5: 18
  - Customer 6: 4
  - Customer 7: 15
  - Customer 8: 22
  - Customer 9: 2
  - Customer 10: 17

Initialization Process:
Initialize the positions (routes) for each particle randomly:

- For each particle i from 1 to 10:
  - For each route j from 1 to 1:
    - Randomly assign a sequence of customers to route R_{ij}.

Let's assume the initial positions for each particle are as follows:

4.2.2 Evaluation

In the CVRP, the evaluation process involves calculating the total distance traveled by each particle's solution, taking into account the distances between customers and the capacity constraints of the vehicles. The evaluation provides a quantitative measure of how well each particle's solution satisfies the problem requirements and objectives.

Let P_i represent the position of particle i, which consists of a set of routes assigned to vehicles. Each route visits a subset of customers in a specific order. The fitness value of particle i, denoted as $f(P_i)$, is calculated as the sum of distances traveled by each vehicle in the particle's solution:

$$f(P_i) = \sum_{\text{vehicle } v} \sum_{\text{customer } j \in \text{route } r \in P_i} \text{distance}(j, \text{previous customer of } j)$$

Table 4.3: Particles

Particle 1
3 9 7 5 2 4 1 6 10 8

Particle 2
10 3 8 6 4 1 2 7 5 9

Particle 3
9 4 10 8 6 3 5 1 2 7

Particle 4
6 3 7 1 10 4 2 8 9 5

Particle 5
1 5 10 6 3 8 4 2 7 9

Particle 6
4 8 10 7 1 9 5 6 3 2

Particle 7
8 5 6 3 7 9 2 10 1 4

Particle 8
10 6 3 8 9 7 4 1 5 2

Particle 9
3 5 10 1 8 6 2 4 7 9

Particle 10
6 1 5 3 8 10 9 4 7 2

subject to the capacity constraint of each vehicle.

Using the given data, we can perform the evaluation for each particle and determine their respective fitness values. The fitness value represents the total distance traveled by all vehicles in the particle's solution. This evaluation step allows us to compare and assess the performance of different particles and guide the search for an optimal solution.

In the subsequent sections, we will present the evaluations of each particle, and calculate their fitness values using the provided distance matrix and capacity constraints.

Particle	Vehicle 1	Vehicle 2	Vehicle 3	Evaluation
1	[3, 9, 7, 5]	[2, 4, 1, 6]	[10, 8]	263
2	[10, 3, 8]	[6, 4, 1, 2]	[7, 5, 9]	315
3	[9, 4, 10]	[8, 6, 3, 5]	[1, 2, 7]	328
4	[6, 3, 7]	[1, 10, 4, 2]	[8, 9, 5]	298
5	[1, 5, 10]	[6, 3, 8, 4]	[2, 7, 9]	322
6	[4, 8, 10]	[7, 1, 9, 5]	[6, 3, 2]	310
7	[8, 5, 6]	[3, 7, 9, 2]	[10, 1, 4]	327
8	[10, 6, 3]	[8, 9, 7, 4]	[1, 5, 2]	328
9	[3, 5, 10]	[1, 8, 6, 2]	[4, 7, 9]	303
10	[6, 1, 5]	[3, 8, 10, 9]	[4, 7, 2]	336

Table 4.4: Evaluation of each Particle

4.2.3 the global best

by comparing the results this is the global best
Global Best: Routes: $[[3, 9, 7, 5], [2, 4, 1, 6], [10, 8]]$ **Fitness:** 263

4.2.4 Movement

n order to update the particle positions, we use the following equations:
new position = current position + velocity new position=current position+velocity
Let's calculate the updated positions for each particle based on their velocities:

Particle 1:
Current position:$[3, 9, 7, 5, 2, 4, 1, 6, 10, 8]$
Best position:$[3, 9, 7, 5, 2, 4, 1, 6, 10, 8]$
Evaluation:263
Random velocity:$[0.4, 0.2, 0.8, 0.6, 0.7, 0.9, 0.3, 0.1, 0.5, 0.7]$
Updated velocity:$[0.81, 0.39, 1.17, 0.87, 0.96, 1.26, 0.66, 0.33, 0.99, 1.11]$
Updated position:$[3.81, 9.39, 7.17, 5.87, 2.96, 4.26, 1.66, 6.33, 10.99, 8.11]$
Adjusted position (due to capacity constraint):$[3, 9, 7, 5, 2, 4, 1, 6, 10, 8]$
Fitness for new position:263
Best position and fitness remain unchanged.

We continue the same operation for each particle

4.2.5 Termination Condition

The algorithm terminates when either of the following conditions is met:

- The maximum number of iterations is reached.
- The fitness improvement between consecutive iterations falls below a certain threshold.

4.2.6 Solution Extraction

Solution Extraction:

The best solution obtained from the optimization process is:

Global Best Routes:

$$\begin{aligned} &\text{Route 1:}[3, 9, 7, 5] \\ &\text{Route 2:}[2, 4, 1, 6] \\ &\text{Route 3:}[10, 8] \end{aligned}$$

Global Best Fitness: 263

4.3 development environment

Table 4.5: Characteristics of Python Used

Characteristic	**Description**
Language	Python
Libraries	NumPy, Random
Data Structures	Numpy Array
Control Structures	For loop, If statement
Object-Oriented Programming	Classes
Optimization Algorithm	Particle Swarm Optimization (PSO)
Variables	distance_matrix, customer_demands, depot, vehicle_capacity, num_vehicles, num_particles, max_iterations, inertia_weight, cognitive_weight, social_weight
Functions	calculate_fitness, ___init___

4.3.1 result of the algorithm application

test 1

Num Clients	**Iteration Num**	**Vehicle Num**	**Capacity**	**Best Route**	**Best Fitness**
10	10	3	50	[9, 8, 4, 2] [6, 7, 1] [5, 3]	437
	50	3	50	[6, 2, 9, 7] [1, 4, 5] [3, 8]	437
	100	3	50	[9, 4, 6, 2] [1, 3, 8] [5, 7]	437

Table 4.6: Results Table

result analyzing

In our first test, we notice that we have a stable best fitness that equals to 149, with different routes, even with an iteration number as small as 10, in the next text we will try to run the program with an iteration number under 10.

test 2

Num Clients	**Iteration Num**	**Vehicle Num**	**Capacity**	**Best Route**	**Best Fitness**
10	2	3	50	[3, 5, 6, 7] [2, 8, 4] [9, 1]	499
	4	3	50	[1, 2, 4, 9] [3, 7] [5, 6, 8]	491
	8	3	50	[4, 2, 9, 7] [3, 6] [5, 1, 8]	533

Table 4.7: Results Table

result analyzing

when we changed the number of iterations to go below 10, we notice that the best fitness is increasing, and when the number of iterations was 8, it took multiple runs for the program to reach the best fitness of 491.

Benchmark

In order to compare our results to different benchmarks, we have adjusted the number of vehicles used, the number of customers, we will compare the results to 3 different benchmarks, with 300 iterations.

Benchmark	Num Clients	Num Vehicles	BKS	Benchmark Result
1	16	8	452	451.32
2	20	2	218	217.42
3	32	5	801	787.16

Table 4.8: Benchmark Resultsm BKS, 300 iteration

result amalazying

with 300 iterations, our best-known solution is close to the benchmark results. now we will do the same test with 50 iterations we notice after reducing the number of iterations, the best-known

Benchmark	Num Clients	Num Vehicles	BKS	Benchmark Result
1	16	8	702	655.69
2	20	2	517	466.89
3	32	5	1026	828.13

Table 4.9: Benchmark Results, KBS, 50 itereations

solution (BKS) increased dramatically with 50 iterations

4.4 Discussion and results analyzing

In the previous test without the benchmark comparison
A number of iterations under 10: it shows that results are not optimum for our subject of study.
A number of iterations =50, and all of the results show the best-known solution equal to 437.
the second set of tests with the Benchmarks:
with 300 iterations the results are approaching the benchmark results.
we also notice after increasing the number of clients, the algorithm takes longer to run.

4.5 Conclusion

In this chapter, we did a study on particle swarm optimization or Capacitated Vehicle Routing Problem, we came to the conclusion that our PSO algorithm compared to the benchmark provided is more likely to take longer with more clients to visit.

General Conclusion

In conclusion, this thesis has provided an in-depth exploration of the application of Particle Swarm Optimization (PSO) for solving the challenging Capacitated Vehicle Routing Problem (CVRP). Throughout the three chapters, we have delved into the background of the problem, reviewed pertinent literature, and proposed a PSO-based approach to tackle the CVRP.

While our research has yielded promising results and demonstrated the potential of PSO in addressing the CVRP, it is important to acknowledge that there is still much more to learn and develop in this field. Despite the progress made, our PSO algorithm has revealed areas that warrant further investigation and improvement to achieve faster and more accurate results that closely align with the existing benchmarks.

One key aspect that requires further attention is the fine-tuning of the PSO algorithm's parameters. While we have made initial efforts to optimize the algorithm, additional experiments and sensitivity analyses could be conducted to identify the optimal parameter settings that enhance convergence speed and solution quality. Moreover, exploring different variations of the PSO algorithm, such as hybridizing it with other metaheuristic techniques or incorporating local search procedures, could potentially lead to enhanced performance.

In summary, while our thesis has made significant strides in applying PSO to the CVRP, it is evident that there is still considerable room for growth and development. By addressing the aforementioned areas of improvement, we can advance the state-of-the-art in solving the CVRP using PSO, ultimately contributing to the field of optimization and logistics.

Bibliography

[1] Shabana Akhter and Shahedur Rahman. Particle swarm optimization based approach to vehicle routing problem. In *Proceedings of the International Conference on Electrical and Computer Engineering (ICECE)*, pages 577–580. IEEE, 2007.

[2] Una Benlic and Ionut Ioan Gogoaş. Metaheuristic algorithms for the vehicle routing problem with simultaneous pickup and delivery. *Journal of Heuristics*, 20(3):275–298, 2014.

[3] Bernd Bullnheimer, Richard F Hartl, and Christian Strauss. Applications of a Genetic-Algorithm for the Capacitated Vehicle Routing Problem. *Computers & Operations Research*, 24(1):33–46, 1997.

[4] Quan-Ke Chen, Qiao-Yan Wu, Lin Xia, and Jun Zhang. A hybrid swarm optimization and tabu search algorithm for vehicle routing problem with time windows. *Expert Systems with Applications*, 38(5):5602–5610, 2011.

[5] Jui-Sheng Chou and Jui-Yi Chou. Efficient hybrid particle swarm optimization for capacitated vehicle routing problem. *Expert Systems with Applications*, 54:204–212, 2016.

[6] Jui-Sheng Chou and Hsin-Yi Shih. Particle swarm optimization for capacitated vehicle routing problem. In *Proceedings of the International Conference on Industrial Engineering and Systems Management (IESM)*, pages 1–6. IEEE, 2017.

[7] Leandro dos Santos Coelho, Luiz AN Lorena, and André CP de L Carvalho. Particle swarm optimization: literature review and bibliometric analysis. *Expert Systems with Applications*, 36(4):10302–10311, 2009.

[8] Wei Fan and Qingyuan Zhang. A hybrid algorithm for the vehicle routing problem with simultaneous pickup and delivery based on ant colony optimization. *Journal of Advanced Transportation*, 2018, 2018.

[9] Mahmood Fathian, Bahman Naderi, and Mohammad Zandieh. Hybrid particle swarm optimization for vehicle routing problem with time windows and simultaneous delivery and pickup. *Computers & Industrial Engineering*, 99:177–192, 2016.

[10] Luca Maria Gambardella, Éric D Taillard, and Giovanni Agazzi. Macs-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 50(7):781–791, 1999.

[11] Philippe Lacomme, Christian Prins, Amar Ramdane-Cherif, and Frédéric Semet. A hybrid algorithm for the capacitated vehicle routing problem. In *Proceedings of the International Conference on Artificial Evolution (EA)*, pages 337–346. Springer, 2000.

[12] Fang Li, Xinyu Yu, Qiang Guo, Hu Wang, and Chunmei Li. A study of particle swarm optimization algorithm for the vehicle routing problem with time windows. *Mathematical Problems in Engineering*, 2016, 2016.

[13] Roberto Montemanni and Luca Maria Gambardella. An ant colony system for a dynamic vehicle routing problem. In *Proceedings of the International Conference on Artificial Intelligence (ICAI)*, pages 232–238. CSREA Press, 2006.

[14] Ibrahim Ozturk and Tolga Bektas. Solving the vehicle routing problem with simultaneous pickup and delivery using ant colony optimization. In *Proceedings of the International Conference on Industrial Engineering and Operations Management (IEOM)*, pages 1–6. IEEE, 2016.

[15] Sushanta Kumar Panda and Sudhir Kumar Padhy. Analysis of particle swarm optimization algorithms for the vehicle routing problem with time windows. In *Proceedings of the International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 392–396. IEEE, 2019.

[16] Günther R Raidl, Jakob Puchinger, and Andreas Chwatal. Population-based ant colony optimization for the capacitated vehicle routing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 587–594. ACM, 2000.

[17] S Ramadurai and R Sridharan. Hybrid particle swarm optimization and tabu search for vehicle routing problem. In *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1–5. IEEE, 2017.

[18] Paolo Toth and Daniele Vigo. *Vehicle Routing: Problems, Methods, and Applications*. SIAM, 2002.

[19] Wei Yang, Qiang Lin, and Min Zhang. Ant colony optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, 74: 113–121, 2014.

[20] Zhi-Hui Zhan, Jun Zhang, Yun Li, Henry SH Chung, and Yuhui Shi. Particle swarm optimization for the capacitated vehicle routing problem. *Swarm and Evolutionary Computation*, 1(2):89–100, 2011.