A thesis on Optimal Vehicle Routing Problem

Project-III (ME57003) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Masters of Technology Dual

in

Mechanical Engineering

by

Akshat Ansh Nayak

(20ME31007)

Under the supervision of

Dr. Dilip Kumar Pratihar

Department of Mechanical Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2024-25

November 11, 2024

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: November 11, 2024 (Akshat Ansh Nayak)

Place: Kharagpur (20ME31007)

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**A thesis on Optimal Vehicle Routing Problem**" submitted by **Akshat Ansh Nayak** (Roll No. 20ME31007) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Masters of Technology Dual in Mechanical Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2024-25.

Date: November 11, 2024
Place: Kharagpur

Dr. Dilip Kumar Pratihar
Department of Mechanical Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Akshat Ansh Nayak** Roll No: **20ME31007**

Degree for which submitted: **Masters of Technology Dual**

Department: **Department of Mechanical Engineering**

Thesis title: **A thesis on Optimal Vehicle Routing Problem**

Thesis supervisor: **Dr. Dilip Kumar Pratihar**

Month and year of thesis submission: **November 11, 2024**

This thesis investigates the application of the Bonobo Optimizer (BO), a novel meta-heuristic nature-inspired algorithm, to address the Vehicle Routing Problem (VRP) with capacity constraints. BO leverages fission-fusion social grouping and varied mating strategies (promiscuous, restrictive, consortship, and extra-group mating), to balance exploration and exploitation within the solution space. For this study, both BO and the Genetic Algorithm (GA) are applied to optimize delivery routes in a simulated city environment, where a vehicle operates under load capacity constraints and returns to a central depot for restocking as necessary. Comparative analysis demonstrates that BO consistently outperforms GA in minimizing total travel distance, converging to lower-cost solutions after a specific number of iterations. However, BO's higher computational demands present a trade-off, highlighting an area for future optimization in terms of runtime efficiency.

Contents

Declaration i

Certificate ii

Abstract iii

Contents iv

List of Figures vi

List of Tables vii

Abbreviations viii

Symbols ix

1 Literature Review 1
1.1 Introduction 1

2 Case Study 4
2.1 Problem Statement 4
2.2 Mathematical Formulation 5
2.2.1 Decision Variables 5
2.2.2 Objective Function 6
2.2.3 Constraints 6

3 Solution Approach 8
3.1 Mathematical Model of the Proposed Bonobo Optimizer (BO) 8
3.1.1 Population Initialization 8
3.1.2 Algorithm Phases: Positive and Negative 9
3.1.3 Social Grouping and Mating Strategies 10
3.1.4 Mating Mechanisms 11
3.1.5 Adaptive Parameter Control 11
3.1.6 Boundary and Acceptance Conditions 12

3.2 Mathematical Model of the Genetic Algorithm (GA) . . . 13
3.2.1 Population Initialization . . . 13
3.2.2 Selection of Parent Chromosomes . . . 13
3.2.3 Crossover Strategies . . . 14
3.2.4 Mutation Strategies . . . 15
3.2.5 Fitness Evaluation . . . 16
3.3 Application of BO and GA to the VRP . . . 16
3.3.1 Handling Infeasible Solutions . . . 16
3.3.2 Fitness Function with Penalty Mechanisms . . . 17
Static Penalty Function . . . 18
Dynamic Penalty Function . . . 18
Adaptive Penalty Function . . . 18

4 Experimentation and Results 19
4.1 Parameter Optimization for Algorithms . . . 19
4.2 Dataset and Experiment Setup . . . 21
4.3 Simulation Results and Analysis . . . 22
4.3.1 Cost Performance Analysis . . . 22
4.3.2 Computation Time Comparison . . . 23
4.4 Conclusion . . . 24

5 Future Scope of Study 25
5.1 Expanding Problem Variations . . . 25
5.2 Incorporating Dynamic Conditions . . . 26
5.3 Optimizing Code with Parallel Computing . . . 27
5.4 Further Research on BO and Comparative Studies . . . 28
5.5 Conclusion . . . 29

Bibliography 30

List of Figures

4.1 Parameter Optimization Results for p_d, p_f, p_{gsm}, and Population Size. Optimal values are indicated by red dashed lines. 20
4.2 Cost vs. Iterations for BO and GA 23

List of Tables

4.1 Initial and Optimal Parameter Values for BO and GA Algorithms . . 19
4.2 Node Parameters for VRP Simulation 21

Abbreviations

GA	**G**enetic **A**lgorithm
BO	**B**onobo **O**ptimizer
VRP	**V**ehicle **R**outing **P**roblem
PSO	**P**article **S**warm **O**ptimization
ACO	**A**nt **C**olony **O**ptimization
DE	**D**ifferential **E**volution
OX	**O**rder **C**rossover
CX	**C**ycle **C**rossover
PMX	**P**artially **M**apped **C**rossover
ERC	**E**dge **R**ecombination **C**rossover
PP	**P**ositive **P**hase
NP	**N**egative **P**hase

Symbols

P	Population set of chromosomes or individuals
p_i	Chromosome or individual in population P
$f(p_i)$	Fitness function evaluating chromosome p_i
N	Population size
p_d	Directional Probability (in BO)
p_f	Phase Probability (in BO)
p_{gsm}	Group Size Multiplier (in BO)
ppc	Positive phase count
npc	Negative phase count
$tsgsfactor$	Temporary subgroup size factor
$tsgsfactor_{\max}$	Maximum subgroup size factor
$tsgs_{\max}$	Maximum size of a temporary social subgroup
C	Constant penalty coefficient
P_i	Penalty term for infeasible solutions
$\phi_{ik}(\mathbf{X})$	Violation measure for the k-th constraint in solution i
$\lambda(t)$	Penalty parameter in adaptive penalty function
t	Iteration or generation number
$F_i(\mathbf{X})$	Fitness of a solution i with penalties applied
$d(a, b)$	Euclidean distance between nodes a and b
X	Solution vector or candidate solution
r_1	Random number uniformly generated in the interval $(0, 1)$
$scab$	Scaling coefficient for alpha bonobo influence
$scsb$	Scaling coefficient for social subgroup influence

Chapter 1

Literature Review

1.1 Introduction

Across industries such as logistics, transportation, manufacturing, and services, a persistent challenge exists: How can limited resources be utilized most effectively? Optimization addresses this question by focusing on maximizing efficiency within specific constraints. Historically, early optimization approaches relied on rigorous mathematical models effective for smaller-scale problems but often inadequate for the complexities of real-world scenarios. Recognizing these limitations, researchers turned to nature for inspiration, leading to the emergence of heuristic and meta-heuristic approaches—flexible, adaptive solutions designed for complex optimization problems Baker and Ayechew (2003); Yang (2010b).

One of the most prominent optimization challenges is the Vehicle Routing Problem (VRP). This classic problem seeks to determine the most cost-effective routes for a fleet of vehicles to serve a set of locations, subject to constraints such as vehicle capacity and route timing. Solving VRP involves not just finding a feasible route but identifying the optimal route among a vast number of possibilities. Due to its

combinatorial nature, VRP is considered NP-hard, making exact solutions computationally prohibitive for large instances. In this context, metaheuristic algorithms have demonstrated substantial success, helping logistics companies reduce time, fuel consumption, and overall costs. Genetic Algorithms (GAs), for instance, have gained popularity for VRP due to their capacity to emulate natural evolution, leveraging selection, crossover, and mutation operators to iteratively refine solutions Wang and Li (2023); Laporte (1992).

However, the increasing complexity of optimization problems has highlighted the limitations of any single algorithm's effectiveness. According to the "No Free Lunch" theorem, no optimization algorithm can universally outperform others across all problem types Wolpert and Macready (1997); Adam et al. (2019). GAs, while efficient in exploring vast solution spaces and preserving solution diversity, can face challenges in convergence speed and require fine-tuning of parameters (e.g., mutation rates) to adapt effectively to a problem's specific characteristics.

In response to these challenges, researchers have broadened their scope, developing algorithms inspired by a range of natural phenomena, including bird flocking, ant foraging, and bacterial behavior Dorigo et al. (1996); Salhi and Fraga (2004). While these approaches have made significant strides, human and animal social structures—characterized by collaborative problem-solving and adaptability—offer additional potential for innovative optimization strategies. This concept has led to the development of the Bonobo Optimizer (BO), a recent metaheuristic based on the unique social dynamics and reproductive behaviors of bonobos, one of humanity's closest relatives Das and Pratihar (2019). Bonobos exhibit a highly adaptable social structure known as fission-fusion dynamics, where group members form and dissolve subgroups flexibly based on environmental conditions. This dynamic adaptability, combined with diverse mating strategies, enables BO to balance exploration (broadly searching for solutions) and exploitation (refining promising solutions) for a robust optimization process.

BO incorporates several social behaviors from bonobo communities to enhance problem-solving effectiveness. Through mechanisms such as promiscuous, restrictive, consortship, and extra-group mating, BO adapts its search strategies dynamically to the problem landscape. This adaptability enables BO to maintain diversity within the search space while achieving rapid convergence on optimal solutions. Unlike GAs, which require predefined parameters that may limit their adaptability, BO's parameters adjust based on performance, making it inherently responsive to the specific demands of each optimization task.

This study applies BO to the VRP and compares its performance with that of the Genetic Algorithm, a well-established method for VRP. By utilizing BO in this context, we aim to demonstrate how BO's adaptive capabilities enhance its ability to handle the complexities and constraints inherent in VRP, potentially leading to faster convergence and greater cost efficiency. Through comparative analysis, this research reveals that BO not only competes with but in certain scenarios surpasses GA in VRP optimization, offering a promising new direction in nature-inspired algorithms.

The following sections provide an in-depth review of the VRP problem, BO's mathematical foundations, and a performance analysis of BO and GA. This chapter establishes the significance of BO as a competitive tool for VRP and paves the way for future exploration of socially-inspired optimization techniques to address complex, real-world challenges.

Chapter 2

Case Study

2.1 Problem Statement

In the context of real-world applications of optimization, the Vehicle Routing Problem (VRP) stands as a classic challenge in logistics, requiring efficient allocation of resources to minimize operational costs. This chapter presents a practical case study on VRP as applied to the operations of RenewHeat, a startup based in Kharagpur that operates a thrift marketplace for pre-loved clothing items. Co-founded by the author and peers, RenewHeat has recently experienced an upsurge in demand, necessitating the optimization of its delivery logistics to enhance efficiency and reduce costs.

Due to limited resources, RenewHeat relies on a single delivery person with a motorcycle, which has a carrying capacity constraint of 50 kg. With 50 delivery orders to be completed within a day, the delivery person must make multiple trips, restocking at a central depot between trips. The objective of this study is to formulate an optimal delivery route that minimizes the total distance traveled, thereby reducing fuel consumption and operational costs.

This scenario aligns well with the VRP with capacity constraints, a well-known NP-hard problem, which highlights the importance of heuristic and metaheuristic approaches for obtaining feasible solutions in a reasonable time frame Laporte (1992). This case study will demonstrate how optimization techniques can be leveraged to improve delivery logistics in a resource-constrained environment.

2.2 Mathematical Formulation

The VRP for RenewHeat's delivery scenario is formulated mathematically as follows:

Let:

- $N = 50$: The total number of delivery locations (nodes).
- $V = \{0, 1, 2, \ldots, N\}$: The set of locations, where 0 represents the depot, and $1, 2, \ldots, N$ are the delivery locations.
- D_i: Demand at each location i, for $i \in \{1, 2, \ldots, N\}$.
- $C = 50$: Maximum carrying capacity of the delivery motorcycle in kg.
- x_i, y_i: Coordinates of each delivery location i, with x_0, y_0 as the coordinates of the depot.
- $d(i, j)$: Euclidean distance between any two locations i and j, calculated as $d(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

2.2.1 Decision Variables

- $x_{ij} \in \{0, 1\}$: A binary variable where $x_{ij} = 1$ if the route includes travel from location i to location j; otherwise $x_{ij} = 0$.
- q_i: The load carried upon arrival at location i, subject to the constraint $0 \leq q_i \leq C$.

2.2.2 Objective Function

The objective is to minimize the total travel distance, which directly correlates with fuel consumption. This can be formulated as:

$$\text{Minimize} \sum_{i=0}^{N} \sum_{j=0}^{N} d(i,j) \cdot x_{ij}$$

2.2.3 Constraints

To ensure feasibility, the following constraints are imposed:

1. **Capacity Constraint**: The total weight of items on the motorcycle at any time cannot exceed C:
$$\sum_{i=1}^{N} D_i \cdot x_{ij} \leq C \quad \forall j$$

2. **Demand Fulfillment**: Each delivery location i must be visited exactly once:
$$\sum_{j=0}^{N} x_{ij} = 1 \quad \forall i \neq 0$$

3. **Depot Return Requirement**: The delivery person must return to the depot to restock when the remaining capacity is insufficient to meet the next location's demand:
$$D_j \leq q_i \quad \text{if } x_{ij} = 1$$

4. **Subtour Elimination Constraint**: To prevent disconnected tours that do not return to the depot, we enforce:
$$x_{ij} + x_{ji} \leq 1 \quad \forall i, j \neq 0$$

This formulation serves as the basis for solving the VRP with capacity constraints at RenewHeat. Using heuristic or metaheuristic algorithms—such as the Bonobo

Optimizer (BO) introduced in Chapter 1—this model enables the exploration of optimal routes that reduce operational costs and meet all delivery demands Dorigo et al. (1996).

By structuring the problem in this manner, we set the stage for applying advanced optimization techniques to RenewHeat's delivery challenges. The next chapter will describe the implementation of BO and Genetic Algorithm (GA) on this formulated problem and analyze their performance.

Chapter 3

Solution Approach

3.1 Mathematical Model of the Proposed Bonobo Optimizer (BO)

BO is a nature-inspired, population-based metaheuristic algorithm that emulates the social behaviors and reproductive strategies of bonobos, particularly their adaptability and complex social interactions Das and Pratihar (2019). In BO, each candidate solution—referred to as a "bonobo"—represents an individual in the population, initialized randomly within the solution space. At each iteration, the fitness values of all bonobos are evaluated, and the bonobo with the best fitness is identified as the "alpha bonobo." This alpha bonobo represents the best solution in the current population and guides the optimization process, similar to the role of leaders in natural populations Waal (1995).

3.1.1 Population Initialization

Let:

- $P = \{p_1, p_2, \ldots, p_N\}$ represent the population of bonobos, where N is the population size.
- Each bonobo $p_i \in P$ is a potential solution in the search space, initialized randomly within the problem's feasible range.

Define $f(p_i)$ as the fitness function evaluating solution p_i. The alpha bonobo, p_α, is the bonobo with the optimal fitness value:

$$f(p_\alpha) = \min\{f(p_i) \mid i = 1, 2, \ldots, N\} \quad \text{for a minimization problem.}$$

3.1.2 Algorithm Phases: Positive and Negative

BO operates in two distinct phases based on the concept of social dynamics in bonobo communities: the Positive Phase (PP) and the Negative Phase (NP), aligning the exploration and exploitation balance in the search space Yang (2010b).

- **Positive Phase (PP)**: PP simulates favorable conditions in the bonobo community, where there is abundant food, security, and mating opportunities. During PP, the algorithm prioritizes **exploitation**, refining solutions around the alpha bonobo. A parameter, the positive phase count (*ppc*), increments with each PP iteration, indicating ongoing improvement.
- **Negative Phase (NP)**: NP simulates adverse conditions, promoting **exploration** as bonobos search for new solutions. A negative phase count (*npc*) increments in each NP iteration, encouraging diversity in the search space and helping avoid local optima.

BO adaptively transitions between PP and NP based on the quality of solutions generated in each phase, maintaining a dynamic balance between exploration and exploitation Holland (1992).

3.1.3 Social Grouping and Mating Strategies

To emulate bonobo social structures, BO incorporates dynamic social grouping and diverse mating strategies that influence solution updates. These strategies are probabilistically selected depending on the current phase, enhancing the search capability and convergence rate.

- **Promiscuous Mating**: Allows multiple bonobos to mate with the alpha bonobo, increasing population diversity through widespread genetic sharing.
- **Restrictive Mating**: Limits mating to high-ranking bonobos, focusing on refining current solutions within an elite group, similar to selection pressure in evolutionary algorithms Das and Pratihar (2018).
- **Consortship Mating**: Represents exclusive mating pairs within subgroups, aiding in controlled exploration within localized regions of the search space.
- **Extra-Group Mating**: Involves mating with bonobos outside the primary social group, promoting wider search diversity and preventing premature convergence.

The **temporary subgroup size factor** ($tsgsfactor$) defines the maximum size of social subgroups in mating, calculated as:

$$tsgsfactor_{\text{initial}} = 0.5 \times tsgsfactor_{\max},$$

where $tsgsfactor_{\max}$ is the predefined maximum subgroup size. The subgroup size for each iteration, $tsgs_{\max}$, is determined as:

$$tsgs_{\max} = \max(2, tsgsfactor \times N),$$

where N is the population size.

3.1.4 Mating Mechanisms

BO utilizes different mathematical formulations for generating new solutions (offspring) based on the mating strategies:

- **Promiscuous and Restrictive Mating**: For promiscuous or restrictive mating, the new bonobo new_bonobo_j (offspring) is generated as follows:

$$\text{new\_bonobo}_j = \text{bonobo}_j^i + r_1 \times \text{scab} \times \left(\alpha_j^{\text{bonobo}} - \text{bonobo}_j^i\right) + (1 - r_1)$$

$$\times \text{scsb} \times \text{flag} \times \left(\text{bonobo}_j^i - \text{bonobo}_j^p\right),$$

  where:

  - scab and scsb are scaling coefficients for the alpha and partner bonobos, respectively.
  - flag is a directional parameter taking values $+1$ or -1.
  - r_1 is a random number uniformly generated in the interval $(0, 1)$.

- **Consortship and Extra-Group Mating**: For consortship or extra-group mating, new bonobos are generated using boundary-driven exploration:

$$\text{new\_bonobo}_j = \text{bonobo}_j^i + \beta_1 \times (\text{Var}_{\text{max},j} - \text{bonobo}_j^i),$$

  or

$$\text{new\_bonobo}_j = \text{bonobo}_j^i - \beta_2 \times (\text{bonobo}_j^i - \text{Var}_{\text{min},j}),$$

  where β_1 and β_2 are random scaling factors, and $\text{Var}_{\text{max},j}$ and $\text{Var}_{\text{min},j}$ denote the upper and lower boundaries for each variable j Whitley et al. (1990).

3.1.5 Adaptive Parameter Control

BO's parameters are dynamically adjusted to guide the search efficiently:

- **Directional Probability (p_d)**: Initialized at 0.5, p_d determines movement direction and is updated based on the phase. Higher values are set during PP for exploitation, while NP favors exploration by reducing p_d.

- **Phase Switching**: If no improvement is observed in the alpha bonobo's fitness, the algorithm adapts by resetting *npc* and increasing *ppc*, adjusting phase-related parameters as follows:

$$npc = 0, \quad ppc = ppc + 1, \quad cp = \min(0.5, ppc \times \text{rcpp}),$$

  where rcpp is the rate of change in phase probability, and *cp* manages subgroup formation.

3.1.6 Boundary and Acceptance Conditions

- **Boundary Conditions**: If new_bonobo_j exceeds the boundary $\text{Var}_{\max,j}$ or falls below $\text{Var}_{\min,j}$, it is clamped to the nearest boundary:

$$\text{new\_bonobo}_j = \max(\min(\text{new\_bonobo}_j, \text{Var}_{\max,j}), \text{Var}_{\min,j}).$$

- **Acceptance Criteria**: A newly created bonobo replaces an existing one if it has a superior fitness value or if it meets probability-based acceptance criteria for diversity, ensuring that the population remains varied while progressing towards the optimal solution Joines and Houck (1994).

This formulation of BO combines adaptive parameter control, boundary handling, and probabilistic acceptance, which collectively enable it to effectively navigate complex optimization landscapes. By mimicking bonobo social dynamics, BO addresses issues of local optima and ensures sustained exploration, making it a promising approach for solving complex combinatorial problems.

3.2 Mathematical Model of the Genetic Algorithm (GA)

GA is an evolutionary optimization technique inspired by the principles of natural selection Holland (1975); Goldberg (1989). GA operates with a population of candidate solutions, termed "chromosomes," which iteratively evolve through selection, crossover, and mutation to produce new generations of solutions. Each chromosome represents a potential solution to the optimization problem. The goal is to identify the optimal chromosome, minimizing (or maximizing) a predefined fitness function across multiple iterations.

3.2.1 Population Initialization

Let:

- $P = \{p_1, p_2, \ldots, p_N\}$ represent the population of chromosomes, where N is the population size.
- Each chromosome $p_i \in P$ is initialized randomly within the feasible solution space.

Define $f(p_i)$ as the fitness function evaluating each chromosome p_i. GA iteratively selects chromosomes based on fitness to create a new generation, expected to be progressively closer to the optimal solution.

3.2.2 Selection of Parent Chromosomes

Selection is the process by which chromosomes are chosen from the current population P for reproduction based on their fitness values. Common selection methods

include **roulette wheel selection**, **tournament selection**, and **rank-based selection** Goldberg and Deb (1991). In this work, tournament selection is applied, where a subset of chromosomes is randomly chosen, and the fittest among them is selected as a parent. This probabilistic method emphasizes fitter chromosomes while maintaining diversity in the population.

3.2.3 Crossover Strategies

Crossover is a crucial GA operator where pairs of parent chromosomes exchange genetic information to produce offspring. This process encourages exploration of new solutions while preserving beneficial traits. For this study, the **Edge Recombination Crossover (ERC)** method is utilized, which prioritizes adjacency information to maintain connectivity, essential in routing problems. Other crossover methods are also discussed for comparison.

1. **Edge Recombination Crossover (ERC)**: ERC, proposed by Whitley et al., is suitable for problems where adjacency and connectivity between nodes are critical Whitley et al. (1990); Spears and De Jong (1991). An edge table is first created, recording all adjacent nodes for each node based on both parent chromosomes.

Let:

- $Pr1$ and $Pr2$ represent the two parent chromosomes.
- An edge table is constructed where each node i in $Pr1$ and $Pr2$ is linked to its adjacent nodes.

The offspring, $Ch1$, is generated using the following steps:

1. Begin with a randomly selected node from $Pr1$.
2. Select the next node based on the least number of connections in the edge table, ensuring continuity in adjacency.

3. Remove each selected node from the edge table to maintain route validity.

This process continues until all nodes are visited, yielding a complete sequence in $Ch1$.

2. **Order Crossover (OX)**: OX is a position-based crossover technique that selects a subset of consecutive genes from one parent and fills the remaining positions with genes from the other parent in their original order Spears and De Jong (1991).

3. **Cycle Crossover (CX)**: CX preserves positional inheritance by identifying cycles between the parents. Nodes that share cycles in $Pr1$ and $Pr2$ are mapped to $Ch1$ to maintain genetic integrity Oliver et al. (1987).

4. **Position-Based Crossover (PBX)**: In PBX, a subset of positions is chosen from one parent, and the rest are filled by the sequence in the second parent, preserving the initial sequence.

5. **Partially Mapped Crossover (PMX)**: PMX maintains the relative positions of nodes in a segment selected from $Pr1$, filling the remaining positions with nodes from $Pr2$ to maintain route consistency Spears and De Jong (1991).

Each crossover method has unique trade-offs. For instance, **OX** might disrupt adjacency in VRP applications, while **CX** ensures absolute positional inheritance but can be computationally intensive in large-scale problems. **ERC** is selected here due to its ability to retain adjacency, which is crucial in vehicle routing applications.

3.2.4 Mutation Strategies

Mutation introduces randomness into the population, preventing premature convergence and ensuring diverse solutions. For VRP, mutation operations modify the sequence of nodes to explore alternative routes. It is applied with a probability $p_m(\approx 0.1)$ to each chromosome in the population Spears and De Jong (1992).

1. **Swap Mutation**: Two nodes in the chromosome are selected randomly and their positions are swapped.

2. **Inversion Mutation**: A segment of nodes is selected and reversed to create a new sequence. This alteration can yield shorter routes by reordering nodes within a confined area.

3. **Insertion Mutation**: A randomly chosen node is removed from its current position and inserted elsewhere in the chromosome, changing the local sequence.

3.2.5 Fitness Evaluation

The fitness of each chromosome p_i is evaluated based on the total travel distance for a given route. The objective is to minimize the total distance, represented as:

$$f(p_i) = \sum_{j=1}^{N-1} d(p_i[j], p_i[j+1]) + d(p_i[N], p_i[1]),$$

where $d(a, b)$ is the Euclidean distance between nodes a and b.

3.3 Application of BO and GA to the VRP

To address the VRP effectively, BO and GA are applied with the goal of minimizing the total route distance while satisfying delivery constraints. Each solution sequence represents a proposed route, structured to ensure each delivery location is visited and the vehicle returns to a central depot as needed due to capacity limits.

3.3.1 Handling Infeasible Solutions

In the context of VRP, certain solution configurations may be infeasible due to the following constraints:

- **Multiple Occurrences of the Same Location**: A feasible solution should visit each location exactly once per delivery round. However, certain evolutionary operations may lead to repeated locations, resulting in redundancy and inefficiency.
- **Capacity Violations**: The vehicle has a fixed carrying capacity. Any solution that exceeds this limit within a single delivery round is infeasible. For example, if the vehicle's capacity is 50 kg and a route segment requires more than this load without returning to the depot, it is deemed infeasible.
- **Skipped Locations**: Each location must be included in the route to ensure complete deliveries. However, some operations might skip locations, resulting in incomplete routes.

To manage infeasible solutions, penalty functions are applied, adjusting fitness values to penalize those violating constraints Deb (1995); Joines and Houck (1994).

3.3.2 Fitness Function with Penalty Mechanisms

The fitness of a solution $F_i(\mathbf{X})$ is given by:

$$F_i(\mathbf{X}) = f_i(\mathbf{X}) + P_i,$$

where $f_i(\mathbf{X})$ represents the objective function (total distance traveled), and P_i is a penalty term applied to infeasible solutions.

Three types of penalty functions are utilized to handle infeasible solutions: static, dynamic, and adaptive.

Static Penalty Function The static penalty function applies a fixed penalty based on constraint violation:

$$P_i = C \sum_{k=1}^{q} \left(\phi_{ik}(\mathbf{X})\right)^2,$$

where C is a constant penalty coefficient, q is the number of constraints, and $\phi_{ik}(\mathbf{X})$ quantifies the violation for the k-th constraint.

Dynamic Penalty Function The dynamic penalty function adjusts over generations to enforce constraints progressively:

$$F_i(\mathbf{X}) = f_i(\mathbf{X}) + (C \times t)^{\alpha} \sum_{k=1}^{q} \left(\phi_{ik}(\mathbf{X})\right)^{\beta},$$

where C, α, and β are constants, and t denotes the generation number. Penalties increase over time, enforcing feasibility in later stages Bean and Hadj-Alouane (1992).

Adaptive Penalty Function The adaptive penalty function updates the penalty parameter $\lambda(t)$ based on solution feasibility:

$$\lambda(t+1) = \begin{cases} \frac{1}{\beta_1} \times \lambda(t), & \text{if feasible (Case 1)} \\ \beta_2 \times \lambda(t), & \text{if infeasible (Case 2)} \\ \lambda(t), & \text{otherwise} \end{cases}$$

where $\beta_1 \neq \beta_2$ and $\beta_1, \beta_2 > 1$.

Both BO and GA iterate over a fixed number of generations, evaluating solution fitness and generating new candidates through crossover and mating strategies. Infeasible solutions are penalized, balancing exploration and exploitation for convergence toward optimized routes.

Chapter 4

Experimentation and Results

4.1 Parameter Optimization for Algorithms

To ensure optimal performance of the Bonobo Optimizer (BO) and Genetic Algorithm (GA), parameter tuning was conducted prior to running the main simulations. Following the methodology outlined in previous sections and best practices in parameter optimization for evolutionary algorithms Eiben and Smith (2003), we initially set the parameters based on common evolutionary algorithm values.

Parameter	**Initial Value**	**Optimal Value**
Directional Probability (p_d)	0.5	0.9
Phase Probability (p_f)	0.7	0.8
Group Size Multiplier (p_{gsm})	0.9	0.8
Population Size	100	140

Table 4.1: Initial and Optimal Parameter Values for BO and GA Algorithms

Each parameter was systematically adjusted across a range of values to observe its impact on the fitness outcomes, with the objective of striking a balance between exploration and exploitation in both algorithms Yang (2010a). These initial values were selected based on heuristics commonly used in evolutionary computation to provide a foundation for systematic adjustment.

The optimal values were derived empirically, based on minimal fitness costs and efficient search behavior. Table 4.1 summarizes the initial and optimized parameters for BO and GA, while Fig. 4.1 illustrates the impact of each parameter on the cost.

Each parameter's optimal value, marked with a red dashed line in Fig. 4.1, was selected to minimize cost effectively, achieving an ideal balance between exploration (broad search) and exploitation (fine-tuning of promising solutions).

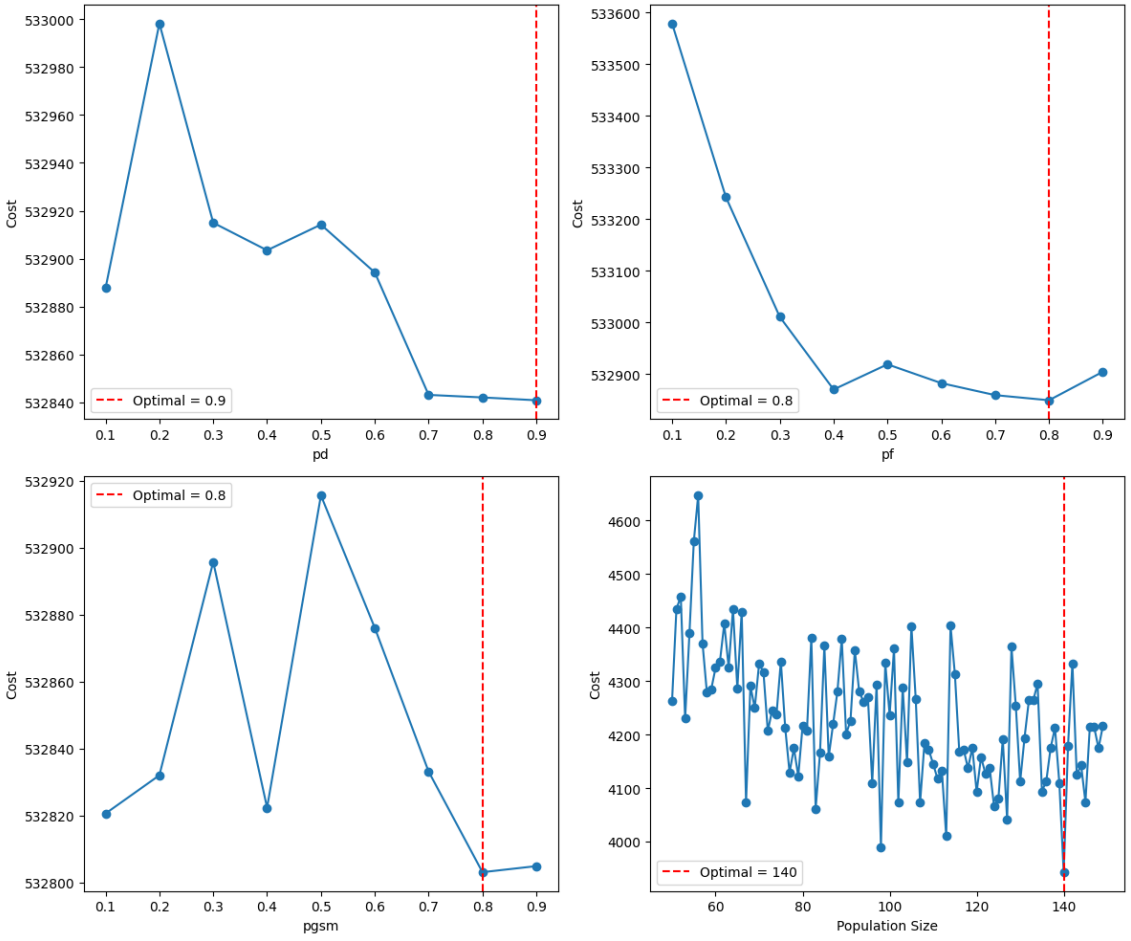

FIGURE 4.1: Parameter Optimization Results for p_d, p_f, p_{gsm}, and Population Size. Optimal values are indicated by red dashed lines.

4.2 Dataset and Experiment Setup

The algorithms were executed on a simulated dataset comprising 50 nodes, representing demand points in a Vehicle Routing Problem (VRP) scenario. The VRP setup aimed to minimize the total travel distance under a vehicle capacity constraint of 50 kg. Each node represents a specific demand location, with coordinates provided for reference (see Table 4.2). The VRP formulation, focusing on optimizing routes for demand nodes, is widely used in logistics and supply chain management research Laporte (1992).

Node	**Capacity**	**X-Coordinate**	**Y-Coordinate**
01	47.844	1.032	41.509
02	45.858	6.035	28.111
03	11.242	83.926	13.504
04	20.283	19.213	47.725
05	16.428	4.520	48.472
06	26.383	72.873	99.481
07	21.076	70.693	0.949
08	12.142	28.986	21.918
09	21.046	54.913	18.029
10	8.055	75.932	0.492
11	3.695	38.550	56.073
12	6.019	59.847	92.361
13	17.525	61.877	47.003
14	4.437	57.150	38.223
15	14.257	76.898	20.498
.	.	.	.
.	.	.	.
.	.	.	.
50	5.942	63.083	27.201

Table 4.2: Node Parameters for VRP Simulation

The BO and GA algorithms utilized adaptive penalty functions to handle infeasible solutions, which arise due to routing and capacity constraints. The adaptive

penalty mechanism dynamically adjusts penalties to guide the search towards feasible solutions that satisfy the vehicle's capacity limit and ensure each location is visited exactly once.

4.3 Simulation Results and Analysis

After optimizing the parameters, the BO and GA algorithms were executed over a fixed number of iterations, with adaptive penalty functions enabled. The evaluation criteria were based on cost (total distance traveled) and computational time.

4.3.1 Cost Performance Analysis

The cost performance for both BO and GA was plotted against the number of iterations, as shown in Fig. 4.2. Initially, both algorithms exhibited a steady reduction in cost; however, BO demonstrated a faster convergence rate after approximately the 23rd iteration. By the 225th iteration, BO achieved a minimum cost of 3716.25, outperforming GA, which reached a minimum cost of 3937.06.

This result demonstrates BO's superiority over GA in minimizing travel distance within the VRP context. The enhanced exploration and exploitation balance in BO, driven by its unique social and mating strategies, contributed to its faster convergence and lower final cost. BO's adaptive dynamics enabled it to escape local minima more effectively than GA, which is critical in combinatorial optimization problems like VRP Yang (2010b).

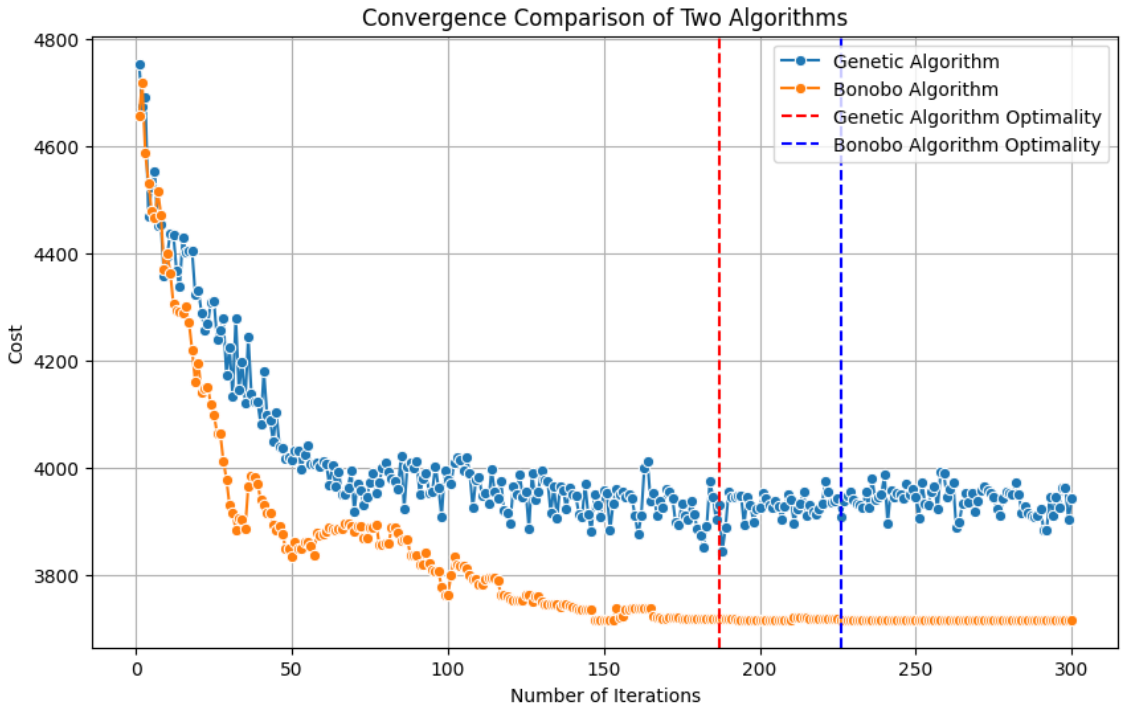

FIGURE 4.2: Cost vs. Iterations for BO and GA

4.3.2 Computation Time Comparison

While BO achieved superior cost optimization, it required significantly more computational time than GA, a factor that may limit its applicability in time-sensitive environments. Specifically:

- Time taken for GA: 28.96 seconds
- Time taken for BO: 3133.90 seconds

The substantial increase in computational time for BO can be attributed to its complex social dynamics and mating strategies, which involve additional calculations relative to GA's simpler operations Holland (1992). This trade-off between solution quality and computation time is often observed in metaheuristic algorithms, where strategies enhancing diversity may increase the computational load Deb (1995).

Therefore, while BO offers a higher-quality solution, its computational demands suggest that it may be better suited for applications where solution quality is prioritized over rapid computation. In contrast, GA may be preferred in real-time or computationally constrained environments where a quicker, albeit suboptimal, solution is acceptable.

4.4 Conclusion

The experimental results illustrate the efficacy of BO in solving the Vehicle Routing Problem by delivering lower travel distances compared to GA. BO's adaptive social dynamics fostered effective exploration and exploitation, yielding superior solutions. However, the time complexity of BO limits its application in scenarios requiring prompt solutions. For high-stakes applications where solution quality is crucial, BO may still be preferred despite its computational demands, whereas GA could be suitable for rapid approximations in dynamic environments.

Chapter 5

Future Scope of Study

Bonobo Optimizer (BO) has demonstrated more promising results than Genetic Algorithm (GA) in addressing the Vehicle Routing Problem (VRP) under controlled conditions with fixed locations and demand constraints. However, there exists substantial potential for extending this study to encompass a broader array of problem scenarios, computational optimizations, and algorithmic enhancements. This chapter discusses potential directions for future research that could improve the performance, applicability, and scalability of the proposed solution approaches.

5.1 Expanding Problem Variations

One immediate extension of this work is to evaluate the performance of BO and GA on more complex VRP variations by scaling the number of locations and altering vehicle capacity constraints. Increasing the complexity of the problem would provide valuable insights into the scalability and adaptability of the algorithms under more realistic conditions. This extension includes:

- **Increasing the Number of Locations**: Expanding the problem to include hundreds or even thousands of delivery points would test the scalability and

robustness of the algorithms. Large-scale VRP instances are common in logistics applications, and such an evaluation could uncover the strengths and limitations of BO in handling more extensive datasets Laporte (1992).

- **Adjusting Vehicle Capacity Constraints**: Varying vehicle capacity could simulate diverse logistical constraints, requiring more sophisticated handling of infeasible solutions. Examining how BO adapts to these constraints could reveal its efficiency in load distribution and capacity management, as well as highlight any necessary adjustments to penalty mechanisms Toth and Vigo (2014).

These problem variations would help validate the adaptability of BO and GA in more complex scenarios, enhancing the insights from this study and potentially identifying specific parameter tuning strategies or algorithm modifications for larger or more constrained VRP instances.

5.2 Incorporating Dynamic Conditions

The current study addresses a static VRP with fixed demand and location parameters. However, many real-world applications involve dynamic conditions where demand varies over time or where new locations are introduced or removed during operation. Future work could incorporate:

- **Variable Demand Over Time**: In real-life scenarios, demand at each location may fluctuate. Implementing a dynamic demand VRP would require modifying the fitness evaluation and penalty functions to accommodate time-varying requirements. This setup would test BO's adaptability under time-dependent constraints, necessitating enhancements in algorithm stability Pillac et al. (2013).

- **Variable Locations**: When locations are not fixed (e.g., mobile service units or pop-up retail), VRP must adapt as new points are introduced or existing points are removed. This added complexity would require the algorithm to dynamically adjust the solution to the evolving problem parameters, making it more relevant for practical applications Gendreau and Potvin (2011).

Addressing dynamic VRP scenarios would involve exploring real-time optimization strategies and adaptive algorithms, broadening the application potential of BO and GA in unpredictable and practical settings. Such developments would result in more resilient and flexible logistics solutions.

5.3 Optimizing Code with Parallel Computing

While BO provides high-quality solutions, its computational complexity due to intricate social dynamics and mating strategies poses a challenge. Future research could address this through parallel computing techniques, potentially improving the feasibility of BO for time-sensitive applications. Potential approaches include:

- **Parallelizing Fitness Evaluation**: Each BO iteration involves evaluating the fitness of multiple individuals, which can be distributed across multiple cores or GPUs to significantly reduce computation time, especially for larger populations Alba (2002).
- **Parallel Execution of Algorithm Phases**: The Positive and Negative Phases in BO could be executed concurrently, allowing both exploration and exploitation to progress simultaneously. This approach could extend to concurrent subgrouping and mating calculations, enhancing efficiency Collet et al. (2002).

- **Distributed Computing for Large-Scale Problems**: For large VRP instances, distributed computing frameworks (e.g., Spark or MPI) could be employed to manage population optimization across multiple machines, enabling BO to scale to massive problem sizes beyond single-machine capabilities Verma et al. (2015).

Parallel computing presents a significant opportunity to reduce BO's execution time, making it more suitable for real-time applications. Implementing these parallelization strategies would provide valuable insights into BO's computational efficiency and expand its potential use cases.

5.4 Further Research on BO and Comparative Studies

BO is a relatively novel algorithm, and there are several avenues for enhancing its performance and applicability. Future research could focus on refining BO's adaptive dynamics and social grouping mechanisms to make it more efficient and versatile. Additionally, systematic comparative studies with other state-of-the-art algorithms could yield insights into its relative strengths and weaknesses. Potential directions include:

- **Improving Social Dynamics in BO**: Exploring alternative social and mating strategies to reduce computational overhead while maintaining solution quality. For instance, experimenting with subgrouping and mating criteria could streamline the optimization process Waal (1995).
- **Comparative Studies with State-of-the-Art Algorithms**: Benchmarking BO against other state-of-the-art algorithms, such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Differential Evolution

(DE), would provide a clearer understanding of where BO excels and where improvements are needed. Such studies could establish BO's position within the broader optimization field Kennedy and Eberhart (1995); Dorigo and Di Caro (1997).

- **Hybridization of BO with Other Algorithms**: Combining BO with elements of other optimization techniques, such as PSO or GA, could create hybrid algorithms that capitalize on the strengths of multiple methods. This hybridization could potentially improve solution quality and convergence rates in complex optimization problems Yang and et al. (2010).

5.5 Conclusion

In summary, BO has demonstrated more effective results in solving the Vehicle Routing Problem in a controlled setup. However, there is significant potential to extend and refine this work. Future studies could focus on scaling the algorithms to handle more complex problem variations, addressing dynamic conditions, leveraging parallel computing for enhanced efficiency, and further enhancing BO's algorithmic design. By conducting comparative studies with other state-of-the-art algorithms and exploring hybridization possibilities, this line of research could yield valuable insights and advancements in optimization for VRP and other logistics applications.

The advancements suggested here could broaden the applicability of BO, providing solutions to more dynamic, large-scale, and computationally intensive VRP problems. These efforts would not only enhance the theoretical understanding of the algorithm but also improve their practical relevance in the fields of logistics and optimization.

Bibliography

Adam, H., Heidari, A. A., Aljarah, I., Faris, H., and Chen, H. (2019). A review of no free lunch theorem: Critical factors and trends in modern optimization. In *Nature-Inspired Computation in Engineering*. Springer.

Alba, E. (2002). Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462.

Baker, E. and Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800.

Bean, J. and Hadj-Alouane, A. (1992). A genetic algorithm for the multiple-choice integer program. In *TR92-50, Department of Industrial and Operations Engineering, The University of Michigan*.

Collet, P., Lutton, E., and Raynal, F. (2002). A parallel genetic programming toolbox applied to an autonomous robotics navigation problem. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 101–106. ACM.

Das, A. K. and Pratihar, D. K. (2018). A directional crossover (dx) operator for real parameter optimization using genetic algorithm. *Applied Intelligence*.

Das, A. K. and Pratihar, D. K. (2019). A new bonobo optimizer (bo) for real-parameter optimization. In *2019 IEEE Region 10 Symposium (TENSYMP)*, pages 108–113.

Deb, K. (1995). Optimization for engineering design. *Prentice-Hall of India Private Limited*, 5:139–148.

Dorigo, M. and Di Caro, G. (1997). Ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 27(1):29–41.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.

Eiben, A. and Smith, J. (2003). *Introduction to Evolutionary Computing*. Springer, Berlin, Heidelberg.

Gendreau, M. and Potvin, J.-Y. (2011). Dynamic vehicle routing and dispatching. *Handbook of Logistics and Supply Chain Management*, pages 439–464.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Goldberg, D. and Deb, K. (1991). A comparison of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA.

Joines, J. and Houck, C. (1994). On the use of non-stationary penalty functions to solve non-linear constrained optimization problems with gas. In *IEEE Conference on Evolutionary Computation*, pages 579–584.

Kennedy, J. and Eberhart, R. (1995). *Particle swarm optimization*. IEEE International Conference on Neural Networks.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358.

Oliver, J., Smith, D., and Holland, J. (1987). A study of permutation crossover operators on the travelling salesman problem. In *Proc. of the Second International Conference on Genetic Algorithms*, pages 224–230.

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.

Salhi, S. and Fraga, E. S. (2004). Nature-inspired optimization approaches in chemical engineering. *Computers & Chemical Engineering*, 28(6):1069–1088.

Spears, W. and De Jong, K. (1991). On the virtues of parameterized uniform crossover. In *Proc. of the Fourth International Conference on Genetic Algorithms*, pages 230–236. Morgan Kaufmann.

Spears, W. and De Jong, K. (1992). Crossover or mutation? In *Foundations of Genetic Algorithms Workshop*, pages 221–237.

Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics.

Verma, A., Pant, M., and Abraham, A. (2015). Efficient parallelization of particle swarm optimization using apache spark. *IEEE Transactions on Parallel and Distributed Systems*, 28(2):447–455.

Waal, F. B. M. D. (1995). Bonobo sex and society. *Scientific American*, 272:82–88.

Wang, J. and Li, J. (2023). An improved genetic algorithm for vehicle routing problem with time windows considering temporal-spatial distance. In *Artificial Intelligence in China*. Springer.

Whitley, D., Starkweather, T., and Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. In *Parallel Computing*, volume 14, page 347–361.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Yang, X.-S. (2010a). Nature-inspired metaheuristic algorithms. *Luniver Press*, 1.

Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm. In González, J., Pelta, D., Cruz, C., Terrazas, G., and Krasnogor, N., editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74. Springer.

Yang, X.-S. and et al. (2010). *Hybrid metaheuristics*. Springer Science & Business Media.