A META-HEURISTIC ALGORITHM FOR VEHICLE ROUTING PROBLEM WITH INTERDICTION

A META-HEURISTIC ALGORITHM FOR VEHICLE ROUTING PROBLEM WITH INTERDICTION

BY

CARLOS HINRICHSEN PICAND, B.Eng., MBA

A THESIS

SUBMITTED TO THE SCHOOL OF COMPUTATIONAL SCIENCE AND ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2024) McMaster University

(Computational Science and Engineering) Hamilton, Ontario, Canada

TITLE: A Meta-Heuristic Algorithm for Vehicle Routing Problem with Interdiction

AUTHOR: Carlos Hinrichsen Picand

B.Eng. (Industrial Engineering),

Universidad de los Andes, Santiago, Chile

MBA (Management Science),

Universidad de Chile, Santiago, Chile

SUPERVISOR: Dr. Kai Huang

NUMBER OF PAGES: xiv, 165

Abstract

This thesis addresses the Vehicle Routing Problem with Interdiction (VRPI), an extension of the classic Vehicle Routing Problem (VRP) that incorporates the risk of route interdiction due to events such as natural disasters, armed conflicts, and infrastructural failures, among others. These interdictions introduce uncertainty and complexity into logistics planning, requiring innovative approaches to the routing process. This research employs both exact methods, using the CPLEX solver, and heuristic methods, particularly using the Greedy Randomized Adaptive Search Procedure (GRASP), to solve VRPI with different instance sizes.

This research's key contributions include successfully implementing the GRASP algorithm on large-scale benchmark instances, representing a significant advancement over prior implementations that focused on smaller, randomly generated instances. A flexible framework was also developed to adapt the GRASP methodology for different VRP variants, including the Capacitated Vehicle Routing Problem (CVRP) and Split Delivery Vehicle Routing Problem (SDVRP), with and without interdiction.

A feasibility analysis for small instances was developed using CPLEX, highlighting the sensitivity of VRPI solutions to interdiction probabilities, particularly in scenarios with

tight capacity constraints. The findings of this analysis are extended to large instances.

Additionally, a 3-fold logic was incorporated in the GRASP implementation—focused on minimizing cost, minimizing interdiction, and minimizing demand—proved critical in facing the VRPI challenges, and provided high-quality solutions with reduced computational effort. Including the minimum demand logic in GRASP was instrumental during the implementation and numerical experimentation for large benchmark instances.

The implications of this thesis are significant for operational research (OR), particularly in high-risk environments where route interdictions can occur. Future research directions include generating more diverse benchmark instances for VRPI, exploring the impact of variability in interdiction probabilities on solution quality and computational time, and applying exact methods like dynamic programming to solve large VRP instances.

Acknowledgements

I want to express my sincere gratitude to my supervisor, Dr. Kai Huang, for his invaluable guidance, support, and encouragement throughout my research. His insights, feedback, and patience were fundamental in developing this work. His guidance has been a critical element of my academic journey in both good times and challenging moments.

I also want to extend my deepest thanks to my wife and children. Without their support and patience, this work would not have been possible. Thank you for being with me every step of the way.

Contents

List of Figures

List of Tables

Notation, and Abbreviations

Notation

V	Set of nodes
A	Set of arcs/directed edges
N	Number of nodes
$\{0\}$	Depot node
i	Indices for demand nodes $i = 1, \ldots, N$
ij	Indices for arcs $ij \in A$
d_i	Node demands $i = 1, \ldots, N$
K	Number of available vehicles
Q	Vehicle capacity
q_{ij}	Probability of interdiction $ij \in A$
p_{ij}	Probability of no interdiction $ij \in A$

Ω	Set of feasible routes
r	A feasible route originating from the depot
c_r	cost of route r
ϕ_{ir}	probability of reaching node i in r
y_{ir}	amount of supply delivered to i in route r
x_r	route r is selected $x_r = \{0,1\}$
a_{ir}	node i is visited by route r $a_{ir} = \{0,1\}$
b_{ijr}	route r uses arc (i,j) $b_{ijr} = \{0,1\}$
y_{ir}	amount delivered to i in r

Abbreviations

CPLEX	IBM/ILOG Optimization Software
CVRP	Capacitated Vehicle Routing Problem
GRASP	Greedy Random Adaptive Search Procedure
GRASP-H	Hinrichsen's GRASP Implementation
GRASP-X	Xu's GRASP Implementation
KNN	k-Nearest Neighbors algorithm

NP-Hard	Non-deterministic Polynomial-time hard
OR	Operational Research
RCL	Restricted Candidate List
SD	Standard Deviation
SDVRP	Split Delivery Vehicle Routing Problem
SVRP	Stochastic Vehicle Routing Problem
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
VRPI	Vehicle Routing Problem with Interdiction
VRPI-1	Vehicle Routing Problem with Interdiction for Cost Minimization

Chapter 1

Introduction

In the Vehicle Routing Problem (VRP), interdiction refers to events that make a transportation route unusable or unsafe, potentially disrupting the delivery process. This concept adds complexity to the classic VRP, similar to the challenges presented by the Split Delivery Vehicle Routing Problem (SDVRP), where multiple vehicles may deliver to a single customer (Archetti and Speranza, 2008).

Examples of interdiction in the context of VRP include:

1. Natural Disasters: A delivery truck en route may be stopped by an earthquake, causing road damage. This event prevents the truck from continuing its journey, blocking supplies from reaching their destination (BBC News, 2016; The Japan Times, 2011).

2. Armed Conflicts: Humanitarian aid vehicles transporting supplies to a conflict zone may face interdiction if routes are closed due to active combat or hazards, forcing vehicles to turn back or seek alternative routes (Al Jazeera, 2018; United Nations, 2020).

3. Infrastructural Failures: The sudden collapse of a bridge on a critical supply route due to poor maintenance, disrupting the supply chain (The Guardian, 2018; BBC News, 2019).

4. Political or Social Events: Delivery routes passing through areas with protests, strikes, or blockades may be interdicted, affecting delivery schedules (CNN, 2019; The New York Times, 2021).

5. Legal Restrictions: New regulations, such as prohibiting heavy vehicles on certain roads or imposing curfews, can cause disruptions (Reuters, 2020; Government of the United Kingdom, 2020).

The uncertainty and its impact on the routing problem are common in these examples, which requires the creation of contingency plans and adaptive routing strategies. In the Vehicle Routing Problem with Interdiction (VRPI), modeling these scenarios involves incorporating the probability of interdiction into route development.

It is important to note that the VRPI has not the same characteristics of the Stochastic Vehicle Routing Problem (SVRP). In VRPI, uncertainty arises from the network's availability due to interdictions, and the routing process must design resilient routes that can serve customers efficiently considering the risk of network disruptions, and minimizing the traveling cost. On the other hand, SVRP uncertainty arises from operational parameters (e.g., customer demand, travel times, among others), and the routing process must design flexible routes that minimize the expected cost of handling random demand or travel time variability.

1.1 Michael Xu's Thesis (2017)

In the M.Sc. thesis of Michael Xu (Xu, 2017), the VRPI problem is investigated, which is an extension of the traditional VRP, introduced by Dantzig and Ramser (1959) in "The Truck Dispatching Problem". Xu's research is especially relevant in humanitarian logistics and military applications, where route stability and reliability are critical due to potential disruptions.

The thesis begins by exploring the VRP, which focuses on optimizing vehicle routes to meet customer demands efficiently. The introduction of the VRPI concept addresses the need to consider route interdiction—the potential for routes to become unusable due to events such as natural disasters, armed conflicts, infrastructural failures, among others.

Xu offers a comprehensive review of the VRP, including variations, such as the CVRP. In the VRPI model, each route in the network is associated with a probability of interdiction. This unpredictability creates challenges in the planning process, as vehicles and their cargo may be lost if a route is interdicted. The thesis explains how the probability of interdiction for each route is determined using pre-defined data or estimations and how this influences routing decisions.

The thesis highlights the computational complexity of VRPI problems, classifying them as NP-hard, and examines analytical properties relevant to VRPI's optimal solutions. To address these challenges, it uses a heuristic algorithm validated through numerical studies, demonstrating its effectiveness in particular scenarios for small instances. Xu's

research is particularly relevant in scenarios with route uncertainty considering small-sized problems. Examples include disaster relief operations, where natural calamities may block routes, and military logistics, where hostile actions could compromise routes.

Michael Xu's thesis on VRPI significantly enhances the understanding of vehicle routing challenges in uncertain environments. By incorporating route interdiction into VRP, the research introduces new strategies for managing routing under uncertain environments.

1.1.1 Vehicle Routing Problem with Interdiction

Michael Xu's thesis on the VRPI addresses two optimization problems and uses the metaheuristic Greedy Randomized Adaptive Search Procedure (GRASP) and CPLEX software to solve them. This thesis focuses specifically on one of these optimization problems: the cost minimization. We will define **GRASP-X** when referring to Xu's GRASP implementation for the cost minimization problem, and **GRASP-H** to this thesis implementation for the cost minimization problem.

Before reviewing the problem formulation, it is important to note the reasons why we chose GRASP as the metaheuristic to solve the VRPI. We selected the GRASP metaheuristic for several key reasons. First, GRASP is a well-suited and widely-used metaheuristic for solving vehicle routing problems due to its flexibility and ease of implementation. Second, the algorithm is relatively easy to understand, as it operates through a straightforward two-phase process: the construction phase, where an initial solution

is built, followed by the local search phase, where the solution is refined. Finally, the VRPI posed significant challenges when attempting to scale for large instances. It became clear that we needed to compare our implementation against the first GRASP implementation to solve the VRPI made by Xu (2017), which was developed for small instances and thus more basic in design. Xu's implementation did not consider the complexities of scaling, making it a suitable baseline for comparison in this study.

A summary of notation is given in Table 1.1. Let us break down each aspect:

Table 1.1: Notations for the VRPI (Xu, 2017)

Indices	
0	index of the depot.
i	index of demand nodes (customers); $i \in V = \{1, \ldots, N\}$.
(i, j)	index of arcs; $(i, j) \in A$.
r, r_k	index of routes; $r, r_k \in \Omega$.
Parameters	
d_i	demand of node i.
c_r	transportation cost of route r.
ϕ_{ir}	probability of arriving at node i without interdiction in route r.
a_{ir}	$a_{ir} = 1$ if node i is in route r; otherwise $a_{ir} = 0$.
K	number of vehicles.
Q	vehicle capacity.
Decision Variables	
y_{ir}	amount of supply delivered to node i in route r.
x_r	$x_r = 1$ if route r is selected; otherwise $x_r = 0$.

Minimization of Total Travel Cost:

In scenarios where total supply exceeds total demand, the focus is on minimizing overall transportation costs. This involves identifying the most cost-effective routes for vehicles, considering the probability of interdiction and its impact on the routing and supply delivery. The cost can be measured in monetary value, distance traveled, or time taken. The main goal is to develop a routing plan that minimizes these costs while meeting demand, considering the risk of route interdictions.

The optimization formulation is shown below:

$$\min \quad \sum_{r\in\Omega} c_r x_r \tag{1.1.1}$$

$$\text{s.t.} \quad \sum_{r\in\Omega} x_r \le K \tag{1.1.2}$$

$$\sum_{i\in V} y_{ir} \le Q \qquad \forall r \in \Omega \tag{1.1.3}$$

$$y_{ir} \le Q a_{ir} x_r \qquad \forall i \in V, \forall r \in \Omega \tag{1.1.4}$$

$$\sum_{r\in\Omega} \phi_{ir} y_{ir} \ge d_i \qquad \forall i \in V \tag{1.1.5}$$

$$x_r \in \{0,1\} \qquad \forall r \in \Omega \tag{1.1.6}$$

$$y_{ir} \ge 0 \qquad \forall i \in V, \forall r \in \Omega \tag{1.1.7}$$

The Cost Minimization problem formulation in Michael Xu's thesis on the VRPI (VRPI-1) aims to minimize the total cost of delivering supplies, given that the total available supply exceeds the total demand.

The objective (1.1.1) aims to minimize the total cost of all selected routes, where Ω represents the set of all feasible routes. Constraint (1.1.2) limits the maximum number of vehicles used. Constraint (1.1.3) ensures that the total supplies on a vehicle do not exceed its capacity Q. Constraint (1.1.4) stipulates that a route can have a positive delivery only if selected. Constraint (1.1.5) ensures that the expected amount delivered to each node meets its demand. Finally, decision variables x_r must be binary, and y_{ir} must be nonnegative, as specified in (1.1.6) and (1.1.7).

In this formulation, the problem focuses on selecting routes and determining the supply to transport on each route to minimize overall transportation costs. The constraints ensure that the solution respects fleet limitations and vehicle capacities while meeting demands of each customer. The complexity arises when incorporating route interdictions, requiring problem-specific modeling and solution techniques to manage this uncertainty.

1.2 Michael Xu's Main Results and Conclusions

The main results and conclusions of Michael Xu's thesis on the VRPI are as follows:

1. General Framework for VRPI:

   - Xu establishes a comprehensive analytical framework for VRPI, extending the foundational principles of Vehicle Routing Problems. This framework is fundamental for understanding and addressing the complexities of routing in environments where routes can be interdicted

- The VRPI models are designed for high-risk areas, such as conflict zones or disaster-stricken regions, where logistics operations are critically impacted by the likelihood of route interdiction

2. Long-term Performance Focus:

- A notable innovation in Xu's approach is optimizing the long-term performance of routing plans. This strategy includes planning for excess supply delivery in specific periods to build a safety stock at nodes, thereby mitigating the impact of future uncertainties and ensuring continuity of supply even when routes are interdicted

3. Properties of VRPI:

3.1 Violation of Triangle Inequality Due to Interdiction: In classical routing problems, the triangle inequality—where the direct path between two points is always the shortest—often holds, simplifying route optimization. However, in VRPI, the probability of interdiction disrupts this assumption. The triangle inequality might not apply when considering a route's effective "cost". Detouring to avoid high-risk interdiction areas could result in a "shorter" or more successful overall path despite being longer in physical distance. This phenomenon adds complexity to route optimization, as direct paths may not always be the most effective

3.2 Adaptation to Interdiction Risks: The optimal structures in VRPI adapt to the risk of interdiction by permitting routes that share multiple demand nodes. This flexibility in

route planning is crucial for navigating areas where specific paths are more likely to be blocked

3.3 Introduction of Detour Nodes: Detour nodes become a strategic component in VRPI, allowing vehicles to travel through nodes without delivering supplies in order to bypass high-risk interdiction areas. This strategy focuses not only on avoiding risk but also on maintaining the reliability of supply chains under uncertain conditions

3.4 Multiple Shared Demand Nodes: Unlike in SDVRP, where optimal routes are designed to have at most one shared demand node, VRPI accommodates solutions where routes may share multiple demand nodes. This property reflects the complex strategic considerations required to navigate the added uncertainty of route interdictions. Nevertheless, it remains true that there is an optimal solution where only one route will deliver to multiple nodes

4. Strategic Use of Detours and Multiple Route Utilization:

- Detours as Strategic Necessities: The use of detour points in routing decisions reflects a strategic adaptation to the risk of interdiction, highlighting the extensive planning needed to navigate uncertainties in route availability effectively
- Multiple Utilizations of Routes: VRPI models show that optimal strategies may involve using the same route multiple times to meet demand nodes' requirements. This highlights the need to balance route efficiency with the probability of successful delivery when facing potential interdictions

5. Efficiency of the Proposed Algorithm: Xu introduces a novel two-phase GRASP meta-heuristic methodology. This algorithm, specifically designed for solving VRPI models, is shown through the thesis to handle small instances of VRPI with success efficiently.

6. Performance Comparison with CPLEX: Compared to the commercial solver CPLEX, Xu's algorithm demonstrated greater efficiency, maintaining an optimality gap within 4% for VRPI on small generated instances.

1.3 Contributions of this thesis

In this thesis, seven main contributions were developed to solve the VRPI successfully:

1. Routes Creation Algorithm:

The testing of benchmark instances will have some limitations due to Xu's optimization model, which aims to obtain an exact solution. The model requires the creation of all feasible routes beforehand, resulting in an exponential increase in the number of routes and computational time as the number of customers grows. Consequently, Xu (2017) used only 5, 6, and 7 customers for numerical experiments, significantly different from the 22 to 100 customers represented by the benchmark instances that better reflect real logistic scenarios.

The GRASP-X algorithm generates an extensive initial set of routes. Then, it filters them

by selecting the first K routes (equal to the number of vehicles) with the highest probability of not being interdicted. This step is computationally expensive for large instances, and the highest probability of not being interdicted does not necessarily yield a better solution, as will be explained in Chapter 3. Therefore, we will create experiments with randomly generated instances for 3, 4, 5, 6, and 7 customers, with and without interdiction, and compare the total cost and computational time for the exact method using CPLEX and GRASP metaheuristics. This step will calibrate the metaheuristic for benchmark instances, where different parameters will be evaluated. Since obtaining an exact solution with the model proposed by Xu (2017) is not computationally feasible, the benchmark instances will be tested only using GRASP with and without interdiction. The computational time and cost will then be computed and compared with the optimal values provided by the benchmark instances bibliography. GRASP-H will use the K-Nearest Neighbors (KNN) algorithm to generate the initial feasible set of routes.

2. Local Search Algorithm Enhancement: Xu's GRASP-X algorithm selects feasible routes

during the local search procedure using a 2-fold logic: choose a route with a lower overall cost or a route with the highest probability of not being interdicted. As shown in Chapter 3, this algorithm does not necessarily identify the best feasible routes. Therefore, GRASP-H will employ a 3-fold logic. In addition to considering cost and the highest probability of not being interdicted, the third aspect of the algorithm evaluates routes with the highest chance of servicing the minimum demand under interdiction. This approach helps prevent saturating the entire network when seeking a feasible solution, as tightness is a critical condition of the problem.

3. Fleet Unnecessary Capacity Utilization: Xu's GRASP-X utilizes the entire fleet capacity when solving VRPI, even oversupplying customers when unnecessary. Over-utilizing vehicle capacity leads to increased transportation costs without added value. Excessive deliveries result in higher fuel consumption, driver labor costs, and vehicle wear and tear. GRASP-H addresses this issue by prioritizing routes with the highest chance of meeting the minimum required demand by customers under interdiction, thus avoiding these indirect costs. For this research, these indirect costs are not measured beyond the travel costs.

4. Parameter Calibration:

Unlike Xu's GRASP-X, parameter calibration is implemented in GRASP-H. Parameter calibration is crucial when using metaheuristics to solve complex optimization problems like VRP, SDVRP, and VRPI. The main advantage of parameter calibration in solving VRPI is the significant improvement in solution quality and computational efficiency. Proper calibration enables the metaheuristic algorithm to find optimal or near-optimal solutions by effectively balancing exploration and exploitation, even when considering interdiction probabilities.

5. GRASP Flexible Framework Development with and without Interdiction:

The development of GRASP to solve the VRPI provides a flexible framework based on parameters. This framework allows users to obtain solutions for various vehicle routing problem variants, with and without interdiction, such as CVRP and SDVRP. Using the same code script, users can select “no interdiction" and “no splits" to solve CVRP,

or “no interdiction" and “splits" to solve SDVRP. If the user selects “interdiction," the metaheuristics will solve VRPI. This flexible framework enables straightforward testing of different benchmark instances for various vehicle routing problem variants in one step. It enhances understanding of GRASP's behavior under different parameter setups, representing a significant advance over Xu (2017).

The user can also evaluate different instances using the previously described 3-fold logic or opt for minimum cost logic (for no interdiction scenarios), minimum interdiction probability, or minimum demand. Additionally, users can choose different local search procedures such as 2-opt, 3-opt, segment-reversal or swap-local search, or select them randomly in each iteration.

6. Problem Infeasibility Analysis:

This thesis addresses a critical aspect of the VRPI: the sensitivity of the problem to interdiction probabilities, especially in scenarios where total demand is very close to the fleet's capacity. Xu (2017) did not fully account for the feasibility challenges posed by real-world scenarios, where balancing total demand and available supply is crucial. In this thesis, we analyze the theoretical feasibility of small-instance scenarios and then understand their implications in large-scale instances and GRASP's capability to identify potential infeasibility scenarios.

This research emphasizes the importance of considering the probability of interdiction in addition to demand and fleet capacity. This approach acknowledges the potential for infeasibility in tightly constrained scenarios. It contributes to a deeper understanding of VRPI's complexities, mainly where demand and supply constraints are critically

balanced.

7. GRASP Implementation for Large-Scale Benchmark Instances:

This thesis applies the GRASP metaheuristic to the VRPI and tests it on large benchmark instances. This approach represents a significant advancement over Michael Xu's methodology, which employed the GRASP metaheuristic only on small, randomly generated instances with 5, 6, and 7 demand nodes or customers.

Using established benchmark instances for testing is a crucial improvement. Benchmark instances, often derived from real-world data or standardized test sets, provide a common ground for evaluating and comparing algorithmic performance. They are essential for validating the effectiveness and generalization of the approach across different scenarios and problem sizes. This contrasts with Xu's use of small, randomly generated instances, which, while helpful, may not adequately represent the range of challenges encountered in practical applications of VRPI.

8. Comprehensive Set of VRPI Examples:

In this thesis, a more comprehensive set of examples was created to give to the reader better tools to understand how VRPI works, and how to incorporate interdiction probabilities in the problem constraints, including all numerical calculations. The set of examples in Xu (2017) was limited and not self-explanatory.

9. Availability of Models Developed:

All models developed in this thesis (using CPLEX and GRASP), are available upon request. Information about how to get access can be found in Numerical Experiments

chapter.

Overall, these enhancements encourage a deeper understanding of VRPI solution strategies and strengthen the validity and applicability of research findings by employing standardized benchmark tests. This approach could lead to more robust, efficient, and applicable solutions for the uncertain scenarios found in the VRPI.

Chapter 2

Literature Review

In computational problem-solving, metaheuristics are critical, offering solutions to complex problems. This chapter explores an overview of metaheuristics, particularly GRASP, including its definition, principles, and reasons for its adoption across various domains (Juan *et al.*, 2015).

Metaheuristics, derived from the Greek words “meta" meaning beyond and “heuriskein" meaning to find or discover, are high-level algorithmic frameworks that provide guidelines or strategies for developing heuristic optimization algorithms (Sörensen and Glover, 2013). Unlike classic optimization methods tailored for specific problems, metaheuristics are designed to be adaptable and flexible, suitable for a wide range of problems under different domains (Talbi, 2009).

The main difference between heuristics and metaheuristics lies in scope and flexibility. Heuristics are problem-specific algorithms, and are designed to find good solutions fast by using, for example, rules of thumb based on the structure of the problem. The main

impact of this approach is that a heuristic will explore a smaller solution space. On the other hand, Metaheuristics are high-level and general-purpose frameworks that can be used to solve a variety of problems, by using iterative and randomized search strategies to explore a larger search space, trying to avoid local optima. This difference is fundamental when solving complex routing problems, because heuristics may struggle with large-scale, while metaheuristics are better suited for finding higher-quality solutions in such cases. This is the main reason why a metaheuristic was implemented to solve the VRPI. Here's a quick comparison:

Aspect	**Heuristic**	**Metaheuristic**
Scope	Problem-specific, designed for a particular type of problem.	General-purpose, can be applied to a wide range of problems.
Solution Quality	Finds good enough solutions but often not optimal.	Capable of finding high-quality (near-optimal) solutions.
Search Space	Explores a smaller portion of the solution space, often locally.	Explores a larger solution space, globally and locally.
Flexibility	Limited to specific problem structures.	Flexible and can handle various constraints and hybrid strategies.
Speed	Typically faster but may get stuck in local optima.	Slower than heuristics but capable of better solutions.
Randomness	Often deterministic and makes greedy decisions.	Uses randomization to escape local optima and explore new areas.
Applicability	Efficient for small to medium-sized problems.	Suitable for large, complex, or highly constrained problems.

Table 2.1: Comparison of Heuristics and Metaheuristics

The essence of metaheuristics is their ability to explore and exploit the search space efficiently. They achieve this by balancing intensification (local search) and diversification (global search), trying to avoid local optima, and increasing the likelihood of finding near-optimal or optimal solutions (Blum and Roli, 2003).

The complexity and nature of real-world problems define the use of metaheuristic.

Many problems in logistics, engineering, economics, and computer science are NP-hard, meaning they cannot be solved optimally within polynomial time as the problem size increases (Gary and Johnson, 1979). Metaheuristics provide an approach to finding sufficiently good solutions within a reasonable timeframe (Koopmans and Beckmann, 1957).

Additionally, metaheuristics' flexibility makes them highly attractive. They can be adapted to different problems and constraints, which is not always possible with traditional optimization methods. This adaptability is crucial in real-world scenarios, where problems are often not well-defined and can change over time (Osman and Laporte, 1996).

Metaheuristics offers versatile and practical solutions to different kinds of problems, particularly considering their adaptive nature, ability to handle complexity, and applicability across various domains. As computational challenges increase, the role of metaheuristics will become more integral to finding solutions to complex problems.

2.1 Metaheuristics for Operational Research

Operational Research (OR) is a discipline focused on applying analytical methods to improve the decision-making process. Within OR, metaheuristics have emerged as a powerful tool for solving complex and computationally challenging problems (Bianchi *et al.*, 2009).

Metaheuristics have become fundamental in OR, as they have provided solutions to previously intractable problems and inspired new lines of research and application. The development of hybrid metaheuristics, has further enhanced their effectiveness and applicability (Blum and Roli, 2008).

Moreover, applying metaheuristics in OR has led to significant cost savings, efficiency improvements, and better decision-making across various sectors. Their role in logistics and supply chain management is increasingly recognized (Juan *et al.*, 2015).

Why Metaheuristics are Crucial in Operational Research?

1. Complexity of Real-World Problems: OR often deals with problems too complex for traditional optimization methods. Metaheuristics provides a flexible and efficient approach to finding suitable solutions for these problems within a reasonable timeframe. They are particularly effective in handling non-linear and high-dimensional problems common in OR (Blum and Roli, 2003; Vidal *et al.*, 2014).

2. Versatility and Adaptability: Unlike problem-specific algorithms, metaheuristics are not tailored to a specific problem type. This versatility allows them to be applied to various OR problems (Osman and Laporte, 1996; Juan *et al.*, 2011).

3. Balancing Exploration and Exploitation: Metaheuristics are designed to balance exploration (searching new areas of the solution space) and exploitation (intensifying the search around promising areas). This balance is critical for trying to avoid local optima and finding near-optimal solutions (Sörensen and Glover, 2013; Talbi, 2009).

As computational capabilities continue to expand, the role of metaheuristics in OR will be more critical, driving efficiency across multiple problems and domains.

2.2 GRASP in Operational Research: A Focus on Vehicle Routing Problem

Metaheuristics like GRASP have gained prominence in OR. This section examines the importance of GRASP in OR, specifically in addressing the VRP, supported by practical examples and academic studies (Toth and Vigo, 2002). In Chapter 4, the principles and definitions of GRASP will be explored in detail, considering metaheuristics in general and their implementations for solving the VRPI.

2.2.1 GRASP

1. Description: GRASP is a multi-phased metaheuristic that combines greedy algorithms with randomized procedures for global optimization. It involves iterative construction and local search procedures to explore and exploit the solution space. GRASP has been effectively used in various combinatorial optimization problems, demonstrating its ability to find high-quality solutions within limited computational time (Feo and Resende, 1989; Resendel and Ribeiro, 2005).

2. Mechanism: Each iteration in GRASP starts with a randomized solution, followed by

an improvement phase using a local search procedure. This process enhances diversification and intensification, essential for trying to avoid local optima (Feo and Resende, 1995; Mladenović *et al.*, 2003).

3. Example: GRASP has been effectively implemented in telecommunications for network design, particularly in maximizing network coverage while minimizing the cost of installation and operation (Feo and Resende, 1989; Resende and Ribeiro, 2010).

2.2.2 Vehicle Routing Problem

1. Description: The VRP is a crucial and extensively studied topic in OR. It involves optimizing the routes of a fleet of vehicles to meet customer demands. The VRP aims to minimize costs—such as travel distance and time—while adhering to constraints like vehicle capacity and customer time windows. Advanced solutions employ techniques ranging from linear and integer programming to heuristic and machine learning methods, reflecting the field's interdisciplinary nature. The problem's complexity and variants, such as the CVRP and SDVRP, underlying its significance in optimizing logistics and distribution systems worldwide.

2. Challenges: The VRP and SDVRP have significant challenges, in particular combinatorial complexity, which grows exponentially with the number of customers, and the need to balance constraints such as vehicle capacity. Dynamic and stochastic elements like traffic and variable demand add further complexity. Additionally, the VRP often involves multiple objectives beyond minimizing costs, including environmental and

customer satisfaction goals. In contrast, SDVRP allows split customer deliveries among different routes. Technological integration, scalability, and the development of efficient heuristic algorithms are crucial for practical implementations.

3. Relevance in Operational Research: VRP and SDVRP represent a class of complex combinatorial optimization problems prevalent in OR, characterized by multiple constraints and objectives. Solving VRP and SDVRP is fundamental in various sectors, including transportation, distribution, and logistics (Drexl and Schneider, 2015).

2.3 Parameter Calibration

Parameter calibration is crucial when using metaheuristics to solve complex optimization problems like the VRP and its variants, such as CVRP and SDVRP. Below is an overview of its importance and how it applies explicitly to GRASP for VRP.

2.3.1 Importance of Parameter Calibration in Meta-Heuristics for Vehicle Routing Problem

1. Performance Optimization: Parameters control the algorithm's behavior, including convergence speed and solution quality. Calibrating these parameters can significantly improve the performance of the metaheuristic, enabling it to find better solutions within a reasonable computational time (Gendreau *et al.*, 2010).

2. Algorithm Robustness: Proper calibration can make the metaheuristic more robust, meaning it can find good solutions across different instances of the problem. This is particularly important in real-world scenarios where problem instances vary significantly (Pisinger and Ropke, 2019).

3. Balancing Exploration and Exploitation: Metaheuristics need to balance exploration (searching new areas of the solution space) and exploitation (refining good areas). Parameters play a critical role in maintaining this balance, and their calibration can ensure that the algorithm does not get stuck in local optima while still efficiently converging to high-quality solutions (Feo and Resende, 1995).

4. Adaptation to Problem Characteristics: VRP characteristics, such as the number of vehicles, delivery locations, and split delivery requirements, can influence the effectiveness of different parameter settings. Calibration allows the algorithm to adapt to these specific problem features, improving solution quality (Golden *et al.*, 2008).

2.3.2 GRASP Meta-Heuristic for Vehicle Routing Problem and Parameter Calibration

GRASP is a multi-start metaheuristic that constructs solutions through a randomized greedy algorithm and then applies local search for improvement. For VRP, calibrating parameters in GRASP is particularly important for several reasons:

1. Construction Phase: The level of greediness and randomness in the construction phase of GRASP is controlled by parameters. Calibrating these parameters affects the

diversity of the initial solutions and the algorithm's ability to explore the solution space effectively (Feo and Resende, 1995).

2. Local Search Intensification: Parameters influencing the local search phase determine how intensively the algorithm exploits the neighborhood of the constructed solutions. Proper calibration balances solution quality and computational effort (Gendreau *et al.*, 2010).

3. Number of Iterations: GRASP iterations directly impact the solution quality and computational time trade-offs. Setting this parameter appropriately ensures the algorithm has enough opportunities to explore and refine solutions without excessive computational cost (Pisinger and Ropke, 2019).

4. Adaptive Mechanisms: GRASP can include adaptive mechanisms where parameters are adjusted dynamically based on the performance of previous iterations. Calibrating these adaptive strategies can enhance the algorithm's performance (Laporte, 2009).

In Chapter 3, the VRPI will be explained through various examples. A vital aspect of this research will be analyzed: the relationship between problem infeasibility, interdiction probabilities, supply, and demand. Additionally, we will examine the implications of choosing routes with a high probability of not being interdicted versus choosing routes with a high chance of meeting the minimum required demand for customers under interdiction. The issue of fleet over-utilization will also be addressed.

Chapter 4 will explore the principles and definitions of GRASP in more detail, considering the GRASP metaheuristic in general and its implementation for solving the VRPI.

Chapter 3

Vehicle Routing Problem with Interdiction

As explained in Chapter 1, interdiction significantly adds significant complexity to the VRP by introducing unpredictable scenarios where transportation routes become unusable or unsafe. This differs from the traditional VRP, which optimizes fleet routes to minimize costs and fulfill customer demand. Interdiction considers events such as natural disasters and infrastructural failures, each capable of disrupting the delivery process along affected routes. This concept is critical OR, particularly in environments where instability can occur in a network. The complexity introduced by interdiction is similar to that in the SDVRP, where multiple deliveries to a single customer via different vehicles are explored. This highlights the need for adaptive and robust routing strategies in the face of uncertainties.

Illustrative examples of interdiction, such as route obstructions due to earthquakes,

armed conflicts, infrastructural collapses, political unrest, or sudden legal restrictions, underline the challenges of maintaining reliable supply chains under uncertain conditions. Each interdiction event highlights the need to create contingency plans and generate flexible routing strategies that adapt to uncertain conditions.

Before going to the examples, four key components need to be assessed when solving the Vehicle Routing Problem with Interdiction:

1. The demand serviced of each customer on each route cannot be more than the vehicle capacity

2. The expected demand serviced of each customer among all routes has to be greater or equal to its required demand

3. A customer can be visited on different routes without servicing any demand to reach another customer. This is due to the interdiction probabilities

4. Cumulative probabilities of interdiction need to be taken into account. For example, suppose the probability of not being interdicted between the depot and customer 1 is 0.9 (90%), and the probability of not being interdicted between customer 1 and customer 2 is 0.8 (80%) in route 0, 1, 2, 0. In that case, the probability of not being interdicted between the depot and customer 1 is the multiplication of the previous probabilities: $0.9 \cdot 0.8 = 0.72$ (72%) due to the events being independent, as defined by Xu (2017)

The following section will explain the VRPI more practically through examples. Additionally, we will examine the implications of choosing routes with a high probability

of not being interdicted versus choosing routes with a high chance of meeting minimum required demand for customers under interdiction, incorporating this logic into GRASP-H. After this section, we will address Xu's issue of fleet over-utilization and explain how it is managed in GRASP-H. Finally, we will analyze the problem of infeasibility considering interdiction probabilities, demand, and supply.

3.1 Examples to understand the Vehicle Routing Problem with Interdiction

Below are two examples from Xu (2017) for the cost minimization problem, which can help you understand the expected results of the VRPI.

Example 1: Let us have a network with five customers and a depot with the following characteristics:

Node	x-coordinate	y-coordinate	Demand
depot (0)	0	0	0
1	-122	132	24
2	-132	-60	24
3	50	108	42
4	-160	62	25
5	-143	-124	21

Table 3.1: VRPI Example (Xu, 2017)

The fleet size is $K = 3$, the vehicle capacity is $Q = 50$, and the network looks like this:

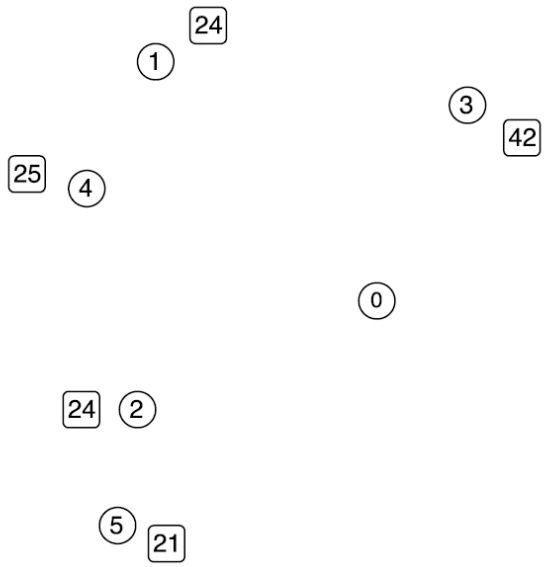

Figure 3.1: VRPI Network Example (Xu, 2017)

First, let us see how tight this network is. The total capacity is $K \cdot Q = 3 \cdot 50 = 150$, and the demand to be serviced is 136. Therefore, the tightness of this network is $\frac{136}{150} = 0.9$ or 90%. For this network, interdiction probabilities were generated between $1\% - 5\%$, resulting in the following matrix:

$$\text{Interdiction Probabilities}: \begin{bmatrix} \sim & 4\% & 3\% & 1\% & 5\% & 1\% \\ 4\% & \sim & 1\% & 2\% & 2\% & 3\% \\ 3\% & 1\% & \sim & 5\% & 5\% & 5\% \\ 1\% & 2\% & 1\% & \sim & 4\% & 3\% \\ 5\% & 2\% & 5\% & 4\% & \sim & 4\% \\ 1\% & 3\% & 5\% & 3\% & 4\% & \sim \end{bmatrix}$$

When solving this problem without interdiction, the optimal value is 1069, yielding the following routes:

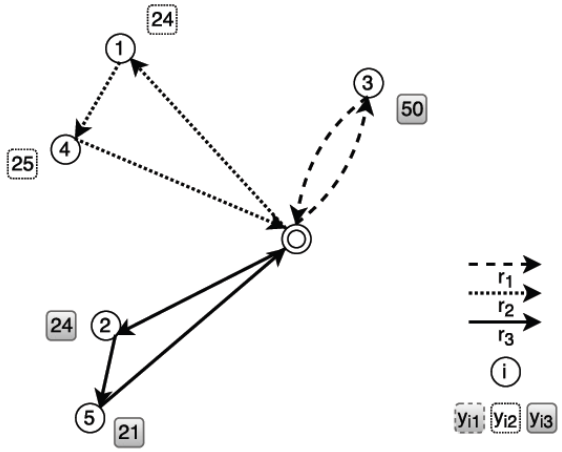

Figure 3.2: Without Interdiction Solution (Xu, 2017)

When solving this problem with interdiction, the optimal value is 1221, yielding the following routes:

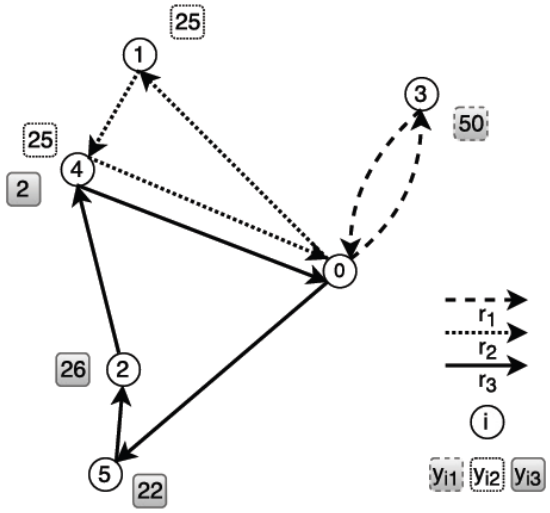

Figure 3.3: With Interdiction Solution (Xu, 2017)

To better understand the solution with interdiction, let us check the critical conditions to be met when solving this problem. First of all, these are the routes and their respective serviced demands:

- $r_1 = \{0,3,0\} : y_{31} = 50$
- $r_2 = \{0,1,4,0\} : y_{12} = 25, y_{42} = 25$
- $r_3 = \{0,5,2,4,0\} : y_{53} = 21.2, y_{23} = 25.5, y_{43} = 1.7$

The critical conditions are the following:

1. The demand serviced by each customer on each route cannot exceed the vehicle capacity. Looking at the solution, this condition is met

2. The expected demand serviced of each customer among all routes has to be greater or equal to its required demand. Let us calculate the expected demand serviced of each customer. We know that ϕ_{ir} is the probability of arriving at node i without interdiction in route r, and y_{ir} is the amount of supply delivered to customer i in route r.

For customer 1, we have the following:

$\phi_{12} \cdot y_{12} \geq d_1 \leftrightarrow 96\% \cdot 25 = 24 \geq 24$

For customer 2, we have the following:

$\phi_{23} \cdot y_{23} \geq d_2 \leftrightarrow 99\% \cdot 95\% \cdot 25.5 \approx 24 \geq 24$

For customer 3, we have the following:

$\phi_{31} \cdot y_{31} \geq d_3 \leftrightarrow 99\% \cdot 50 = 49.5 \geq 42$

For customer 4, we have the following:

$\phi_{42} \cdot y_{42} + \phi_{43} \cdot y_{43} \geq d_4 \leftrightarrow 96\% \cdot 98\% \cdot 25 + 99\% \cdot 95\% \cdot 95\% \cdot 1.7 = 25.0389075 \geq 25$

Finally, for customer 5, we have the following:

$\phi_{53} \cdot y_{53} \geq d_5 \leftrightarrow 99\% \cdot 21.2 \approx 21 \geq 21$

Therefore, all conditions were met.

3. A customer can be visited on different routes without servicing any demand to reach another customer. This is due to the interdiction probabilities. Looking at the solution, this condition was not necessary with the current structure of the interdiction probabilities

Example 2: Let us have a network with 3 customers and a depot. The fleet size is $K = 2$, and the vehicle capacity is $Q = 5$. First, let's see how tight is this network. The total capacity is $K \cdot Q = 2 \cdot 5 = 10$, and the demand to be serviced is 8. Therefore, the tightness of this network is $\frac{8}{10} = 0.8$ or 80%, which is more tight than the one in Example 1. The network looks like this:

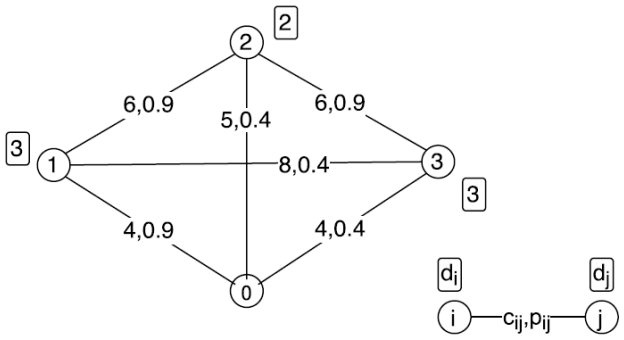

Figure 3.4: VRPI Network Example (Xu, 2017)

When solving this problem with interdiction, the optimal routes are:

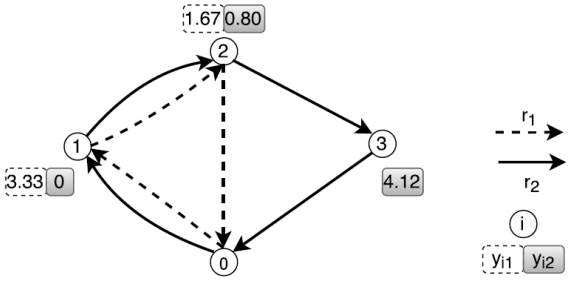

Figure 3.5: With Interdiction Solution (Xu, 2017)

To better understand the solution with interdiction, let us check the critical conditions to be met when solving this problem. First of all, these are the routes and their respective serviced demands:

- $r_1 = \{0,1,2,0\} : y_{11} = 3.3\overline{3}, y_{21} = 1.6\overline{6}$
- $r_2 = \{0,1,2,3,0\} : y_{12} = 0, y_{22} = 0.8, y_{32} = 4.12$

The critical conditions are the following:

1. The demand serviced by each customer on each route can be, at most, the vehicle capacity. Looking at the solution, this condition is met.

2. The expected demand serviced of each customer among all routes has to be greater or equal to its required demand. Let us calculate the expected demand serviced of each customer. We know that ϕ_{ir} is the probability of arriving at node i without interdiction in route r, and y_{ir} is the amount of supply delivered to node i in route r.

For customer 1, we have the following:

$\phi_{11} \cdot y_{11} + \phi_{12} \cdot y_{12} \geq d_1 \leftrightarrow 90\% \cdot 3.3\overline{3} + 90\% \cdot 0 = 3 \geq 3$

For customer 2, we have the following:

$\phi_{21} \cdot y_{21} + \phi_{22} \cdot y_{22} \geq d_2 \leftrightarrow 90\% \cdot 90\% \cdot 1.6\overline{6} + 90\% \cdot 90\% \cdot 0.8 \approx 2 \geq 2$

For customer 3, we have the following:

$\phi_{32} \cdot y_{32} \geq d_3 \leftrightarrow 90\% \cdot 90\% \cdot 90\% \cdot 4.12 = 3.00348 \geq 3$

Therefore, all conditions were met.

3. In different routes, you can visit a customer without servicing any demand, to reach another customer. This is due to the interdiction probabilities. Looking at the solution, in route 2, customer 1 was visited but not serviced to reach customer 2 due to interdiction probabilities

3.2 Minimum Demand Logic

As explained in Chapter 1, Xu (2017) uses a 2-fold logic to create feasible routes after creating an initial set of routes. The algorithm can use a minimum cost logic to find a feasible route or a minimum interdiction logic to find a feasible route. On the other hand, this thesis with GRASP-H uses a 3-fold logic to create feasible routes after creating an initial set of routes. Besides the minimum cost and minimum interdiction logic to find a feasible route, the new proposal adds the minimum demands to be serviced with interdiction logic. Besides this addition, GRASP-H is more exhaustive when checking the feasibility of the routes, not only after the local search improvements but also after the initial sets of routes created using KNN and the restricted candidate list (RCL). Both algorithms can be seen in Figures 3.6 and 3.7:

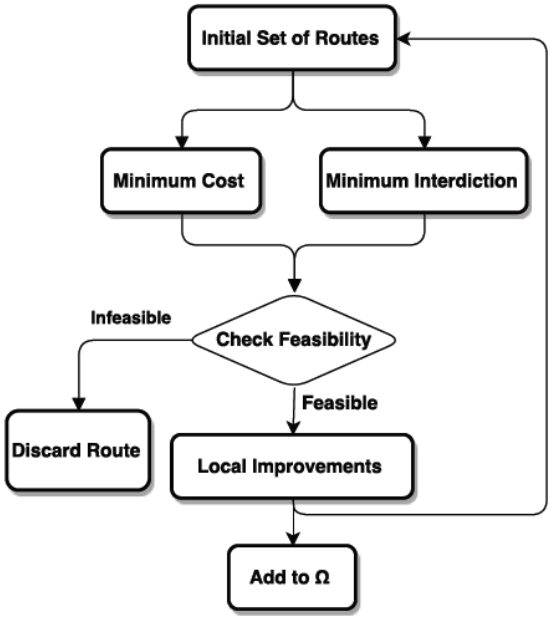

Figure 3.6: GRASP-X Local Search Algorithm (Xu, 2017)

It is important to note that Xu's GRASP representation shows what could happen in each iteration during GRASP-X execution. Additionally, the figure does not show the termination criteria implemented by Xu (2017).

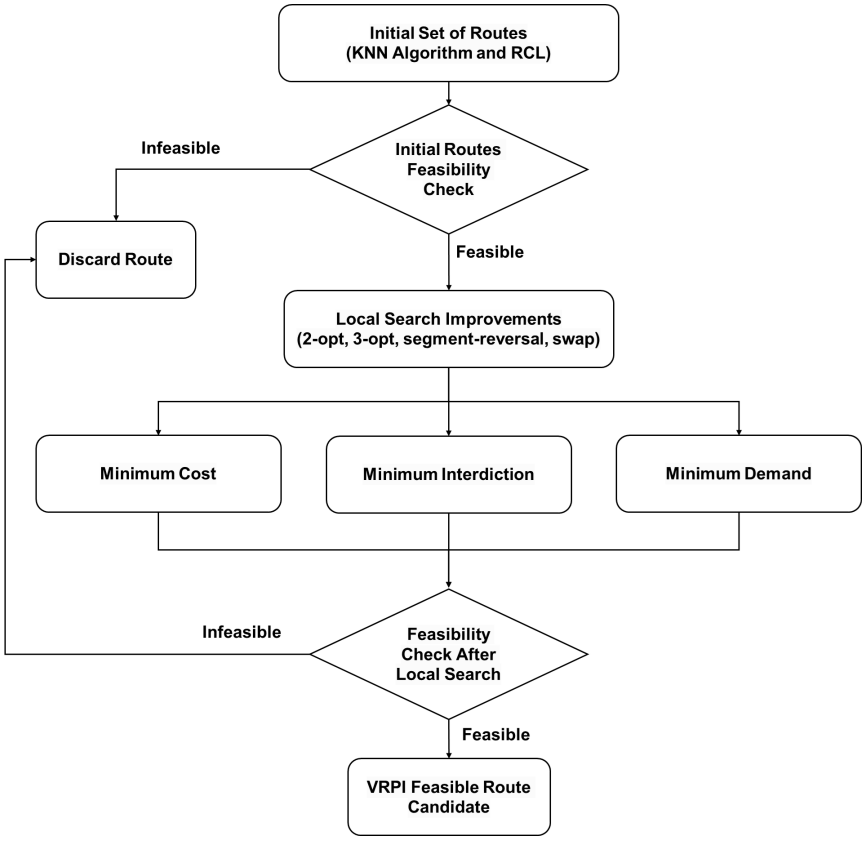

Figure 3.7: GRASP-H Local Search Algorithm

On the other hand, this thesis GRASP representation (GRASP-H) shows what could happen in each iteration during execution. The termination criteria is not being shown too, but basically considers a stagnation level to be set by the user. This means, if we cannot find any feasible solution by a certain number of iterations (set by the user), the code finishes resulting in an Infeasible solution. If at some point we find a feasible solution, and if there is no improvement of the solution by a certain number of iterations (set by the user), the code finishes resulting in the best Feasible solution found. This is the basic logic of the termination criteria, which gives to the user an extra layer of flexibility

to end up the code early.

Let us understand the differences in the 3 logic to create feasible routes (minimum cost, minimum interdiction, and minimum demand). We will explain the differences with a simple example:

Example 3: Let us have a network with 2 customers and a depot, with the following network and demands to be serviced. For simplicity, let us imagine only two potential solutions, as shown below:

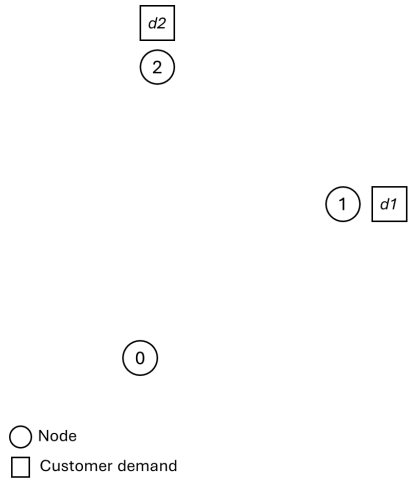

Figure 3.8: Example 3 Network

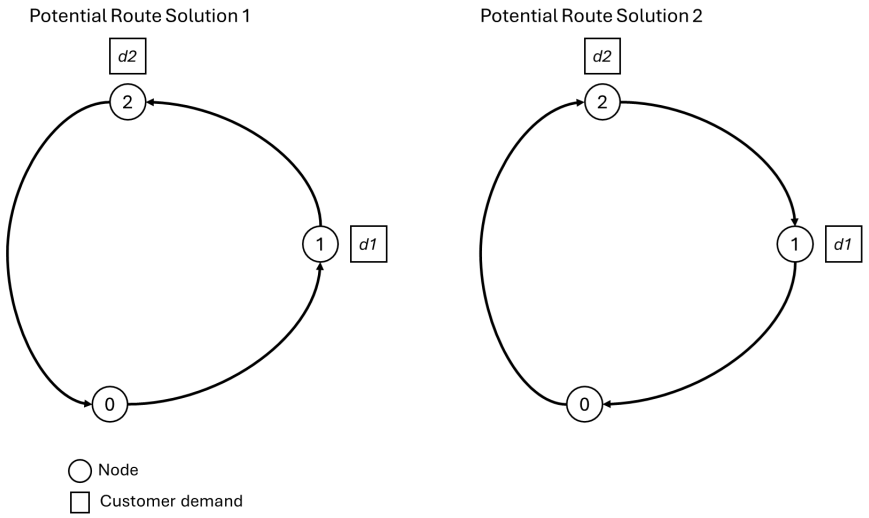

Figure 3.9: Example 3 Network Potential Route Solutions

The fleet size is $K = 1$, and the vehicle capacity is Q. Knowing that q_{ij} is the probability of interdiction between node i and node j (thus $p_{ij} = 1 - q_{ij}$ is the probability of no interdiction between node i and node j). For this network, interdiction probabilities are:

$$\text{Interdiction Probabilities}: \begin{bmatrix} \sim & q_{01} & q_{02} \\ q_{10} & \sim & q_{12} \\ q_{20} & q_{21} & \sim \end{bmatrix}$$

For symmetry, we have $q_{01} = q_{10}$, $q_{02} = q_{20}$, and $q_{12} = q_{21}$. Therefore, the non-interdiction probabilities are:

$$\text{Non-Interdiction Probabilities}: \begin{bmatrix} \sim & p_{01} & p_{02} \\ p_{01} & \sim & p_{12} \\ p_{02} & p_{12} & \sim \end{bmatrix}$$

Finally, the cost matrix is shown below:

$$\text{Cost Matrix}: \begin{bmatrix} \sim & c_{01} & c_{02} \\ c_{01} & \sim & c_{12} \\ c_{02} & c_{12} & \sim \end{bmatrix}$$

The main difference between the potential solution 1 and the potential solution 2 of this network is the direction of the route. To move from r_1 to r_1^*, a 2-opt or segment

reversal local search procedures must be done. These are the routes:

- $r_1 = \{0,1,2,0\}$
- $r_1^* = \{0,2,1,0\}$

Now let us review the 3-fold logic to get a feasible solution:

Minimum Cost

After making the local search procedure, let us calculate the cost of the initial route r_1 and the candidate route r_1^*:

- $r_1 = \{0,1,2,0\} : C_t = c_{01} + c_{12} + c_{20}$
- $r_1^* = \{0,2,1,0\} : C_t^* = c_{02} + c_{21} + c_{10} = C_t$

We can see that in terms of cost, the candidate route has no benefit in comparison to the initial route, yielding the exact cost ($C_t = C_t^*$).

Minimum Interdiction

After making the local search procedure, let us calculate the probability of not being interdicted on each route, starting and ending at the depot. We have to remember that the probability of not being interdicted in each pair of nodes is independent because those interdiction events are independent:

- $r_1 = \{0,1,2,0\} : NI_p = p_{01} * p_{12} * p_{20}$
- $r_1^* = \{0,2,1,0\} : NI_p^* = p_{02} * p_{21} * p_{10} = NI_p$

We can see that, in terms of interdiction, the candidate route has no benefit over the initial route, yielding the same minimum interdiction ($NI_t = NI_t^*$).

The main problem with this approach is that it does not consider the effect of the interdiction probabilities on the serviced demands when performing the 2-opt or segment reversal. Let us see this impact.

Consider that the demand serviced in r_1 for customer 1 is $y_{11} = d_1^{'}$, and for customer 2 is $y_{21} = d_2^{'}$. Due to Xu's GRASP-X methodology, the network (or, in this case, this route) is saturated, meaning that $d_1^{'} + d_2^{'} = Q$. Due to interdiction probabilities, we also have that $d_1 + d_2 \leq d_1^{'} + d_2^{'} = Q$.

On the other, we know that the probability of not being interdicted between the depot and customer 1 is p_{01}, and the probability of not being interdicted between the customer 1 and customer 2 is p_{12}. Therefore, due to interdiction events being independent, the probability of not being interdicted between the depot and customer 2 is the multiplication of the two probabilities described before, $p_{01} \cdot p_{12}$.

Finally, we know that the expected demand to be serviced to customer 1 has to be at least d_1, and the expected demand to be serviced to customer 2 has to be at least d_2. Mathematically speaking, this means for customer 1: $d_1^{'} \cdot p_{01} \geq d_1$. For customer 2, this expression is: $d_2^{'} \cdot p_{01} \cdot p_{12} \geq d_2$. Adding these two expressions together, we have:

$d_1^{'} \cdot p_{01} + d_2^{'} \cdot p_{01} \cdot p_{12} \geq d_1 + d_2$. Knowing that $d_1^{'} + d_2^{'} = Q$ and doing some arithmetic, we have the following expressions that have to be met for the route r_1 to be feasible:

$$d_1^{'} \leq \frac{Q \cdot p_{01} - d_1 - d_2 - Q \cdot p_{01} \cdot (1 - p_{12})}{p_{01} \cdot (1 - p_{12})}$$

$$d_2^{'} \leq \frac{Q \cdot p_{01} - d_1 - d_2}{p_{01} \cdot (1 - p_{12})}$$

If $Q = 10, d_1 = 3, d_2 = 2, p_{01} = p_{10} = 0.9, p_{02} = p_{20} = 0.95$ and $p_{12} = p_{21} = 0.85$, we have that $d_1^{'} \leq 4.81481$ and $d_2^{'} \leq 14.81481$. For this example, let us choose $d_1^{'} = 5.5$ and $d_2^{'} = 4.5$. Let us review if these conditions have met the expected demand to be serviced to each customer:

- Customer 1: $d_1^{'} \cdot p_{01} \geq d_1 \Rightarrow 4.5 \cdot 0.9 = 4.05 \geq d_1 = 4$. Condition met
- Customer 2: $d_2^{'} \cdot p_{01} \cdot p_{12} \geq d_2 \Rightarrow 5.5 \cdot 0.9 \cdot 0.85 = 4.2075 \geq d_2 = 3$. Condition met

After performing the 2-opt or segment reversal local search procedure, we will have the following:

- $r_1^* = \{0, 2, 1, 0\}, y_{22}^{'} = 5.5, y_{12}^{'} = 4.5$

After the local search procedure, for r_1^* the probability of reaching customer 2 without interdiction is $p_{02} = 0.95$, and the probability of reaching customer 1 without interdiction is $p_{02} \cdot p_{21} = 0.95 \cdot 0.85 = 0.8075$ .Let us review if the expected demand to be serviced to each customer changes:

- Customer 1: $d_1^{'} \cdot p_{02} \cdot p_{21} \geq d_1 \Rightarrow 4.5 \cdot 0.95 \cdot 0.85 = 3.63375 < d_1 = 4$. Condition not met
- Customer 2: $d_2^{'} \cdot p_{02} \geq d_2 \Rightarrow 5.5 \cdot 0.95 = 5,225 \geq d_2 = 3$. Condition met

The impact of the interdiction probabilities over the demand to be serviced after the local search procedure can make a new set of routes infeasible. If this new route is infeasible, and if we have additional routes in the solution (let us say r_1, r_2, among others), it will be impossible to generate a combination where the route $r_1^{'}$ is part of a feasible or optimal solution (considering that might be the case). The main reason for this issue is that servicing a higher expected demand to each customer than needed, which makes the network (or, in this case, this route) very saturated, which will have an impact on following steps of local search procedures (for example, if we want to perform a swap-local search procedure with other routes). In r_1, the expected demand to be serviced to customer 1 was very close to the required demand (4.05 when 4 was needed), but for customer 2, we were serving 40.25% more goods that required (4.2075 when 3 was needed). In this example, this over-servicing generates infeasibility because GRASP-X always tries to use all the available capacity when generating the routes, even when there is no need. We will introduce a third logic to prevent this effect, giving GRASP-H a bigger space to find feasible solutions. This logic is called Minimum Demand.

Minimum Demand

We know that the condition that addresses that the expected demand to be serviced has to be greater or equal to the required demand is:

$$d_i^{'} \cdot \phi_i \geq d_i$$

Where d_i is the required demand to be serviced to customer i, $d_i^{'}$ is the demand serviced to customer i, and ϕ_i is the probability of reaching customer i without interdiction. Basically, $d_i^{'} \cdot \phi_i$ is the expected demand to be serviced, and it has to be at least equal to d_i. In GRASP-H, we are creating more flexible routes, serving the minimum amount of demand to be serviced to each customer considering interdiction probabilities:

$$d_{min_i}^{'} = \frac{d_i}{\phi_i}$$

Let us comprehend this logic with the previous example:

After performing the 2-opt or segment reversal local search procedure over r_1, we will have:

- $r_1^* = \{0,2,1,0\}$

Now, let us compute the minimum amount of demand to be serviced to each customer:

- Customer 1: $y_{min_{12}}^{'} = \frac{d_1}{p_{02} \cdot p21} = \frac{4}{0.95 \cdot p0.85} = 4.95356$
- Customer 2: $y_{min_{22}}^{'} = \frac{d_2}{p_{02}} = \frac{3}{0.95} = 3.15789$

Adding up these 2 demands, we have 8.111455, which is less than the solution under GRASP-X, which is 10. With this new logic, our new route is feasible, but it only uses 81% of the vehicle capacity instead of 100%, providing more flexibility when performing further local searches. With this new logic, the new route $r_1^{'}$ can move forward as a feasible candidate toward finding the feasible or optimal solution to the problem. It is important to emphasize that GRASP-H performs these minimum demands to be serviced calculations every time a local search procedure(2-opt, 3-opt, segment-reversal, and/or swap-local search) is performed. This new logic is fundamental for scaling GRASP-H to solve large benchmark instances, logic not implemented in GRASP-X.

3.3 Fleet Over-Utilization

As understood with the previous example, Xu's GRASP-X tends to over-service customers, even when there is no need to. Before analyzing the fleet over-utilization under Xu's GRASP-X, let us review the primary considerations and impacts of reaching this condition in real-world scenarios:

1. Cost Inefficiency: Over-utilizing vehicle capacity can lead to unnecessary transportation costs. When a vehicle delivers more than the required amount to a customer, the additional trips or over-deliveries increase fuel consumption, driver labor costs, and vehicle wear and tear without adding value.

2. Customer Dissatisfaction: Delivering more than a customer's demand can lead to dissatisfaction or logistical issues on the customer's end. Customers may not have the

storage capacity for excess deliveries, disrupting their operations and leading to a negative perception of the service provider.

3. Inefficient Use of Resources: By unnecessarily utilizing vehicles' total capacity, resources are not optimally allocated. This can result in fewer deliveries being made on a single trip, leading to more trips overall and reducing the overall efficiency of the delivery operation.

4. Environmental Impact: More trips and higher fuel consumption due to over-utilization of vehicle capacity contribute to higher carbon emissions.

5. Increased Complexity in Inventory Management: Over-delivering to customers complicates inventory management for both the delivery service and the customer. It might require additional handling, creating discrepancies in inventory records.

6. Opportunity Cost: When a vehicle's capacity is fully utilized without considering actual demand, it prevents that capacity from being used to service other customers who might have immediate needs.

7. Operational Flexibility: Maintaining flexibility in vehicle capacity allows for better handling unexpected demands or last-minute changes. Over-utilizing capacity reduces this flexibility, making adapting to dynamic conditions and urgent delivery requests harder.

8. Impact on Service Quality: Optimal service quality often requires balancing delivery efficiency with meeting customer needs precisely. Over-utilizing vehicle capacity can reduce the ability to provide timely, accurate, and reliable deliveries.

Although these indirect and direct costs were not included in the transportation costs in the numerical experiments, they were considered in the algorithm when seeking a feasible solution. Let us revisit Example 1 from this chapter and examine the solution when implementing the logic of serving the Minimum Demand under Interdiction. As a reminder, these are the characteristics of the network:

Node	x-coordinate	y-coordinate	Demand
depot (0)	0	0	0
1	-122	132	24
2	-132	-60	24
3	50	108	42
4	-160	62	25
5	-143	-124	21

Table 3.2: VRPI Example (Xu, 2017)

The fleet size is $K = 3$, the vehicle capacity is $Q = 50$, and the interdiction probabilities are:

$$\text{Interdiction Probabilities}: \begin{bmatrix} \sim & 4\% & 3\% & 1\% & 5\% & 1\% \\ 4\% & \sim & 1\% & 2\% & 2\% & 3\% \\ 3\% & 1\% & \sim & 5\% & 5\% & 5\% \\ 1\% & 2\% & 1\% & \sim & 4\% & 3\% \\ 5\% & 2\% & 5\% & 4\% & \sim & 4\% \\ 1\% & 3\% & 5\% & 3\% & 4\% & \sim \end{bmatrix}$$

Xu's Example 1 had the following results for the network:

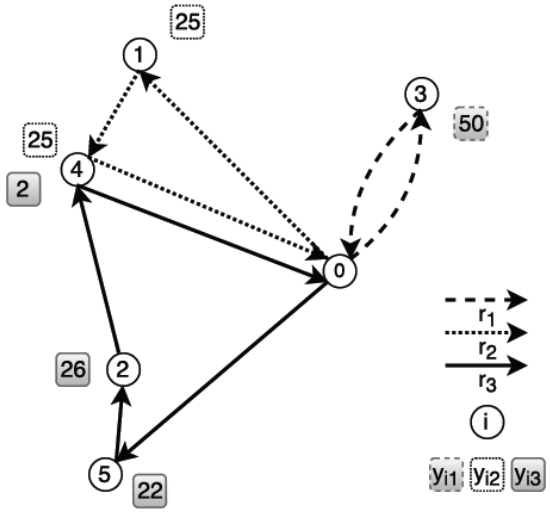

Figure 3.10: With Interdiction Solution (Xu, 2017)

In Xu's solution, we had the following logic comprehending the expected demand to be serviced condition. For customer 1, we have the following:

$\phi_{12} \cdot y_{12} \geq d_1 \leftrightarrow 96\% \cdot 25 = 24 \geq 24$

For customer 2, we have the following:

$\phi_{23} \cdot y_{23} \geq d_2 \leftrightarrow 99\% \cdot 95\% \cdot 25.5 \approx 24 \geq 24$

For customer 3, we have the following:

$\phi_{31} \cdot y_{31} \geq d_3 \leftrightarrow 99\% \cdot 50 = 49.5 \geq 42$

For customer 4, we have the following:

$\phi_{42} \cdot y_{42} + \phi_{43} \cdot y_{43} \geq d_4 \leftrightarrow 96\% \cdot 98\% \cdot 25 + 99\% \cdot 95\% \cdot 95\% \cdot 1.7 = 25.0389075 \geq 25$

Finally, for customer 5, we have the following:

$\phi_{53} \cdot y_{53} \geq d_5 \leftrightarrow 99\% \cdot 21.2 \approx 21 \geq 21$

The results show that the network is saturated with a total capacity of 150, equal to the serviced demand. For Customer 3, the expected demand is 49.2, but the required demand is 42, which is an increment of approximately 20%. For this particular customer, the minimum demand to be serviced is (addition of the 3-fold logic in GRASP-H):

$$y_{min_{31}} = \frac{d_3}{\phi_{31}} = \frac{41}{99\%} = 42.42 \leq 49.5$$

With this new serviced demand, the solution remains optimal. However, the network saturation or tightness is now 95% instead of 100%, which gives more flexibility if new solutions are needed to be found. Additionally, this new solution does not incur the over-utilization costs described above.

3.4 Infeasibility Analysis

As explained in Chapter 1, Xu (2017) did not study how interdiction probabilities can affect the feasibility of VRPI, which is critical when the total supply is similar to the customer demands, conditions close to the real-world scenarios and aligned to the large benchmark instances used in this thesis. To understand the roles of interdiction probabilities, supply, and demand in the feasibility of the VRPI, let us review a simple example:

Example 4:

- The fleet is just 1 vehicle with capacity Q
- There is only 1 customer to be serviced, with demand d_1
- There is only 1 possible route from the depot: $\{0,1,0\}$
- The probability of reaching customer 1 in route 1 without interdiction is ϕ_{11}

We know that the demand to be serviced to customer 1 in route 1, y_{11} has to be less or equal to vehicle capacity Q, or $y_{11} \leq Q$. Conversely, the expected serviced demand has to be at least the required demand of customer 1, or $\phi_{11} \cdot y_{11} \geq d_1$. Doing some replacements, we have:

$$Q \geq y_{11} \geq \frac{d_1}{\phi_{11}} \Longrightarrow \phi_{11} \geq \frac{d_1}{Q}, \text{ for the problem to be feasible}$$

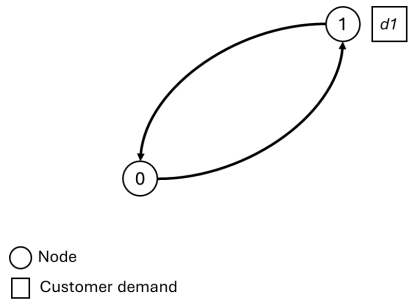

Figure 3.11: Example 4 Network and Route

Reviewing the feasibility condition described above, if the problem is entirely tight (i.e., $d_1 = Q \rightarrow \frac{d_1}{Q} = 1$ or 100%), any interdiction probability greater than 0 will make the problem infeasible. Therefore, there is an essential relationship between the feasibility of the problem and interdiction probabilities, demand, and supply.

For this example, we find that this is an *if-and-only-if* condition. In other words, **the problem is infeasible** $\iff \phi_{11} < \frac{d_1}{Q}$

Let's review another example:

Example 5: We will use the same network structure as Example 3:

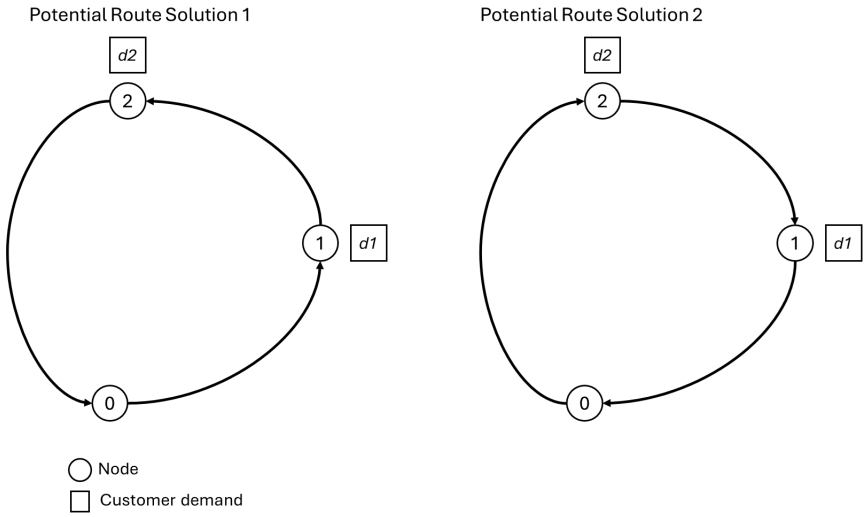

Figure 3.12: Example 5 Network Potential Route Solutions

- The fleet is just 1 vehicle with a capacity of Q
- There are only 2 customers to be serviced, with demand d_1 and d_2
- There are only 2 possible routes from the depot: $\{0,1,2,0\}$ or $\{0,2,1,0\}$
- The probability of reaching customer 1 in route 1 without interdiction is ϕ_{11}, and the probability of reaching customer 2 in route 1 without interdiction is ϕ_{21}
- The probability of reaching customer 1 in route 2 without interdiction is ϕ_{12}, and the probability of reaching customer 2 in route 2 without interdiction is ϕ_{22}

Route 1 and Route 2 will have the same mathematical approach, so let us analyze only Route 1. We know that the demand to be serviced to customer 1 in route 1, y_{11}, has

to be less or equal to vehicle capacity Q, or $y_{11} \leq Q$. Additionally, the demand to be serviced to customer 2 in route 1, y_{21}, and it has to be less or equal to vehicle capacity Q, or $y_{21} \leq Q$. On the other side, the expected serviced demand of customer 1 has to be at least the required demand of customer 1, or $\phi_{11} \cdot y_{11} \geq d_1$, and the expected serviced demand of customer 2 has to be at least the required demand of customer 2, or $\phi_{21} \cdot y_{21} \geq d_2$. Finally, the demand to be serviced in Route 1 cannot exceed the vehicle capacity, i.e., $y_{11} + y_{21} \leq Q$.

$$\phi_{11} \cdot y_{11} \geq d_1 \Longrightarrow y_{11} \geq \frac{d_1}{\phi_{11}}$$

$$\phi_{21} \cdot y_{21} \geq d_2 \Longrightarrow y_{21} \geq \frac{d_2}{\phi_{21}}$$

$$Q \geq y_{11} + y_{21} \geq \frac{d_1}{\phi_{11}} + \frac{d_2}{\phi_{21}}$$

$$\Longrightarrow Q \cdot \phi_{11} \cdot \phi_{21} \geq \phi_{21} \cdot d_1 + \phi_{11} \cdot d_2$$

$$\Longrightarrow \phi_{21} \cdot (Q \cdot \phi_{11} - d_1) \geq \phi_{11} \cdot d_2$$

Due to the interdiction event related to ϕ_{11} is independent to the interdiction event related to ϕ_{21}, in Route 1 $\phi_{11} \geq \phi_{21}$ (for Route 2, $\phi_{21} \geq \phi_{11}$).

$$\Longrightarrow \phi_{11} \geq \phi_{21} \geq \frac{\phi_{11} \cdot d_2}{Q \cdot \phi_{11} - d1} = \frac{\phi_{11} \cdot d_2}{\phi_{11} \cdot (Q - \frac{d_1}{\phi_{11}})} \geq \frac{d_2}{Q - d_1}$$

Therefore, we can see that the problem is infeasible when $\phi_{21} < \frac{d_2}{Q-d_1}$. If the problem is entirely tight, i.e., $d_1 + d_2 = Q$, we will have $\frac{d_2}{Q-d_1} = 1$, so any interdiction event will make the problem infeasible.

For this example, this is an *if-and-only-if* condition. In other words, **the problem is infeasible** $\iff \phi_{21} < \frac{d_2}{Q-d_1}$

When more complex examples have multiple customers and fleet numbers, it is not easy to identify potential feasible solutions and routes. Therefore, the analysis and infeasibility conditions developed with the previous examples are complex to determine. For this reason, the following examples will be solved using different interdiction scenarios, flight size, and fleet capacity.

Example 6:

For the following example, we will use a network of 3 customers plus a depot, with fleet sizes of 2 and 3 vehicles, with 3 different fleet capacities: 7, 7.5, and 8. For simplicity, for each set, we will generate the same probabilities of interdiction between each pair of arcs (i.e., if we set a probability of interdiction equal to 5%, that number will apply between each par or arcs for that specific setting). To solve this problem, we will use CPLEX for the cost minimization problem discussed in Chapter 1 (VRPI-1).

We will use this network as a base, with the 3 customers and depot and their respective demands:

Node	x-coordinate	y-coordinate	Demand
depot (0)	0	0	0
1	4	1	4
2	2	3	6
3	0	-1	4

Table 3.3: VRPI Example Testing for Infeasibility

Let us remember that tightness is defined by the total demand divided by the total capacity of the fleet ($\frac{\sum_i d_i}{K \cdot Q}$), where K is the fleet size, Q is the capacity of the vehicles, and d_i is the demand of customer i. For the experiments, the demands and position of the clients remained unchanged. These are the different settings run and their respective results:

Setting	Fleet Size	Capacity	Tightness	Interdiction Prob.	Feasible Solution	Cost
1	2	7.0	100%	0%	Optimal	18.7
2	2	7.0	100%	≥ 1%	Infeasible	NA
3	2	7.5	93.3%	1%	Optimal	18.7
4	3	7.0	66.6%	1%	Optimal	17.5
5	2	7.5	93.3%	2.5%	Optimal	18.7
6	2	7.5	93.3%	5.0%	Optimal	18.7
7	2	7.5	93.3%	≥ 7.5%	Infeasible	NA
8	3	7.5	62.2%	7.5%	Optimal	17.5
9	2	8.0	87.5%	7.5%	Optimal	18.7
10	2	8.0	87.5%	≥ 10.0%	Infeasible	NA

Table 3.4: Results VRPI Example Testing for Infeasibility

The results show that depending on the problem's characteristics (fleet size and capacity), the probability of interdiction can make it infeasible. This analysis was fundamental because some benchmark instances for implementing GRASP have a tightness close to 100%, probably leading to infeasible solutions.

Chapter 4

Greedy Randomized Adaptive Search Procedure

4.1 GRASP Metaheuristic Overview

GRASP is a well-known metaheuristic algorithm used for complex combinatorial optimization problems, which combines greedy algorithms and randomization techniques to explore and navigate the solution space.

Heuristic Nature: GRASP is an algorithm designed to find solutions within a reasonable time frame, particularly for problems where finding the optimal solution is computationally difficult or even impractical. Unlike exact algorithms that guarantee optimality, GRASP seeks a balance between solution quality and computational time.

Metaheuristic Framework: GRASP goes beyond basic heuristic algorithms by incorporating a higher-level framework or “meta-strategy" that leads the search process. This strategy helps the algorithm try to avoid local optima and explore the solution space extensively. GRASP is versatile and can be adapted to different problems and domains. A study by Pisinger and Ropke (2007) highlights the adaptability of GRASP in various OR contexts, emphasizing its robustness across different problem settings.

4.1.1 Key Components of GRASP

1. Greedy Construction Phase: This phase generates an initial feasible solution by a series of greedy choices. However, GRASP uses randomness in this process instead of always choosing the absolute best (greediest) option.

2. Randomized Selection: To avoid the myopic nature of pure greedy algorithms, GRASP introduces randomness by creating a Restricted Candidate List (RCL) containing several good, but not necessarily the best, options. The algorithm randomly selects from this list, balancing greedy selection and exploration. Binato *et al.* (2001) elaborates on the efficacy of the RCL in GRASP, particularly in large-scale industrial applications.

3. Local Search: GRASP employs a local search procedure to improve the solution after its initial solution generation iteratively, trying to find local optimality.

4. Iterative Process: GRASP iterates through these phases multiple times. Each iteration starts by constructing a new solution and refining it using a local search procedure. The best solution found across all iterations is retained as the final solution.

4.1.2 Historical Context

The inception of GRASP can be traced back to the late 1980s, primarily to the work of Thomas A. Feo and Mauricio G.C. Resende, laying the foundation of one of the most versatile and applied metaheuristics.

Feo and Resende initially introduced the GRASP methodology in the context of set covering problems. Their 1989 paper was fundamental in defining the basic structure and approach of GRASP.

Following its success in covering problems, GRASP's applicability was quickly recognized in other areas. Researchers adapted the algorithm to tackle combinatorial optimization problems, such as the Traveling Salesman Problem (TSP) and facility location problems.

The versatility of GRASP was further improved through its integration with other optimization techniques. Hybrid approaches, combining GRASP with path-relinking strategies, as discussed by Resende and Ribeiro, showcased improved performance and solution quality (Resende and Ribeiro, 2010). Over time, improvements in the algorithmic structure of GRASP, such as more effective local search strategies and adaptive mechanisms, have been developed.

GRASP's growing recognition in the academic world is evident through its frequent presence in top-tier OR journals. Articles and studies on GRASP have become staples in journals such as Operations Research, European Journal of Operational Research,

and Computers and Operations Research. Additionally, the inclusion of GRASP in academic curriculums and textbooks attests to its importance and influence in OR. It is a fundamental learning tool for students and researchers entering the field.

4.1.3 Role in Problem Solving in Operational Research

GRASP metaheuristic has significantly impacted the field of OR, offering versatile and robust solutions to a diverse range of complex problems and domains. In this context, GRASP has emerged as a powerful tool, particularly for addressing combinatorial optimization problems, which are prevalent in OR.

Combinatorial optimization problems are challenging. GRASP addresses this complexity by providing a flexible, structured approach. One of GRASP's key characteristics is its versatility, which allows it to be adapted to various problem types in OR.

4.1.4 Advantages Over Exact Methods

OR included a variety of exact optimization methods, such as linear programming, integer programming, and dynamic programming. While effective in specific scenarios, these methods have limitations, particularly when dealing with complex, non-linear, and large-scale problems. This is where GRASP offers several advantages.

Flexibility and Adaptability

Unlike exact methods that are often rigid and problem-specific, GRASP can be easily tailored to a wide range of problem types and domains. This flexibility is exemplified in a study by Resende and Ribeiro (2010), demonstrating GRASP's adaptability across various OR problems.

Handling of Non-Linearity and Discreteness

GRASP does not require the problem to adhere to linearity or continuity, making it ideal for real-world applications where these conditions are often unmet.

Efficiency in Large-Scale Problems

GRASP's randomized approach remains efficient even as the problem size increases. This efficiency is particularly beneficial in industries where decisions must be made quickly and on a large scale.

Robustness and Quality of Solutions

GRASP consistently finds high-quality solutions across different runs and problem instances.

Comparative Studies and Evaluations

GRASP has been the subject of numerous comparative studies, often outperforming exact methods in terms of solution quality and computational efficiency. Binato *et al.* (2001) evaluated GRASP against integer programming in power system expansion planning. The study concluded that GRASP not only provided comparable or better solutions but did so in significantly less computational time.

Transportation and Routing Problems

In transportation, GRASP's ability to handle real-world constraints such as time windows and vehicle capacities has been a significant advantage. Taillard *et al.* (1997) demonstrates GRASP's effectiveness in this domain, surpassing traditional routing algorithms.

4.1.5 Components of GRASP Metaheuristic

Greedy Algorithm Component

The Greedy Algorithm phase is the first step in the GRASP methodology, characterized by making consecutive myopic decisions and choosing the locally optimal solution at each stage to approximate a global optimum.

The fundamental premise of the Greedy Algorithm is that choosing the best short-term option at each iteration leads to the best overall solution. However, this approach does not always guarantee a global optimum, which is where GRASP introduces its unique elements. In GRASP, the Greedy Algorithm phase serves as the initial solution construction method. It quickly builds a feasible solution from which the subsequent local search phase can start, influencing the algorithm's overall effectiveness.

Characteristics of the Greedy Algorithm in GRASP

Iterative Selection: The Greedy Algorithm iteratively adds elements to the solution, choosing the element that seems best at each step.

Myopic Decision-Making: The decisions made are myopic, focusing on immediate benefit rather than long-term implications.

Speed and Simplicity: The Greedy Algorithm is valued for its speed and simplicity, making it an efficient method for initial solution construction.

GRASP enhances the traditional Greedy Algorithm by introducing randomness. Instead of selecting the absolute best option at each step, GRASP creates a list of good candidates and randomly picks one, adding diversification. This Greedy Algorithm component provides a quick and efficient method for initial solution construction.

Randomization Technique

The randomization technique is a critical component of GRASP's success. This technique introduces a stochastic element enhances the algorithm's ability to explore the solution space more comprehensively and avoid local optima.

In the GRASP framework, randomization is implemented by creating a Restricted Candidate List (RCL), which includes several of the best elements identified by the greedy function, not just the single best element. A candidate from this list is then randomly selected to be part of the solution.

Benefits of Randomization in GRASP

Avoiding Local Optima: Randomization reduces the likelihood of the algorithm getting trapped in local optima.

Enhanced Exploration: The stochastic nature of the selection process ensures a more extensive solution space exploration.

Balanced Search: Randomization in GRASP balances intensive local search and global exploration.

Binato *et al.* (2001) found that the randomized component of GRASP provided solutions that were not only competitive but often superior to those obtained from exact methods, particularly in complex, multi-dimensional problems.

Local Search Procedure

Local search in GRASP is a critical phase that follows the initial solution construction via the greedy randomized approach. It enhances solution quality by iteratively exploring the neighborhood of the current solution and seeking improvements, helping to escape local optima and move toward more globally optimal solutions.

Implementation of Local Search in GRASP

The implementation of local search in GRASP involves evaluating and exploring neighboring solutions. This process typically includes:

Defining a Neighborhood Structure: The neighborhood of a solution is defined based on the problem's context and consists of solutions that are slightly different or "close" to the current solution.

Exploration Strategy: The search within the neighborhood can be exhaustive, evaluating all neighbors or randomly selecting a subset of neighbors for evaluation.

Solution Update: The current solution is updated if a better solution is found in the neighborhood. This process continues until no further improvements can be found.

Binato *et al.* (2001) found that the local search component of GRASP significantly improved solutions over pure greedy algorithms.

Integrating local search with other metaheuristic techniques often results in a more robust search process that effectively navigates complex solutions.

Iterative Process

GRASP's iterative process is fundamental to to solve complex optimization problems effectively. It consists of repeated cycles, each with two main phases: the construction phase, where a feasible solution is generated, and the local search phase, where this solution is improved. This repetitive cycle is key to GRASP's effectiveness in exploring the solution space and enhancing solution quality over time.

Role of Iteration in GRASP

The iterative process in GRASP serves several crucial roles:

1. Diversification of Solutions: Each iteration in GRASP starts from a different starting point, allowing the exploration of diverse regions of the solution space.

2. Balance Between Exploration and Exploitation: The iterations balance exploring new solutions (exploration) and enhancing existing ones (exploitation), essential for preventing convergence to local optima.

3. Convergence to Optimal Solutions: Through successive iterations, GRASP progressively converges toward optimal or near-optimal solutions by refining its search based on the outcomes of earlier iterations.

In implementing its iterative process, GRASP typically follows these steps:

1. Initialization: The process starts with an initial setup that involves defining parameters such as the maximum number of iterations and stopping criteria.

2. Greedy Randomized Construction: Each iteration begins with constructing a new solution using a greedy randomized approach.

3. Local Search Enhancement: The constructed solution is subjected to a local search procedure looking for enhancements.

4. Repeat Cycle: The cycle iterates, with each repetition using results from prior iterations.

The iterative process is critical in the GRASP metaheuristic, balancing exploration and exploitation through repeated cycles of solution construction and refinement.

4.1.6 Algorithmic Form of General GRASP

The GRASP metaheuristic is a robust and adaptable method extensively employed in OR for solving complex optimization problems. Its algorithmic representation, often in pseudo-code, is crucial when understanding and applying GRASP.

Pseudo-code of GRASP

The pseudo-code for GRASP delineates its fundamental phases: the construction phase, the local search phase, and the iterative process. Below is a comprehensive representation of these components.

Algorithm 1 Greedy Randomized Adaptive Search Procedures (GRASP)

Require: ProblemInstance, MaxIterations, Alpha
Ensure: BestSolution
1: BestSolution ← ∅
2: **for** $i = 1$ **to** MaxIterations **do**
3: Solution ← GreedyRandomizedConstruction(ProblemInstance, Alpha)
4: Solution ← LocalSearch(Solution)
5: **if** Cost(Solution) < Cost(BestSolution) **then**
6: BestSolution ← Solution
7: **end if**
8: **end for**
9: **return** BestSolution

Algorithm 2 GreedyRandomizedConstruction

Require: ProblemInstance, Alpha
Ensure: Solution
Solution ← ∅
while Solution is not complete **do**
  RCL ← ConstructRCL(ProblemInstance, Solution, Alpha)
  Element ← SelectRandomly(RCL)
  Solution ← Solution ∪ {Element}
end while
return Solution

Algorithm 3 LocalSearch

Require: Solution
Ensure: Solution
Improved ← true
while Improved **do**
  Improved ← false
  for each NeighborSolution in Neighborhood(Solution) **do**
    if Cost(NeighborSolution) < Cost(Solution) **then**
      Solution ← NeighborSolution
      Improved ← true
      break
    end if
  end for
end while
return Solution

The algorithmic representation of GRASP in pseudo-code offers a structured and systematic framework for implementing this robust metaheuristic.

Explanation of Each Step

Initialization and Preliminary Setup

The GRASP algorithm begins by establishing a benchmark for comparison, often initializing the best solution found so far as null or assigning it an infinitely high cost. This step is vital for ensuring that any feasible solution identified during the iterative process can be recognized as an improvement.

Iterative Cycle: Core of GRASP

1. Greedy Randomized Construction Phase: This initial phase builds a feasible solution incrementally, starting from an empty state and progressively adding components until a complete solution is achieved. The RCL is populated using a greedy evaluation function alongside a parameter alpha (α) that introduces randomness.

Mechanism: For each element added to the solution, the algorithm computes a greedy value (e.g., cost reduction) and selects candidates within the top percentile defined by alpha (α). This approach ensures the selection process remains greedy while incorporating randomness.

2. Local Search Phase: After constructing a feasible solution, the local search phase aims to refine it by exploring its immediate neighborhood.

Mechanism: The local search phase typically evaluates all or a substantial subset of neighboring solutions, implementing changes like element swapping, reordering, or component replacement. If a neighbor with a better objective value is identified, the current solution is updated to this improved neighbor.

Solution Assessment and Iteration Update

After each iteration, the algorithm compares the solution obtained from the local search phase with the best solution from previous iterations. If an improvement is identified, the best solution is updated.

4.2 GRASP for the Vehicle Routing Problem with Interdiction

The VRP and its variant, the SDVRP, present significant challenges in logistics and OR. This section explores the definitions, complexities, and distinct challenges associated with the VRP and SDVRP, emphasizing the SDVRP due to its similarity to the VRPI, particularly when considering split deliveries.

4.2.1 Vehicle Routing Problem

The VRP focuses on designing optimal routes from a depot to different customers, subject to constraints such as vehicle capacity, customer demand, service time windows,

route length, among others. The objective is to minimize total traveled distance, cost, or the number of vehicles used.

Complexity and Challenges

The VRP is NP-hard, indicating that it is computationally challenging to find an exact solution in reasonable time for its most complex cases. This complexity suggests that solving VRP optimally becomes impractical as the problem size increases. Challenges include the combinatorial explosion of possible routes as customer numbers increase and the difficulty of meeting multiple constraints simultaneously.

4.2.2 Split Delivery Vehicle Routing Problem and Vehicle Routing Problem with Interdiction

The SDVRP extends the VRP by allowing multiple deliveries to the same customer, breaking the traditional constraint of a single visit per customer. This variant is relevant to the VRPI, where deliveries may be split due to interdiction events. According to Xu (2017), the VRPI, which also permits split deliveries under such conditions, is NP-hard.

Definition and Challenges

In VRPI, the objective remains to minimize the overall cost or distance. However, the ability to split deliveries introduces new dimensions to the problem, including decisions on how to split orders, which deliveries to combine, and how to schedule multiple visits to the same customer without significantly increasing the total route length or

violating other constraints. This is fundamental because due to the probability of interdiction in different routes, now we should compute the expected amount of deliveries to be served for each customer. 4 key components need to be assessed when executing GRASP:

1. The demand serviced of each customer on each route cannot be more than the vehicle capacity

2. The expected demand serviced of each customer among all routes has to be greater or equal to its required demand

3. In VRPI, a customer can be visited without servicing any demand to reach another customer, depending on interdiction probabilities along different routes.

4. Cumulative probabilities of interdiction need to be taken into account. For example, if the probability of not being interdicted between the depot and customer 1 is 0.9 (90%), and the probability of not being interdicted between customer 1 and customer 2 is 0.8 (80%) in route 0, 1, 2, 0, then the probability of not being interdicted between the depot and customer 1 is the multiplication of the previous probabilities: 0.9*0.8 = 0.72 (72%), due to the events are independent as defined by Xu (2017)

Split deliveries offer potential cost savings and increased flexibility but also introduce additional complexity. Key challenges include determining the optimal split of deliveries and dynamically adjusting routes.

4.2.3 Relevance of GRASP

The GRASP metaheuristic is fundamental for solving complex optimization problems, including the VRP, SDVRP, and VRPI, due to its characteristics make it well-suited to address the challenges of these problems.

GRASP's dual-phase approach, combining greedy randomized construction with local search, aligns well with the nature of SDVRP and VRPI, due to these problems require balancing exploration of the solution space and exploitation of high-quality solutions.

1. Greedy Randomized Construction for Diverse Solution Generation

The greedy randomized construction phase of GRASP is critical for generating feasible solutions for SDVRP and VRPI. By introducing controlled randomness into the selection process, GRASP avoids prematurely converging to local optima.

2. Local Search for Solution Refinement

The local search phase in GRASP refines each constructed solution by exploring its neighborhood for potential improvements.

Why GRASP Excels in Split Delivery Vehicle Routing Problem and Vehicle Routing Problem with Interdiction?

1. Adaptability to Complex Constraints

SDVRP and VRPI are characterized by complex constraints, including vehicle capacities and the possibility of split deliveries. GRASP's framework can incorporate these constraints into the construction and local search phases.

2. Effective Exploration and Exploitation Balance

The balance GRASP maintains between exploring new solutions and exploiting known good ones is crucial for SDVRP and VRPI. This balance ensures the search process remains dynamic, uncovering and refining solutions to find optimal or near-optimal routes.

3. Handling Dynamic and Large-scale Problems

GRASP's iterative nature makes it well-suited to handle large-scale aspects of SDVRP and VRPI.

GRASP's relevance to solving SDVRP and VRPI lies in its unique combination of greedy randomized construction, intensive local search, adaptability, and balance between exploration and exploitation, making it ideal for addressing the complexity, constraints, and dynamic nature of these problems.

GRASP's Advantages in Split Delivery Vehicle Routing Problem and Vehicle Routing Problem with Interdiction

1. Scalability: GRASP scales effectively with problem size, maintaining good performance without exponential increases in computation time.

2. Solution Quality: While not always guaranteeing optimality, GRASP consistently finds near-optimal solutions, often outperforming classical heuristics and other metaheuristics.

3. Flexibility and Adaptability: GRASP's structure allows adaptation to various problem constraints and configurations.

4. Simplicity and Ease of Implementation: Compared to some metaheuristics that require complex parameter calibration, GRASP's simpler structure facilitates more straightforward implementation.

Despite its strengths, GRASP can be further enhanced, particularly in computational efficiency and escaping local optima in highly constrained scenarios. Future research could focus on integrating GRASP with machine learning techniques for adaptive parameter calibration or hybridizing it with other metaheuristics to improve exploration capabilities.

4.2.4 Algorithmic Form of GRASP for the Vehicle Routing Problem with Interdiction

Considering the critical components discussed in this chapter for the implementation of GRASP to solve the VRPI, a pseudo-code will be constructed. One key element for this grasp implementation is the ability to handle a 3-fold during the local search procedure, as explained in Chapter 3. Xu's GRASP-X algorithm selects feasible routes during the local search procedure using a 2-fold logic: either choose a route with a lower overall cost or choose a route with the highest probability of not being interdicted. This GRASP-X implementation does not necessarily identify the best feasible routes, in particular

when scaling to large instances. Therefore, GRASP-H employs a 3-fold logic. In addition to considering cost and the highest probability of not being interdicted, the third aspect of the algorithm evaluates routes with the highest chance of servicing the minimum required demand for customers under interdiction. This approach helps prevent saturating the entire network when seeking a feasible solution. Below, you can see this new local search approach:

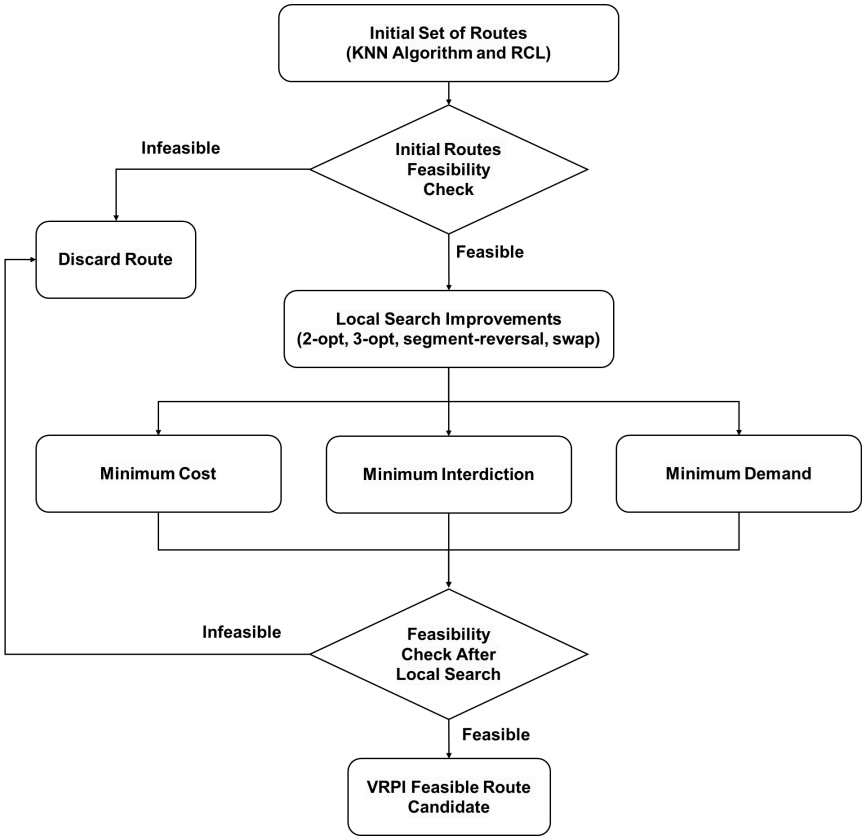

Figure 4.1: GRASP-H Local Search Algorithm

This thesis GRASP representation (GRASP-H) shows what could happen in each iteration during execution. The termination criteria is not being shown too, but basically considers a stagnation level to be set by the user. This means, if we cannot find any feasible solution by a certain number of iterations (set by the user), the code finishes resulting in an Infeasible solution. If at some point we find a feasible solution, and if there is no improvement of the solution by a certain number of iterations (set by the user), the code finishes resulting in the best Feasible solution found. This is the basic logic of the termination criteria, which gives to the user an extra layer of flexibility to end up the code early.

We need to explain that GRASP implementation used to solve the VRPI, being a metaheuristic rather than an exact method, has inherent limitations in its ability to guarantee feasibility and optimality. Specifically, if the GRASP result is infeasible, we cannot definitively conclude that the underlying problem itself is infeasible, as the algorithm may not have fully explored the solution space. On the other hand, if the GRASP solution is feasible, there is no guarantee that it is optimal—it simply represents a high-quality solution found through the heuristic process. To prove infeasibility or optimality, the problem would need to be solved using an exact method, for example using an optimization solver like CPLEX. However, this is often technically impractical or impossible for large-scale instances due to the significant computational resources required by exact methods to explore all possible solutions

4.2.5 Pseudo-code of GRASP for the Vehicle Routing Problem with Interdiction:

The pseudo-code for GRASP outlines its core phases: the construction phase, the local search phase, and the iterative process. The following is a detailed representation:

Algorithm 4 Greedy Randomized Adaptive Search Procedures (GRASP) for VRPI

Require: ProblemInstance, MaxIterations, Alpha
Ensure: BestSolution
BestSolution ← ∅
for $i = 1$ **to** MaxIterations **do**
  Solution ← KNNBasedConstruction(ProblemInstance, Alpha)
  if FeasibilityCheck(Solution) **then**
    Solution ← MultiObjectiveLocalSearch(Solution)
    if FeasibilityCheck(Solution) **and** ObjectiveFunction(Solution) < ObjectiveFunction(BestSolution) **then**
      BestSolution ← Solution
    end if
  end if
end for
return BestSolution

Algorithm 5 KNNBasedConstruction

Require: ProblemInstance, Alpha
Ensure: Solution
Solution ← ∅
while Solution is not complete **do**
  RCL ← ConstructRCL(ProblemInstance, Solution, Alpha)
  Element ← SelectRandomly(RCL)
  Solution ← Solution ∪ {Element}
end while
return Solution

Algorithm 6 MultiObjectiveLocalSearch

Require: Solution
Ensure: Solution
Improved ← true
while Improved **do**
  Improved ← false
  for each NeighborSolution in Neighborhood(Solution) **do**
    Objective ← SelectObjective() {Choose between Cost, Interdiction, and Demand}
    if ObjectiveFunction(NeighborSolution, Objective) < ObjectiveFunction(Solution, Objective) **then**
      Solution ← NeighborSolution
      Improved ← true
      break
    end if
  end for
end while
return Solution

Explanation of each Part of the GRASP Metaheuristic for the Vehicle Routing Problem with Interdiction

Initialization and Preliminary Setup

The GRASP algorithm for VRPI begins by initializing the best solution as an empty set or assigning it an infinitely high cost. This step is crucial for ensuring that any feasible solution found during the iterations is recognized as an improvement. Additionally, initial routes are generated using the KNN algorithm to ensure a feasible starting point, considering the specific constraints and objectives of the VRPI.

Iterative Cycle: Core of GRASP

GRASP's strength lies in its iterative cycle, which consists of two distinct phases that collaboratively explore and exploit the solution space.

1. KNN-Based Initial Solution Phase: This phase begins by constructing an initial feasible solution using the KNN algorithm and the RCL. The RCL is populated using a greedy evaluation function alongside a parameter alpha (α) that introduces randomness.

Mechanism: For each element added to the solution, the algorithm computes a greedy value (cost reduction) and selects candidates within the top percentile defined by alpha (α). This approach ensures the selection process remains greedy while incorporating randomness, facilitating diverse solution pathways. Additionally, the KNN algorithm helps select initial routes where the customers are spatially close to each other and split the customer demands if necessary when creating the neighborhoods, ensuring feasibility and efficiency from the start.

2. Multi-Objective Local Search Phase: After constructing a feasible solution, the local search phase refines it by exploring its immediate neighborhood. This phase involves iterating through potential neighboring solutions—variations of the current solution with minor modifications—to identify improvements. Unlike the general GRASP, this implementation incorporates a multi-objective approach (min cost, min interdiction, min demand).

Mechanism: The local search phase evaluates all or a substantial subset of neighboring solutions, implementing changes like element swapping, reordering, or component replacement. The unique aspect here is the 3-fold logic.

- Minimize Cost: Searches for neighbor solutions that reduce the total cost
- Minimize Interdiction Probability: Searches for solutions that reduce the probability of interdiction
- Minimize Demand: Searches for solutions that minimize the demand to be serviced considering interdiction probabilities

The objective selection is dynamic, allowing the algorithm to effectively balance different aspects of the problem.

Solution Assessment and Iteration Update

After each iteration, the algorithm compares the cost of the solution obtained from the local search phase with the best solution from previous iterations. The best solution is updated if an improvement (cost reduction) is identified.

Detailed Algorithmic Explanation

1. Initialization and Preliminary Setup

- Initialize the best solution as an empty set or with an infinitely high cost
- Set up parameters such as MaxIterations and α

2. KNN-Based Initial Solution Phase

- Construct an initial feasible solution using the KNN algorithm
- Populate the RCL based on a greedy evaluation function and alpha (α)
- Randomly select elements from the RCL to incrementally build a complete solution

3. Multi-Objective Local Search Phase

- Perform a local search on the constructed solution
- Evaluate neighbor solutions based on the chosen objective: min cost, min interdiction probability, or min demand
- Update the current solution to the improved neighbor solution if a better one is found
- Ensure the solution remains feasible after each improvement

4. Solution Assessment and Iteration Update

- Compare the current solution to the best solution found so far
- Update the best solution if the current solution is better

5. Final Output

- Return the best solution found after completing the specified number of iterations

By integrating these steps, GRASP for VRPI effectively balances the multi-objective nature of the problem, ensuring robust and adaptable solutions to complex routing challenges under interdiction scenarios.

4.3 Parameters in GRASP

The GRASP metaheuristic could be influenced by various parameters. These parameters are crucial for determining GRASP's performance and effectiveness. Xu (2017) did not explore this topic, as performance and computational time were not critical for small instances. However, this thesis addresses large benchmark instances, requiring parameter calibration. The strategy for this calibration will be described in the Numerical Results chapter.

On the they key objectives to calibrate GRASP parameters is to generate or improve your domain knowledge of the problem. Le us imagine that you are trying to solve a new problem, for example, the VRPI. At the beginning you do not know how sensitive can

be your implementation if some of the key parameters change. When you realize which set of parameters have the best performance in your metaheuristic implementation to solve your problem, then you can use that set of parameters for future runs. You will not need to calibrate the parameters every time you want to solve different instances of the same problem. In the particular case of this thesis, this process was very valuable, because we could understand that the minimum demand logic was the most successful when solving large scale instances with interdiction under tight network, which was the purpose of the introduction of that logic. In cases without interdiction, the minimum cost logic was overall the most successful, which makes sense because in those cases you are just trying to solve a minimum cost optimization problem called the SDVRP.

It is important to note when making the parameter calibration on small instances, we hope that those parameters are going to be the best parameters for large instances as well. There is no 100% guarantee that this might be the case, because large instances might have different and more complex network structures than small instances. That is why is critical to perform the parameter calibration with an extensive set of instances that, even though they are small in size, they can reflect different kinds of problems in terms of network structure, interdiction probabilities, tightness of the network, number of customers, number of vehicles, depot location, among others. In this thesis, in order to accomplish this idea, we tested 180 different set of instances, which are detailed in Numerical Experiments chapter.

4.3.1 List of Key Parameters:

1. Greediness Factor (Alpha or α)

The greediness factor, commonly denoted as alpha (α), is a crucial parameter in GRASP. It controls the balance between greediness and randomness during the solution construction phase.

Role: Alpha determines the size of the RCL, influencing how greedy or random the solution construction will be.

Range: Typically, alpha ranges from 0 (purely greedy) to 1 (purely random).

Impact: An optimal value of alpha can significantly improve the quality of the initial solution and the algorithm's overall performance.

2. Perturbation Percentage

The perturbation percentage is another important parameter in the GRASP metaheuristic. It controls the degree of diversification during the local search phase.

Role: The perturbation percentage determines how much the current solution is altered or “perturbed" to escape local optima and explore another parts of the solution space.

Range: Typically, the perturbation percentage ranges from 0% (no perturbation) to 100% (completely random perturbation). A higher percentage introduces more randomness to the solution, while a lower percentage maintains more of the current solution's structure.

Impact: An optimal perturbation percentage can help the algorithm avoid getting stuck in local optima and improve the chances of finding a better overall solution. Too little perturbation might lead to premature convergence, while too much could result in a loss of solution quality.

3. Number of Iterations

The number of iterations is another critical parameter that determines how many times the GRASP procedure will be executed.

Role: It sets the number of independent runs of the algorithm, each starting with a different initial solution.

Impact: A higher number of iterations generally increases the chances of finding better solutions but also requires more computational time.

4. Local Search Parameters

Local search parameters define the depth and scope of the local search phase in GRASP.

Components: These may include the maximum number of local search iterations, the type of neighborhood structure, and the criteria for accepting new solutions.

Impact: Fine-calibrating these parameters can lead to a more thorough exploitation of the solution space, enhancing the quality of the final solution.

5. Termination Criteria

Termination criteria define when the GRASP algorithm should stop. This can be based on a fixed number of iterations, a time limit, or a convergence criterion.

Role: It prevents the algorithm from running indefinitely and can be used to control the computational effort.

Impact: Appropriate termination criteria ensure the algorithm runs long enough to explore the solution space adequately but does not overextend computational resources.

6. Randomization Seed

The randomization seed is used to initialize the random number generator in GRASP.

Role: It ensures the reproducibility of results, which is vital for experimental analysis and comparison.

Impact: Different seeds can lead to different solution paths and outcomes, highlighting the algorithm's stochastic nature.

GRASP's parameters significantly influence its performance and effectiveness, where optimal parameter configuration depends on the specific problem context and requires empirical analysis, experimentation, and domain knowledge.

4.3.2 Calibration Strategies

The GRASP metaheuristic's effectiveness is due to fine-calibrating its parameters. Calibrating GRASP's parameters involves adjusting factors like greediness, perturbation

percentage, number of iterations, local search parameters, termination criteria, and randomization seed to enhance the algorithm's efficiency and solution quality.

1. Manual Calibration

Manual calibration involves adjusting parameters based on intuition, experience, and trial and error. It requires a profound understanding of domain of the problem, and the effects of each parameter on the algorithm's performance (Bergstra and Bengio, 2012).

2. Automated Calibration Automated calibration involves using algorithms to find the optimal set of parameters (Snoek *et al.*, 2012).

3. Adaptive Calibration

Adaptive calibration refers to dynamically adjusting parameters during the algorithm's execution based on its performance. This approach can lead to more responsive and effective parameter settings (Hutter *et al.*, 2011).

4. Machine Learning-Based Calibration

Machine learning techniques can be employed to calibrate GRASP's parameters. Methods like reinforcement learning or neural networks can learn the most effective parameters based on historical performance data (Zoph and Le, 2016).

5. Metaheuristic Calibration

Metaheuristic algorithms can also be used to calibrate GRASP's parameters. These metaheuristics can explore the parameter space to find the most effective combinations (Fukunaga, 2008).

The calibration of parameters in GRASP is an important process that significantly impacts the algorithm's efficacy in solving OR problems. The ongoing advancements in calibration methodologies, particularly in harnessing machine learning and metaheuristics, present exciting prospects for further enhancing GRASP's capabilities in OR.

For this thesis, an Automated Calibration will be performed, and the calibration strategy will be explained in Chapter 5.

Chapter 5

Numerical Experiments

The main purpose of this chapter is to describe and analyze the different numerical results derived from the evaluation of GRASP metaheuristic and CPLEX models with different instances, with and without interdiction. All models and algorithms were coded using Python version 3.11.7, and IBM ILOG CPLEX version 22.1.1. The computations were performed using an MS Windows computer with a 64-bit operating system, x64-based processor, Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz, and 32.0 GB of installed RAM.

All CPLEX and GRASP models developed and evaluated in this thesis, among all the instances, are stored in the following private Github repository: `https://github.com/chinrichsen/mcmaster_cse_thesis_carlos_hinrichsen`. Permission to access the repository can be granted upon request. Please send an email to `hinrichc@mcmaster.ca` to request access.

The Numerical Results Chapter is divided into six parts:

1. Description of Small Randomly Generated Instances and their interdiction probabilities

Small randomly generated instances are VRPI problem sets created using random generation methods to produce a variety of scenarios with fewer customers and vehicles compared to large benchmark instances. These instances are used for testing and calibrating algorithms in a controlled environment. Interdiction probabilities in these instances are also randomly assigned, simulating different levels of uncertainty across the routes. This randomness helps evaluate the algorithms' flexibility and adaptability to different levels of route disruption.

2. Feasibility Analysis for Small Randomly Generated Instances using CPLEX

Feasibility analysis involves determining whether a given VRPI instance can be solved within the constraints using the CPLEX optimization software. By applying CPLEX to small randomly generated instances, we can verify if feasible solutions exist that meet all problem constraints.

3. CPLEX and GRASP comparison for Small Randomly Generated Instances using CPLEX

This comparison involves evaluating the performance of the GRASP algorithm against CPLEX on small randomly generated VRPI instances. Key metrics for comparison include solution quality (e.g., total cost, interdiction probability), computational time, and the ability to find feasible solutions. CPLEX is an exact solver, which means that it

provides a benchmark for optimal solutions. At the same time, GRASP, as a metaheuristic method, is assessed on its efficiency and effectiveness in approaching these optimal solutions within reasonable time limits.

4. GRASP Parameter Calibration for Small Randomly Generated Instances

Parameter calibration involves adjusting the GRASP algorithm's parameters, such as the greediness factor (α) and the number of iterations, among others, to optimize its performance on small, randomly generated instances. The purpose of this process is to find the best combination of parameters that yields high-quality solutions with minimal computational effort.

5. Description of Large Benchmark Instances and their interdiction probabilities

Large benchmark instances refer to pre-defined sets of VRPI problems commonly used in research to evaluate algorithm performance. These instances typically feature a large number of customers and vehicles with complex routing requirements. Interdiction probabilities in these instances represent the likelihood of a route being disrupted, adding a layer of complexity to the routing problem.

6. GRASP results on Large Benchmark Instances

This part analyzes the outcomes of applying the GRASP-H algorithm to large benchmark instances of the VRPI based on the Parameter Calibration performed on small instances. The results are analyzed considering solution quality, solution robustness against interdiction, and computational efficiency. Key performance indicators include

the total cost of the routes, and the time taken to reach a solution. These results help validate the effectiveness of GRASP-H in solving large VRPI instances.

5.1 Description of Small Randomly Generated Instances and Their Interdiction Probabilities

5.1.1 Small Instances

Following Xu (2017) methodology, we randomly generated instances with $3,4,5,6$, and 7 customers, number of vehicles of 2, 3, or 4, fixed vehicle capacity equal to 10. Each customer was positioned randomly in a $200x200$ grid. The depot is located in the middle of the grid, and customer demands were generated with a Uniform Distribution, with a minimum of 2 and a maximum of 10. One main difference with Xu (2017) is that for each combination of a number of customers and a number of vehicles, we are setting different levels of tightness: 60%, 70%, 80%, 90%, 95% and 99%. Let us review the definition of tightness.

Tightness is a crucial metric in SDVRP and VRPI, indicating how constrained the problem or the network is concerning available vehicle capacity. It helps understand a given instance's difficulty and feasibility under various constraints. Tightness is defined as the ratio of total demand to total vehicle capacity, calculated using the formula:

$$\textbf{Tightness} = \frac{\text{Total Demand}}{\text{Total Vehicle Capacity}} \tag{5.1.1}$$

Where:

Total Demand is the sum of the demands of all customer nodes in the network, and Total Vehicle Capacity is the product of the number of vehicles and the capacity of each vehicle.

From Appendix A, the table describes a diverse set of instances characterized by different numbers of customers, vehicles, and capacities, as well as differing tightness levels and interdiction probability parameters. This extensive range of parameters ensures a comprehensive algorithm performance evaluation across different scenarios. The instances include small to medium-sized scenarios with 3 to 7 customers and vehicle capacities consistently set at 10 units. The tightness levels range from 60% to 99%, reflecting different degrees of constraint on the routes. Each instance is further diversified by interdiction probability parameters, ranging from 0% to 1% – 40%.

The main focus to solve the VRPI on small instances, involved to generate instances up to 7 customers. Basically, as the number of customers increased, the computational challenges increased as well. Specifically, with exact methods using CPLEX, the solver started to demand a significant amount of computational time to generate all possible routes and to generate the associated variables. This increased level of complexity resulted in extensive memory usage, ultimately leading to a memory crash due to the extensive volume of routes and constraints that needed to be managed. Consequently, it became impractical to extend the exact methods to larger instances without encountering substantial computational limitations.

When generating the interdiction probabilities for small generated instances and large benchmark instances, we created all the samples by trying to recreate their theoretical mean and standard deviation (SD) as closely as possible. Being a the min value and b the max value of a Uniform distribution, the theoretical mean and SD to recreate are:

$$\text{Mean} = \frac{a+b}{2}$$

$$\text{SD} = \sqrt{\frac{1}{12} \cdot (b-a)^2}$$

The mean interdiction probability values demonstrate the expected likelihood of interdiction occurring, while the standard deviation provides the variability of this probability. These characteristics collectively impact the complexity and solution quality for each instance.

5.1.2 Interdiction Probabilities

Following Xu (2017) methodology, we will assess six different interdiction probability scenarios between each pair of nodes, recreating their theoretical mean and SD.

- Scenario 1: 0% interdiction probability
- Scenario 2: Between 1% – 5% interdiction probability, Uniformly distributed
- Scenario 3: Between 1% – 10% interdiction probability, Uniformly distributed
- Scenario 4: Between 1% – 20% interdiction probability, Uniformly distributed

- Scenario 5: Between 1% – 30% interdiction probability, Uniformly distributed
- Scenario 6: Between 1% – 40% interdiction probability, Uniformly distributed

Considering the combination of the instances and the interdiction probabilities, the total number of small generated instances with interdiction probabilities to assess is 180. The characteristics of these 180 scenarios can be seen in Appendix A. The Interdiction Probability statistics of each of the 180 scenarios can also be found in Appendix A.

5.2 Feasibility Results for Small Randomly Generated Instances using CPLEX

The feasibility results can be found in Appendix B. The most important insights of this analysis are:

1. High Interdiction Probability: Instances with higher interdiction probabilities (e.g., 1% – 40%) are more challenging and often result in infeasibility. This is observed consistently across different levels of tightness and vehicle numbers.

2. Tightness Levels: Higher tightness levels (e.g., 95% or 99%) increase the likelihood of infeasible problems. This is particularly evident when combined with higher interdiction probabilities.

3. Capacity Constraints: Lower vehicle capacities also contribute to infeasibility, especially when tightness and interdiction probabilities are high.

4. Interdiction Probability Parameters: As interdiction probability parameters increase from 1% – 5% to 1% – 40%, the feasibility of the instances decreases significantly. This is true across various configurations of tightness, number of vehicles, and capacity.

5. Split Deliveries: Instances allowing split deliveries generally exhibit better feasibility outcomes. However, even with split deliveries, higher interdiction probabilities, and tightness levels can still lead to infeasibility.

Finally, we can understand all these insights graphically, where the blue dots represent optimal/feasible solutions, and the red dots represent infeasible solutions:

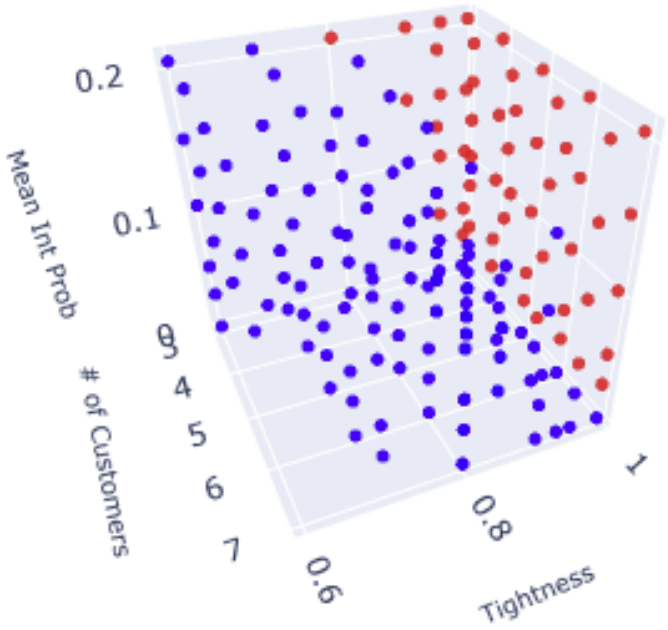

Figure 5.1: Feasibility Analysis 3D Plot

A useful rule of thumb for feasibility analysis in the VRPI is based on the relationship between the tightness of the network and the expected value of interdiction probabilities. Generally, when the network tightness approximates the expected value of interdiction probabilities, the problem sits at the feasibility frontier, meaning it is on the borderline between being feasible and infeasible. If the tightness of the network and/or the expected value interdiction probabilities increased, the problem tends to become infeasible. Conversely, by decreasing the network tightness and/or lowering the expected value of the interdiction probabilities, the problem shifts toward feasibility. These patterns are particularly evident in small instances, but in larger instances or more complex network structures, these relationships may exhibit slight deviations due to the additional variables and complexity introduced.

5.3 CPLEX and GRASP Comparison for Small Randomly Generated Instances

CPLEX results have already been computed and shown in Appendix B. For GRASP results, the modeling was performed with the following fixed parameters:

- Number of Iterations: 100
- Alpha (α): 50%
- Multi-Objective Local Procedure: In each iteration, choose randomly between minimum cost, minimum interdiction probability, and minimum demand

- Perturbation Percentage: Mechanism used to introduce diversification into the search process. Equal to 50%.

GRASP results can be shown in Appendix C. First, let us review how good GRASP-H implementation is without Parameter Calibration to label CPLEX results correctly. We can see these results in the following contingency table:

CPLEX \GRASP	Optimal/Feasible	Infeasible
Optimal/Feasible	84	42
Infeasible	0	54

Table 5.1: Contingency table comparing CPLEX and GRASP results

We can calculate the Accuracy, precision, recall, and F1 score:

- Accuracy = $\frac{\text{True Positives+True Negative}}{\text{Total Instances}} = \frac{138}{180} = 76.7\%$
- Precision = $\frac{\text{True Positives}}{\text{True Positives+False Positives}} = \frac{84}{126} = 66.7\%$
- Recall = $\frac{\text{True Positives}}{\text{True Positives+False Negatives}} = \frac{84}{84} = 100\%$
- F1-Score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision+Recall}} = 2 \cdot \frac{0.667}{1.667} = 80\%$

Some insights:

- The accuracy is relatively high at 76.7%, indicating that the model correctly identifies a majority of the cases as either “Optimal/Feasible" or “Infeasible"

- A precision of approximately 66.7% suggests that when the model predicts "Optimal/Feasible," it is correct about 66.7% of the time. This indicates a moderate level of confidence in positive predictions
- A recall of 1.0 means that the model can identify all actual “Optimal/Feasible" cases without missing any. This indicates that the model has a perfect recall for the positive class
- The F1-score, which is the harmonic mean of precision and recall, is approximately 80%. This value balances the trade-off between precision and recall, suggesting that the model’s performance in identifying “Optimal/Feasible" cases is quite good

The current model shows high recall and accuracy, with a good F1-score but moderate precision. This indicates that while the model is excellent at identifying all true “Optimal/Feasible" cases (no false negatives), it also generates some “Infeasible" cases were there were actually “Optimal/Feasible". There is room for improvement in precision, which could be achieved by further calibration of the model.

The following analysis is to understand the GAP between GRASP and CPLEX regarding time elapsed and solution quality. Considering the results of Appendix B and Appendix C, we have the following plots:

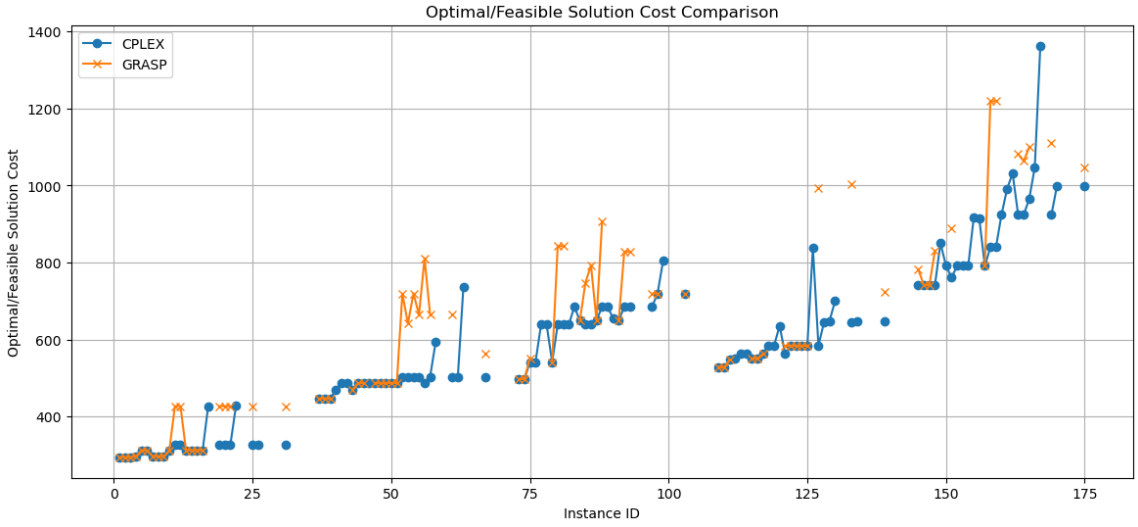

Figure 5.2: Solution Comparison CPLEX vs GRASP for VRPI

Looking at the solution comparison, we identify:

- GRASP algorithm generally achieves solutions with no GAP (0%) in smaller instances
- As the instance size increases, the GAP for GRASP remains zero until it starts to show feasible solutions with a higher cost compared to CPLEX in larger instances
- CPLEX consistently finds optimal solutions with lower costs compared to GRASP, especially as the instance size increases
- The increase in instance size slightly increases the solution cost found by CPLEX, but it maintains an optimal status in most cases

Looking at the elapsed time comparison, we identify:

- The time elapsed for GRASP remains significantly lower than CPLEX, especially in smaller instances
- As the instance size increases, GRASP's time increases but remains lower than CPLEX
- CPLEX takes more time to find optimal solutions, with time increasing significantly as the instance size increases
- The time complexity of CPLEX shows a steeper increase compared to GRASP as the number of customers increases

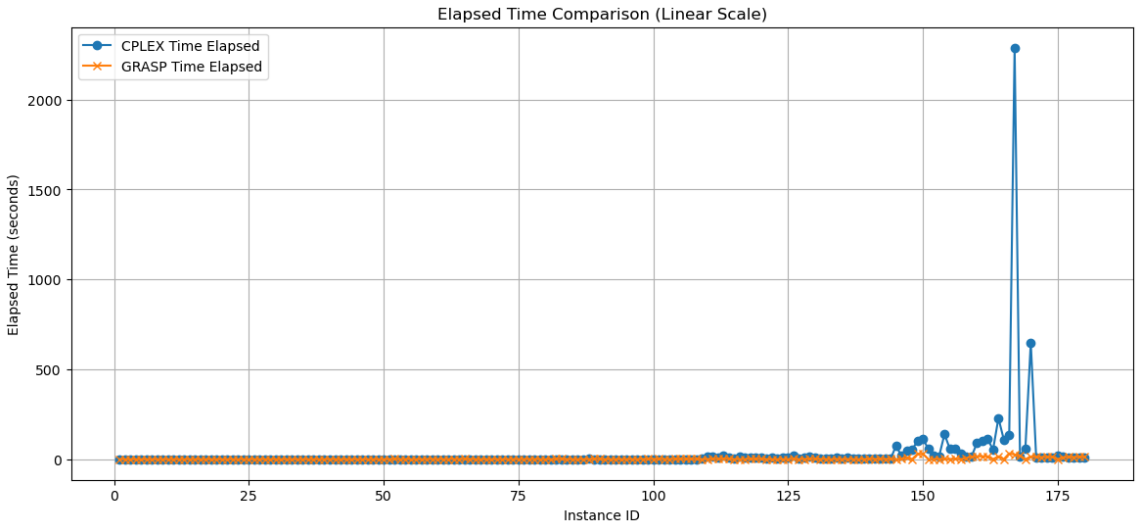

Figure 5.3: Elapsed Time Comparison CPLEX vs GRASP for VRPI (Linear Scale)

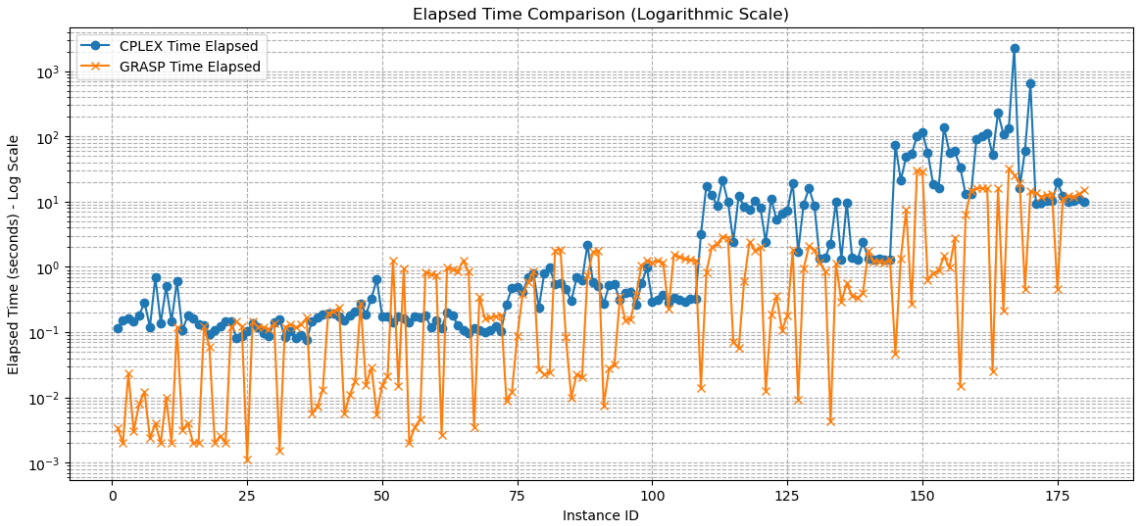

Figure 5.4: Elapsed Time Comparison CPLEX vs GRASP for VRPI (Logarithmic Scale)

Overall:

- GRASP achieves feasible solutions quickly but might not always match the optimal solutions of CPLEX, especially in larger instances
- The efficiency of GRASP in terms of time is notable, but there is a trade-off in solution quality as the instance size increases
- CPLEX consistently finds optimal solutions, but it requires significantly more computation time
- The optimality of solutions found by CPLEX comes at the cost of higher time consumption, especially for larger instances

- GRASP is more time-efficient and provides quick, feasible solutions with minimal GAP in smaller instances. However, as the problem size increases, the solution quality might deviate from the optimal
- CPLEX is more reliable in finding optimal solutions but with higher computation time, particularly as the instance size increases

5.4 GRASP Parameter Calibration for Small Randomly Generated Instances

Reviewing the results of GRASP without Parameter Calibration, we could identify in the contingency table that the method has the opportunity for improvement to reach better solutions in some scenarios. This analysis will be fundamental, as we will use the best parameter derived from this analysis to run the Benchmark Instances. The instances where we are going to perform Parameter Calibration are going to be the ones on the previous section where:

- GRASP did not reach an optimal solution, but a feasible one
- GRASP identifies the solution as “Infeasible" when it was “Optimal/Feasible".

The number of scenarios to be calibrated was 81. The parameters to be calibrated are the following:

- Iterations = {250, 500}
- Multi Local Search Objective = {min cost, min interdiction prob, min demand, random}
- Alpha (α) = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}
- Perturbation Percentage = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}

Therefore, the total number of scenarios to be run equals the 81 instances multiplied by all combination parameters. In this case, we have $81 \cdot 2 \cdot 4 \cdot 9 \cdot 9 = 52488$ runs. With the Parameter Calibration, we are trying to find if any of the “Infeasible" solutions turn to “Optimal/Feasible" or if we can find a better solution to the ones already labeled as “Feasible". This analysis is fundamental to understanding the best set of parameters to run the Benchmark Instances.

The results of the Parameter Calibration can be found in Appendix D. Looking at the results, we can see that around 20% of the 81 scenarios ended with a better solution in terms of cost. Those improved scenarios were translated to a mean improvement of 12% of the cost, where the most extreme case was about 40% cost improvement. Around 10% of the scenarios changed the solution status from “Infeasible" to “Feasible", and around 70% did not have any improvement. These results have shown the importance of Parameter Calibration when finding a better solution for developing domain knowledge in a problem like VRP. The most beneficial parameters to be used on the Benchmark Instance runs are:

- Iterations = {500}

- Multi Local Search Objective = {min cost, min demand, random}
- Alpha (α) = {0.5, 0.8}
- Perturbation Percentage = {0.5, 0.8}

Considering these parameters, the number of combinations to be used for the Benchmark Instance considering the 150 instances are: $150 \cdot 1 \cdot 3 \cdot 2 \cdot 2 = 1800$ runs.

Before going to the last chapter, let us review the improvement of the contingency table due to the Parameter Calibration:

CPLEX \GRASP	Optimal/Feasible	Infeasible
Optimal/Feasible	89	37
Infeasible	0	54

Table 5.2: Contingency table comparing CPLEX and GRASP results after Parameter Calibration

We can calculate the Accuracy, Precision, Recall, and F1-score:

- Accuracy = $\frac{\text{True Positives+True Negative}}{\text{Total Instances}} = \frac{143}{180} = 79.4\%$
- Precision = $\frac{\text{True Positives}}{\text{True Positives+False Positives}} = \frac{89}{126} = 70.6\%$
- Recall = $\frac{\text{True Positives}}{\text{True Positives+False Negatives}} = \frac{84}{84} = 100\%$
- F1-Score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision+Recall}} = 2 \cdot \frac{0.706}{1.706} = 82.7\%$

Some insights:

The model has improved Accuracy, Precision, and F1-score while maintaining a perfect Recall. The increase in precision is particularly significant, indicating that the model is making fewer incorrect "Optimal/Feasible" predictions. These improvements suggest that the model is better calibrated to differentiate between "Optimal/Feasible" and "Infeasible" solutions, leading to more reliable predictions overall.

5.5 Description of Large Benchmark Instances and Their Interdiction Probabilities

5.5.1 Benchmark Instances

The capability for split deliveries is a key characteristic of VRPI and SDVRP. Therefore, the benchmark instances from Dror *et al.* (1994), as outlined in the article “Vehicle Routing with Split Deliveries," were chosen for this study. These instances share properties with VRPI, particularly regarding split delivery capabilities, making them ideal for our numerical experiments. Additionally, as Xu (2017) noted, VRPI is a variant of SDVRP.

To help us better understand the benchmark instances used in this study, we will present a table detailing their characteristics. This table will include essential features of each

instance, such as name, number of nodes (customers plus depot), total demand, number of available vehicles, vehicle capacity limits, instance tightness, whether split deliveries were part of the optimal solution (yes/no), and the optimal solution value for each instance.

Below, the benchmark instances table is shown:

Name	Nodes	TotDemand	VehNum	VehCap	Tightness	Split	OptimalValue
eil22	22	22500	4	6000	0.937	No	375
eil23	23	10189	3	4500	0.754	No	569
eil30	30	12750	3	4500	0.944	Yes	510
eil33	33	29370	4	8000	0.917	No	835
eil51	51	777	5	160	0.971	No	521
eilA76	76	1364	10	140	0.974	Yes	832
eilA101	101	1458	8	200	0.911	No	817
eilB76	76	1364	14	100	0.974	Yes	1023
eilB101	101	1458	14	112	0.929	Yes	1077
eilC76	76	1364	8	180	0.947	Yes	735
eilD76	76	1364	7	220	0.885	No	683
S51D1	51	402	3	160	0.837	No	458
S51D2	51	1415	9	160	0.982	Yes	726
S51D3	51	2275	15	160	0.947	Yes	972
S51D4	51	4317	27	160	0.999	Yes	1677
S51D5	51	3645	23	160	0.990	Yes	1440
S51D6	51	6459	41	160	0.984	Yes	2327
S76D1	76	614	4	160	0.959	No	594
S76D2	76	2383	15	160	0.992	Yes	1147
S76D3	76	3542	23	160	0.962	Yes	1474
S76D4	76	5765	37	160	0.973	Yes	2257
S101D1	101	788	5	160	0.985	No	716
S101D2	101	3064	20	160	0.957	Yes	1393
S101D3	101	4841	31	160	0.976	Yes	1975
S101D5	101	7679	48	160	0.999	Yes	2915

Table 5.3: SDVRP Benchmark Instances (Dror *et al.*, 1994)

The instances range from small-medium-sized instances with 22 nodes to larger instances with 101 nodes. Total demand and vehicle capacities vary widely, affecting each problem's complexity. Some instances allow split deliveries, impacting the solution strategy and optimal value.

The descriptive statistics of the tightness of the benchmark instances are described below:

- Mean: 0.940
- SD: 0.059
- Min: 0.754
- 25%: 0.917
- 50% (Median): 0.947
- 75%: 0.976
- Max: 0.999

These statistics provide a better understanding of the networks' saturation levels. The high mean and median values indicate that many instances are indeed quite tight, suggesting that the addition of interdiction probabilities will impact the feasibility of these instances.

5.5.2 Interdiction Probabilities

As the Small Randomly Generated Instances section describes, we will assess six different interdiction probability scenarios between each pair of nodes:

- Scenario 1: 0% interdiction probability
- Scenario 2: Between 1% – 5% interdiction probability, Uniformly distributed
- Scenario 3: Between 1% – 10% interdiction probability, Uniformly distributed
- Scenario 4: Between 1% – 20% interdiction probability, Uniformly distributed
- Scenario 5: Between 1% – 30% interdiction probability, Uniformly distributed
- Scenario 6: Between 1% – 40% interdiction probability, Uniformly distributed

Considering that we have 25 different benchmark instances and six different interdiction probabilities scenarios, the total number of benchmark instances with interdiction probabilities to assess is $25 \cdot 6 = 150$. The characteristics of these 150 scenarios can be seen in Appendix E.

5.6 GRASP Results on Large Benchmark Instances

The results of GRASP-H applied to Large Benchmark Instances can be found in Appendix F. Due to there are no Benchmark results for the cases with Interdiction, we will make plots with and without Interdiction to compare the results.

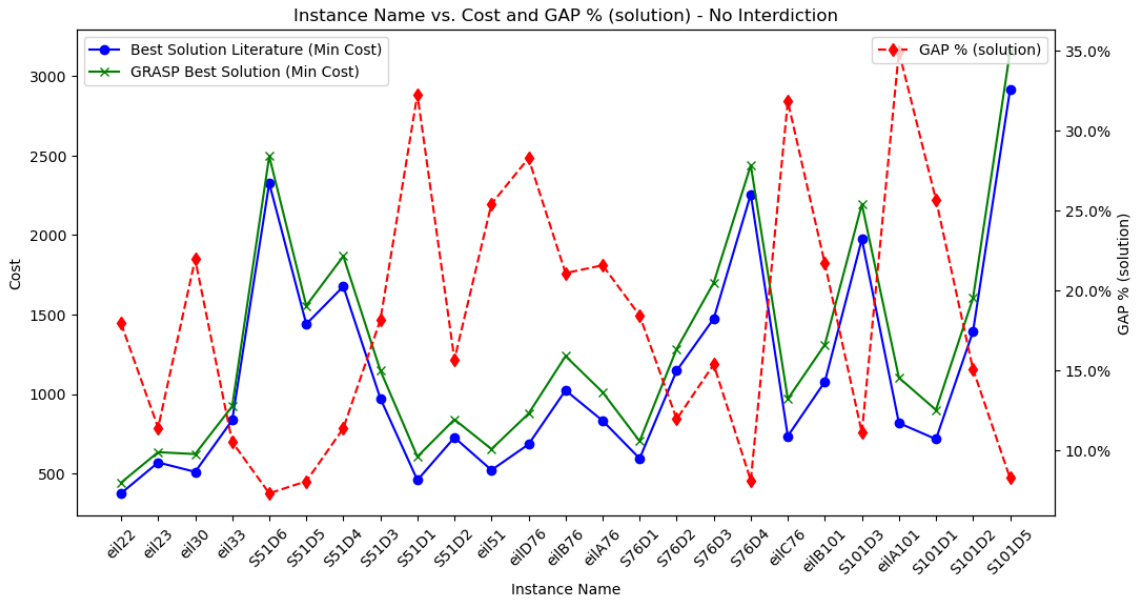

Figure 5.5: Solution Quality Benchmark without Interdiction

Some insights:

- The cost values (both from literature and GRASP) generally increase as the instance size increases. This is expected because larger instances typically involve more complex routing and demand satisfaction challenges, leading to higher overall costs
- The literature solutions provide a benchmark for the lowest known costs. GRASP closely matches or even reaches the literature solutions for smaller instances. However, as the instance size increases, the difference between the literature and GRASP solutions becomes more pronounced
- GRASP generally performs well for smaller instances, closely aligning with the

solutions in the literature. But for larger instances, the GRASP solutions tend to diverge from the literature, indicating that the heuristic may struggle more with larger, more complex problems

- The GAP between the literature solutions and GRASP's best solutions tends to increase as the instance size increases. This suggests that GRASP is effective for smaller instances, but its efficiency diminishes as the problem grows. This could be due to the increased complexity and the larger solution space that GRASP must navigate

- There are specific points where the GAP is particularly large, even when the instance size is only moderately increased. These outliers might indicate that specific instances have characteristics (e.g., higher tightness, complex customer distributions, specific demands, or network structure) that make them particularly challenging for GRASP

- As instances grow larger, the GRASP algorithm appears to face scalability issues. The algorithm might not be as effective at exploring the solution space thoroughly or might require more sophisticated adjustments (e.g., more iterations, different local search methods) to maintain performance comparable to the literature solutions

- The increasing GAP suggests room for improvement in the GRASP approach, particularly in handling larger instances. This could involve better Parameter Calibration, more advanced local search techniques, or hybrid methods that combine GRASP with other optimization strategies

Now, instead of plotting the Solution GAP, let us choose the tightness as the secondary Y-axis:

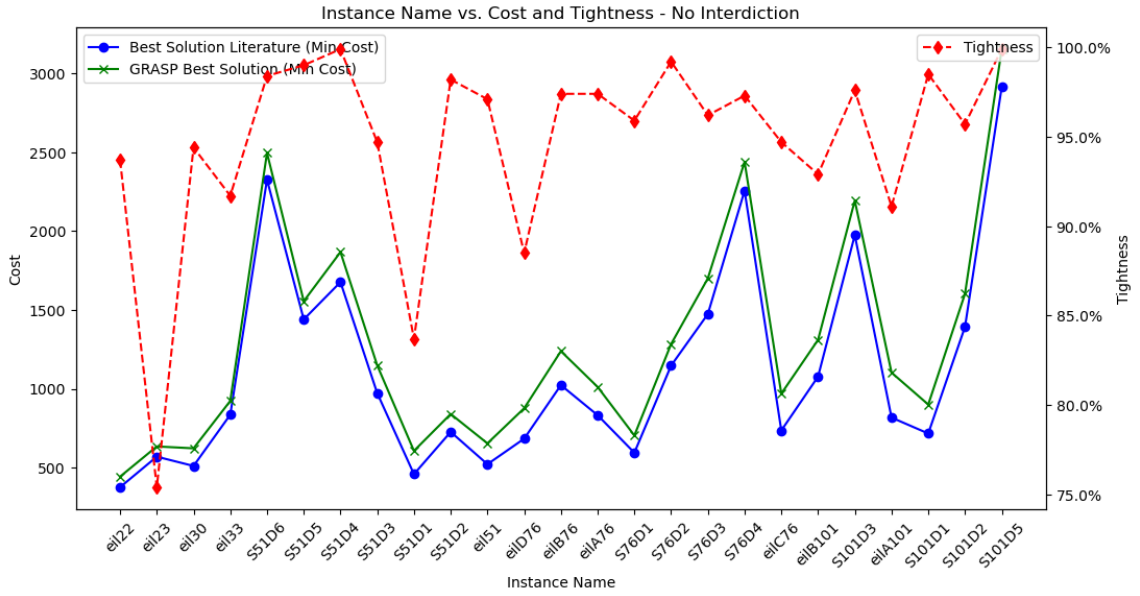

Figure 5.6: Solution Quality Benchmark without Interdiction and Tightness

In the figure, we can see a relationship between the solution and tightness. To confirm this trend, let us calculate the Spearman Correlation and its p-value. We are using Spearman instead of Pearson due to its stronger assumptions:

- Does not assume a linear relationship
- Does not require the data to be normally distributed
- Measures the strength and direction of the monotonic relationship (variables move in the same direction, but not necessarily at a constant rate)

When calculating the correlation, the coefficient between tightness and GRASP's best solution is 60%, which is strong, according to Cohen (1988). Its p-value is 0.0018, indicating strong evidence against the null hypothesis, suggesting that there is a significant monotonic relationship between the two variables. The observed correlation is unlikely to have occurred by chance.

Now, if we analyze the time elapsed vs cost, we yield the following plot:

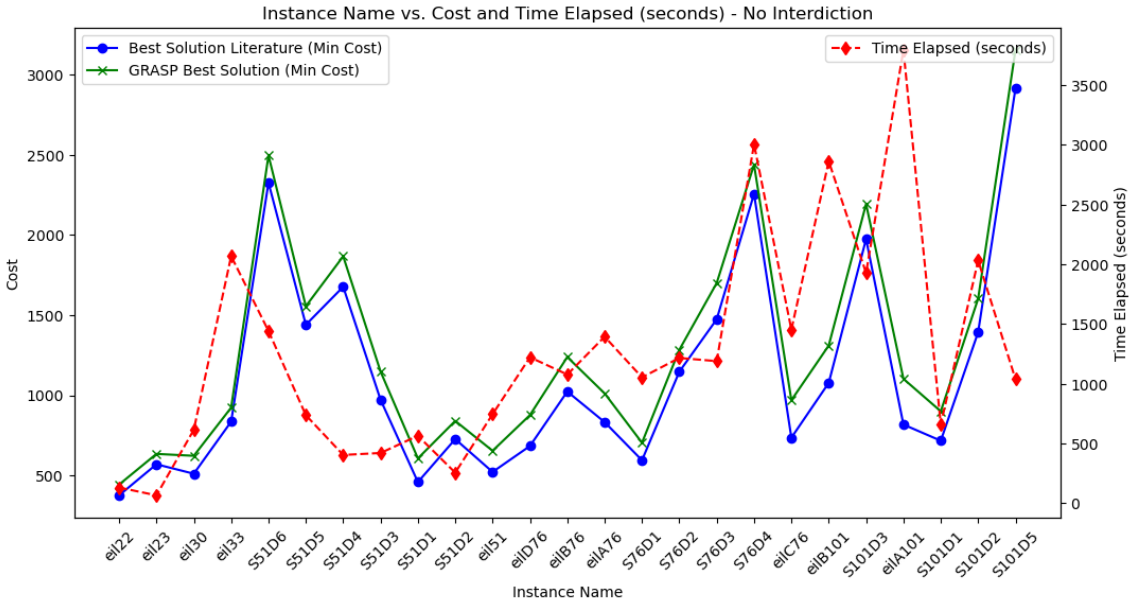

Figure 5.7: Elapsed Time Benchmark without Interdiction

Some insights:

- The time elapsed for smaller instances is relatively low, indicating that GRASP

efficiently solves smaller problems quickly. This efficiency is crucial in scenarios where computation time is a constraint

- As the instances grow in size, the time required to reach a solution increases significantly. There are instances with a high rise in time elapsed, reaching 3500 seconds. This rise suggests that while GRASP remains competitive in finding good solutions, the computational effort required increases substantially as problem complexity grows

- There appears to be a correlation between the time elapsed and the complexity of the instance. Generally, as the instances become larger and more complex, the cost and the time taken increase. However, there are cases where the cost does not increase as much despite the rise in time elapsed, which might suggest diminishing returns in terms of cost improvement for the time invested

When calculating the correlation between GRASP's best solution and time elapsed, the coefficient is 50%, which is strong according to Cohen (1988). Its p-value is 0.013, indicating strong evidence against the null hypothesis, suggesting that there is a significant monotonic relationship between the two variables. The observed correlation is unlikely to have occurred by chance.

Now, we plot the quality solution with and without interdiction and the GAP %:

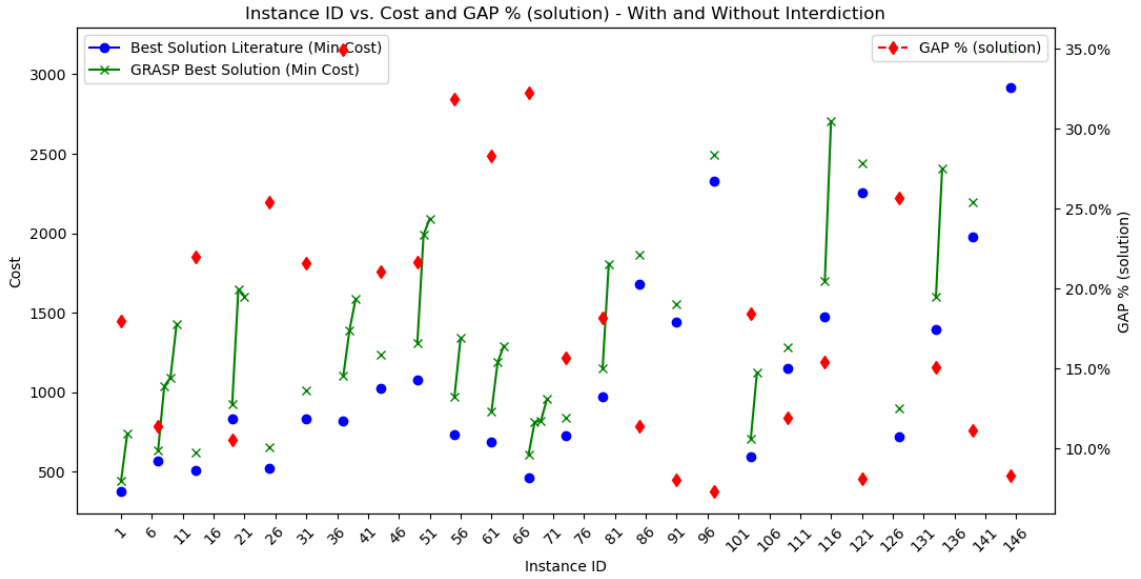

Figure 5.8: Solution Quality Benchmark with Interdiction

The only thing the plot clearly shows is that the GAP tends to increase when interdiction probabilities are present, and the instance size increases as well. Let's generate another plot showing tightness instead of GAP and calculate the correlation between the quality solution and tightness.

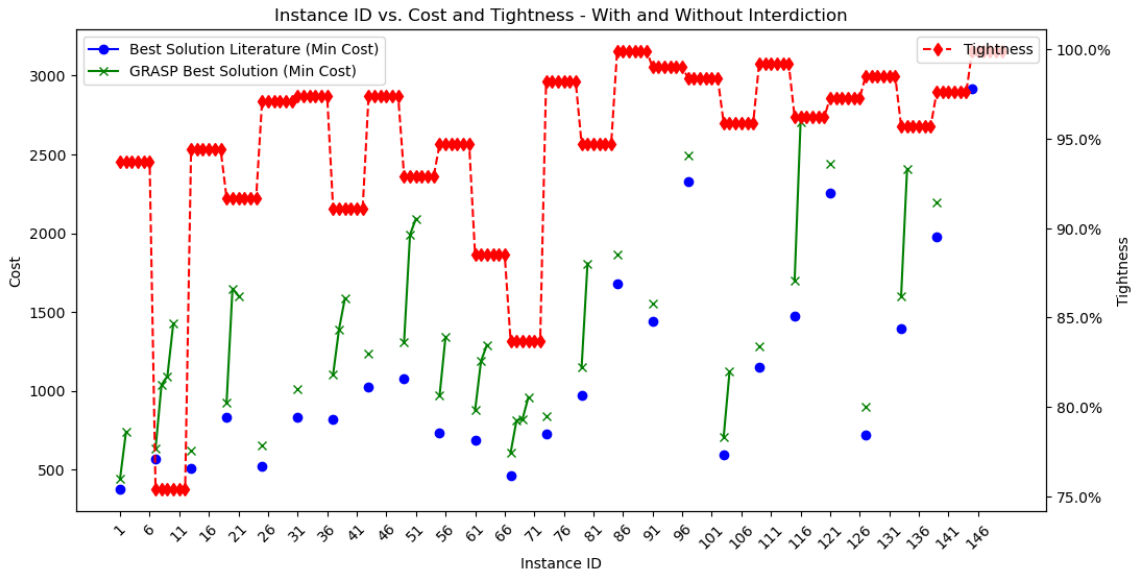

Figure 5.9: Solution Quality Benchmark with Interdiction and Tightness

There seems to be a relationship between GRASP's best solution and Tightness. Let's calculate the Spearman correlation and its p-value. The correlation was 40%, which is moderate by Cohen (1988), and the p-value was 0.0145, indicating strong evidence against the null hypothesis. This suggests that there is a significant monotonic relationship between the two variables. The observed correlation is unlikely to have occurred by chance.

Finally, let's plot the quality of the solution against the time it takes to achieve it using GRASP:

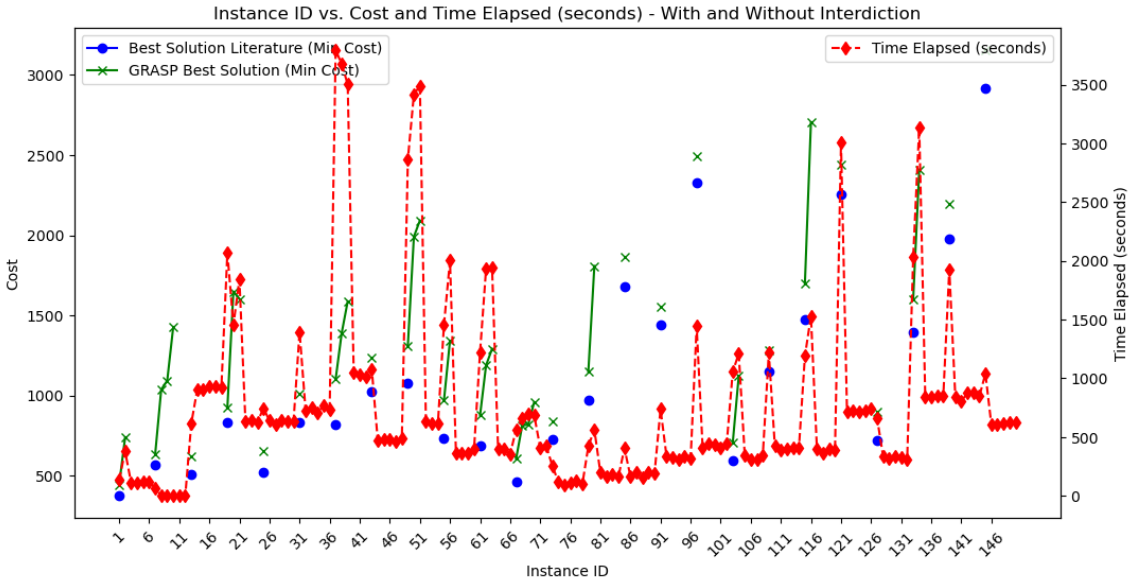

Figure 5.10: Elapsed Time Benchmark without Interdiction

Some insights:

- The time required to find the GRASP solution, shown by the red dashed line, generally increases with instance size. However, there are significant fluctuations in time, suggesting variability in the computational effort required across different instances of similar sizes
- Some larger instances have a higher increase in time elapsed, indicating that GRASP may require much more time as complexity increases, especially where the cost is also higher
- The instances with higher costs, particularly those where GRASP deviates significantly from the literature's best solutions, tend to have higher computation times.

This correlation suggests that more challenging instances not only lead to worse solutions but also require more computational effort

- The plot shows instances where GRASP closely matches the literature in cost but with significantly lower computation time, indicating efficient performance on certain problem types
- As the instances grow, the gap between GRASP and literature solutions widens. This trend suggests that while GRASP is effective for smaller or less complex problems, its effectiveness diminishes for more extensive and complex scenarios, both in solution quality and time efficiency

Like in previous plots, there seems to be a relationship between the time elapsed and the quality of the GRASP solution. Let us calculate this correlation. The Spearman correlation of these two variables is 50%, which is strong according to Cohen (1988), and its p-value is 0.0006, indicating strong evidence against the null hypothesis and suggesting a significant monotonic relationship between the two variables. The observed correlation is unlikely to have occurred by chance.

Finally, reviewing the results in Appendix F, we can see that cost minimization was the most used Multi Local Search Objective for scenarios without interdiction. This is intuitive when there are no interdiction events because the objective is to minimize the traveling costs. On the other hand, when interdiction is present, the most common Multi Local Search Objective used was the minimum demand, reaffirming the importance of developing this logic during the local search procedure in GRASP-H.

Chapter 6

Conclusions and Future Research Directions

6.1 Main Conclusions

This thesis presented a comprehensive approach to solving the VRPI, both using exact and heuristic methods. The research's primary contributions include implementing the GRASP metaheuristic on large-scale benchmark instances, developing a more comprehensive local search procedure with a 3-fold logic-minimizing cost, minimizing interdiction, and minimizing demand- which was instrumental to scaling the GRASP-H implementation to solve large benchmark instances, conducting a detailed feasibility analysis for small instances, and creating a flexible framework for various VRP variants.

The results indicate that:

1. Efficiency of GRASP: The GRASP algorithm demonstrated a reasonably robust performance on both small and large instances of VRPI. It provided high-quality solutions with minimal computational resources, particularly compared to CPLEX, which struggles with larger instances.

2. Benchmarking and Generalization: This study validated the generalization of GRASP-H implementation to large-scale benchmark instances in most cases. This is a significant advancement over previous implementations focused on small, randomly generated instances.

3. Impact of Interdiction Probabilities: The analysis highlighted the sensitivity of VRPI solutions to interdiction probabilities, particularly in scenarios with tight capacity constraints. The introduction of a 3-fold logic in GRASP-H (minimizing cost, minimizing interdiction, and minimizing demand) was essential facing these challenging scenarios, particularly for Benchmark Instances where the minimum demand logic was the most successful overall.

The findings from this research have significant implications in OR. Academically speaking, developing a flexible GRASP framework for various VRP variants offers a valuable tool for further research. Additionally, the insights generated from this study can be directly applied to logistics and supply chain management, particularly in high-risk environments where potential route interdictions are a real concern.

6.2 Future Research Directions

The analysis presented in this thesis offers valuable insights into the potential future research directions in the VRPI domain. One promising direction is the creation of benchmark instances designed specifically for VRPI. These instances will enable a more robust comparison between GRASP solutions and those found in the existing literature. This comparison would provide evidence of the effectiveness of GRASP-H in solving VRPI, highlighting its strengths and identifying areas for improvement.

Another significant direction for future research is analyzing how the variability of interdiction probabilities—characterized by a higher standard deviation—would affect both solution quality and computational time. Understanding this relationship is critical, as it can help develop more accurate models that account for the variability in interdiction probabilities across different customers in a network. This understanding could lead to the formulation of more robust routing strategies that could perform well, even under conditions of high uncertainty due to interdiction.

Moreover, another relevant direction would be exploring exact methods, such as Dynamic Programming and branch-and-price, for solving large instances of VRPI. These methods could complement the heuristic approaches by providing optimal solutions or strong bounds that can be used to validate the effectiveness of GRASP-H and other metaheuristic algorithms. It is important to note that for small instances, Dynamic Programming could be implemented successfully, but for large instances there are several challenges to take into account:

- State explosion, where the number of states grows exponentially due to the combination of customers, routes, and interdiction scenarios
- The complexity of evaluating interdiction scenarios, as each possible interdiction creates a new routing subproblem
- The increasing complexity of the Bellman equation due to the involvement of interdiction probabilities

For more information about Dynamic Programming, please refer to Bertsekas (2012).

Beyond future research directions, a potential business opportunity can be explored. The GRASP development implemented in this thesis offers a significant business opportunity, especially in dealing with real-time routing challenges. If it is scaled properly to handle larger instances, the implementation can be packaged and deployed through an API that connects to real-time data. By doing this, the algorithm would be able to dynamically respond to disruptions, like road closures or delays, and subsequently generate alternative routes quickly. This capability would enable a delivery truck to efficiently adjust its path and continue serving customers with minimal delays, therefore offering substantial value in industries that relies on timely deliveries. Consequently, it would contribute to improving operational efficiency and resilience against unexpected disruptions.

In conclusion, future research directions should focus on generating benchmark instances, investigating the impact of interdiction probability variability, and applying

exact methods to solve large VRPI instances. These directions will improve the understanding of VRPI and contribute to developing more effective solution approaches to solving this problem.

Appendix A

Small Randomly Generated Instances and Interdiction Probabilities

ID	Customers	# Vehicles	Capacity	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
1	3	2	10	60%	0%	0.00%	0.00%
2	3	2	10	60%	1%-5%	3.04%	1.16%
3	3	2	10	60%	1%-10%	5.51%	2.39%
4	3	2	10	60%	1%-20%	10.58%	5.91%
5	3	2	10	60%	1%-30%	15.50%	8.91%
6	3	2	10	60%	1%-40%	20.64%	10.82%
7	3	2	10	70%	0%	0.00%	0.00%
8	3	2	10	70%	1%-5%	3.04%	1.16%
9	3	2	10	70%	1%-10%	5.51%	2.39%
10	3	2	10	70%	1%-20%	10.58%	5.91%
11	3	2	10	70%	1%-30%	15.50%	8.91%
12	3	2	10	70%	1%-40%	20.64%	10.82%
13	3	2	10	80%	0%	0.00%	0.00%
14	3	2	10	80%	1%-5%	3.04%	1.16%
15	3	2	10	80%	1%-10%	5.51%	2.39%
16	3	2	10	80%	1%-20%	10.58%	5.91%
17	3	2	10	80%	1%-30%	15.50%	8.91%
18	3	2	10	80%	1%-40%	20.64%	10.82%
19	3	2	10	90%	0%	0.00%	0.00%
20	3	2	10	90%	1%-5%	3.04%	1.16%

Continued on next page

Table A.1 – continued from previous page

ID	Customers	# Vehicles	Capacity	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
21	3	2	10	90%	1%-10%	5.51%	2.39%
22	3	2	10	90%	1%-20%	10.58%	5.91%
23	3	2	10	90%	1%-30%	15.50%	8.91%
24	3	2	10	90%	1%-40%	20.64%	10.82%
25	3	2	10	95%	0%	0.00%	0.00%
26	3	2	10	95%	1%-5%	3.04%	1.16%
27	3	2	10	95%	1%-10%	5.51%	2.39%
28	3	2	10	95%	1%-20%	10.58%	5.91%
29	3	2	10	95%	1%-30%	15.50%	8.91%
30	3	2	10	95%	1%-40%	20.64%	10.82%
31	3	2	10	99%	0%	0.00%	0.00%
32	3	2	10	99%	1%-5%	3.04%	1.16%
33	3	2	10	99%	1%-10%	5.51%	2.39%
34	3	2	10	99%	1%-20%	10.58%	5.91%
35	3	2	10	99%	1%-30%	15.50%	8.91%
36	3	2	10	99%	1%-40%	20.64%	10.82%
37	4	2	10	90%	0%	0.00%	0.00%
38	4	2	10	90%	1%-5%	3.00%	1.19%
39	4	2	10	90%	1%-10%	5.48%	2.53%
40	4	3	10	60%	1%-20%	10.53%	5.88%
41	4	3	10	60%	1%-30%	15.51%	9.25%
42	4	3	10	60%	1%-40%	20.73%	11.17%
43	4	3	10	70%	0%	0.00%	0.00%
44	4	3	10	70%	1%-5%	3.00%	1.19%
45	4	3	10	70%	1%-10%	5.48%	2.53%
46	4	3	10	70%	1%-20%	10.53%	5.88%
47	4	3	10	70%	1%-30%	15.51%	9.25%
48	4	3	10	70%	1%-40%	20.73%	11.17%
49	4	3	10	80%	0%	0.00%	0.00%
50	4	3	10	80%	1%-5%	3.00%	1.19%
51	4	3	10	80%	1%-10%	5.48%	2.53%
52	4	3	10	80%	1%-20%	10.53%	5.88%
53	4	3	10	80%	1%-30%	15.51%	9.25%
54	4	3	10	80%	1%-40%	20.73%	11.17%
55	4	3	10	90%	0%	0.00%	0.00%
56	4	3	10	90%	1%-5%	3.00%	1.19%
57	4	3	10	90%	1%-10%	5.48%	2.53%
58	4	3	10	90%	1%-20%	10.53%	5.88%
59	4	3	10	90%	1%-30%	15.51%	9.25%
60	4	3	10	90%	1%-40%	20.73%	11.17%
61	4	3	10	95%	0%	0.00%	0.00%
62	4	3	10	95%	1%-5%	3.00%	1.19%
63	4	3	10	95%	1%-10%	5.48%	2.53%

Continued on next page

Table A.1 – continued from previous page

ID	Customers	# Vehicles	Capacity	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
64	4	3	10	95%	1%-20%	10.53%	5.88%
65	4	3	10	95%	1%-30%	15.51%	9.25%
66	4	3	10	95%	1%-40%	20.73%	11.17%
67	4	3	10	99%	0%	0.00%	0.00%
68	4	3	10	99%	1%-5%	3.00%	1.19%
69	4	3	10	99%	1%-10%	5.48%	2.53%
70	4	3	10	99%	1%-20%	10.53%	5.88%
71	4	3	10	99%	1%-30%	15.51%	9.25%
72	4	3	10	99%	1%-40%	20.73%	11.17%
73	5	2	10	90%	0%	0.00%	0.00%
74	5	2	10	90%	1%-5%	3.01%	1.20%
75	5	2	10	90%	1%-10%	5.49%	2.72%
76	5	3	10	60%	1%-20%	10.52%	5.60%
77	5	3	10	60%	1%-30%	15.61%	9.08%
78	5	3	10	60%	1%-40%	20.40%	11.81%
79	5	3	10	70%	0%	0.00%	0.00%
80	5	3	10	70%	1%-5%	3.01%	1.20%
81	5	3	10	70%	1%-10%	5.49%	2.72%
82	5	3	10	70%	1%-20%	10.52%	5.60%
83	5	3	10	70%	1%-30%	15.61%	9.08%
84	5	3	10	70%	1%-40%	20.40%	11.81%
85	5	3	10	80%	0%	0.00%	0.00%
86	5	3	10	80%	1%-5%	3.01%	1.20%
87	5	3	10	80%	1%-10%	5.49%	2.72%
88	5	3	10	80%	1%-20%	10.52%	5.60%
89	5	3	10	80%	1%-30%	15.61%	9.08%
90	5	3	10	80%	1%-40%	20.40%	11.81%
91	5	3	10	90%	0%	0.00%	0.00%
92	5	3	10	90%	1%-5%	3.01%	1.20%
93	5	3	10	90%	1%-10%	5.49%	2.72%
94	5	3	10	90%	1%-20%	10.52%	5.60%
95	5	3	10	90%	1%-30%	15.61%	9.08%
96	5	3	10	90%	1%-40%	20.40%	11.81%
97	5	3	10	95%	0%	0.00%	0.00%
98	5	3	10	95%	1%-5%	3.01%	1.20%
99	5	3	10	95%	1%-10%	5.49%	2.72%
100	5	3	10	95%	1%-20%	10.52%	5.60%
101	5	3	10	95%	1%-30%	15.61%	9.08%
102	5	3	10	95%	1%-40%	20.40%	11.81%
103	5	3	10	99%	0%	0.00%	0.00%
104	5	3	10	99%	1%-5%	3.01%	1.20%
105	5	3	10	99%	1%-10%	5.49%	2.72%
106	5	3	10	99%	1%-20%	10.52%	5.60%

Continued on next page

Table A.1 – continued from previous page

ID	Customers	# Vehicles	Capacity	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
107	5	3	10	99%	1%-30%	15.61%	9.08%
108	5	3	10	99%	1%-40%	20.40%	11.81%
109	6	2	10	90%	0%	0.00%	0.00%
110	6	2	10	90%	1%-5%	3.10%	1.17%
111	6	2	10	90%	1%-10%	5.60%	2.72%
112	6	3	10	60%	1%-20%	10.46%	5.79%
113	6	3	10	60%	1%-30%	15.52%	8.84%
114	6	3	10	60%	1%-40%	20.58%	11.53%
115	6	3	10	70%	0%	0.00%	0.00%
116	6	3	10	70%	1%-5%	3.10%	1.17%
117	6	3	10	70%	1%-10%	5.60%	2.72%
118	6	3	10	70%	1%-20%	10.46%	5.79%
119	6	3	10	70%	1%-30%	15.52%	8.84%
120	6	3	10	70%	1%-40%	20.58%	11.53%
121	6	3	10	80%	0%	0.00%	0.00%
122	6	3	10	80%	1%-5%	3.10%	1.17%
123	6	3	10	80%	1%-10%	5.60%	2.72%
124	6	3	10	80%	1%-20%	10.46%	5.79%
125	6	3	10	80%	1%-30%	15.52%	8.84%
126	6	3	10	80%	1%-40%	20.58%	11.53%
127	6	3	10	90%	0%	0.00%	0.00%
128	6	3	10	90%	1%-5%	3.10%	1.17%
129	6	3	10	90%	1%-10%	5.60%	2.72%
130	6	3	10	90%	1%-20%	10.46%	5.79%
131	6	3	10	90%	1%-30%	15.52%	8.84%
132	6	3	10	90%	1%-40%	20.58%	11.53%
133	6	3	10	95%	0%	0.00%	0.00%
134	6	3	10	95%	1%-5%	3.10%	1.17%
135	6	3	10	95%	1%-10%	5.60%	2.72%
136	6	3	10	95%	1%-20%	10.46%	5.79%
137	6	3	10	95%	1%-30%	15.52%	8.84%
138	6	3	10	95%	1%-40%	20.58%	11.53%
139	6	3	10	99%	0%	0.00%	0.00%
140	6	3	10	99%	1%-5%	3.10%	1.17%
141	6	3	10	99%	1%-10%	5.60%	2.72%
142	6	3	10	99%	1%-20%	10.46%	5.79%
143	6	3	10	99%	1%-30%	15.52%	8.84%
144	6	3	10	99%	1%-40%	20.58%	11.53%
145	7	3	10	80%	0%	0.00%	0.00%
146	7	3	10	80%	1%-5%	3.03%	1.17%
147	7	3	10	80%	1%-10%	5.55%	2.62%
148	7	3	10	80%	1%-20%	10.48%	5.62%
149	7	4	10	60%	1%-30%	15.67%	8.28%

Continued on next page

Table A.1 – continued from previous page

ID	Customers	# Vehicles	Capacity	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
150	7	4	10	60%	1%-40%	20.58%	11.73%
151	7	3	10	93%	0%	0.00%	0.00%
152	7	4	10	70%	1%-5%	3.03%	1.17%
153	7	4	10	70%	1%-10%	5.55%	2.62%
154	7	4	10	70%	1%-20%	10.48%	5.62%
155	7	4	10	70%	1%-30%	15.67%	8.28%
156	7	4	10	70%	1%-40%	20.58%	11.73%
157	7	4	10	80%	0%	0.00%	0.00%
158	7	4	10	80%	1%-5%	3.03%	1.17%
159	7	4	10	80%	1%-10%	5.55%	2.62%
160	7	4	10	80%	1%-20%	10.48%	5.62%
161	7	4	10	80%	1%-30%	15.67%	8.28%
162	7	4	10	80%	1%-40%	20.58%	11.73%
163	7	4	10	90%	0%	0.00%	0.00%
164	7	4	10	90%	1%-5%	3.03%	1.17%
165	7	4	10	90%	1%-10%	5.55%	2.62%
166	7	4	10	90%	1%-20%	10.48%	5.62%
167	7	4	10	90%	1%-30%	15.67%	8.28%
168	7	4	10	90%	1%-40%	20.58%	11.73%
169	7	4	10	95%	0%	0.00%	0.00%
170	7	4	10	95%	1%-5%	3.03%	1.17%
171	7	4	10	95%	1%-10%	5.55%	2.62%
172	7	4	10	95%	1%-20%	10.48%	5.62%
173	7	4	10	95%	1%-30%	15.67%	8.28%
174	7	4	10	95%	1%-40%	20.58%	11.73%
175	7	4	10	99%	0%	0.00%	0.00%
176	7	4	10	99%	1%-5%	3.03%	1.17%
177	7	4	10	99%	1%-10%	5.55%	2.62%
178	7	4	10	99%	1%-20%	10.48%	5.62%
179	7	4	10	99%	1%-30%	15.67%	8.28%
180	7	4	10	99%	1%-40%	20.58%	11.73%

Appendix B

Feasibility Results on Small Randomly Generated Instances with CPLEX

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries
1	294.738711	0.114847183	Optimal	No
2	294.738711	0.151495934	Optimal	No
3	294.738711	0.162949324	Optimal	No
4	297.3297176	0.147966146	Optimal	No
5	312.8078168	0.182559013	Optimal	No
6	312.8078168	0.284761429	Optimal	No
7	297.3297176	0.120128393	Optimal	No
8	297.3297176	0.693101883	Optimal	No
9	297.3297176	0.137248278	Optimal	No
10	312.8078168	0.500146866	Optimal	No
11	327.4588615	0.145730972	Optimal	Yes
12	327.4588615	0.606101513	Optimal	Yes
13	312.8078168	0.105967999	Optimal	No
14	312.8078168	0.178212881	Optimal	No
15	312.8078168	0.160049438	Optimal	No
16	312.8078168	0.133422613	Optimal	No
17	427.4129121	0.12820363	Optimal	Yes
18	NA	0.093062639	Infeasible	NA
19	327.4588615	0.108874798	Optimal	Yes
20	327.4588615	0.123351097	Optimal	Yes

Continued on next page

Table B.1 – continued from previous page

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries
21	327.4588615	0.146835566	Optimal	Yes
22	427.6003776	0.147563934	Optimal	Yes
23	NA	0.081897974	Infeasible	NA
24	NA	0.085934401	Infeasible	NA
25	327.4588615	0.102074623	Optimal	Yes
26	327.4588615	0.137180567	Optimal	Yes
27	NA	0.121040344	Infeasible	NA
28	NA	0.095350981	Infeasible	NA
29	NA	0.088197708	Infeasible	NA
30	NA	0.143119097	Infeasible	NA
31	327.4588615	0.155112505	Optimal	Yes
32	NA	0.083840847	Infeasible	NA
33	NA	0.104273558	Infeasible	NA
34	NA	0.081512213	Infeasible	NA
35	NA	0.09070611	Infeasible	NA
36	NA	0.076552153	Infeasible	NA
37	447.0861519	0.148379087	Optimal	No
38	447.0861519	0.170269728	Optimal	No
39	447.0861519	0.187201738	Optimal	No
40	469.419194	0.195721865	Optimal	No
41	487.5257403	0.192256212	Optimal	No
42	487.5257403	0.174651384	Optimal	No
43	469.419194	0.153213978	Optimal	No
44	487.5257403	0.178795338	Optimal	No
45	487.5257403	0.206089735	Optimal	No
46	487.5257403	0.27561307	Optimal	No
47	487.5257403	0.187109232	Optimal	No
48	487.5257403	0.330114126	Optimal	No
49	487.5257403	0.6440413	Optimal	No
50	487.5257403	0.174844503	Optimal	No
51	487.5257403	0.177227974	Optimal	No
52	503.3677074	0.142822504	Optimal	Yes
53	503.3677074	0.176370621	Optimal	Yes
54	503.3677074	0.162853241	Optimal	Yes
55	503.3677074	0.143475533	Optimal	Yes
56	487.5257403	0.174844503	Optimal	No
57	503.3677074	0.170776367	Optimal	Yes
58	593.6224088	0.183358908	Optimal	Yes
59	NA	0.117836714	Infeasible	NA
60	NA	0.149119139	Infeasible	NA
61	503.3677074	0.115825176	Optimal	Yes
62	503.3677074	0.198811293	Optimal	Yes
63	736.6880611	0.180307627	Optimal	Yes

Continued on next page

Table B.1 – continued from previous page

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries
64	NA	0.128530264	Infeasible	NA
65	NA	0.106292963	Infeasible	NA
66	NA	0.096591949	Infeasible	NA
67	503.3677074	0.113938808	Optimal	Yes
68	NA	0.107517958	Infeasible	NA
69	NA	0.098725319	Infeasible	NA
70	NA	0.108216524	Infeasible	NA
71	NA	0.123418331	Infeasible	NA
72	NA	0.104197979	Infeasible	NA
73	497.7004479	0.266986847	Optimal	No
74	497.7004479	0.479599476	Optimal	No
75	541.8781896	0.484283447	Optimal	No
76	541.8781896	0.412248611	Optimal	No
77	638.8653775	0.683982611	Optimal	Yes
78	638.8653775	0.803528786	Optimal	Yes
79	541.8781896	0.236293554	Optimal	No
80	638.8653775	0.802109718	Optimal	Yes
81	638.8653775	0.969558477	Optimal	Yes
82	638.8653775	0.543526888	Optimal	Yes
83	685.2671764	0.569363117	Optimal	Yes
84	649.853598	0.46057415	Optimal	No
85	638.8653775	0.301715136	Optimal	Yes
86	638.8653775	0.684202433	Optimal	Yes
87	649.853598	0.629189253	Optimal	No
88	685.2671764	2.200120449	Optimal	Yes
89	685.2671764	0.582645178	Optimal	Yes
90	655.063012	0.511090517	Optimal	Yes
91	649.853598	0.272328377	Optimal	No
92	685.2671764	0.524552822	Optimal	Yes
93	685.2671764	0.536679029	Optimal	Yes
94	NA	0.308453321	Infeasible	NA
95	NA	0.399936914	Infeasible	NA
96	NA	0.416132927	Infeasible	NA
97	685.2671764	0.260633945	Optimal	Yes
98	719.5395481	0.560606718	Optimal	Yes
99	804.8739643	0.964958191	Optimal	Yes
100	NA	0.296108246	Infeasible	NA
101	NA	0.317147493	Infeasible	NA
102	NA	0.36856389	Infeasible	NA
103	719.5395481	0.279083014	Optimal	Yes
104	NA	0.333434582	Infeasible	NA
105	NA	0.312095642	Infeasible	NA
106	NA	0.293560028	Infeasible	NA

Continued on next page

Table B.1 – continued from previous page

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries
107	NA	0.322124004	Infeasible	NA
108	NA	0.320747614	Infeasible	NA
109	527.5973838	3.138249397	Optimal	No
110	527.5973838	17.44124818	Optimal	No
111	548.610515	12.76395917	Optimal	No
112	550.1773497	8.650572538	Optimal	No
113	563.1718898	21.47652841	Optimal	No
114	563.1718898	9.992511988	Optimal	No
115	551.0572991	2.449659109	Optimal	No
116	551.0572991	12.36286187	Optimal	No
117	563.1718898	8.335483551	Optimal	No
118	584.185021	7.566887617	Optimal	No
119	584.185021	10.29633451	Optimal	No
120	635.8645069	8.129265308	Optimal	No
121	563.1718898	2.371087551	Optimal	No
122	584.185021	11.19156718	Optimal	No
123	584.185021	5.291562557	Optimal	No
124	584.185021	6.681310892	Optimal	No
125	584.185021	7.33804512	Optimal	No
126	839.143073	19.24132967	Optimal	Yes
127	584.185021	1.71133399	Optimal	No
128	644.4538876	9.027630806	Optimal	No
129	648.5165937	15.99040771	Optimal	Yes
130	700.356807	8.724375248	Optimal	Yes
131	NA	1.34403801	Infeasible	NA
132	NA	1.376410961	Infeasible	NA
133	644.4538876	2.282662392	Optimal	No
134	648.5165937	9.967606783	Optimal	Yes
135	NA	1.313554764	Infeasible	NA
136	NA	9.761516571	Infeasible	NA
137	NA	1.36908555	Infeasible	NA
138	NA	1.309030533	Infeasible	NA
139	648.5165937	2.394912481	Optimal	Yes
140	NA	1.343605757	Infeasible	NA
141	NA	1.30096221	Infeasible	NA
142	NA	1.336131334	Infeasible	NA
143	NA	1.28545332	Infeasible	NA
144	NA	1.301508665	Infeasible	NA
145	741.4192694	73.566185	Optimal	No
146	741.4192694	21.4252367	Optimal	No
147	741.4192694	48.3084681	Optimal	No
148	741.4192694	53.6300509	Optimal	No
149	850.7519016	101.4203174	Optimal	No

Continued on next page

Table B.1 – continued from previous page

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries
150	791.3263079	115.5473423	Optimal	No
151	761.3174774	55.8280468	Optimal	Yes
152	791.3263079	18.43054104	Optimal	No
153	791.3263079	16.174685	Optimal	No
154	791.3263079	139.6958947	Optimal	No
155	916.7734361	56.42913508	Optimal	No
156	913.6814307	59.0814178	Optimal	Yes
157	791.3263079	33.76885748	Optimal	No
158	839.989866	12.90901566	Optimal	Yes
159	839.989866	12.98634601	Optimal	Yes
160	924.0339497	90.25597215	Optimal	Yes
161	990.7865498	99.64564395	Optimal	Yes
162	1032.06248	113.5510592	Optimal	Yes
163	924.0339497	52.39133358	Optimal	Yes
164	924.0339497	228.7071111	Optimal	Yes
165	965.5572091	108.573333	Optimal	Yes
166	1047.091978	133.7166901	Optimal	Yes
167	1363.063983	2288.204947	Optimal	Yes
168	NA	16.03871179	Infeasible	NA
169	924.0339497	61.01000333	Optimal	Yes
170	999.0891811	648.085897	Optimal	Yes
171	NA	9.273626566	Infeasible	NA
172	NA	9.475574017	Infeasible	NA
173	NA	10.2217288	Infeasible	NA
174	NA	10.17548871	Infeasible	NA
175	999.0891811	19.61138177	Optimal	Yes
176	NA	12.31623363	Infeasible	NA
177	NA	9.866729498	Infeasible	NA
178	NA	10.42457247	Infeasible	NA
179	NA	11.08601046	Infeasible	NA
180	NA	9.941188574	Infeasible	NA

Appendix C

GRASP Results on Small Randomly Generated Instances

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries	GRASP GAP % (solution)	GRASP GAP % (time)
1	294.738711	0.003398418	Optimal	No	0.00%	-97.04%
2	294.738711	0.002001047	Optimal	No	0.00%	-98.68%
3	294.738711	0.023235321	Optimal	No	0.00%	-85.74%
4	297.3297176	0.003050566	Optimal	No	0.00%	-97.94%
5	312.8078168	0.007917166	Optimal	No	0.00%	-95.66%
6	312.8078168	0.012206078	Optimal	No	0.00%	-95.71%
7	297.3297176	0.002357006	Optimal	No	0.00%	-98.04%
8	297.3297176	0.00399828	Optimal	No	0.00%	-99.42%
9	297.3297176	0.002005816	Optimal	No	0.00%	-98.54%
10	312.8078168	0.009782314	Optimal	No	0.00%	-98.04%
11	427.4129121	0.00201416	Feasible	Yes	30.52%	-98.62%
12	427.4129121	0.118281126	Feasible	Yes	30.52%	-80.48%
13	312.8078168	0.00312376	Optimal	No	0.00%	-97.05%
14	312.8078168	0.00404954	Optimal	No	0.00%	-97.73%
15	312.8078168	0.002007246	Optimal	No	0.00%	-98.75%
16	312.8078168	0.001999617	Optimal	No	0.00%	-98.50%
17	NA	0.121643305	Infeasible	NA	NA	-5.12%
18	NA	0.060529947	Infeasible	NA	NA	-34.96%
19	427.4129121	0.001999617	Feasible	Yes	30.52%	-98.16%
20	427.4129121	0.002581358	Feasible	Yes	30.52%	-97.91%
21	427.4129121	0.001999378	Feasible	Yes	30.52%	-98.64%

Continued on next page

Table C.1 – continued from previous page

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries		
22	NA	0.11707592	Infeasible	NA	NA	-20.66%
23	NA	0.144191027	Infeasible	NA	NA	76.06%
24	NA	0.122408152	Infeasible	NA	NA	42.44%
25	427.4129121	0.001129627	Feasible	Yes	30.52%	-98.89%
26	NA	0.145313501	Infeasible	NA	NA	5.93%
27	NA	0.120705128	Infeasible	NA	NA	-0.28%
28	NA	0.119689465	Infeasible	NA	NA	25.53%
29	NA	0.10966754	Infeasible	NA	NA	24.34%
30	NA	0.135070324	Infeasible	NA	NA	-5.62%
31	427.4129121	0.001520157	Feasible	Yes	30.52%	-99.02%
32	NA	0.120195866	Infeasible	NA	NA	43.36%
33	NA	0.130620718	Infeasible	NA	NA	25.27%
34	NA	0.118313551	Infeasible	NA	NA	45.15%
35	NA	0.13193655	Infeasible	NA	NA	45.45%
36	NA	0.16578269	Infeasible	NA	NA	116.56%
37	447.0861519	0.00566864	Optimal	No	0.00%	-96.18%
38	447.0861519	0.007341862	Optimal	No	0.00%	-95.69%
39	447.0861519	0.013188839	Optimal	No	0.00%	-92.95%
40	NA	0.196428776	Infeasible	NA	NA	0.36%
41	NA	0.203747272	Infeasible	NA	NA	5.98%
42	NA	0.239966393	Infeasible	NA	NA	37.40%
43	469.419194	0.00565362	Optimal	No	0.00%	-96.31%
44	487.5257403	0.011056185	Optimal	No	0.00%	-93.82%
45	487.5257403	0.017591238	Optimal	No	0.00%	-91.46%
46	NA	0.276374578	Infeasible	NA	NA	0.28%
47	487.5257403	0.015542269	Optimal	No	0.00%	-91.69%
48	487.5257403	0.028864384	Optimal	No	0.00%	-91.26%
49	487.5257403	0.005546331	Optimal	No	0.00%	-99.14%
50	487.5257403	0.015582561	Optimal	No	0.00%	-91.09%
51	487.5257403	0.021350384	Optimal	No	0.00%	-87.95%
52	719.7411575	1.26612258	Feasible	Yes	42.99%	786.50%
53	641.7810893	0.015180111	Feasible	Yes	27.50%	-91.39%
54	719.7411575	0.934924364	Feasible	Yes	42.99%	474.09%
55	664.8870894	0.001999617	Feasible	Yes	32.09%	-98.61%
56	811.1448115	0.003492832	Feasible	Yes	66.38%	-98.00%
57	664.8870894	0.004600525	Feasible	Yes	32.09%	-97.31%
58	NA	0.822085619	Infeasible	NA	NA	348.35%
59	NA	0.777799368	Infeasible	NA	NA	560.07%
60	NA	0.749403238	Infeasible	NA	NA	402.55%
61	664.8870894	0.002676964	Feasible	Yes	32.09%	-97.69%
62	NA	0.96553278	Infeasible	NA	NA	385.65%
63	NA	0.929165125	Infeasible	NA	NA	415.32%
64	NA	0.864661932	Infeasible	NA	NA	572.73%
65	NA	1.246846676	Infeasible	NA	NA	1073.03%
66	NA	0.840579987	Infeasible	NA	NA	770.24%

Continued on next page

Table C.1 – continued from previous page

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries		
67	563.2212294	0.003503323	Feasible	Yes	11.89%	-96.93%
68	NA	0.35121417	Infeasible	NA	NA	226.66%
69	NA	0.163201571	Infeasible	NA	NA	65.31%
70	NA	0.165969849	Infeasible	NA	NA	53.37%
71	NA	0.175273657	Infeasible	NA	NA	42.02%
72	NA	0.182790041	Infeasible	NA	NA	75.43%
73	497.7004479	0.008999348	Optimal	No	0.00%	-96.63%
74	497.7004479	0.01202035	Optimal	No	0.00%	-97.49%
75	552.0379054	0.086966753	Feasible	No	1.87%	-82.04%
76	NA	0.385063887	Infeasible	NA	NA	-6.59%
77	NA	0.579434633	Infeasible	NA	NA	-15.29%
78	NA	0.854026556	Infeasible	NA	NA	6.28%
79	541.8781896	0.026904583	Optimal	No	0.00%	-88.61%
80	842.8639679	0.02239871	Feasible	Yes	31.93%	-97.21%
81	842.8639679	0.02420783	Feasible	Yes	31.93%	-97.50%
82	NA	1.791688442	Infeasible	NA	NA	229.64%
83	NA	1.853954792	Infeasible	NA	NA	225.62%
84	649.853598	0.083445549	Optimal	No	0.00%	-81.88%
85	747.8496126	0.01000762	Feasible	Yes	17.06%	-96.68%
86	793.1109804	0.022629976	Feasible	Yes	24.14%	-96.69%
87	649.853598	0.020301104	Optimal	No	0.00%	-96.77%
88	907.1114696	0.740440369	Feasible	Yes	32.37%	-66.35%
89	NA	1.698547363	Infeasible	NA	NA	191.52%
90	NA	1.775528193	Infeasible	NA	NA	247.40%
91	649.853598	0.007530928	Optimal	No	0.00%	-97.23%
92	828.5237832	0.02808094	Feasible	Yes	20.91%	-94.65%
93	828.5237832	0.032538176	Feasible	Yes	20.91%	-93.94%
94	NA	0.344077826	Infeasible	NA	NA	11.55%
95	NA	0.152555704	Infeasible	NA	NA	-61.86%
96	NA	0.158984423	Infeasible	NA	NA	-61.79%
97	719.5395481	0.366784334	Feasible	Yes	5.00%	40.73%
98	719.5395481	1.061498165	Optimal	Yes	0.00%	89.35%
99	NA	1.238837004	Infeasible	NA	NA	28.38%
100	NA	1.153033495	Infeasible	NA	NA	289.40%
101	NA	1.258708239	Infeasible	NA	NA	296.88%
102	NA	1.159288168	Infeasible	NA	NA	214.54%
103	719.5395481	0.232941389	Optimal	Yes	0.00%	-16.53%
104	NA	1.524906397	Infeasible	NA	NA	357.33%
105	NA	1.422627449	Infeasible	NA	NA	355.83%
106	NA	1.316708088	Infeasible	NA	NA	348.53%
107	NA	1.275863647	Infeasible	NA	NA	296.08%
108	NA	1.251493931	Infeasible	NA	NA	290.18%
109	529.3143368	0.014111996	Feasible	No	0.33%	-99.55%
110	527.5973838	0.819176435	Optimal	No	0.00%	-95.30%
111	548.610515	2.010409832	Optimal	No	0.00%	-84.25%

Continued on next page

Table C.1 – continued from previous page

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries		
112	NA	2.223486185	Infeasible	NA	NA	-74.30%
113	NA	2.91607213	Infeasible	NA	NA	-86.42%
114	NA	2.795580864	Infeasible	NA	NA	-72.02%
115	551.0572991	0.070666552	Optimal	No	0.00%	-97.12%
116	551.0572991	0.056832075	Optimal	No	0.00%	-99.54%
117	563.1718898	0.606825829	Optimal	No	0.00%	-92.72%
118	NA	2.426347017	Infeasible	NA	NA	-67.93%
119	NA	1.789558411	Infeasible	NA	NA	-82.62%
120	NA	1.995904684	Infeasible	NA	NA	-75.45%
121	584.185021	0.012388229	Feasible	No	3.73%	-99.48%
122	584.185021	0.183636189	Optimal	No	0.00%	-98.36%
123	584.185021	0.358308077	Optimal	No	0.00%	-93.23%
124	584.185021	0.107360125	Optimal	No	0.00%	-98.39%
125	584.185021	0.179905653	Optimal	No	0.00%	-97.55%
126	NA	1.82946682	Infeasible	NA	NA	-90.49%
127	993.4146308	0.009306431	Feasible	Yes	70.05%	-99.46%
128	NA	0.934769869	Infeasible	NA	NA	-89.65%
129	NA	2.086862326	Infeasible	NA	NA	-86.95%
130	NA	1.813490152	Infeasible	NA	NA	-79.21%
131	NA	1.147241116	Infeasible	NA	NA	-14.64%
132	NA	0.865254879	Infeasible	NA	NA	-37.14%
133	1002.836783	0.004240751	Feasible	Yes	55.61%	-99.81%
134	NA	1.139447212	Infeasible	NA	NA	-88.57%
135	NA	0.294408083	Infeasible	NA	NA	-77.59%
136	NA	0.557134628	Infeasible	NA	NA	-94.29%
137	NA	0.359592199	Infeasible	NA	NA	-73.73%
138	NA	0.330717325	Infeasible	NA	NA	-74.74%
139	724.5268817	0.400824785	Feasible	Yes	11.72%	-83.26%
140	NA	1.758160114	Infeasible	NA	NA	30.85%
141	NA	1.210644722	Infeasible	NA	NA	-6.94%
142	NA	1.213045359	Infeasible	NA	NA	-9.21%
143	NA	1.182906628	Infeasible	NA	NA	-7.98%
144	NA	1.300034046	Infeasible	NA	NA	-0.11%
145	782.4725366	0.046911478	Feasible	No	5.54%	-99.94%
146	741.4192694	1.354097605	Optimal	No	0.00%	-93.68%
147	741.4192694	7.495004892	Optimal	No	0.00%	-84.49%
148	830.7495902	0.269727468	Feasible	No	12.05%	-99.50%
149	NA	29.79738617	Infeasible	NA	NA	-70.62%
150	NA	29.2103858	Infeasible	NA	NA	-74.72%
151	889.9121094	0.634518385	Feasible	Yes	16.89%	-98.86%
152	NA	0.803798676	Infeasible	NA	NA	-95.64%
153	NA	0.871045828	Infeasible	NA	NA	-94.61%
154	NA	1.461205006	Infeasible	NA	NA	-98.95%
155	NA	0.970113516	Infeasible	NA	NA	-98.28%
156	NA	2.735004902	Infeasible	NA	NA	-95.37%

Continued on next page

Table C.1 – continued from previous page

ID	Optimal Solution (Min Cost)	Time Elapsed (seconds)	Feasible or Infeasible	Split Deliveries		
157	791.3263079	0.014783382	Optimal	No	0.00%	-99.96%
158	1218.832906	6.339544535	Feasible	Yes	45.10%	-50.89%
159	1219.097579	14.76925802	Feasible	Yes	45.13%	13.73%
160	NA	16.13575816	Infeasible	NA	NA	-82.12%
161	NA	15.98366046	Infeasible	NA	NA	-83.96%
162	NA	16.03126001	Infeasible	NA	NA	-85.88%
163	1081.660235	0.025531769	Feasible	Yes	17.06%	-99.95%
164	1063.809206	16.0694077	Feasible	Yes	15.13%	-92.97%
165	1099.172877	0.216792583	Feasible	Yes	13.84%	-99.80%
166	NA	32.6824882	Infeasible	NA	NA	-75.56%
167	NA	25.64211535	Infeasible	NA	NA	-98.88%
168	NA	19.14689589	Infeasible	NA	NA	19.38%
169	1109.926884	0.466546535	Feasible	Yes	20.12%	-99.24%
170	NA	14.39113712	Infeasible	NA	NA	-97.78%
171	NA	13.5519433	Infeasible	NA	NA	46.13%
172	NA	11.93197346	Infeasible	NA	NA	25.92%
173	NA	12.47369409	Infeasible	NA	NA	22.03%
174	NA	13.33650684	Infeasible	NA	NA	31.07%
175	1045.778169	0.464526176	Feasible	Yes	4.67%	-97.63%
176	NA	10.9949913	Infeasible	NA	NA	-10.73%
177	NA	12.22589779	Infeasible	NA	NA	23.91%
178	NA	11.9643867	Infeasible	NA	NA	14.77%
179	NA	13.1002667	Infeasible	NA	NA	18.17%
180	NA	14.87932849	Infeasible	NA	NA	49.67%

Appendix D

Parameter Calibration Results

D.0.1 First Part of the Table

ID	Iterations	Best Solution (Min Cost)	Time Elapsed (seconds)	Split Deliveries	Feasible or Infeasible	Multi-Objective Option	Local Search Option
11	500	427.412912	0.002006	Yes	Feasible	random	random
12	500	427.412912	0.54534	Yes	Feasible	random	random
17	NA	NA	NA	NA	Infeasible	NA	NA
19	500	427.412912	0.000998	Yes	Feasible	cost	random
20	500	427.412912	0.001505	Yes	Feasible	random	random
21	500	427.412912	0.001004	Yes	Feasible	random	random
22	NA	NA	NA	NA	Infeasible	NA	NA
25	500	427.412912	0.001012	Yes	Feasible	cost	random
26	NA	NA	NA	NA	Infeasible	NA	NA
31	500	427.412912	0.001999	Yes	Feasible	cost	random
40	NA	NA	NA	NA	Infeasible	NA	NA
41	NA	NA	NA	NA	Infeasible	NA	NA
42	NA	NA	NA	NA	Infeasible	NA	NA
46	NA	NA	NA	NA	Infeasible	NA	NA
52	500	719.741158	5.120704	Yes	Feasible	random	random
53	500	641.781089	0.016168	Yes	Feasible	random	random
54	500	719.741158	8.120221	Yes	Feasible	random	random
55	500	661.339756	0.001999	Yes	Feasible	cost	random
56	500	661.339756	0.007275	Yes	Feasible	random	random
57	500	664.887089	0.001999	Yes	Feasible	random	random
58	NA	NA	NA	NA	Infeasible	NA	NA
61	500	664.887089	0.001999	Yes	Feasible	cost	random
62	500	699.94	0.002982378	Yes	Feasible	random	random

Continued on next page

Table D.1 – continued from previous page

ID	Iterations	Best Solution (Min Cost)	Time Elapsed (seconds)	Split Deliveries	Feasible or Infeasible	Multi-Objective Option	Local Search Option
63	NA	NA	NA	NA	Infeasible	NA	NA
67	500	563.221229	0.423949	Yes	Feasible	cost	random
75	500	552.037905	2.508154	No	Feasible	random	random
76	NA	NA	NA	NA	Infeasible	NA	NA
77	NA	NA	NA	NA	Infeasible	NA	NA
78	NA	NA	NA	NA	Infeasible	NA	NA
80	500	842.863968	0.023001	Yes	Feasible	random	random
81	500	807.503287	0.013999	Yes	Feasible	random	random
82	500	1036.592	0.06511402	Yes	Feasible	random	random
83	NA	NA	NA	NA	Infeasible	NA	NA
85	500	691.4271045	0.006729603	Yes	Feasible	cost	random
86	500	793.11098	0.022497	Yes	Feasible	random	random
88	500	857.435691	9.744831	Yes	Feasible	random	random
89	500	907.11147	15.493548	Yes	Feasible	random	random
90	NA	NA	NA	NA	Infeasible	NA	NA
92	500	828.523783	0.048334	Yes	Feasible	random	random
93	500	828.523783	0.043534	Yes	Feasible	random	random
97	500	719.539548	1.283548	Yes	Feasible	cost	random
99	NA	NA	NA	NA	Infeasible	NA	NA
109	500	529.314337	1.381036	No	Feasible	cost	random
112	NA	NA	NA	NA	Infeasible	NA	NA
113	NA	NA	NA	NA	Infeasible	NA	NA
114	NA	NA	NA	NA	Infeasible	NA	NA
118	NA	NA	NA	NA	Infeasible	NA	NA
119	NA	NA	NA	NA	Infeasible	NA	NA
120	NA	NA	NA	NA	Infeasible	NA	NA
121	500	584.185021	0.012002	No	Feasible	cost	random
126	NA	NA	NA	NA	Infeasible	NA	NA
127	500	774.2361643	0.023146868	Yes	Feasible	cost	random
128	500	839.23	1.710773	Yes	Feasible	random	random
129	500	913.79	1.766835	Yes	Feasible	random	random
130	NA	NA	NA	NA	Infeasible	NA	NA
133	500	809.500617	0.008004	Yes	Feasible	cost	random
134	NA	NA	NA	NA	Infeasible	NA	NA
139	500	724.526882	1.238964	Yes	Feasible	cost	random
145	500	782.472537	0.023146	No	Feasible	cost	random
148	500	741.419269	3.744898	No	Optimal	random	random
149	NA	NA	NA	NA	Infeasible	NA	NA
150	NA	NA	NA	NA	Infeasible	NA	NA
151	500	889.912109	0.017775	Yes	Feasible	cost	random
152	NA	NA	NA	NA	Infeasible	NA	NA
153	NA	NA	NA	NA	Infeasible	NA	NA
154	NA	NA	NA	NA	Infeasible	NA	NA
155	NA	NA	NA	NA	Infeasible	NA	NA
156	NA	NA	NA	NA	Infeasible	NA	NA

Continued on next page

Table D.1 – continued from previous page

ID	Iterations	Best Solution (Min Cost)	Time Elapsed (seconds)	Split Deliveries	Feasible or Infeasible	Multi-Objective Option	Local Search Option
158	500	1212.836482	134.865072	Yes	Feasible	random	random
159	500	1212.836482	23.860708	Yes	Feasible	random	random
160	NA	NA	NA	NA	Infeasible	NA	NA
161	NA	NA	NA	NA	Infeasible	NA	NA
162	NA	NA	NA	NA	Infeasible	NA	NA
163	500	1008.021063	0.010616779	Yes	Feasible	cost	random
164	500	1017.312773	146.832886	Yes	Feasible	random	random
165	500	1079.143212	66.316303	Yes	Feasible	random	random
166	NA	NA	NA	NA	Infeasible	NA	NA
167	NA	NA	NA	NA	Infeasible	NA	NA
169	500	979.7103481	0.022599697	Yes	Feasible	cost	random
170	NA	NA	NA	NA	Infeasible	NA	NA
175	500	1045.778169	2.245906	Yes	Feasible	cost	random

D.0.2 Second Part of the Table

ID	Split Solution	RCL Factor	Pertub Percentage	GAP % (solution	GAP % (time)	Improvement Solution
11	Yes	0.8	0.8	30.52%	-98.62%	None
12	Yes	0.8	0.8	30.52%	-10.02%	None
17	NA	NA	NA	NA	NA	None
19	Yes	0.8	0.8	30.52%	-99.08%	None
20	Yes	0.8	0.8	30.52%	-98.78%	None
21	Yes	0.8	0.8	30.52%	-99.32%	None
22	NA	NA	NA	NA	NA	None
25	Yes	0.8	0.8	30.52%	-99.01%	None
26	NA	NA	NA	NA	NA	None
31	Yes	0.8	0.8	30.52%	-98.71%	None
40	NA	NA	NA	NA	NA	None
41	NA	NA	NA	NA	NA	None
42	NA	NA	NA	NA	NA	None
46	NA	NA	NA	NA	NA	None
52	Yes	0.8	0.8	42.99%	3485.36%	None
53	Yes	0.5	0.5	27.50%	-90.83%	None
54	Yes	0.8	0.8	42.99%	4886.22%	None
55	Yes	0.8	0.8	31.38%	-98.61%	0.70%
56	Yes	0.8	0.8	35.65%	-95.84%	30.73%
57	Yes	0.5	0.5	32.09%	-98.83%	None
58	NA	NA	NA	NA	NA	None
61	Yes	0.5	0.5	32.09%	-98.27%	None
62	Yes	0.8	0.8	39.05%	-98.50%	From NA to Solution
63	NA	NA	NA	NA	NA	None
67	Yes	0.8	0.8	11.89%	272.08%	None
75	No	0.8	0.8	1.87%	417.91%	None

Continued on next page

Table D.2 – continued from previous page

ID	Split Solution	RCL Factor	Pertub Percentage	GAP % (solution	GAP % (time)	Improvement Solution
76	NA	NA	NA	NA	NA	None
77	NA	NA	NA	NA	NA	None
78	NA	NA	NA	NA	NA	None
80	Yes	0.5	0.5	31.93%	-97.13%	None
81	Yes	0.8	0.8	26.40%	-98.56%	5.53%
82	Yes	0.8	0.8	62.26%	-88.02%	From NA to Solution
83	NA	NA	NA	NA	NA	None
85	Yes	0.5	0.8	8.23%	-97.77%	8.83%
86	Yes	0.5	0.5	24.14%	-96.71%	None
88	Yes	0.8	0.8	25.12%	342.92%	7.25%
89	Yes	0.8	0.8	32.37%	2559.17%	From NA to Solution
90	NA	NA	NA	NA	NA	None
92	Yes	0.5	0.5	20.91%	-90.79%	None
93	Yes	0.5	0.5	20.91%	-91.89%	None
97	Yes	0.8	0.8	5.00%	392.47%	None
99	NA	NA	NA	NA	NA	None
109	Yes	0.5	0.5	0.33%	-55.99%	None
112	NA	NA	NA	NA	NA	None
113	NA	NA	NA	NA	NA	None
114	NA	NA	NA	NA	NA	None
118	NA	NA	NA	NA	NA	None
119	NA	NA	NA	NA	NA	None
120	NA	NA	NA	NA	NA	None
121	No	0.8	0.8	3.73%	-99.49%	None
126	NA	NA	NA	NA	NA	None
127	Yes	0.8	0.5	32.53%	-98.65%	37.52%
128	Yes	0.8	0.8	30.22%	-81.05%	From NA to Solution
129	No	0.8	0.8	40.90%	-88.95%	From NA to Solution
130	NA	NA	NA	NA	NA	None
133	Yes	0.8	0.8	25.61%	-99.65%	30.00%
134	NA	NA	NA	NA	NA	None
139	Yes	0.8	0.8	11.72%	-48.27%	None
145	No	0.9	0.8	5.54%	-99.97%	None
148	No	0.8	0.2	0.00%	-93.02%	12.05%
149	NA	NA	NA	NA	NA	None
150	NA	NA	NA	NA	NA	None
151	No	0.9	0.8	16.89%	-99.97%	None
152	NA	NA	NA	NA	NA	None
153	NA	NA	NA	NA	NA	None
154	NA	NA	NA	NA	NA	None
155	NA	NA	NA	NA	NA	None
156	NA	NA	NA	NA	NA	None
158	Yes	0.8	0.5	44.39%	944.74%	0.71%
159	Yes	0.8	0.8	44.39%	83.74%	0.75%
160	NA	NA	NA	NA	NA	None

Continued on next page

Table D.2 – continued from previous page

ID	Split Solution	RCL Factor	Pertub Percentage	GAP % (solution	GAP % (time)	Improvement Solution
161	NA	NA	NA	NA	NA	None
162	NA	NA	NA	NA	NA	None
163	Yes	0.5	0.5	9.09%	-99.98%	7.97%
164	Yes	0.8	0.8	10.09%	-35.80%	5.03%
165	No	0.8	0.8	11.76%	-38.92%	2.07%
166	NA	NA	NA	NA	NA	None
167	NA	NA	NA	NA	NA	None
169	Yes	0.8	0.5	6.03%	-99.96%	14.09%
170	NA	NA	NA	NA	NA	None
175	Yes	0.8	0.8	4.67%	-88.55%	None

Appendix E

Benchmark Instances and Interdiction Probabilities

ID	Name	Nodes	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
1	eil22	22	93.7%	0%	0.00%	0.00%
2	eil22	22	93.7%	1%-5%	3.00%	1.14%
3	eil22	22	93.7%	1%-10%	5.59%	2.59%
4	eil22	22	93.7%	1%-20%	10.53%	5.52%
5	eil22	22	93.7%	1%-30%	15.64%	8.36%
6	eil22	22	93.7%	1%-40%	20.54%	11.78%
7	eil23	23	75.4%	0%	0.00%	0.00%
8	eil23	23	75.4%	1%-5%	3.02%	1.14%
9	eil23	23	75.4%	1%-10%	5.49%	2.57%
10	eil23	23	75.4%	1%-20%	10.57%	5.45%
11	eil23	23	75.4%	1%-30%	15.53%	8.32%
12	eil23	23	75.4%	1%-40%	20.55%	11.26%
13	eil30	30	94.4%	0%	0.00%	0.00%
14	eil30	30	94.4%	1%-5%	3.02%	1.15%
15	eil30	30	94.4%	1%-10%	5.53%	2.61%
16	eil30	30	94.4%	1%-20%	10.49%	5.50%
17	eil30	30	94.4%	1%-30%	15.50%	8.45%
18	eil30	30	94.4%	1%-40%	20.54%	11.32%
19	eil33	33	91.7%	0%	0.00%	0.00%
20	eil33	33	91.7%	1%-5%	3.04%	1.15%

Continued on next page

Table E.1 – continued from previous page

ID	Name	Nodes	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
21	eil33	33	91.7%	1%-10%	5.54%	2.58%
22	eil33	33	91.7%	1%-20%	10.54%	5.54%
23	eil33	33	91.7%	1%-30%	15.51%	8.40%
24	eil33	33	91.7%	1%-40%	20.45%	11.30%
25	eil51	51	97.1%	0%	0.00%	0.00%
26	eil51	51	97.1%	1%-5%	3.02%	1.15%
27	eil51	51	97.1%	1%-10%	5.52%	2.60%
28	eil51	51	97.1%	1%-20%	10.51%	5.49%
29	eil51	51	97.1%	1%-30%	15.50%	8.31%
30	eil51	51	97.1%	1%-40%	20.54%	11.31%
31	eilA76	76	97.4%	0%	0.00%	0.00%
32	eilA76	76	97.4%	1%-5%	3.01%	1.15%
33	eilA76	76	97.4%	1%-10%	5.53%	2.60%
34	eilA76	76	97.4%	1%-20%	10.50%	5.47%
35	eilA76	76	97.4%	1%-30%	15.48%	8.38%
36	eilA76	76	97.4%	1%-40%	20.50%	11.34%
37	eilA101	101	91.1%	0%	0.00%	0.00%
38	eilA101	101	91.1%	1%-5%	3.01%	1.16%
39	eilA101	101	91.1%	1%-10%	5.51%	2.63%
40	eilA101	101	91.1%	1%-20%	10.50%	5.47%
41	eilA101	101	91.1%	1%-30%	15.49%	8.41%
42	eilA101	101	91.1%	1%-40%	20.50%	11.30%
43	eilB76	76	97.4%	0%	0.00%	0.00%
44	eilB76	76	97.4%	1%-5%	3.01%	1.14%
45	eilB76	76	97.4%	1%-10%	5.50%	2.60%
46	eilB76	76	97.4%	1%-20%	10.51%	5.47%
47	eilB76	76	97.4%	1%-30%	15.50%	8.28%
48	eilB76	76	97.4%	1%-40%	20.50%	11.22%
49	eilB101	101	92.9%	0%	0.00%	0.00%
50	eilB101	101	92.9%	1%-5%	3.00%	1.15%
51	eilB101	101	92.9%	1%-10%	5.52%	2.59%
52	eilB101	101	92.9%	1%-20%	10.51%	5.49%
53	eilB101	101	92.9%	1%-30%	15.51%	8.30%
54	eilB101	101	92.9%	1%-40%	20.53%	11.23%
55	eilC76	76	94.7%	0%	0.00%	0.00%
56	eilC76	76	94.7%	1%-5%	3.00%	1.14%
57	eilC76	76	94.7%	1%-10%	5.50%	2.61%
58	eilC76	76	94.7%	1%-20%	10.50%	5.51%
59	eilC76	76	94.7%	1%-30%	15.51%	8.36%
60	eilC76	76	94.7%	1%-40%	20.52%	11.20%
61	eilD76	76	88.5%	0%	0.00%	0.00%
62	eilD76	76	88.5%	1%-5%	2.98%	1.15%
63	eilD76	76	88.5%	1%-10%	5.52%	2.61%

Continued on next page

Table E.1 – continued from previous page

ID	Name	Nodes	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
64	eilD76	76	88.5%	1%-20%	10.51%	5.55%
65	eilD76	76	88.5%	1%-30%	15.52%	8.36%
66	eilD76	76	88.5%	1%-40%	20.52%	11.27%
67	S51D1	51	83.7%	0%	0.00%	0.00%
68	S51D1	51	83.7%	1%-5%	2.99%	1.16%
69	S51D1	51	83.7%	1%-10%	5.47%	2.57%
70	S51D1	51	83.7%	1%-20%	10.50%	5.55%
71	S51D1	51	83.7%	1%-30%	15.51%	8.39%
72	S51D1	51	83.7%	1%-40%	20.52%	11.29%
73	S51D2	51	98.2%	0%	0.00%	0.00%
74	S51D2	51	98.2%	1%-5%	2.99%	1.16%
75	S51D2	51	98.2%	1%-10%	5.50%	2.58%
76	S51D2	51	98.2%	1%-20%	10.50%	5.54%
77	S51D2	51	98.2%	1%-30%	15.51%	8.44%
78	S51D2	51	98.2%	1%-40%	20.52%	11.28%
79	S51D3	51	94.7%	0%	0.00%	0.00%
80	S51D3	51	94.7%	1%-5%	2.99%	1.14%
81	S51D3	51	94.7%	1%-10%	5.50%	2.56%
82	S51D3	51	94.7%	1%-20%	10.50%	5.44%
83	S51D3	51	94.7%	1%-30%	15.50%	8.36%
84	S51D3	51	94.7%	1%-40%	20.56%	11.24%
85	S51D4	51	99.9%	0%	0.00%	0.00%
86	S51D4	51	99.9%	1%-5%	3.01%	1.17%
87	S51D4	51	99.9%	1%-10%	5.50%	2.57%
88	S51D4	51	99.9%	1%-20%	10.52%	5.47%
89	S51D4	51	99.9%	1%-30%	15.51%	8.28%
90	S51D4	51	99.9%	1%-40%	20.51%	11.34%
91	S51D5	51	99.0%	0%	0.00%	0.00%
92	S51D5	51	99.0%	1%-5%	3.01%	1.16%
93	S51D5	51	99.0%	1%-10%	5.51%	2.62%
94	S51D5	51	99.0%	1%-20%	10.50%	5.57%
95	S51D5	51	99.0%	1%-30%	15.52%	8.26%
96	S51D5	51	99.0%	1%-40%	20.49%	11.33%
97	S51D6	51	98.4%	0%	0.00%	0.00%
98	S51D6	51	98.4%	1%-5%	3.00%	1.18%
99	S51D6	51	98.4%	1%-10%	5.51%	2.59%
100	S51D6	51	98.4%	1%-20%	10.50%	5.40%
101	S51D6	51	98.4%	1%-30%	15.52%	8.38%
102	S51D6	51	98.4%	1%-40%	20.49%	11.43%
103	S76D1	76	95.9%	0%	0.00%	0.00%
104	S76D1	76	95.9%	1%-5%	3.01%	1.16%
105	S76D1	76	95.9%	1%-10%	5.50%	2.57%
106	S76D1	76	95.9%	1%-20%	10.51%	5.46%

Continued on next page

Table E.1 – continued from previous page

ID	Name	Nodes	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
107	S76D1	76	95.9%	1%-30%	15.51%	8.39%
108	S76D1	76	95.9%	1%-40%	20.50%	11.10%
109	S76D2	76	99.2%	0%	0.00%	0.00%
110	S76D2	76	99.2%	1%-5%	3.00%	1.15%
111	S76D2	76	99.2%	1%-10%	5.49%	2.60%
112	S76D2	76	99.2%	1%-20%	10.50%	5.54%
113	S76D2	76	99.2%	1%-30%	15.50%	8.36%
114	S76D2	76	99.2%	1%-40%	20.50%	11.42%
115	S76D3	76	96.2%	0%	0.00%	0.00%
116	S76D3	76	96.2%	1%-5%	3.01%	1.16%
117	S76D3	76	96.2%	1%-10%	5.52%	2.59%
118	S76D3	76	96.2%	1%-20%	10.50%	5.50%
119	S76D3	76	96.2%	1%-30%	15.50%	8.43%
120	S76D3	76	96.2%	1%-40%	20.50%	11.33%
121	S76D4	76	97.3%	0%	0.00%	0.00%
122	S76D4	76	97.3%	1%-5%	3.00%	1.17%
123	S76D4	76	97.3%	1%-10%	5.50%	2.59%
124	S76D4	76	97.3%	1%-20%	10.50%	5.41%
125	S76D4	76	97.3%	1%-30%	15.51%	8.31%
126	S76D4	76	97.3%	1%-40%	20.49%	11.39%
127	S101D1	101	98.5%	0%	0.00%	0.00%
128	S101D1	101	98.5%	1%-5%	3.00%	1.16%
129	S101D1	101	98.5%	1%-10%	5.50%	2.57%
130	S101D1	101	98.5%	1%-20%	10.51%	5.46%
131	S101D1	101	98.5%	1%-30%	15.50%	8.40%
132	S101D1	101	98.5%	1%-40%	20.50%	11.05%
133	S101D2	101	95.7%	0%	0.00%	0.00%
134	S101D2	101	95.7%	1%-5%	2.99%	1.15%
135	S101D2	101	95.7%	1%-10%	5.50%	2.59%
136	S101D2	101	95.7%	1%-20%	10.50%	5.46%
137	S101D2	101	95.7%	1%-30%	15.50%	8.39%
138	S101D2	101	95.7%	1%-40%	20.50%	11.32%
139	S101D3	101	97.6%	0%	0.00%	0.00%
140	S101D3	101	97.6%	1%-5%	3.00%	1.16%
141	S101D3	101	97.6%	1%-10%	5.50%	2.61%
142	S101D3	101	97.6%	1%-20%	10.51%	5.42%
143	S101D3	101	97.6%	1%-30%	15.50%	8.35%
144	S101D3	101	97.6%	1%-40%	20.50%	11.19%
145	S101D5	101	99.9%	0%	0.00%	0.00%
146	S101D5	101	99.9%	1%-5%	3.00%	1.16%
147	S101D5	101	99.9%	1%-10%	5.49%	2.60%
148	S101D5	101	99.9%	1%-20%	10.49%	5.48%
149	S101D5	101	99.9%	1%-30%	15.50%	8.38%

Continued on next page

Table E.1 – continued from previous page

ID	Name	Nodes	Tightness	Interdiction Probability Parameters	Mean Interdiction Probability	SD Interdiction Probability
150	S101D5	101	99.9%	1%-40%	20.50%	11.29%

Appendix F

GRASP Results on Benchmark Instances

F.0.1 First Part of the Table

ID	Best Solution Literature (Min Cost)	Best Solution (Min Cost)	Time Elapsed (seconds)	Split Deliveries	Feasible or Infeasible	Multi Objective Option
1	375	442.1955688	131.4901023	Yes	Feasible	random
2	NA	736.5115947	376.1166553	Yes	Feasible	demand
3	NA	NA	110.1700673	NA	Infeasible	demand
4	NA	NA	106.62381444	NA	Infeasible	demand
5	NA	NA	120.60272187	NA	Infeasible	demand
6	NA	NA	120.15064707	NA	Infeasible	demand
7	569	633.8238344	65.96057153	Yes	Feasible	random
8	NA	1036.536797	0.686771154	Yes	Feasible	demand
9	NA	1092.194257	0.697322607	Yes	Feasible	demand
10	NA	1427.903004	0.557892561	Yes	Feasible	demand
11	NA	NA	0.624038085	NA	Infeasible	demand
12	NA	NA	0.66990346	NA	Infeasible	demand
13	510	621.8816426	614.5795112	Yes	Feasible	random
14	NA	NA	906.210113	NA	Infeasible	demand
15	NA	NA	907.9980481	NA	Infeasible	demand
16	NA	NA	926.4804907	NA	Infeasible	demand
17	NA	NA	928.2116715	NA	Infeasible	demand
18	NA	NA	922.3472672	NA	Infeasible	demand
19	835	922.6546174	2068.195897	Yes	Feasible	cost
20	NA	1648.07893	1453.71778	Yes	Feasible	demand
21	NA	1602.869507	1841.528442	Yes	Feasible	demand
22	NA	NA	633.2698921	NA	Infeasible	demand

Continued on next page

Table F.1 – continued from previous page

ID	Best Solution Literature (Min Cost)	Best Solution (Min Cost)	Time Elapsed (seconds)	Split Deliveries	Feasible or Infeasible	Multi Objective Option
23	NA	NA	640.340842	NA	Infeasible	demand
24	NA	NA	626.6728863	NA	Infeasible	demand
25	521	653.3325994	745.291903	Yes	Feasible	cost
26	NA	NA	638.0994248	NA	Infeasible	demand
27	NA	NA	608.0995936	NA	Infeasible	demand
28	NA	NA	637.909057	NA	Infeasible	demand
29	NA	NA	634.0669971	NA	Infeasible	demand
30	NA	NA	635.7728911	NA	Infeasible	demand
31	832	1011.371644	1393.905025	Yes	Feasible	cost
32	NA	NA	726.3096816	NA	Infeasible	demand
33	NA	NA	746.7578841	NA	Infeasible	demand
34	NA	NA	709.0639978	NA	Infeasible	demand
35	NA	NA	767.6857413	NA	Infeasible	demand
36	NA	NA	733.9601483	NA	Infeasible	demand
37	817	1102.433772	3793.884239	Yes	Feasible	cost
38	NA	1389.17345	3675.642096	Yes	Feasible	demand
39	NA	1586.994689	3501.555506	Yes	Feasible	demand
40	NA	NA	1049.363572	NA	Infeasible	demand
41	NA	NA	1029.474908	NA	Infeasible	demand
42	NA	NA	1013.876467	NA	Infeasible	demand
43	1023	1238.529391	1077.475019	Yes	Feasible	cost
44	NA	NA	467.9306319	NA	Infeasible	demand
45	NA	NA	475.2060307	NA	Infeasible	demand
46	NA	NA	481.0287361	NA	Infeasible	demand
47	NA	NA	463.3541846	NA	Infeasible	demand
48	NA	NA	487.6062481	NA	Infeasible	demand
49	1077	1310.374391	2862.385229	Yes	Feasible	cost
50	NA	1990.705947	3416.068105	Yes	Feasible	demand
51	NA	2091.811196	3490.333206	Yes	Feasible	demand
52	NA	NA	635.3402689	NA	Infeasible	demand
53	NA	NA	619.0545537	NA	Infeasible	demand
54	NA	NA	617.3452253	NA	Infeasible	demand
55	735	968.8701212	1453.604385	Yes	Feasible	random
56	NA	1341.548349	2001.146505	Yes	Feasible	demand
57	NA	NA	359.6987239	NA	Infeasible	demand
58	NA	NA	361.005931	NA	Infeasible	demand
59	NA	NA	363.8290206	NA	Infeasible	demand
60	NA	NA	394.4925259	NA	Infeasible	demand
61	683	876.0515535	1221.854937	Yes	Feasible	random
62	NA	1188.027728	1931.543744	Yes	Feasible	demand
63	NA	1291.142327	1939.113838	Yes	Feasible	demand
64	NA	NA	396.4095498	NA	Infeasible	demand
65	NA	NA	398.1854183	NA	Infeasible	demand

Continued on next page

Table F.1 – continued from previous page

ID	Best Solution Literature (Min Cost)	Best Solution (Min Cost)	Time Elapsed (seconds)	Split Deliveries	Feasible or Infeasible	Multi Objective Option
66	NA	NA	353.9838747	NA	Infeasible	demand
67	458	605.7408083	564.6257231	Yes	Feasible	random
68	NA	811.513545	655.780587	Yes	Feasible	demand
69	NA	819.203259	693.730644	Yes	Feasible	demand
70	NA	957.8845103	685.1661688	Yes	Feasible	demand
71	NA	NA	407.2936418	NA	Infeasible	demand
72	NA	NA	424.185641	NA	Infeasible	demand
73	726	839.6778672	255.7757089	Yes	Feasible	cost
74	NA	NA	119.5798073	NA	Infeasible	demand
75	NA	NA	86.87583797	NA	Infeasible	demand
76	NA	NA	106.0364612	NA	Infeasible	demand
77	NA	NA	129.3763045	NA	Infeasible	demand
78	NA	NA	96.18037799	NA	Infeasible	demand
79	972	1148.245226	420.4983733	Yes	Feasible	random
80	NA	1807.944865	562.8971668	Yes	Feasible	demand
81	NA	NA	195.1249213	NA	Infeasible	demand
82	NA	NA	162.5961337	NA	Infeasible	demand
83	NA	NA	183.4171645	NA	Infeasible	demand
84	NA	NA	161.7570845	NA	Infeasible	demand
85	1677	1867.380558	404.2405553	Yes	Feasible	cost
86	NA	NA	162.8322791	NA	Infeasible	demand
87	NA	NA	196.7333502	NA	Infeasible	demand
88	NA	NA	152.6119041	NA	Infeasible	demand
89	NA	NA	195.0798505	NA	Infeasible	demand
90	NA	NA	188.9348046	NA	Infeasible	demand
91	1440	1555.707027	736.918854	Yes	Feasible	cost
92	NA	NA	334.218054	NA	Infeasible	demand
93	NA	NA	323.4953924	NA	Infeasible	demand
94	NA	NA	304.6834789	NA	Infeasible	demand
95	NA	NA	336.8772256	NA	Infeasible	demand
96	NA	NA	314.031097	NA	Infeasible	demand
97	2327	2496.524332	1441.740751	Yes	Feasible	random
98	NA	NA	407.6530604	NA	Infeasible	demand
99	NA	NA	444.2545785	NA	Infeasible	demand
100	NA	NA	430.5491891	NA	Infeasible	demand
101	NA	NA	403.0051229	NA	Infeasible	demand
102	NA	NA	442.5265516	NA	Infeasible	demand
103	594	703.3958445	1053.63763	Yes	Feasible	cost
104	NA	1121.417654	1210.299719	Yes	Feasible	demand
105	NA	NA	345.8206975	NA	Infeasible	demand
106	NA	NA	306.59459	NA	Infeasible	demand
107	NA	NA	305.291038	NA	Infeasible	demand
108	NA	NA	346.4678694	NA	Infeasible	demand

Continued on next page

Table F.1 – continued from previous page

ID	Best Solution Literature (Min Cost)	Best Solution (Min Cost)	Time Elapsed (seconds)	Split Deliveries	Feasible or Infeasible	Multi Objective Option
109	1147	1283.77499	1216.135388	Yes	Feasible	cost
110	NA	NA	425.3203295	NA	Infeasible	demand
111	NA	NA	387.9814675	NA	Infeasible	demand
112	NA	NA	400.7110827	NA	Infeasible	demand
113	NA	NA	406.6298747	NA	Infeasible	demand
114	NA	NA	405.4850604	NA	Infeasible	demand
115	1474	1700.327111	1189.392405	Yes	Feasible	cost
116	NA	2704.072049	1527.881179	Yes	Feasible	demand
117	NA	NA	393.9435242	NA	Infeasible	demand
118	NA	NA	358.8987124	NA	Infeasible	demand
119	NA	NA	394.0947128	NA	Infeasible	demand
120	NA	NA	386.9815872	NA	Infeasible	demand
121	2257	2439.552339	3004.079062	Yes	Feasible	random
122	NA	NA	711.3436779	NA	Infeasible	demand
123	NA	NA	727.0201507	NA	Infeasible	demand
124	NA	NA	712.8957615	NA	Infeasible	demand
125	NA	NA	727.1739897	NA	Infeasible	demand
126	NA	NA	741.5986541	NA	Infeasible	demand
127	716	899.7562082	658.6731527	Yes	Feasible	demand
128	NA	NA	334.1333626	NA	Infeasible	demand
129	NA	NA	316.1831912	NA	Infeasible	demand
130	NA	NA	330.9975815	NA	Infeasible	demand
131	NA	NA	321.6218571	NA	Infeasible	demand
132	NA	NA	312.0390199	NA	Infeasible	demand
133	1393	1602.653346	2032.427178	Yes	Feasible	cost
134	NA	2404.274267	3133.736674	Yes	Feasible	demand
135	NA	NA	843.054945	NA	Infeasible	demand
136	NA	NA	837.242364	NA	Infeasible	demand
137	NA	NA	845.1256628	NA	Infeasible	demand
138	NA	NA	847.6218511	NA	Infeasible	demand
139	1975	2193.79042	1927.209679	Yes	Feasible	cost
140	NA	NA	841.759282	NA	Infeasible	demand
141	NA	NA	800.8643501	NA	Infeasible	demand
142	NA	NA	877.5709165	NA	Infeasible	demand
143	NA	NA	876.6600262	NA	Infeasible	demand
144	NA	NA	847.0492505	NA	Infeasible	demand
145	2915	3155.224533	1038.584538	Yes	Feasible	cost
146	NA	NA	601.7299446	NA	Infeasible	demand
147	NA	NA	603.4374689	NA	Infeasible	demand
148	NA	NA	611.1357954	NA	Infeasible	demand
149	NA	NA	624.2315796	NA	Infeasible	demand
150	NA	NA	625.8748994	NA	Infeasible	demand

F.0.2 Second Part of the Table

ID	Local Search Option	RCL Factor	Pertub Percentage	GAP % (solution)
1	random	0.8	0.8	17.92%
2	random	0.5	0.8	NA
3	random	0.8	0.8	NA
4	random	0.8	0.8	NA
5	random	0.8	0.8	NA
6	random	0.8	0.8	NA
7	random	0.5	0.8	11.39%
8	random	0.5	0.5	NA
9	random	0.5	0.5	NA
10	random	0.8	0.8	NA
11	random	0.8	0.8	NA
12	random	0.8	0.8	NA
13	random	0.8	0.8	21.94%
14	random	0.8	0.8	NA
15	random	0.8	0.8	NA
16	random	0.8	0.8	NA
17	random	0.8	0.8	NA
18	random	0.8	0.8	NA
19	random	0.5	0.8	10.50%
20	random	0.8	0.5	NA
21	random	0.8	0.5	NA
22	random	0.8	0.8	NA
23	random	0.8	0.8	NA
24	random	0.8	0.8	NA
25	random	0.8	0.5	25.40%
26	random	0.8	0.8	NA
27	random	0.8	0.8	NA
28	random	0.8	0.8	NA
29	random	0.8	0.8	NA
30	random	0.8	0.8	NA
31	random	0.5	0.8	21.56%
32	random	0.8	0.8	NA
33	random	0.8	0.8	NA
34	random	0.8	0.8	NA
35	random	0.8	0.8	NA
36	random	0.8	0.8	NA
37	random	0.5	0.8	34.94%
38	random	0.8	0.5	NA
39	random	0.8	0.5	NA
40	random	0.8	0.8	NA
41	random	0.8	0.8	NA
42	random	0.8	0.8	NA

Continued on next page

Table F.2 – continued from previous page

ID	Local Search Option	RCL Factor	Pertub Percentage	GAP % (solution)
43	random	0.8	0.5	21.07%
44	random	0.8	0.8	NA
45	random	0.8	0.8	NA
46	random	0.8	0.8	NA
47	random	0.8	0.8	NA
48	random	0.8	0.8	NA
49	random	0.8	0.8	21.67%
50	random	0.8	0.5	NA
51	random	0.8	0.5	NA
52	random	0.8	0.8	NA
53	random	0.8	0.8	NA
54	random	0.8	0.8	NA
55	random	0.5	0.5	31.82%
56	random	0.8	0.5	NA
57	random	0.8	0.8	NA
58	random	0.8	0.8	NA
59	random	0.8	0.8	NA
60	random	0.8	0.8	NA
61	random	0.5	0.8	28.27%
62	random	0.8	0.5	NA
63	random	0.8	0.5	NA
64	random	0.8	0.8	NA
65	random	0.8	0.8	NA
66	random	0.8	0.8	NA
67	random	0.5	0.8	32.26%
68	random	0.8	0.5	NA
69	random	0.8	0.5	NA
70	random	0.8	0.5	NA
71	random	0.8	0.8	NA
72	random	0.8	0.8	NA
73	random	0.8	0.5	15.66%
74	random	0.8	0.8	NA
75	random	0.8	0.8	NA
76	random	0.8	0.8	NA
77	random	0.8	0.8	NA
78	random	0.8	0.8	NA
79	random	0.5	0.8	18.13%
80	random	0.8	0.5	NA
81	random	0.8	0.8	NA
82	random	0.8	0.8	NA
83	random	0.8	0.8	NA
84	random	0.8	0.8	NA
85	random	0.5	0.8	11.35%

Continued on next page

Table F.2 – continued from previous page

ID	Local Search Option	RCL Factor	Pertub Percentage	GAP % (solution)
86	random	0.8	0.8	NA
87	random	0.8	0.8	NA
88	random	0.8	0.8	NA
89	random	0.8	0.8	NA
90	random	0.8	0.8	NA
91	random	0.5	0.8	8.04%
92	random	0.8	0.8	NA
93	random	0.8	0.8	NA
94	random	0.8	0.8	NA
95	random	0.8	0.8	NA
96	random	0.8	0.8	NA
97	random	0.5	0.8	7.29%
98	random	0.8	0.8	NA
99	random	0.8	0.8	NA
100	random	0.8	0.8	NA
101	random	0.8	0.8	NA
102	random	0.8	0.8	NA
103	random	0.8	0.8	18.42%
104	random	0.8	0.8	NA
105	random	0.8	0.8	NA
106	random	0.8	0.8	NA
107	random	0.8	0.8	NA
108	random	0.8	0.8	NA
109	random	0.5	0.5	11.92%
110	random	0.8	0.8	NA
111	random	0.8	0.8	NA
112	random	0.8	0.8	NA
113	random	0.8	0.8	NA
114	random	0.8	0.8	NA
115	random	0.8	0.8	15.35%
116	random	0.8	0.8	NA
117	random	0.8	0.8	NA
118	random	0.8	0.8	NA
119	random	0.8	0.8	NA
120	random	0.8	0.8	NA
121	random	0.8	0.8	8.09%
122	random	0.8	0.8	NA
123	random	0.8	0.8	NA
124	random	0.8	0.8	NA
125	random	0.8	0.8	NA
126	random	0.8	0.8	NA
127	random	0.8	0.8	25.66%
128	random	0.8	0.8	NA

Continued on next page

Table F.2 – continued from previous page

ID	Local Search Option	RCL Factor	Pertub Percentage	GAP % (solution)
129	random	0.8	0.8	NA
130	random	0.8	0.8	NA
131	random	0.8	0.8	NA
132	random	0.8	0.8	NA
133	random	0.5	0.8	15.05%
134	random	0.5	0.8	NA
135	random	0.8	0.8	NA
136	random	0.8	0.8	NA
137	random	0.8	0.8	NA
138	random	0.8	0.8	NA
139	random	0.8	0.8	11.08%
140	random	0.8	0.8	NA
141	random	0.8	0.8	NA
142	random	0.8	0.8	NA
143	random	0.8	0.8	NA
144	random	0.8	0.8	NA
145	random	0.5	0.5	8.24%
146	random	0.8	0.8	NA
147	random	0.8	0.8	NA
148	random	0.8	0.8	NA
149	random	0.8	0.8	NA
150	random	0.8	0.8	NA

Bibliography

Al Jazeera (2018). Aid convoy to syria's eastern ghouta postponed amid violence. *Al Jazeera*. Accessed: 26 August 2024.

Archetti, C. and Speranza, M. G. (2008). The split delivery vehicle routing problem: A survey. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 103–122. Springer, Boston, MA.

BBC News (2016). Japan earthquake: Survivors rescued after two days. *BBC News*. Accessed: 26 August 2024.

BBC News (2019). Bridge collapse in italy: impact on transport. *BBC News*. Accessed: 26 August 2024.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, **13**, 281–305.

Bertsekas, D. (2012). *Dynamic programming and optimal control: Volume I*. Athena scientific, 4 edition.

Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, **8**, 239–287.

Binato, S., de Oliveira, G. C., and de Araujo, J. L. (2001). A greedy randomized adaptive search procedure for transmission expansion planning. *IEEE Transactions on Power Systems*, **16**(2), 247–253.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, **35**(3), 268–308.

Blum, C. and Roli, A. (2008). Hybrid metaheuristics: An introduction. In C. Blum, M. J. B. Aguilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 114 of *Studies in Computational Intelligence*, pages 1–30. Springer, Berlin, Heidelberg.

CNN (2019). How the yellow vest protests impacted paris. *CNN*. Accessed: 26 August 2024.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd edition.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, **6**(1), 80–91.

Drexl, M. and Schneider, M. (2015). A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, **241**(2), 283–308.

Dror, M., Laporte, G., and Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics*, **50**(3), 239–254.

Feo, T. A. and Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, **8**(2), 67–71.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, **6**, 109–133.

Fukunaga, A. S. (2008). Automated discovery of local search heuristics for satisfiability testing. *Evolutionary Computation*, **16**(1), 31–61.

Gary, M. R. and Johnson, D. S. (1979). Computers and intractability: A guide to the theory of np-completeness.

Gendreau, M., Potvin, J.-Y., *et al.* (2010). *Handbook of Metaheuristics*, volume 2. Springer.

Golden, B. L., Raghavan, S., and Wasil, E. A. (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43. Springer Science & Business Media.

Government of the United Kingdom (2020). New traffic regulations for heavy vehicles. GOV.UK. Accessed: 26 August 2024.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 507–523. Springer.

Juan, A. A., Faulín, J., Jorba, J., Riera, D., Masip, D., and Barrios, B. (2011). On the use of monte carlo simulation, cache and splitting techniques to improve the clarke and

wright savings heuristics. *Journal of the Operational Research Society*, **62**(6), 1085–1097.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., and Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, **2**, 62–72.

Koopmans, T. C. and Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation science*, **43**(4), 408–416.

Mladenović, N., Labbé, M., and Hansen, P. (2003). Solving the p-center problem with tabu search and variable neighborhood search. *Networks: An International Journal*, **42**(1), 48–64.

Osman, I. H. and Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations research*, **63**, 511–623.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, **34**(8), 2403–2435.

Pisinger, D. and Ropke, S. (2019). Large neighborhood search. *Handbook of Metaheuristics*, pages 99–127.

Resende, M. G. and Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. *Handbook of Metaheuristics*, pages 283–319.

Resendel, M. G. and Ribeiro, C. C. (2005). Grasp with path-relinking: Recent advances and applications. *Metaheuristics: progress as real problem solvers*, pages 29–63.

Reuters (2020). Curfews across cities impact logistics. *Reuters*. Accessed: 26 August 2024.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, **25**.

Sörensen, K. and Glover, F. (2013). Metaheuristics. *Encyclopedia of operations research and management science*, **62**, 960–970.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, **31**(2), 170–186.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. John Wiley & Sons, Hoboken, NJ.

The Guardian (2018). Genoa bridge collapse: at least 39 dead as rescuers search for survivors. *The Guardian*. Accessed: 26 August 2024.

The Japan Times (2011). Tohoku earthquake and tsunami. *The Japan Times*. Accessed: 26 August 2024.

The New York Times (2021). India's farmers' protests and their impact. *The New York Times*. Accessed: 26 August 2024.

Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM.

United Nations (2020). Humanitarian aid in conflict zones. United Nations Report. Accessed: 26 August 2024.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, **234**(3), 658–673.

Xu, M. (2017). *Vehicle Routing Problem with Interdiction*. Master's thesis, McMaster University, Hamilton, Canada. Accessed: 26 August 2024.

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.