Pontifícia Universidade Católica
do Rio de Janeiro

Bruno Guimarães de Castro

Improved Hybrid Genetic Search for the Inventory Routing Problem

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor : Prof. Hélio Côrtes Vieira Lopes
Co-advisor: Prof. Marcus Vinicius Soledade Poggi de Aragão
Co-advisor: Prof. Rafael Martinelli Pinto

Rio de Janeiro
September 2023

Pontifícia Universidade Católica do Rio de Janeiro

Bruno Guimarães de Castro

Improved Hybrid Genetic Search for the Inventory Routing Problem

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Hélio Côrtes Vieira Lopes
Advisor
Departamento de Informática – PUC-Rio

Prof. Marcus Vinicius Soledade Poggi de Aragão
Co-advisor
Departamento de Informática – PUC-Rio

Prof. Rafael Martinelli Pinto
Co-advisor
Pontífica Universidade Católica do Rio de Janeiro – PUC-Rio

Prof. Magnus Stålhane
NTNU

Prof. Pedro Augusto Munari Júnior
UFSCar

Rio de Janeiro, September 6th, 2023

Bruno Guimarães de Castro

Obtained a bachelor's degree in Computer Science (2012) from Federal University of Juiz de Fora (UFJF, Juiz de Fora, Brazil). Earned a scholarship by CAPES for the masters.

Bibliographic Data

Castro, Bruno

Improved Hybrid Genetic Search for the Inventory Routing Problem / Bruno Guimarães de Castro; advisor: Hélio Côrtes Vieira Lopes; co-advisores: Marcus Vinicius Soledade Poggi de Aragão, Rafael Martinelli Pinto. – 2023.

126 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. keywordpre – Teses. 2. keywordpre – Teses. 3. Problema de Roteamento de Inventário. 4. Busca Genética Híbrida. 5. Metaheuristics. 6. Gerenciamento de Inventário pelo Fornecedor. I. Lopes, Hélio. II. Poggi, Marcus. III. Martinelli, Rafeal. IV. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. V. Título.

CDD: 004

To my family, my loving fiancée, dedicated professors, supportive friends, and all who shaped my journey—this work is dedicated to you

Acknowledgments

I would like to express my deep gratitude to all the individuals and institutions that played a significant role in the completion of this research work and my academic journey as a whole.

First and foremost, I would like to thank Professors Marcus Poggi and Rafael Martinelli for their constant guidance, patience, and encouragement throughout this process. Their valuable guidance and insights were instrumental in the development of this work. I would also like to express my gratitude to Professor Hélio Lopes for joining the team.

To my family, for their unwavering support over the years, I am profoundly grateful. Without your love and support, this achievement would not have been possible.

To my fiancée, Ludmila, your constant presence, unconditional love, and unwavering support have been the driving force behind everything I have accomplished. Your understanding, patience, and encouragement during challenging times were essential for me to focus on my research. I am grateful to have you by my side on this journey.

To my friends and fellow classmates who shared in the joys and challenges of this academic journey with me, I thank you for your friendship and support.

To the funding sources and scholarships that made this project viable, my sincere gratitude.

I would also like to extend my thanks to the faculty and staff at PUC-Rio for providing a high-quality academic environment and resources that were essential for the completion of this work.

Lastly, I would like to thank everyone who, in one way or another, contributed to my personal and academic growth throughout this journey. Your influences were invaluable.

I am aware that this list of acknowledgments may not fully encompass all the people and institutions that were important to me during this period, but each of you had a significant impact on my academic journey.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Abstract

Castro, Bruno; Lopes, Hélio (Advisor); Poggi, Marcus (Co-Advisor); Martinelli, Rafeal (Co-Advisor). **Improved Hybrid Genetic Search for the Inventory Routing Problem**. Rio de Janeiro, 2023. 126p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Theme: This study investigates the Inventory Routing Problem (IRP) within the context of Vendor-Managed Inventory (VMI), a prevalent supply chain practice where suppliers assume responsibility for replenishment. The IRP, a combinatorial problem that has been widely studied for almost 40 years, encompasses three distinct subproblems: delivery scheduling, inventory management, and vehicle routing. **Problem:** Despite its age, the IRP continues to attract industry and academia attention. The recent 12^{th} DIMACS Implementation Challenge dedicated a track to the IRP, and among the commonly used benchmarks, 401 instances still lack optimal solutions, particularly in the challenging Large instance subset. **Hypothesis and Justification:** The HGS framework proposed by Vidal et al. (2012) emerged as a prominent tool used successfully by numerous teams in the competition. However, to the best of our knowledge, the HGS framework has not been tested for the IRP. This study proposes a method combining the HGS framework with an efficient local search strategy, namely NSIRP proposed by Diniz et al. (2020), to tackle the IRP. **Methodology:** We implemented the proposed method and compared its performance to 21 existing methods using the literature benchmarks. **Summary of Results:** Our approach identified 79 new Best Known Solutions (BKS) out of 1100 instances. If applied under the same rules as the DIMACS competition, our method would have secured the first place. **Contributions and Impacts:** This work contribute to the ongoing development of IRP methods, offering an efficient and competitive approach that may inspire further research and practical applications in the realm of inventory management and vehicle routing.

Keywords

Inventory Routing Problem; Hybrid Genetic Search; Metaheuristics; Vendor Managed Inventory.

Resumo

Castro, Bruno; Lopes, Hélio; Poggi, Marcus; Martinelli, Rafeal. **Melhoria de Busca Genética Híbrida para o Problema de Roteamento de Inventário**. Rio de Janeiro, 2023. 126p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Tema: Este estudo investiga o Problema de Roteamento de Inventário (IRP) no contexto do Gerenciamento de Inventário pelo Fornecedor (VMI), uma prática comum na cadeia de suprimentos onde os fornecedores assumem a responsabilidade pela reposição. O IRP, um problema combinatório estudado amplamente há quase 40 anos, engloba três subproblemas distintos: programação de entregas, gerenciamento de estoque e roteamento de veículos. **Problema:** Apesar de sua idade, o IRP continua a atrair a atenção da indústria e da academia. O recente 12º Desafio de Implementação DIMACS dedicou uma categoria ao IRP, e entre os benchmarks comumente utilizados, 401 instâncias ainda não possuem soluções ótimas, especialmente no desafiador subconjunto de instâncias grandes. **Hipótese e Justificativa:** O framework HGS proposto por Vidal et al. (2012) surgiu como uma ferramenta proeminente utilizada por várias equipes de forma satisfatória na competição. No entanto, até onde sabemos, o framework HGS não foi testada para o IRP. Este estudo propõe uma solução que combina o framework HGS com uma estratégia de busca local eficiente, o método NSIRP proposto por Diniz et al. (2020), para abordar o IRP. **Metodologia:** Implementamos a solução proposta e comparamos seu desempenho com 21 abordagens existentes, utilizando os benchmarks da literatura. **Resumo dos Resultados:** Nossa abordagem identificou 79 novas Melhores Soluções Conhecidas (BKS) dentre 1100 instâncias. Se aplicada sob as mesmas regras da competição DIMACS, nossa solução teria garantido o primeiro lugar. **Contribuições e Impactos:** Este trabalho contribui para o desenvolvimento contínuo de soluções para o IRP, oferecendo uma abordagem eficiente e competitiva que pode inspirar futuras pesquisas e aplicações práticas no campo do gerenciamento de estoque e roteamento de veículos.

Palavras-chave

Problema de Roteamento de Inventário; Busca Genética Híbrida; Metaheuristics; Gerenciamento de Inventário pelo Fornecedor.

Table of Contents

List of Figures

List of Tables

List of Acronym

ALNS	Adaptative Large Neighborhood Search
B&C	Branch and Cut
BP&C	Branch Price and Cut
BKLB	Best Known Lower Bound
BKS	Best Known Solution
CVRP	Capacitated Vehicle Routing Problem
DSIRP	Dynamic and Stochastic IRP
FFNS	Fast Flow Network Simplex
GA	Genetic Algorithm
HGS	Hybrid Genetic Search
HGS-CVRP	Hybrid Genetic Search for the CVRP
IRP	Inventory Routing Problem
ILS	Iterated Local Search
ILS-RVND	Iterated Local Search with Variable Neighborhood Descent with Random Neighborhood
LB	Lower Bound
ML	Maximum-Level
MCFP	Minimum Cost Flow Problem
MILP	Mixed Integer Linear Programming
NSIRP	Network Simplex IRP
OU	Order-Up-to-Level

PRP	Production Routing Problem
PTNU	Periodic-Tour-Non-Uniform
RVND	Variable Neighborhood Descent with Random Neighborhood
SA	Simulated Annealing
SIRP	Stochastic IRP
TAF	Time Adjust Factor
TSP	Traveling Salesperson Problem
UB	Upper Bound
VRP	Vehicle Routing Problem
VMI	Vender-Managed Inventory

List of Symbols

n – A client n

N – The total number of clients

$\mathcal{N}$ – The set of clients

0 – The supplier n

t – A period t

T – The total number of periods T

$\mathcal{T}$ – The set of periods

k – A vehicle

K – The total number of vehicles K

$\mathcal{K}$ – The set of vehicles $\mathcal{K}$

Q – The vehicle capacity Q

c_{ij} – The transportation cost from client i to a client j

h_n – The inventory cost from a client n

I_n^0 – The initial inventory level from a client n

I_n^t – The inventory level from a client n at period t

L_n – The minimum inventory capacity from a client n

U_n – The maximum inventory capacity from a client n

d_n^t – The product consumption from a client n at period t

s – A solution s

r – A route r

$\mathcal{R}(s)$ – The set of routes from a solution s

$nbIter$ – The number of iterations without improvement until termination

T_{MAX} – The CPU time limit until termination (in seconds)

$\mathcal{P}_{fes}$ – The feasible subpopulation $\mathcal{P}_{fes}$

$\mathcal{P}_{inf}$ – The infeasible subpopulation $\mathcal{P}_{inf}$

α – The initial population size

$\mathcal{P}$ – The entire population $\mathcal{P}$

$|\mathcal{P}|$ – The size of population $\mathcal{P}$

s – The individual s

s_o – The offspring generated after crossovers s_1 and s_2

s_1 – A first parent

s_2 – A second parent

π_n^t – The chromosome π – If the client n is attended or not at period t

τ^t – The chromosome τ – A giant tour of period t represented by connecting all routes from all $\mathcal{K}$

κ_k^t – The chromosome κ – The routing the k go makes during period t

δ_n^t – The chromosome δ – The amount delivered to client n at period t

ω^Q – The penalty for exceeding the vehicle capacity

1 Introduction

The Inventory Routing Problem (IRP) is a well-known combinatorial problem in the literature that combines three distinct challenges: delivery scheduling, inventory management, and vehicle routing. This problem often arises in the context of Vender-Managed Inventory (VMI), a prevalent business practice in supply chain management where the supplier assumes responsibility for replenishing a customer's inventory based on their supply chain policies.

To successfully manage VMI, the supplier must make three critical decisions: (1) when to serve their clients, (2) the appropriate quantity of products to deliver, and (3) the optimal routing strategy for their vehicles. These decisions are highly interdependent and must take into account a range of complex factors, such as transportation costs, inventory carrying costs, customer consumption, and vehicle capacity.

By integrating inventory management and routing decisions, companies can reduce their overall costs associated with inventory holding, transportation, and production. This results in increased operational efficiency and profitability. The IRP is particularly relevant in industries such as food and beverage, retail, and transportation, where the management of inventory and logistics plays a crucial role in achieving customer satisfaction and minimizing operational costs.

Despite the age of the IRP, it continues to attract attention from both industry and academia. In the recent 12$^{\text{th}}$ DIMACS Implementation Challenge (DIMACS, 2022) a track was dedicated to the IRP, and among the commonly used benchmarks, 401 instances still lack optimal solutions, particularly in the challenging Large instance subset. Developing effective methods to the IRP will not only contribute to the academic literature but also provide valuable tools for practitioners in various industries, ultimately improving supply chain performance and customer satisfaction.

In recent years, various methods have been proposed to address the IRP, with two particular approaches capturing our interest. The first method is the Network Simplex IRP (NSIRP), proposed by Diniz et al. (2020). NSIRP is specifically tailored for the IRP and boasts a highly effective local search strategy. The second approach is the Hybrid Genetic Search (HGS), introduced by Vidal et al. (2012), which has gained significant prominence in the context of the Vehicle Routing Problem (VRP). Notably, the open-source implementation by Vidal (2022) has played a key role in numerous successful algorithms during the 12^{th} DIMACS Implementation Challenge. While HGS has not been originally developed for the IRP, it has shown great potential for adaptation to this problem domain, given its success in the broader VRP field.

Given the previous state, we formulate the following research question: Can the combination of the HGS framework and the NSIRP local search strategy yield an effective method for the Inventory Routing Problem?

We hypothesize that the combination of the HGS framework and the NSIRP local search strategy will result in a competitive and effective method for the IRP. This hypothesis is based on the previous success of the HGS framework in the competition and the potential synergy between the HGS framework and the NSIRP local search strategy.

We implemented the proposed method to address the research question and compared its performance to 21 existing methods using the literature benchmarks.

The main contributions expected from this study are developing an effective and competitive method for the IRP and identifying new Best Known Solution (BKS) for the literature benchmark using the proposed method.

In order to achieve the research objective, the following specific objectives have been established:

- Develop a new implementation that combines the HGS framework with the NSIRP local search strategy for the IRP.
- Evaluate the performance of the proposed method using benchmark problem instances from the literature.
- Compare the performance of the proposed method to existing methods in terms of method quality and computational efficiency.
- Contribute to the ongoing development of IRP methods and inspire further research in the field of inventory management and vehicle routing.

Numerous studies have been conducted in the last decade, addressing a wide variety of IRPs and their applications. These IRPs differ mainly in terms of the time horizon (finite or infinite), the structure of the distribution network (one-to-one, one-to-many, many-to-many), inventory replenishment policy (the two most common are the ML and OU), fleet size (single, multi-vehicle, or unconstrained), fleet composition (homogeneous or heterogeneous vehicles), and information on customer consumption (deterministic or stochastic).

This work focuses on the basic variant, as defined in Coelho et al. (2014), where a single supplier distributes a single product over a finite time horizon, using a fleet of homogeneous vehicles to serve clients with deterministic consumption. This scope delimitation was chosen based on numerous published papers and the possibility of comparison using a popular set of benchmark problem instances.

The remainder of this document is organized as follows:

Chapter 2: Literature Review - This chapter presents the problem statement, the exact and approximate methods used to solve the IRP, and the current solution status of the most widely used literature benchmark instances.

Chapter 3: Background - This chapter provides an overview of the two methods that inspired this work: the NSIRP and the HGS framework. It discusses their respective approaches and the reasons for their selection in the development of the proposed method.

Chapter 4: Proposed Methodology - This chapter outlines the main ideas of this work, describing how the integration of the NSIRP and HGS methods was made possible, and discussing three new improvements introduced to enhance their performance in solving the IRP.

Chapter 5: Computational Experiments and Analysis - This chapter describes the experiments conducted to evaluate the proposed method, detailing the classical benchmark instances used, the experimental setup, and the performance measures adopted to assess the effectiveness and efficiency of the proposed method.

Chapter 6: Conclusions - This final chapter revisits the main contributions of this work, summarizing the findings and the impact of the proposed method on the IRP literature. It also suggests possible future directions for further research and improvements in the field of inventory management and vehicle routing.

2
Literature review

The IRP has received significant attention in the operations research and logistics literature due to its practical relevance and inherent difficulty. This chapter provides a comprehensive review of the existing literature on the IRP, highlighting the most relevant and recent contributions regarding exact and approximate methods, surveys, and the classical benchmark instances in together with its current performance state.

The chapter is organized as follows:

– Section 2.1: This section introduces the IRP, its history, main variations, and basic variant.

– Section 2.2: We present a comprehensive review of the literature methods for resolving the IRP. This section will specifically examine both exact and approximate approaches, providing a thorough understanding of the methods used to address this problem.

– Section 2.3: In this section, we provide an overview of the surveys on the IRP, focusing on their different perspectives, such as applications, characteristics, modeling approaches, and methodological aspects.

– Section 2.4: This section describes the widely used benchmark instances for the IRP, including those proposed by Archetti et al. (2007) and Archetti et al. (2012). It also discusses the extension of these instances to the multi-vehicle IRP by Coelho et al. (2012a). The section provides a detailed breakdown of the instances in terms of the number of clients, vehicles, inventory cost types, and periods.

– Section 2.5: Lastly, we outline the current state of the classical benchmark instances, highlighting solved and open instances, as well as the overall performance of the methods in terms of optimality and Duality Gap.

2.1 Problem statement

The IRP is a complex optimization problem that integrates inventory management, vehicle routing, and delivery scheduling decisions. It originated from the seminal paper by Bell et al. (1983) in the context of Vender-Managed Inventory (VMI), which aims to reduce logistics costs and add business value by having suppliers manage product replenishment for customers based on specific inventory and supply chain policies. Early studies on the IRP, such as those by Federgruen and Zipkin (1984), Blumenfeld et al. (1985), and Dror and Ball (1987), adapted Vehicle Routing Problem (VRP) models and heuristics to consider inventory costs, production setup costs, and stochastic consumption environments.

However, these initial contributions faced challenges in integrating distribution and inventory problems due to limited computing power and the difficulty of handling large combinatorial problems.

2.1.1 Main variations

Coelho et al. (2014) classifies the IRPs according to its structural variants and the availability of consumption information. Structural criteria include time horizon, structure, routing, inventory policy, inventory decisions, fleet composition, and fleet size. The time horizon can be finite or infinite, while the structure varies from one-to-one, one-to-many, or many-to-many, depending on the number of suppliers and customers. Routing can be direct, multiple, or continuous. Inventory policies include Maximum-Level (ML) and Order-Up-to-Level (OU) policies. Inventory decisions involve back-ordering, lost sales, or nonnegative. Fleet composition can be homogeneous or heterogeneous, and fleet size can be fixed (single or multi-vehicle) or unconstrained. The second classification, availability of consumption information, covers deterministic, stochastic (SIRP), dynamic, and dynamic and stochastic inventory-routing problems (DSIRP). This classification scheme separates problem structure from information availability, allowing for clearer distinctions between models and algorithms.

2.1.2 The basic variant

The basic variant of the IRP, as described by Coelho et al. (2014), considers a one-to-many structure, where there is a single supplier denoted as *0*, and a set of customers represented by $\mathcal{N} = \{n \text{ is integer} \mid 1 \leq n \leq N\}$. The planning horizon is finite, defined by the set of time periods $\mathcal{T} = \{t \text{ is integer} \mid 1 \leq t \leq T\}$, and a set of vehicles is given by $\mathcal{K} = \{k \text{ is integer} \mid 1 \leq k \leq K\}$.

Consumption is assumed to be deterministic, with the supplier producing a specified quantity of product d_0^t at each time period t, and each customer n consuming d_n^t. The vehicles are assumed to be homogeneous with a specified capacity Q, and each customer should be visited by no more than one vehicle per period.

The initial inventory level for the supplier is denoted as I_0^0, and for each customer n, it is I_n^0. The inventory level for a customer n in a period t is given by $I_n^t = I_n^{t-1} + q_n^t - d_n^t$, where q_n^t denotes the quantity delivered to customer n at period t.

The total quantity of products that a vehicle k can carry must not exceed Q, and the inventory levels I_n^t for each customer n in each period t must remain within the bounds of L_n and $U_n - d_n^t$.

The Figure 2.1 depicts the sequence of events. The quantity delivered q_n^t to customer n at period t occur prior to the customer's consumption d_n^t.

This imposes that $I_n^{t-1} + q_n^t \leq U_n$. In other words, the sum of the last period's remaining inventory and the current delivery cannot exceed the customer's maximum storage capacity.

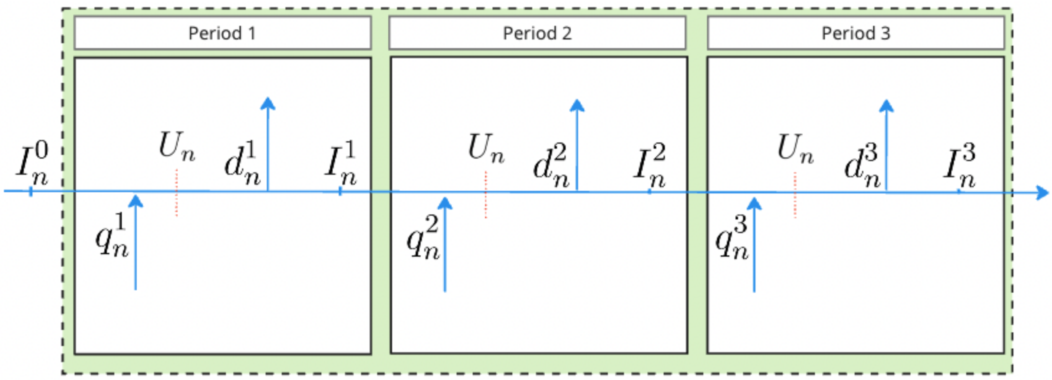

Figure 2.1: Sequence of events for a client n in IRP. The delivery must occur prior to the customer's consumption

The IRP is subject to two types of costs: inventory and transportation costs. The supplier incurs an inventory cost per product unit defined by h_0, while customers incur a cost of h_n. The transportation cost is given by c_{ij}, where i and j denote customers or the supplier.

The decision-maker has knowledge of the current inventory levels of the supplier and customers, as well as the consumption of each customer for every time period. The objective is to minimize the total inventory distribution cost while satisfying constraints such as maximum inventory capacity, non-negative inventory levels, vehicle routing, and vehicle capacities.

The basic variant of the IRP is considered NP-hard since it includes the classical VRP. However, effective algorithms have been proposed for this problem, such as Mixed Integer Linear Programming (MILP) models and heuristic algorithms, including simulated annealing, and tabu search. These approaches provide solutions to the basic IRP and its variants and have contributed significantly to the development of practical inventory routing systems.

2.2 Literature methods

The purpose of this section is to provide a comprehensive overview of the literature methods used to solve the IRP. The section will cover the following topics:

- Subsection 2.2.1: This section presents an overview of the exact methods for solving the IRP, including the single-vehicle and multi-vehicle cases.
- Subsection 2.2.2: Here, we discuss the approximate methods for the IRP, emphasizing the motivation for using approximates in larger instances, where exact methods may not provide optimal solutions within a reasonable time frame. The section covers single-vehicle algorithms, multi-vehicle algorithms, and recent innovative algorithms that have gained attention in the literature.

2.2.1 Exact methods

This section presents the literature on exact methods for solving the IRP.

Archetti et al. (2007) were the first to propose a Branch and Cut (B&C) approach for the single-vehicle IRP. They introduced the first set of valid inequalities and demonstrated the advantages of using the ML inventory policy over the OU policy. Ǒguz Solyali and Süral (2011) later improved upon this work with a stronger formulation employing shortest-path networks representing customer replenishments and a heuristic to provide an initial upper bound for the B&C approach. Furthermore, Avella et al. (2015) suggested a novel two-index vehicle-flow formulation, applicable when the inventory capacity at each customer is an integer multiple of consumption.

For the multi-vehicle IRP, Coelho and Laporte (2013) adapted the valid inequalities introduced by Archetti et al. (2007). Subsequently, Coelho and Laporte (2014) proposed a three-index formulation and new valid inequalities to bound the minimum number of visits per customer over consecutive periods of the horizon. Adulyasak et al. (2014) presented a two-index arc-flow formulation, adding capacitated sub-tour elimination constraints dynamically. Avella et al. (2018) adapted the work of Avella et al. (2015) for the multi-vehicle case and introduced a new family of generic valid inequalities. Guimarães et al. (2020) suggested new mechanisms for enhancing IRP solution feasibility that can be incorporated into exact methods and heuristics, employing two techniques to find primal solutions during the B&C search.

Four recent studies have gained prominence in the field. Two of these works focus on two-index vehicle-flow formulations. In one, Manousakis et al. (2021) expands the formulations to accommodate a two-commodity flow, whereas Skålnes et al. (2022) employs a customer schedule reformulation and modifies the capacity inequalities put forth by Desaulniers et al. (2016). In another notable study, Schenekemberg et al. (2023) utilizes the two-index B&C approach, complemented by a three-index B&C and a matheuristic method that runs parallelly and shares information and stopping criteria. Lastly, Skålnes et al. (2023b) achieves significant results with a branch-and-cut embedded metaheuristic framework, which fuses a construction heuristic with an improvement heuristic to generate and optimize routes, thereby surpassing previously established methods for tackling the problem.

Distinct among the exact methods for the IRP is the Branch Price and Cut (BP&C) algorithm proposed by Desaulniers et al. (2016). The authors introduced a column generation algorithm embedded within a B&C approach, achieving better lower and upper bounds for more complex instances.

Table 2.1 summarizes the exact methods discussed above, listing them chronologically from the earliest to the most recent publications. The table provides information on the reference, year of publication, and approach used for solving the IRP.

2.2.2 Approximate methods

This subsection presents the literature on approximate methods for solving the IRP.

None of the exact methods described in Section 2.2.1 have been able to solve larger multi-vehicle instances to optimality. As instance size increases, the Duality Gap can become very large, often resulting in no feasible solution found within a reasonable time frame when using an exact method.

Archetti et al. (2012) proposed the first approximate method for the classical IRP and introduced a large set of widely used benchmark instances. Their Tabu search algorithm features an improvement phase that solves MILP

Table 2.1: Summary of exact methods for solving the IRP in the literature

Refference	Year	Approach
Archetti et al. (2007)	2007	B&C
Ǒguz Solyali and Süral (2011)	2011	B&C
Coelho and Laporte (2013)	2014	B&C
Adulyasak et al. (2014)	2014	B&C
Coelho and Laporte (2014)	2014	B&C
Avella et al. (2015)	2015	B&C
Desaulniers et al. (2016)	2016	BP&C
Avella et al. (2018)	2018	B&C
Guimarães et al. (2020)	2020	B&C
Manousakis et al. (2021)	2021	B&C
Skålnes et al. (2022)	2022	B&C
Schenekemberg et al. (2023)	2022	B&C
Skålnes et al. (2023b)	2023	B&C

problems. The algorithm starts from a feasible solution and explores the neighborhood of the current solution while performing occasional jumps to new regions of the search space. Two Adaptative Large Neighborhood Search (ALNS) methods followed: Coelho et al. (2012b) for a single vehicle, and its extended version Coelho et al. (2012a), which introduced the multi-vehicle case for the benchmark set.

After Coelho et al. (2012a), new approximate methods for multi-vehicle instances emerged. Adulyasak et al. (2014) also proposed an ALNS method, while Santos et al. (2016) suggested an Iterated Local Search (ILS) with a hybrid multi-start. Archetti et al. (2017) extended their previous work Archetti et al. (2012) for the multi-vehicle case. In Alvarez et al. (2018), two heuristics were proposed: a Simulated Annealing (SA) and an ILS. Chitsaz et al. (2019) proposed a three-phase decomposition. Alvarez et al. (2020) suggested a hybrid heuristic using ILS and MILPs for perishable products that could be adapted to the IRP.

More recently, several works have gained attention. Diniz et al. (2020) proposed an effective local search based on a modification of the network simplex, which will be discussed in Chapter 3. Archetti et al. (2021) suggested a kernel search that uses information gathered by a Tabu search to create a sequence of MILPs. Vadseth et al. (2021) developed a method where the initial route set is created from a giant tour using a split algorithm, and then iteratively solves a route-based MILP, altering the set of routes between each iteration. Solyalı and Süral (2022) introduced an algorithm that sequentially solves different mixed integer linear programs. Achamrah et al. (2022) presented a two-phase matheuristic, combining MILP and hybridizing Genetic Algorithm (GA) and SA. Lastly, Vadseth et al. (2023) described an algorithm that constructs a feasible starting solution using a traditional decomposition approach and improves it with a path-flow-inspired model.

Table 2.2 outlines the approximate methods described previously. The methods are listed chronologically, from the earliest to the most recent publications. The table provides information on the reference, year of publication, and approach used for solving the IRP.

Table 2.2: Summary of approximate methods for solving the IRP in the literature

Reference	Year	Approach
Archetti et al. (2012)	2012	TABU + MILP
Coelho et al. (2012b)	2012	ALNS
Coelho et al. (2012a)	2012	ALNS
Adulyasak et al. (2014)	2014	ALNS
Santos et al. (2016)	2016	ILS-RVND
Archetti et al. (2017)	2017	TABU + MILP
Alvarez et al. (2018)	2018	SA
Alvarez et al. (2018)	2018	ILS
Chitsaz et al. (2019)	2019	DECOMPOSITION
Alvarez et al. (2020)	2020	ILS + MILP
Diniz et al. (2020)	2020	ILS-RVND + NS
Archetti et al. (2021)	2021	KERNEL
Vadseth et al. (2021)	2021	DECOMPOSITION
Sakhri et al. (2022)	2021	GA + VNS
Solyalı and Süral (2022)	2022	DECOMPOSITION
Achamrah et al. (2022)	2022	GA+SA+MILP
Vadseth et al. (2023)	2023	DECOMPOSITION

2.3 Surveys

Several surveys have been written on the IRP scope. Andersson et al. (2010) focused on different applications of the IRP. Bertazzi and Speranza (2012) and Bertazzi and Speranza (2013) classify the characteristics of an IRP and present different models and policies for the IRP. Coelho et al. (2014) studied the methodological aspects. More recently, Roldán et al. (2017) studies stochastic versions.

2.4
Classical benchmark instances

The instances proposed by Archetti et al. (2007) are among the most widely used in the literature and consist of 160 instances in total. These instances have different numbers of customers, ranging from 5 to 50, and two different types of inventory costs (low and high). There are also two time horizons: a three-period and a six-period horizon. The latter is only available for instances with the number of customers from 5 to 30.

Later, Archetti et al. (2012) proposed 60 new instances with the number of customers of 50, 100, and 200, and a planning horizon of six-time periods. These instances are collectively referred to as the Large set, while the instances from Archetti et al. (2007) are referred to as the Small set.

In 2012, Coelho et al. (2012a) extended the instances to a multi-vehicle IRP by dividing the original vehicle capacity by the number of vehicles and rounding to the nearest integer. This resulted in 640 small instances and 240 large instances for vehicles ranging from two to five.

Tables 2.3 and 2.4 summarize the different IRP instances used in the literature. Table 2.3 gives an overview of the total number of single-vehicle and multi-vehicle instances, separated into Small and Large categories, while Table 2.4 provides a detailed breakdown of the instances from the three main works, including the number of clients, vehicles, inventory cost type, and the number of periods. There are a total of 1100 instances, with 220 single-vehicle instances and 880 multi-vehicle instances.

Table 2.3: Overview of IRP instances in the classical benchmark, categorized by size and vehicle count

	Single-Vehicle	**Multi-Vehicle**	**Total**
Small	160	640	800
Large	60	240	300
Total	220	880	1100

2.5
Literature methods on classical benchmark instances

In this section, we demonstrate the current state of classical benchmark instances by using the literature methods previously discussed. Given the 29 total studies, we excluded 8 from our analysis because they did not provide detailed results for each instance. The 21 remaining methods used to calculate the current state can be seen on Table 2.5.

Table 2.4: Detailed breakdown of IRP instances in the literature benchmark by paper, type, client count, vehicle count, inventory cost, and periods

Paper	**Type**	**Instances**	**#Clients**	**#Vehicles**	**Inv. Cost Type**	**Periods**	**Total**
Archetti et al. (2007)	Small	5	5, 10, 15, 20, 25, 30	1	High, Low	3, 6	120
			35, 40, 45, 50			3	40
Archetti et al. (2012)	Large	10	50, 100, 200	1	High, Low	6	60
Coelho et al. (2012a)	Small	5	5, 10, 15, 20, 25, 30	2, 3, 4, 5	High, Low	3, 6	480
			35, 40, 45, 50			3	160
	Large	10	50, 100, 200			6	240

Table 2.5: Summary of 21 literature methods used to calculate the current state of the IRP classical benchmark instances

Reference	Type
Archetti et al. (2012)	Approximate
Coelho and Laporte (2013)	Exact
Coelho and Laporte (2014)	Exact
Adulyasak et al. (2014)	Exact
Desaulniers et al. (2016)	Exact
Archetti et al. (2017)	Approximate
Alvarez et al. (2018)	Approximate
Avella et al. (2018)	Exact
Chitsaz et al. (2019)	Approximate
Alvarez et al. (2020)	Approximate
Diniz et al. (2020)	Approximate
Guimarães et al. (2020)	Exact
Manousakis et al. (2021)	Exact
Archetti et al. (2021)	Approximate
Vadseth et al. (2021)	Approximate
Skålnes et al. (2022)	Exact
Solyalı and Süral (2022)	Approximate
Achamrah et al. (2022)	Approximate
Schenekemberg et al. (2023)	Exact
Vadseth et al. (2023)	Approximate
Skålnes et al. (2023b)	Exact

Table 2.6 comprises the results of the 1100 classical benchmark instances. However, two of them were found to be unfeasible, leaving us with 1098 feasible instances.

When we analyzed all the studies and compares the best Upper Bound (UB) and the best Lower Bound (LB) identified by the exact methods, we found an odd occurrence - 148 instances showed the LB to be larger than the UB. This isn't something that should happen normally, but upon closer scrutiny, it was revealed that 134 of these had a difference of less than 0.3 cost units. In these cases, we treated the LB and UB as equal.

There were 14 results with a LB that formed a discrepancy larger than 0.3. These results were left out of this analysis and can be found detailed in Appendix B.

Of the total 1098 instances, 697 were solved optimally, leaving 401 instances still unresolved. It's important to mention that none of the larger instances involving multiple vehicles were optimally solved. The **Duality Gap**, a measure of the difference between the best possible solution, and the actual best-known solution, is 0.80%. The calculation for this percentage is as follows:

$$\frac{UB - LB}{LB}$$

Table 2.6: Performance of literature on IRP classical benchmark instances categorized by size and vehicle count

		Single-Vehicle		**Multi-Vehicle**	
		Small	**Large**	**Small**	**Large**
Instances	**1098 (100%)**	160	60	638	240
Optimals	**697 (63.48%)**	160	23	514	0
Open	**401 (36.52%)**	0	37	124	240
Avg. Duality Gap	**0.80%**	**0%**	**0.80%**	**0.16%**	**3.03%**

3
Background

This chapter provides an overview of the two methods that inspired this work: Network Simplex IRP (NSIRP) and the Hybrid Genetic Search (HGS) framework. We start with a brief overview of the NSIRP algorithm, which is specifically designed for the IRP and features a highly efficient local search strategy. We present its main components and how they are integrated to solve the IRP. Next, we introduce the HGS framework, a popular approximate algorithm for the Vehicle Routing Problem (VRP) that has been successfully applied in the literature. We describe the main ideas of the HGS framework and its main components, including its advanced population management. Finally, we discuss the reasons behind the selection of these two methods and their potential synergies in solving the IRP.

The chapter is organized as follows:

- Section 3.1: This section presents an overview of the NSIRP algorithm, discussing its main components and how they are integrated to solve the IRP.
- Section 3.2: Here, we introduce the HGS framework, discussing its main components and how they are applied to solve the VRP.
- Section 3.3: Lastly, we discuss the reasons for the selection of these two methods and their potential synergies in solving the IRP.

3.1
Network Simplex IRP (NSIRP)

The NSIRP algorithm, as proposed in Diniz et al. (2020), is an Iterated Local Search with Variable Neighborhood Descent with Random Neighborhood (ILS-RVND) (Subramanian, 2012) specifically designed to efficiently address the IRP. The work demonstrates significant improvements, achieving better upper bounds for 113 out of 640 instances in the small multi-vehicle category from the classical benchmark.

One of the primary strengths of the NSIRP algorithm lies in its highly effective local search, which significantly diminishes the search space required for scheduling and routing decisions. Inventory decisions are solved optimally by modeling them as a Minimum Cost Flow Problem (MCFP) and subsequently employing a network simplex method to obtain the solution. Since the network simplex is an exact method, it can require significant computational resources. However, the NSIRP algorithm improves the network simplex by introducing modifications that greatly enhance its efficiency. This results in a performance improvement of ten times compared to the most advanced network simplex implementations currently available.

3.1.1 Outline the algorithm

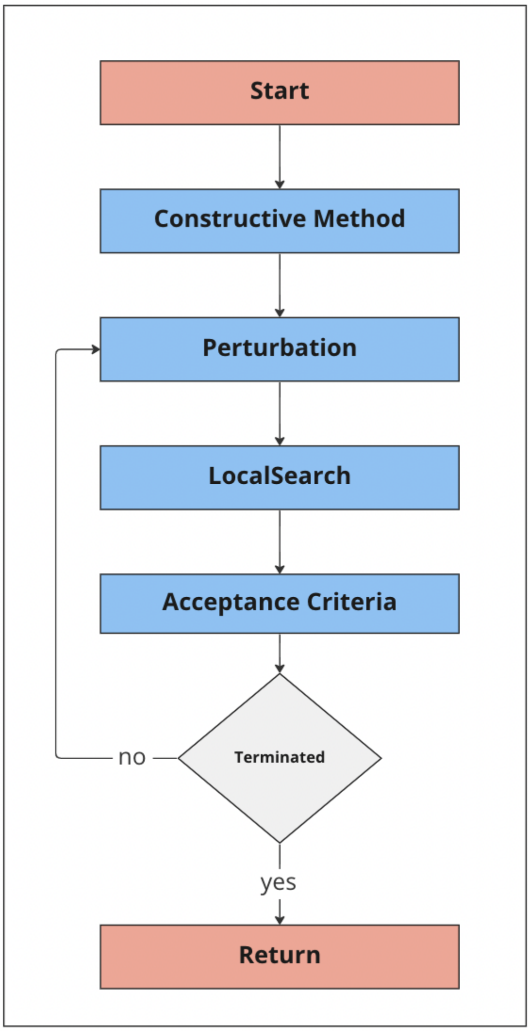

Figure 3.1: NSIRP Outline

The NSIRP algorithm, as illustrated in Figure 3.1, comprises several key components, including the generation of the initial solution, local search, perturbation, and acceptance criterion. The following sections provide an overview of each block in the NSIRP algorithm.

Initial solution generation: The first step involves creating a preliminary solution, essentially an empty solution with no clients attended. Although the NSIRP algorithm does not have a robust constructive heuristic for generating initial solutions, its local search operators effectively enhance the solution quality.

Perturbation: Once the initial solution is generated, it enters the main loop of the Iterated Local Search (ILS) framework, which consists of perturbation, local search, and acceptance criterion. Perturbations introduce diversity into the search process and help escape local optima. The NSIRP employs a simple yet effective perturbation strategy by selecting a random neighborhood of the current incumbent solution and attempting 15 times to find an improved or slightly worse solution. The slight deviation is controlled by a simulated annealing process.

Local search: The incumbent solution then undergoes the local search step, executed by the Variable Neighborhood Descent with Random Neighborhood (RVND) algorithm. RVND comprises multiple neighborhoods to effectively explore the search space. It performs a local search on each neighborhood, with the order of neighborhood exploration determined randomly. Whenever a neighborhood identifies a superior solution, the process restarts. The algorithm concludes when no improvements are found across all neighborhoods.

Acceptance criterion: The final component of the ILS framework is the acceptance criterion, which determines if a newly generated candidate solution should replace the incumbent solution. In the NSIRP, the initial acceptance criterion allows for a 20% probability of accepting solutions that are 20% worse. As the algorithm progresses, the acceptance criterion becomes more stringent; after 500 iterations of the ILS, the probability of accepting solutions 1% worse decreases to 1%.

3.1.2 Local search neighborhoods

The IRP involves three types of decisions: scheduling, routing, and inventory. The large neighborhoods resulting from these variable decisions can be reduced by using an indirect representation of the solution. This approach explores smaller neighborhoods, and each solution is mapped to a complete solution through a decoder.

The NSIRP uses scheduling and routing as the solution representation. During the local search, for each solution representation, the algorithm determines the optimal inventory by modeling it as a MCFP. This process is illustrated on Figure 3.2.

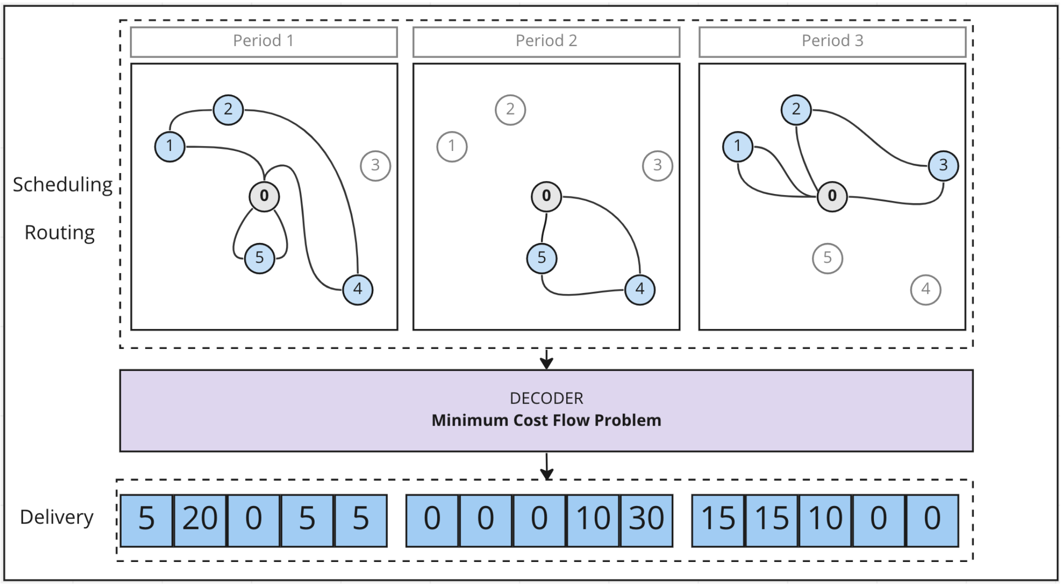

Figure 3.2: Example the use of the MCFP as a decoder to map an incomplete IRP solution based on two decision variables, Scheduling, and Routing to a complete solution with the Delivery decision variables

The NSIRP incorporates a total of six neighborhoods: Insert, Remove, Relocate, Swap, Shift, and 2-Opt.

Insert and Remove neighborhoods have similar sizes, both with a complexity of $\mathcal{O}(T * K * N)$. The Insert operation adds a client n in a period t using a vehicle k. In contrast, the Remove operation eliminates the client n from the period t and vehicle k. Figure 3.3 and Figure 3.4 exemplifies both neighborhoods.

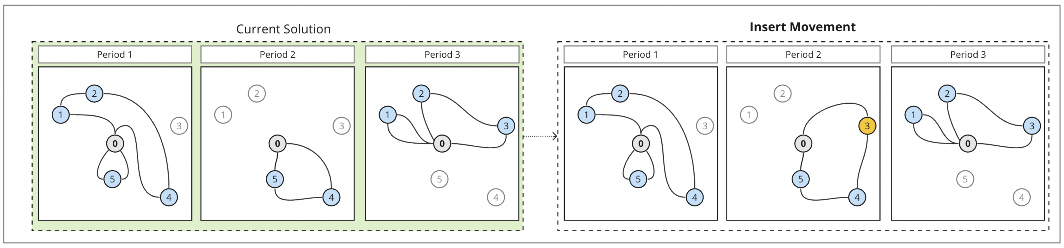

Figure 3.3: Example of Insert movement. Client 3 is inserted in Vehicle 1 from Period 2.

Relocate and Swap neighborhoods also exhibit similar sizes, both with a complexity of $\mathcal{O}(T^2 * K^2 * N^2)$. The Relocate operation selects a client n from period t_1 using vehicle k_1 and moves it to period t_2 in route k_2. This operation can be viewed as a combination of Remove and Insert operations. Similarly, the Swap operation functions like Relocate but also reassigns the customer

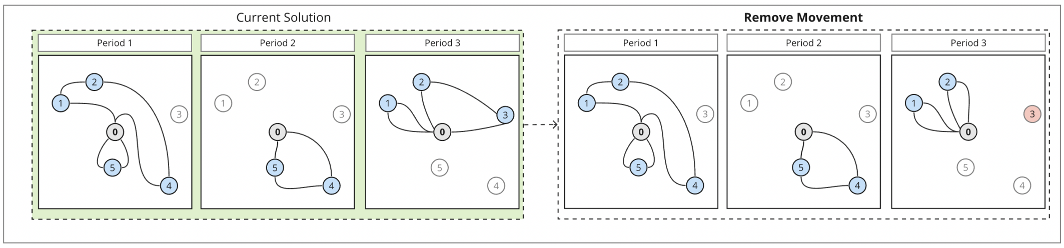

Figure 3.4: Example of Remove movement. Client 3 is removed from Vehicle 2 from Period 3.

from t_2 and route k_2 back to t_1 using vehicle k_1. Figure 3.5 and Figure 3.6 exemplifies both neighborhoods.

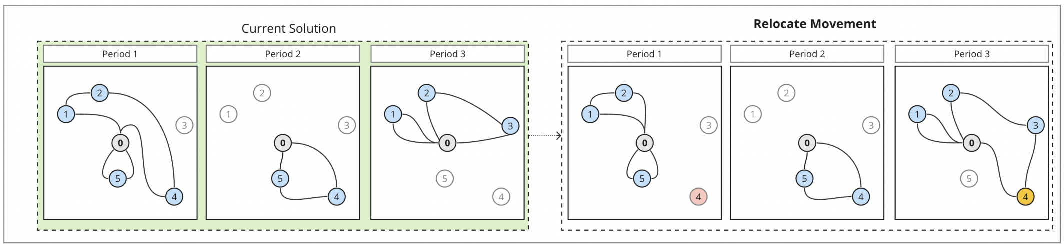

Figure 3.5: Example of Relocate movement. Client 4 is removed from Route 1 from Period 1 and inserted in Vehicle 2 from Period 3.

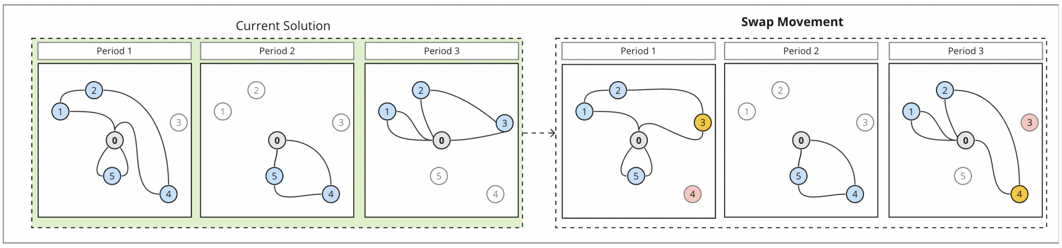

Figure 3.6: Example Swap movement. Client 4 is removed from Route 1 from Period 1, and swapped with Client 3 from Route 2 in Period 3.

Shift and 2-Opt neighborhoods share a common characteristic: both involve intra-route movements, which means that the inventory solution for the neighboring solution remains unchanged. The Shift operation has a complexity of $\mathcal{O}(T * K * N)$. It selects a client n from period t using vehicle k and moves it one position forward. The 2-Opt operation has a complexity of $\mathcal{O}(T * K * N^2)$. It selects clients n_1 and n_2 from period t using vehicle k and reverses the sub-tour between them. Figures 3.7 and 3.8 exemplify these neighborhoods.
Overall, these neighborhoods enable the NSIRP algorithm to explore various solution spaces effectively. By employing a combination of intra-route and inter-route movements, the local search process is capable of finding high-quality solutions while maintaining efficiency. The indirect representation of

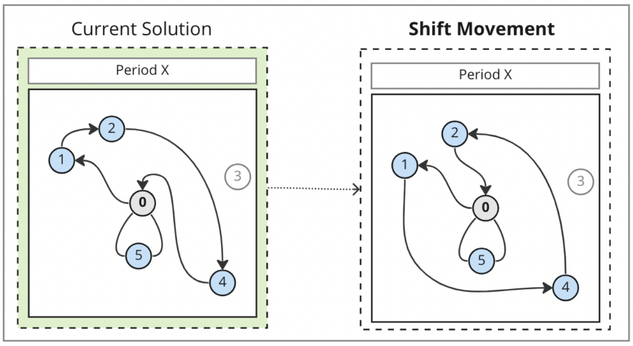

Figure 3.7: Example of Shift movement. Client 2 in Vehicle 1 was shift and the route changed from $1-2-4$ to $1-4-2$.

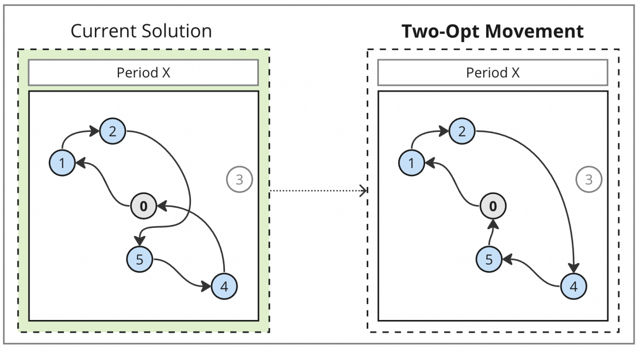

Figure 3.8: Example of 2-Opt movement. The route was changed from $1-2-5-4$ to $1-2-4-5$.

the solution further contributes to the algorithm's effectiveness by reducing the search space size and allowing the decoder to map partial solutions to complete ones.

3.1.3 Modeling with Minimum Cost Flow Problem (MCFP)

Based on Orlin's (1983) ideas, the NSIRP algorithm models the inventory decisions as a MCFP, which allows it to efficiently find its optimal inventory for a given routing and scheduling solution. The proposed network flow model consists of the following components, as illustrated in Figure 3.9:

Supplier Nodes: A node s_t is created for every period $t \in \mathcal{T}$. Each supplier node represents the production of d_s^t units of products. To handle the initial inventory in the supplier, the s_1 supplies $I_s^0 + d_s^1$ units of products.

Client Nodes: A node $c_{n,t}$ is created for every client $n \in \mathcal{N}$ and for every period $t \in \mathcal{T}$. Each client node represents the consumption for d_n^t units of products. To handle the initial inventory in the clients, the $c_{n,1}$ consumes $d_n^1 - I_n^0$ units of products.

Vehicle Nodes: A node $v_{k,t}$ is created for every vehicle $k \in \mathcal{K}$ and for every period $t \in \mathcal{T}$. Each vehicle node represents a vehicle that is available to transport products with no consumption for products itself.

Excess Product Node: A single node e is created to represent the excess of product that may exist at the end of the time horizon. This node has a consumption equal to the difference between the total supply and the total consumption.

Arcs for Vehicle Transport: An arc is created from each supplier node s_t to each vehicle node $v_{k,t}$. These arcs have a cost of 0 and a limit equal to the vehicle's capacity, Q.

Arcs for Vehicle Routing: An arc is created from each vehicle node $v_{k,t}$ to each client node $c_{n,t}$. The cost of these arcs is 0 if the client n is attended by vehicle k; otherwise, the cost is infinite. The limit of these arcs is equal to Q.

Arcs for Client Inventory: An arc is created from each client node $c_{n,t}$ to the corresponding client node in the next period, $c_{n,t+1}$. These arcs have a cost of h_n and a limit of $U_n - d_n^t$. This limit guarantees that the delivery occurs prior to the consumption as seen on Figure 2.1.

Arcs for Excess Product: An arc is created from each client node at the last period $c_{n,T}$ to the excess product node e. These arcs have a cost of 0 and an unlimited limit.

By modeling the IRP's inventory decision as an MCFP using these components, the NSIRP can efficiently find the optimal inventory for a given routing and scheduling solution, reducing the size of the search space and enabling better exploration of the solution space.

3.1.4
Fast Flow Network Simplex (FFNS)

The Coelho et al. (2012a) also tried the inventory decision decomposition as a MCFP, but they reported that approximately 65% of the total time was spent solving only this subproblem. The Diniz et al.'s (2020) work focused on how to speed up the resolution of the MCFP.

The NSIRP continues to employ the network simplex as the exact method for tackling the MCFP. Notably, it introduces a novel enhancement within its implementation that skillfully obviates the necessity of reinitiating the entire algorithm each time a new execution is carried out.. Termed the Fast Flow Network Simplex (FFNS), this enhancement has demonstrated an exceptional speedup, being found to be 10 times faster.

The network simplex method is a version of the simplex method tailored for Network Flow problems. It utilizes a spanning tree structure composed of

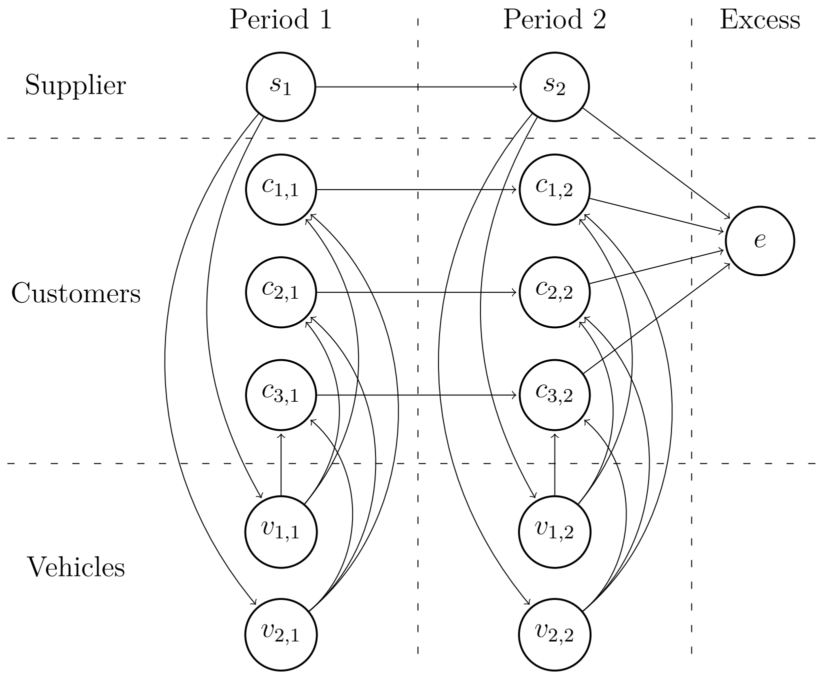

Figure 3.9: Proposed Network Flow Model for 3 Customers, 2 Vehicles, and 2 Periods

three sets of arcs: T, L, and U. The set T forms the basis of the solution and represents the spanning tree. Meanwhile, L and U are non-basic variables, with L containing arcs with zero flow and U containing arcs with flow equal to their capacity.

The algorithm iterates with the aim of finding non-basic arcs with negative reduced costs to enter the basis. To compute the reduced costs, dual variable values are needed for each node.

An improvement to this process is by proposing a procedure that iterates over all nodes, starting from the root and continuing node by node in the order used for tree construction (often referred to as the "thread order"). This enhancement streamlines the algorithm's execution, resulting in more efficient performance overall.

Usually, any change on the MCFP would require executing the entire algorithm from the beginning, the Diniz et al. (2020) paper modeled the MCFP in a specific way that the algorithm only needs to adjust the arcs related to the vehicle routing for subsequent solutions during the local search, rather than recreate the entire model from scratch.

3.1.5 Participation in 12th DIMACS Implementation Challenge

The group behind the NSIRP algorithm participated in the 12th DIMACS Implementation Challenge competition under the team name PUC-Rio and implemented four improvements to their algorithm, which are incorporated into this work.

In contrast to the paper by Diniz et al. (2020), the team used an **initial constructor** proposed in this work and described in Section 4.2. The following two modifications, **candidate arcs lookup** and **routing arcs update** aim to reduce the running time of the FFNS algorithm by limiting the search for entering arcs and the lookup of non-basic arcs, respectively. The fourth modification, **maximum solution degradation**, introduces a threshold for solutions evaluated during each local search iteration to avoid running the exact method on suboptimal solutions, thus reducing computational time. This last modification inspired an improvement where the threshold is automatically detected and included in this work, discussed in Section 4.5.3.

3.2 Hybrid Genetic Search (HGS)

The HGS framework, introduced by Vidal et al. (2012), addresses three variations of the VRP, namely the multi-depot, the periodic, and the multi-depot periodic VRPs. As a memetic algorithm, HGS combines the principles of genetic algorithms with neighborhood-based metaheuristics. The HGS framework has gained significant attention in the literature due to its high performance in terms of solution quality, convergence speed, and conceptual simplicity. A recent implementation of HGS, the Hybrid Genetic Search for the CVRP (HGS-CVRP) demonstrates its continued relevance and effectiveness as a leading metaheuristic (Vidal, 2022).

As highlighted by Vidal (2022), three key characteristics contribute to the effectiveness of the HGS framework:

- The integration of crossover-based and neighborhood-based methods facilitates a balanced approach to exploration and exploitation. The crossover operation introduces diversification within the solution space, while the neighborhood search aggressively refines the solutions.
- Building on the insights from Glover and Hao (2011), as well as Vidal et al. (2015), optimal solutions can often be found at the boundary between feasible and infeasible solutions. Consequently, the HGS framework enables controlled exploration of infeasible solutions.
- Contrary to many genetic algorithms, the selection of parents and survivors in HGS is not solely based on solution quality. Solution diversity also plays a significant role in the selection process. This approach allows the genetic algorithm to maintain the best and most diverse solutions,

while the neighborhood search exploits the solutions in pursuit of optimal results.

Algorithm 1 presents an overview of the HGS algorithm. The algorithm begins by defining the initial penalty cost, which will be explained in greater detail in Section 3.2.5. Next, the initial population is constructed, with a further explanation provided in Section 3.2.1. The algorithm runs while there is no improvement on the best individual from the population, during the last $nbIter$ iterations. In each iteration, two parents are selected from the population, as described in Section 3.2.3. Subsequently, a crossover operator is applied, generating a new offspring individual.

This new individual undergoes an education phase, comprising local search operators that enhance its quality while using the penalty to enable the exploration of infeasible solutions. If the educated offspring remains infeasible, there is a 50% chance that it will be repaired by re-running the local search with a penalty 10 times larger. This process will be further discussed in Section 3.2.4. The offspring are then added to the population, and once the maximum size is reached, some individuals are removed, as detailed in Section 3.2.2. Finally, the algorithm adjusts the penalty value to maintain a target percentage of feasible solutions within the population, which will be further clarified in Section 3.2.5.

Algorithm 1: HGS Outline

$penalty \leftarrow maxDistance/maxConsumption$;

// **Population Initialization**
$population.Initialize(penalty)$;

while *population has no improvement on last nbIter* **do**
    // **Parent Selection**
    $parent1, parent2 \leftarrow SelectParents(population)$;

    // **Genetic Operators**
    $offspring \leftarrow Crossover(parent1, parent2)$;

    // **Education**
    $LocalSearch(offspring, penalty)$;
    if $offspring.IsInfeasible()$ *and with 50% probability* **then**
        $LocalSearch(offspring, 10 * penalty)$;

    // **Population Management**
    $population.Insert(offspring)$;

    // **Infeasible Solution Management**
    $penalty \leftarrow ManagePenalties(population)$;
end

$best \leftarrow population.GetBestIndividual()$;

3.2.1 Population initialization

Algorithm 2 describes the process of initializing the population for the HGS algorithm. The algorithm continues constructing the initial population until its size reaches $4\times\ \mu$. During this process, the algorithm first generates random individuals, which are subsequently refined using local search and repair operators. Each refined individual is then added to the population. The following sections will provide a more detailed discussion of these components.

Algorithm 2: HGS Population Initialization Outline

```
while population.Size() ≤ 4 ∗ μ do
    individual ← RandomIndividual();

    // Education
    LocalSearch(offspring, penalty);
    if offspring.IsInfeasible() and with 50% probability then
        LocalSearch(offspring, 10 ∗ penalty);

    // Population Management
    population.Insert(offspring);
end
```

3.2.2 Population management

In the HGS algorithm, the population is divided into two distinct subpopulations: one for feasible solutions and another for infeasible solutions. Algorithm 3 illustrates how offspring are added to the appropriate subpopulation based on their feasibility. Following this, the biased fitness of each offspring is calculated, which will be further discussed in Section 3.2.2.1.

When a subpopulation's size reaches the maximum limit of $\mu + \lambda$, the algorithm proceeds to remove the λ worst individuals, as determined by their biased fitness values. This survivor selection process will be elaborated upon in Section 3.2.2.2.

Algorithm 3: HGS Population Management Outline

```
if offspring.IsFeasible then
    subPopulation ← population.GetFeasible();
else
    subPopulation ← population.GetInfeasible();

subPopulation.Insert(offspring);

// Biased Fitness Calculation
offspring.CalculateBiasedFitness(subPopulation);

// Survivor Selection
if subPopulation.Size() > μ + λ then
    while subPopulation.Size() > μ do
        subPopulation.RemoveWorstBiasedFitness();
    end
```

3.2.2.1 Biased Fitness Calculation

A distinguishing feature of the HGS from other genetic algorithms is its fitness calculation criterion. Each individual is ranked based on the entire sub-population. The rank is calculated using the following formula:

$$f_{\mathcal{P}}(s) = f_{\mathcal{P}}^{\phi}(s) + \left(1 - \frac{n^{\text{ELITE}}}{|\mathcal{P}|}\right) f_{\mathcal{P}}^{\text{DIV}}(s)$$

The first term, $f_{\mathcal{P}}^{\phi}(s)$, represents the rank of individual s within the sub-population $\mathcal{P}$ with respect to solution quality. The second term, $f_{\mathcal{P}}^{\text{DIV}}(s)$, denotes the rank of the individual s within the sub-population $\mathcal{P}$ in terms of diversification quality. To calculate diversification, the HGS computes the broken-pair distances for the n^{CLOSE} most similar solutions in the sub-population $\mathcal{P}$. Lastly, a smaller weight is applied to the second term to ensure that the top n^{ELITE} best individuals are preserved throughout the search process.

3.2.2.2 Survivor Selection

Maintaining diversity and avoiding premature convergence are significant challenges in population-based algorithms, particularly when education intensifies the parent selection tends to favor individuals with good characteristics. This reduction in genetic material diversity within the population can hinder the algorithm's exploration capabilities. To address this challenge, the HGS framework employs a two-component mechanism for survivor selection, which aims to preserve the most promising solution characteristics and maintain diversity in both subpopulations.

The first component of this mechanism consists of the biased-fitness function definition and the explicit consideration of diversity during parent selection (as discussed in Section 3.2.3). The second component, known as the Survivor Selection procedure, is activated when one of the two subpopulations reaches the maximum size $\mu + \lambda$. This procedure identifies the μ individuals that will proceed to the next generation, ensuring that the population diversity, in terms of visit patterns, is preserved, and elite individuals in terms of cost are protected. The λ discarded individuals are either clones or considered unfavorable with respect to cost and diversity contribution based on their biased fitness values.

3.2.3 Parent selection

The parent selection step in the HGS algorithm involves two rounds of a binary tournament. In each tournament round, two individuals are randomly selected from the entire population. Their fitness values, $f_{\mathcal{P}}(s)$, are then compared, and the individual with the better fitness wins the round. This process is conducted twice in order to select both parents for the crossover operation.

The Figure 3.10 illustrates this binary tournament process. In the first tournament round, individual c from the feasible sub-population and individual k from the infeasible sub-population is selected. Since individual c has a better fitness value, it is chosen as the first parent. Next, a second tournament round is conducted, in which individual i and individual m, both from the infeasible sub-population, are selected. In this round, individual m prevails due to its superior fitness. Consequently, the two parents selected for the crossover operation are individual c and individual m.

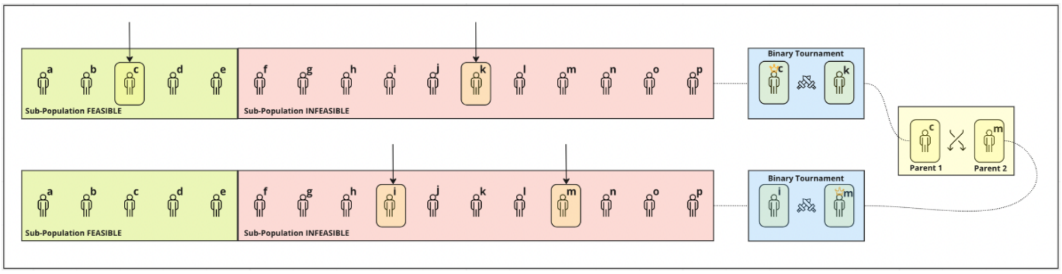

Figure 3.10: This figure presents an example of Parent Selection using binary tournaments. For the first tournament, two individuals, labeled as c and k, are randomly chosen. Through the course of the tournament, the first parent is identified. The same procedure is repeated for the second parent, with individuals i and m participating in the second binary tournament.

3.2.4 Education (local search)

The education step is controlled by local search operators. The original HGS paper Vidal et al. (2012) includes two stages: route improvement (RI) and pattern improvement (PI). In the updated HGS-CVRP Vidal (2022), the PI phase is excluded, and an additional neighborhood called Swap* is included in the RI phase.

An efficient local search is applied to each solution resulting from the crossover and Split algorithms. The RI local search uses Swap and Relocate moves, generalized to sequences of two consecutive nodes, as well as 2-Opt and 2-Opt*. The neighborhoods are limited to moves involving geographically close node pairs (i, j) such that j belongs to the Γ closest clients from i. The granularity parameter Γ therefore limits the neighborhoods' size to $O(\Gamma n)$. The exploration of the moves is organized in random order of the indices i and j, and any improving move is immediately applied. This process is pursued until attaining a local minimum.

The classical Swap neighborhood exchanges two customers in place, meaning one replaces the other and vice-versa. This neighborhood is typically used for intra-route and inter-route improvements. However, the proposed Swap* neighborhood differs as it involves exchanging two customers v and v' from different routes r and r' without an in-place insertion. In this process, v can be inserted in any position of r', and v' can likewise be inserted in any position of r.

Evaluating all the Swap* moves would take computational time proportional to $\Theta(n^3)$ with direct implementation. However, more efficient search strategies exist, which can significantly reduce the computational complexity and improve the overall performance of the algorithm.

Due to the controlled exploration of infeasible solutions, it is possible for a solution to remain infeasible after the local search. When this happens, a Repair operation is applied with 50% probability. This operation consists of running the local search with (10×) higher penalty coefficients, aiming to recover a feasible solution.

3.2.5
Infeasible solution management

A notable advantage of the HGS framework is its ability to explore infeasible solutions, which is achieved by defining a penalty cost for each excess product and incorporating this additional cost into the overall solution cost. The initial penalty cost is determined as the ratio between the maximum distance between two clients or between a client and the suppliers, and the maximum consumption. This penalty cost is updated to maintain the percentage of feasible solutions in the entire population equal to ξ.

If the percentage of feasible solutions is smaller than ξ, indicating a higher proportion of infeasible solutions than expected, the algorithm increases the penalty cost by 20% without exceeding a maximum value of 100,000. Conversely, if the percentage of feasible solutions is larger than ξ, implying a higher number of feasible solutions than anticipated, the algorithm decreases the penalty cost by 15% until it reaches a minimum value of 0.1.

By dynamically adjusting the penalty cost, the HGS framework encourages a controlled exploration of infeasible solutions, which may lead to the discovery of optimal solutions at the boundary between feasible and infeasible regions. This approach contributes to the overall effectiveness of the HGS algorithm in solving VRP variants.

3.3 Synergy

Integrating the NSIRP algorithm with the HGS framework presents a promising approach to enhance performance in addressing the IRP. The NSIRP algorithm effectively narrows the search space and facilitates superior exploration of the solution space by representing the inventory decisions from the IRP as an MCFP. This methodology enables the efficient computation of optimal inventory levels for a given routing and scheduling solution. Meanwhile, the HGS framework, an advanced memetic algorithm, has exhibited remarkable performance in solving various VRP variations, including capacitated, multi-depot, periodic, and multi-depot periodic VRPs. The HGS's forces reside in its capacity to balance exploration and exploitation through crossover-based and neighborhood-based techniques.

The NSIRP and HGS algorithms can be synergistically combined by incorporating the NSIRP's efficient local search within the HGS local search. The NSIRP local search will rapidly ascertain the optimal inventory levels for vehicle routing and scheduling solutions generated by the HGS algorithm. As a result, the HGS algorithm can concentrate on optimizing vehicle routing and scheduling while capitalizing on the NSIRP's capability to swiftly compute optimal inventory levels. This amalgamation has the potential to achieve superior solutions in a reduced time frame, as it leverages the strengths of both algorithms.

4
Proposed methodology

In this chapter, we present the $HGSIRP$ methodology for addressing the IRP by combining the strengths of the Hybrid Genetic Search (HGS) framework and the local search operators from the Network Simplex IRP (NSIRP) algorithm. To adapt the HGS for the IRP, we propose a new solution representation, discussed in Section 4.1. The complete algorithm is outlined in Algorithm 4, which highlights both differences and similarities with the original HGS.

Algorithm 4: $HGSIRP$ Outline

```
penalty ← maxDistance/maxConsumption;

// Population Initialization – (HGS) w/ Constructive
population.Initialize(penalty);

while population has no improvement on last nbIter do
    // Parent Selection – (HGS)
    parent1, parent2 ← SelectParents(population);

    // Genetic Operators
    offspring ← Crossover(parent1, parent2);

    // Education
    LocalSearch(offspring, penalty);
    if offspring.IsInfeasible() and with 50% probability then
        LocalSearch(offspring, penalty);

    // Population Management – (HGS)
    population.Insert(offspring);

    // Infeasible Solution Management – (HGS)
    penalty ← ManagePenalties(population);
end
best ← population.GetBestIndividual();
```

The **Population Initialization** is similar to the HGS, but utilizes a new constructive heuristic described in Section 4.2. **Parent Selection** employs the previously discussed binary tournament method. The **Genetic Op-**

erators introduces a novel crossover method, which is detailed in Section 4.3. The **Education** phase retains its overall structure, including the repair operation, but now incorporates IRP neighborhoods from the NSIRP algorithm, further elaborated in Section 4.4. Lastly, the **Population Management** and **Infeasible Solution Management** steps remain consistent with the original HGS approach, as covered in the previous chapter. Three new optimization proposals are made in this work and will be discussed in Section 4.5.

4.1 Solution representation

The solution representation used in $HGSIRP$ consists of four chromosomes:

– $chrom\boldsymbol{P}$(or shortly π): This chromosome represents the visit schedule for each client. Each gene π_n^t in π is a binary digit that indicates whether a specific client n is visited or not during a specific period t. If the gene value is 1, the client is visited during the period, while a value of 0 means that the client is not visited.

– $chrom\boldsymbol{R}$(or shortly κ): This chromosome represents the route for each vehicle. Each gene κ_k^t in κ encodes the route taken by vehicle k during a specific period t. The gene value is an integer that represents the index of the client in the solution. The order of clients visited by the vehicle is encoded by the order of genes in the chromosome.

– $chrom\boldsymbol{T}$(or shortly τ): This chromosome represents a giant tour that connects all routes from all vehicles in $\mathcal{K}$ for a specific period t. The gene τ^t concatenates the routes of all vehicles for a specific period. Each gene value in τ is an integer that represents the index of the client in the solution. The order of genes in the chromosome represents the order in which the clients are visited in the giant tour.

– $chrom\boldsymbol{D}$(or shortly δ): This chromosome represents the delivery for each client. Each gene δ_n^t in δ represents the amount delivered to a specific client n during a specific period t. The gene value is a real number that represents the quantity delivered to the client.

It's worth noting that while the $chrom\boldsymbol{P}$ and $chrom\boldsymbol{T}$ chromosomes are not absolutely essential for representing an IRP solution, they can be obtained from $chrom\boldsymbol{R}$. However, this redundancy is required because genetic operators are applied based on the chromosome format. For instance, a mutation operator that switches a 0 to 1 can only be used on a binary chromosome, such as $chrom\boldsymbol{P}$.

An example solution is shown in Figure 4.1. The yellow box shows a solution for an instance with five clients, three periods, and two vehicles. Each route starts and ends at the supplier *0*, and the amount delivered to each client q_n^t is shown in parentheses. The next three boxes illustrate the scheduling and routing for each period, and the four chromosomes are detailed with their specific genes.

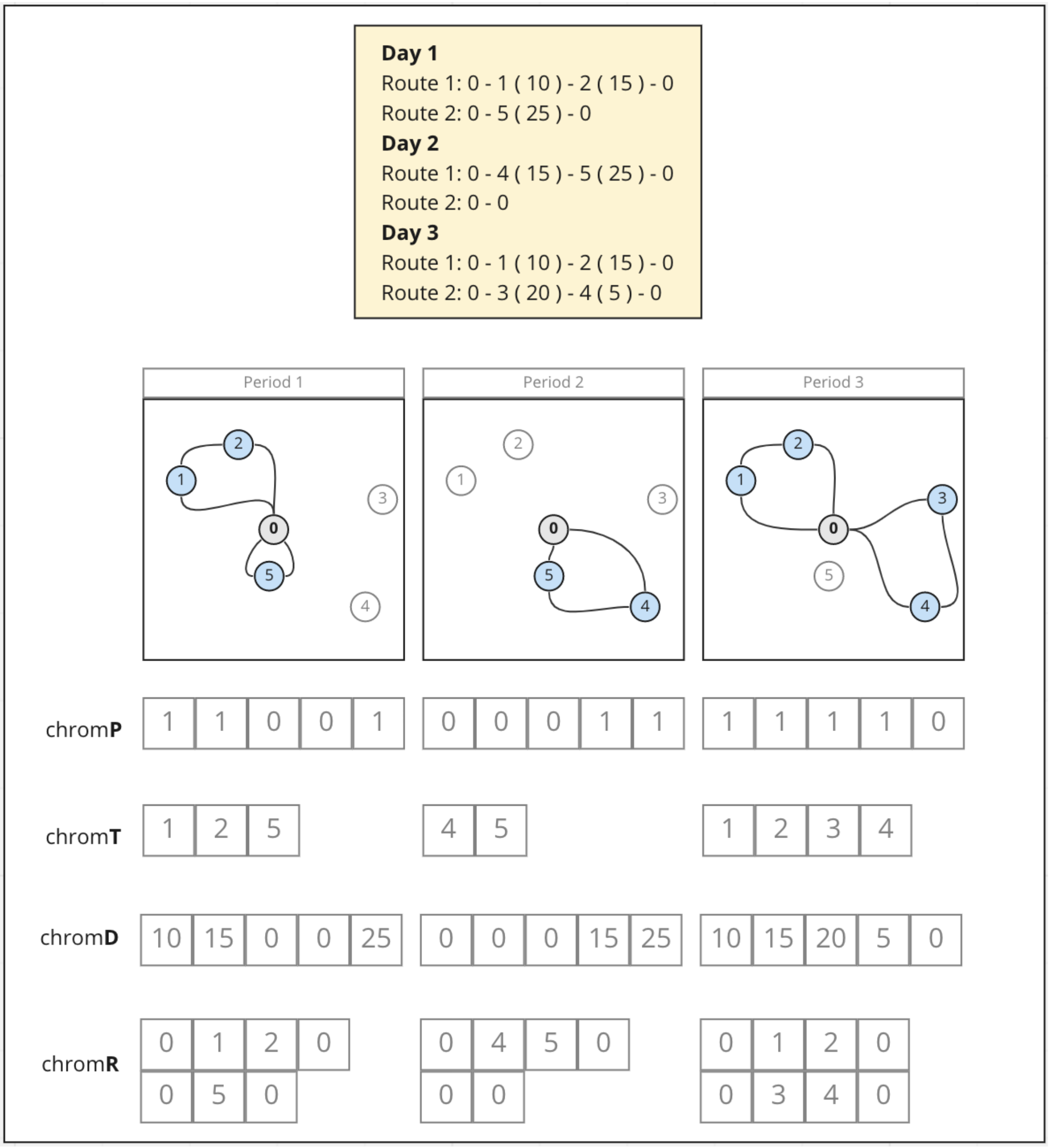

Figure 4.1: This figure provides a detailed representation of a solution structure. The uppermost yellow box displays an overall solution. Below it, three separate boxes provide a more visually intuitive representation of the same solution, showcasing the three periods alongside their associated scheduling and routing details. The final section, comprising four listed boxes, demonstrates the quartet of chromosomes that encapsulate the same solution. This multi-perspective view offers a comprehensive understanding of the solution's composition and functionality.

4.2 Constructive Heuristic

To generate a new individual, we employed a constructive heuristic based on the approach presented by Alvarez et al. (2018). This heuristic, known as the Two-Phase method, is delineated in Algorithm 5.

The overarching heuristic entails the simulation of multiple scenarios, relying on two key variables: the *look_ahead* parameter, which determines the number of periods into the future for identifying clients facing stockouts, and the *ratio_consumption* parameter, which signifies the proportion of a client's consumption utilized in the calculation of deliveries.

Using these parameters, the initial phase establishes two distinct client sets: set $C1$, which includes clients that will run out of stock if not serviced during the current period, and set $C2$, which includes clients that are good candidates for service. In addition, the procedures also determine the amount consumed by each client represent as $D1$ and $D2$, respectively for each set.

Subsequently, the second phase employs the insights gleaned from the first phase to construct delivery routes. In contrast to the approach detailed in Alvarez et al.'s (2018) work, we adopted a more intricate constructive heuristic for the Vehicle Routing Problem (VRP), as proposed by Subramanian (2012). This heuristic, outlined in Algorithm 6, incorporates a slight adaptation to accommodate non-obligatory clients. The algorithm employs an insertion technique that randomly selects between two insertion strategies and two insertion criteria, considering the number of vehicles and their capacities. First, the heuristic creates routes by inserting the obligatory clients from set $C1$. Then, it tries to insert as many non-obligatory clients from set $C2$ into the routes as possible.

This iterative process is carried out for each period, culminating in the creation of a fresh solution. Additionally, we applied the Fast Flow Network Simplex (FFNS) algorithm to optimize inventory-related expenses.

The constructive algorithm executes this simulation across various iterations, varying the *look_ahead* and *ratio_consumption* parameters. Ultimately, the most favorable solution is selected from among these iterations.

Algorithm 5: Outline of Alvarez et al.'s (2018) Two-Phase constructive heuristic – with adaptations highlighted.

```
best ← ∅;
for ratio_consumption ← 100% to 10% do
    for look_ahead ← 1 to T/2 do
        Initialize s as an empty solution;
        for t ← 1 to T do
            // Phase 1
            c1, d1 ← obligatoryClients(t);
            c2, d2 ←
             candidateClients(t, ratio_consumption, look_ahead);
            // Phase 2
            r ← createCapacitedRoutes(c1, d1, c2, d2);
            // Build Routing – chromR (or shortly κ)
            s.κ^t ← r;
        end
        // Build Inventory – chromD (or shortly δ)
        s.δ ← FFNS(s.κ);
        // Build Scheduling – chromP (or shortly π)
        s.π ← derived from s.κ;
        // Build Giant Tour – chromT (or shortly τ)
        s.τ ← derived from s.κ;
        Update best with s if it is better;
    end
end
return best;
```

Algorithm 6: Outline of Subramanian's (2012) constructive heuristic for the VRP – with adaptations highlighted.

$c1 \leftarrow$ list of obligatory clients;
$d1 \leftarrow$ deliveries of obligatory clients;
$c2 \leftarrow$ list of candidates clients;
$d2 \leftarrow$ deliveries of candidates clients;

// **Insert obligatory clients firsts**
do
  $insertCriterion \leftarrow$ MCFIC or NFIC (choose at random);
  $insertStrategy \leftarrow$ SIS or PIS (choose at random);
  $s \leftarrow constructSolution(c1, d1, insertCritera, insertStrategy)$;
while *(s is not feasible) and (attempt <= MAX_ATTEMPTS)*;

// **Update solution with candidates clients**
$s \leftarrow updateSolution(s, c2, d2, insertCriterion, insertStrategy)$;

Overall, the Two-Phase heuristic allows for the efficient generation of new individuals that take into account the inventory costs and client consumption constraints.

4.3 Genetic operator (crossover)

The crossover operation serves as a vital genetic operator within the realm of Genetic Algorithm (GA). This operation is employed to forge a fresh solution by utilizing two other solutions, commonly referred to as offspring and parents. Its primary objective is to bolster diversity within the population. This is achieved by crafting a novel individual, or solution, through the amalgamation of two existing individuals, or parents. Importantly, this fusion is carried out in a manner that upholds the distinctive characteristics of the parent solutions. This approach prevents the generation of notably subpar solutions, while simultaneously introducing innovative combinations that open up new avenues of possibility.

As discussed on 2.1, the IRP encompasses three main decisions, the scheduling, the routing, and the inventory. This work modeled these three decisions into two main chromosomes, the $chrom\boldsymbol{D}$ for inventory and $chrom\boldsymbol{R}$ for routing. But as said on 4.1, the $chrom\boldsymbol{P}$, encoding scheduling, and the $chrom\boldsymbol{T}$, encoding the giant tour, plays an important role during the crossover operation. As we will discuss in the following.

The fundamental concept of the crossover operation applied to the IRP is broken down into four key steps, illustrated in Figure 4.2:

1. **Mixing parental scheduling decisions:** The scheduling chromosome $chrom\boldsymbol{P}$ is formed by combining $chrom\boldsymbol{P}$ from both parental solutions s_1 and s_2.

2. **Preserving parental visit order:** The giant tour chromosome $chrom\boldsymbol{T}$ is constructed using the created $chrom\boldsymbol{P}$ and elements from both parental solutions.

3. **Formulating an effective:** The deliveries chromosome $chrom\boldsymbol{D}$ is generated using a variant of the modeling used for the Minimum Cost Flow Problem (MCFP) for a single vehicle, with $chrom\boldsymbol{T}$ as input. This step yields the optimal inventory, given the giant tour.

4. **Formulating an effective routing:** The routing chromosome $chrom\boldsymbol{R}$ is obtained by applying the Split algorithm (Vidal, 2016) to $chrom\boldsymbol{T}$ and $chrom\boldsymbol{D}$.

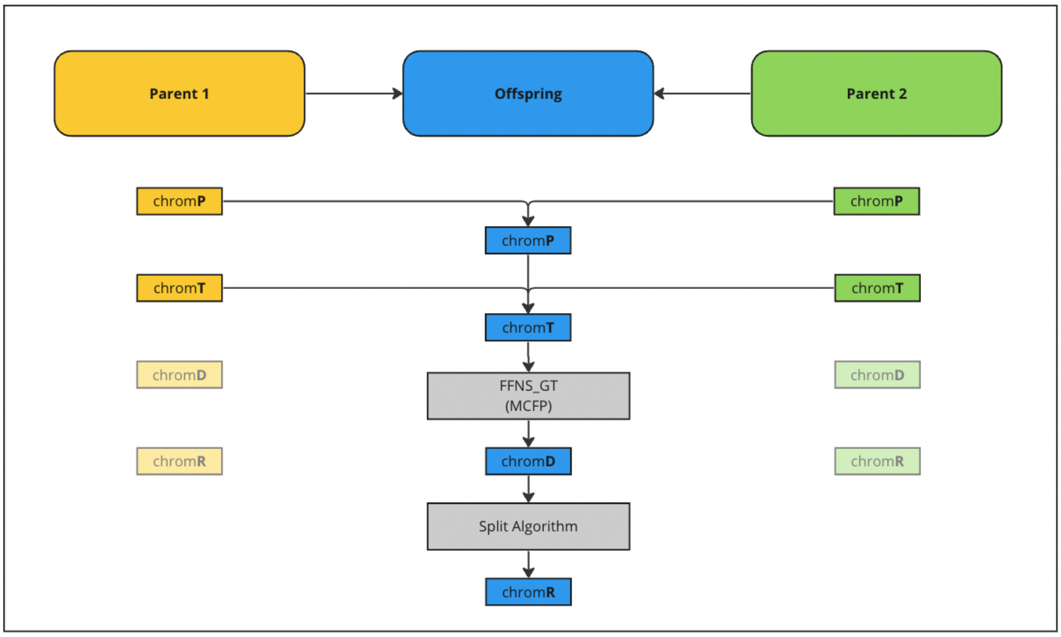

Figure 4.2: This figure illustrates the crossover process of the PTNU algorithm. Here, 'Parent 1' and 'Parent 2', depicted in yellow and green respectively, are utilized to generate a new 'Offspring', illustrated in blue. Both $chrom\boldsymbol{P}$ and $chrom\boldsymbol{T}$ chromosomes are derived from these parental inputs. Then, the FFNS algorithm is applied to $chrom\boldsymbol{T}$, resulting in the generation of the $chrom\boldsymbol{D}$ chromosome. Following this, the Split algorithm is applied to $chrom\boldsymbol{D}$, thereby creating the $chrom\boldsymbol{R}$ chromosome.

Algorithm 7: Detailed Outline of PTNU Crossover

Initialize s_o as an empty individual;

// **1. Build Scheduling – *chromP* (or shortly π)**

$cut \leftarrow$ pick a random integer in $(0, N)$;

$\mathcal{N}_1 \leftarrow$ select cut random clients from $\mathcal{N}$;

$\mathcal{N}_2 \leftarrow \mathcal{N} \setminus \mathcal{N}_1$;

for $t \in \mathcal{T}$, $n \in \mathcal{N}$ **do**

  $s_o.\pi_n^t \leftarrow s_1.\pi_n^t$ if $n \in \mathcal{N}_1$, otherwise $s_2.\pi_n^t$;

end

// **2. Build Giant Tour – *chromT* (or shortly τ)**

for $t \in \mathcal{T}$ **do**

  while $n_1 \in s_1.\tau^t$ *or* $n_2 \in s_2.\tau^t$ **do**

    if $n_1 \in \mathcal{N}_1$ *and* $n_2 \in \mathcal{N}_2$ **then**

      $s_o.\tau^t$ append $\langle n_1, n_2 \rangle$ in random order;

    else if $n_1 \in \mathcal{N}_1$ **then**

      $s_o.\tau^t$ append $\langle n_1 \rangle$;

    else if $n_2 \in \mathcal{N}_2$ **then**

      $s_o.\tau^t$ append $\langle n_2 \rangle$;

    end

  end

end

// **3. Build Inventory – *chromD* (or shortly δ)**

$s_o.\delta \leftarrow FFNS_{GT}(s_o.\tau)$;

// **4. Build Routing – *chromR* (or shortly κ)**

$s_o.\kappa \leftarrow Split(s_o.\tau, s_o.\delta)$;

Algorithm 7 presents a detailed explanation of each step. The algorithm starts by initializing an empty offspring solution s_o and the four chromosomes *chrom**P***, *chrom**T***, *chrom**D*** and *chrom**R***.

The construction of the scheduling chromosome *chrom**P*** (or shortly π) begins with the selection of a random integer *cut* in the range $(0, N)$. This value determines the portion of $s_1.\pi$ that will be copied to $s_o.\pi$. Then, the client set $\mathcal{N}$ is divided into two sets: set $\mathcal{N}_1$, consisting of *cut* random clients, and set $\mathcal{N}_2$, which is the complementary set. Finally, $s_o.\pi$ is constructed by iterating over all periods $t \in \mathcal{T}$ and all clients $n \in \mathcal{N}$. If $n \in \mathcal{N}_1$, $s_1.\pi_n^t$ is copied to $s_o.\pi_n^t$, otherwise, $s_2.\pi_n^t$ is copied.

The construction of the giant tour chromosome, *chrom**T*** (or shortly τ) involves iterating over both $s_1.\tau$ and $s_2.\tau$ simultaneously. The goal is to create an $s_o.\tau$ that follows the scheduling $s_o.\pi$ and preserves the parental visit order. This is achieved by checking if the current clients from both parents, n_1 and n_2, were both selected for their respective sets $\mathcal{N}_1$ and $\mathcal{N}_2$. If this is the case, n_1 and n_2 are randomly selected and added to $s_o.\tau$. If both n_1 and n_2 are in $\mathcal{N}_1$, n_1 is added to $s_o.\tau$. If both are in $\mathcal{N}_2$, n_2 is added. If the clients are exchanged, meaning one is in $\mathcal{N}_1$ and the other is $\mathcal{N}_2$, they are ignored and evaluated at a later time. This process is repeated for each t in $\mathcal{T}$. It is important to highlight that the sets $\mathcal{N}_1$ and $\mathcal{N}_2$ are disjoint sets, so since we append to the giant tour only in cases when the client n is found on one of these sets, we cannot append same customer several times.

The delivery chromosome *chrom**D*** (or shortly δ) is constructed by solving a MCFP based on the giant tour stored in $s_o.\tau$ using an adapted FFNS. In this context, the giant tour represents the route assigned to a single vehicle, which has a capacity equal to $K * Q$. However, even though the vehicle capacity is quite large, the maximum amount that can be delivered to a client is constrained to Q. This ensures that the feasibility of the original problem is maintained, where a single vehicle is limited to Q.

Finally, the routing chromosome *chrom**R*** (or shortly κ) is constructed using the Split algorithm Vidal (2016). The algorithm takes as input the giant tour chromosome $s_o.\tau$ and the delivery chromosome $s_o.\delta$, and performs a single execution for every period $t \in \mathcal{T}$. The output of the algorithm is the routing chromosome $s_o.\kappa$ that represents the detailed route of the vehicles, including their delivery and pick-up operations, for each period.

The Periodic-Tour-Non-Uniform (PTNU) crossover method used in this work allows the genetic algorithm to generate offspring solutions that maintain the desirable features of the parent solutions, such as the scheduling and the routing. The method also ensures good quality solutions, since the FFNS method and the Split algorithm will respect the vehicle capacity using the penalty and the inventory capacity constraint. The Figure 4.3 provides an example of how the two parents are used to generate the offspring solution.

4.4 Education (local search)

This section delves into the education phase, outlined in Algorithm 8. The education phase comprises two distinct steps: the execution of the CVRP local search, followed by the IRP local search.

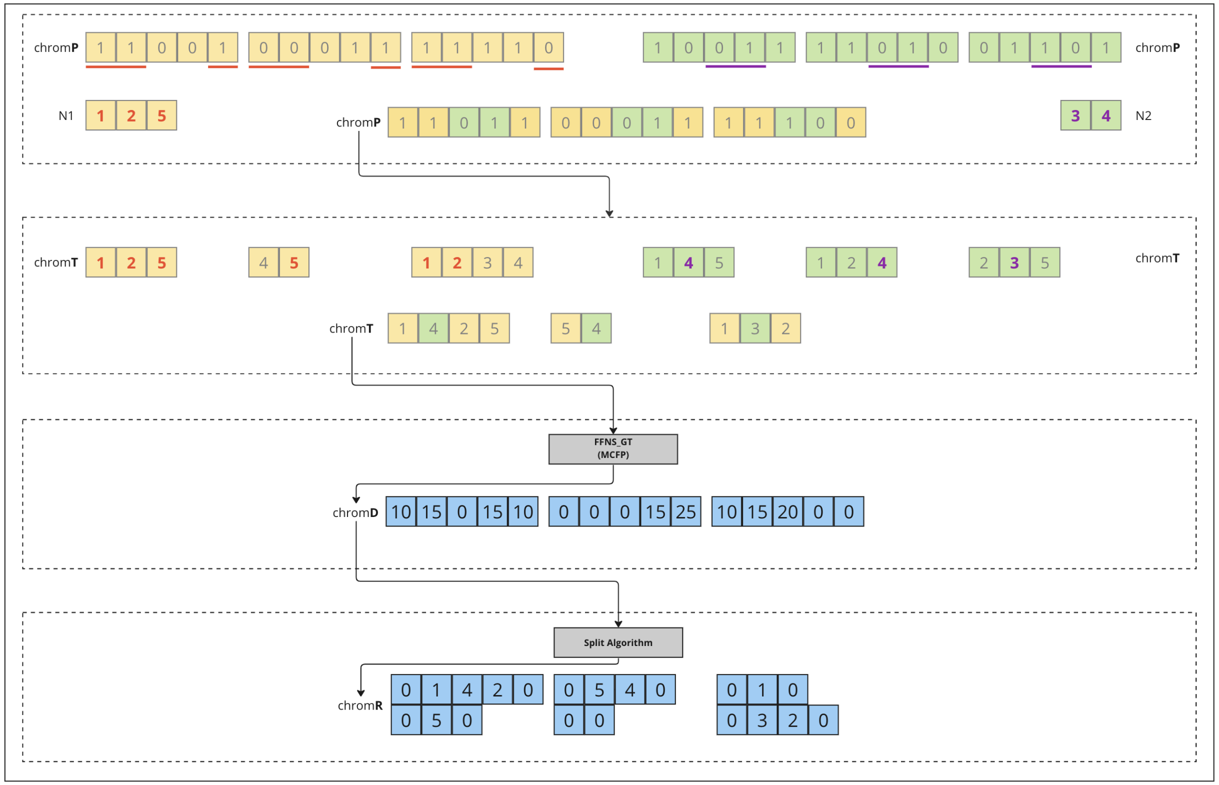

Figure 4.3: This figure illustrates an example of the PTNU crossover operation. The first box portrays the construction of the *chrom**P***chromosome, where the yellow and green boxes represent genes inherited from 'Parent 1' and 'Parent 2', respectively. The associated sets $N1$ and $N2$ are indicated on the side. The process then proceeds to the formation of the *chrom**T***chromosome, demonstrated in the second step. Following this, the *chrom**T***chromosome serves as an input to the FFNS algorithm, which generates the *chrom**D***chromosome. Finally, the *chrom**R***chromosome is produced by applying the Split algorithm to *chrom**D***.

Algorithm 8: $HGSIRP$ Education Outline

```
// Local Search – (HGS) per period
for t ∈ 𝒯 do
    LocalSearchCVRP(offspring.ForPeriod(t), penalty);
end

// Local Search – (NSIRP) w/ penalty
LocalSearchIRP(offspring, penalty);
```

Both steps draw from the implementations in the HGS and NSIRP algorithms, respectively, with a modification made in the former. To accommodate the incorporation of the Capacitated Vehicle Routing Problem (CVRP) local search from the HGS algorithm, we begin by recognizing the IRP as a generalization of the VRP, encompassing its routing aspect. By locking the scheduling and delivered quantities in the offspring solution, we execute a CVRP local search for each period. This localized search focuses solely on refining the routing while keeping the inventory untouched. A visual representation of this process is illustrated in Figure 4.4.

4.5 Optimization Proposals (OP)

This section outlines optimization proposals made to the $HGSIRP$ algorithm to enhance its efficiency and effectiveness in finding better solutions. The first optimization (Section 4.5.1) involves adding a CVRP local search before the IRP local search, which provides a better routing solution and enables the IRP local search to focus on refining the scheduling and amount delivered to each client. The second optimization (Section 4.5.2) allows the search into infeasible vehicle capacity solutions by incorporating a new node representing an auxiliary vehicle, which represents the excess of the associated vehicle in the solution. Finally, the third optimization Section 4.5.3 incorporates a heuristic to calculate the maximum inventory cost degradation, avoiding expensive, unnecessary calculations. These modifications have proven valuable in improving the performance of the $HGSIRP$ algorithm.

4.5.1 CVRP local search (OP1)

The decision to incorporate the local search for the CVRP in this work was deliberate, even though its usage is not strictly obligatory. For that reason we consider it as a optimization proposal. Our investigation revealed that its implementation enhances the quality of the solutions, as demonstrated in

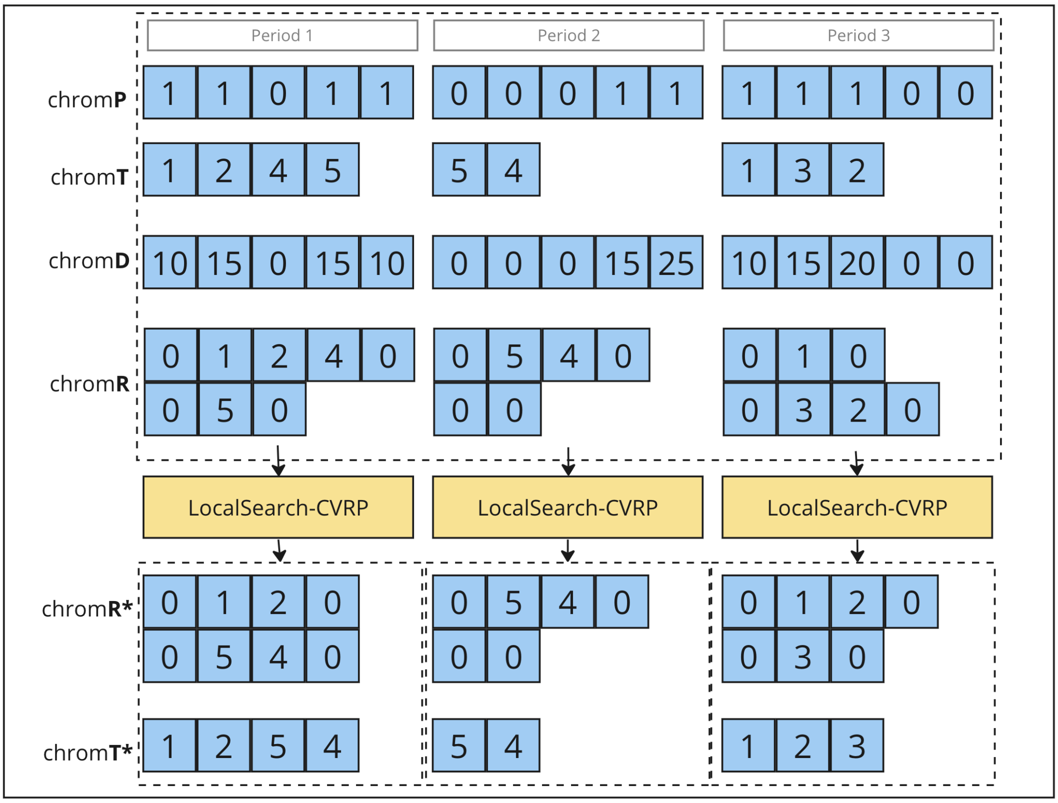

Figure 4.4: This figure illustrates the integration of the LocalSearch-CVRP method. By conceptualizing each period as a standalone solution for a CVRP, we can subject each of these solutions to the LocalSearch-CVRP process. Through this, only the routing, $chrom\boldsymbol{R}$, and the giant tour chromosomes, $chrom\boldsymbol{T}$, are enhanced, demonstrating the selective optimization of solution components.

the subsequent chapter. The primary rationale behind this choice lies in our observation that the local search methods employed in the context of the IRP tend to focus primarily on movements within individual periods, potentially lacking efficiency in terms of overall routing improvement. Consequently, the $HGSIRP$ algorithm integrates the CVRP local search as a precursor to the IRP local search for each period, thereby enhancing the routing aspect exclusively. This strategic sequencing results in a more streamlined and effective IRP local search process, leading to the discovery of improved solutions.

4.5.2 Infeasible vehicle capacity (OP2)

Incorporating the ability to explore infeasible solutions during the local search is a key characteristic of HGS. This capability enables the metaheuristic to navigate between structurally different feasible solutions. To leverage this characteristic and improve the performance of the $HGSIRP$ algorithm, it was essential to extending its capability to allow the search into infeasible vehicle capacity solutions.

To extend the capability of $HGSIRP$ to allow the exploration of infeasible solutions, we modified the IRP model presented in Diniz et al. (2020) by adding a new node representing an auxiliary vehicle. The quantity represented by this additional vehicle represents the excess of the associated vehicle in the solution.

Figure 4.5 shows an example of the methodology to handle infeasible capacities in the vehicle routing problem. The diagram presents the network flow with source nodes s_1 and s_2, clients $c_{i,j}$, and vehicles $v_{i,j}$. The auxiliary vehicles $vi_{i,j}$ are introduced to handle the infeasible capacities of vehicles $v_{i,j}$. The arcs between the source nodes and the vehicles have a cost of the current penalty value ω^Q and a capacity of Q, while the arcs between the source nodes and the auxiliary vehicles have a cost of zero and an infinite capacity. The arcs between the auxiliary vehicles and vehicles have zero cost and infinite capacity. The arc between the vehicles and the clients has now a capacity as $2 \times Q$ allowing the clients to receive the excess capacity.

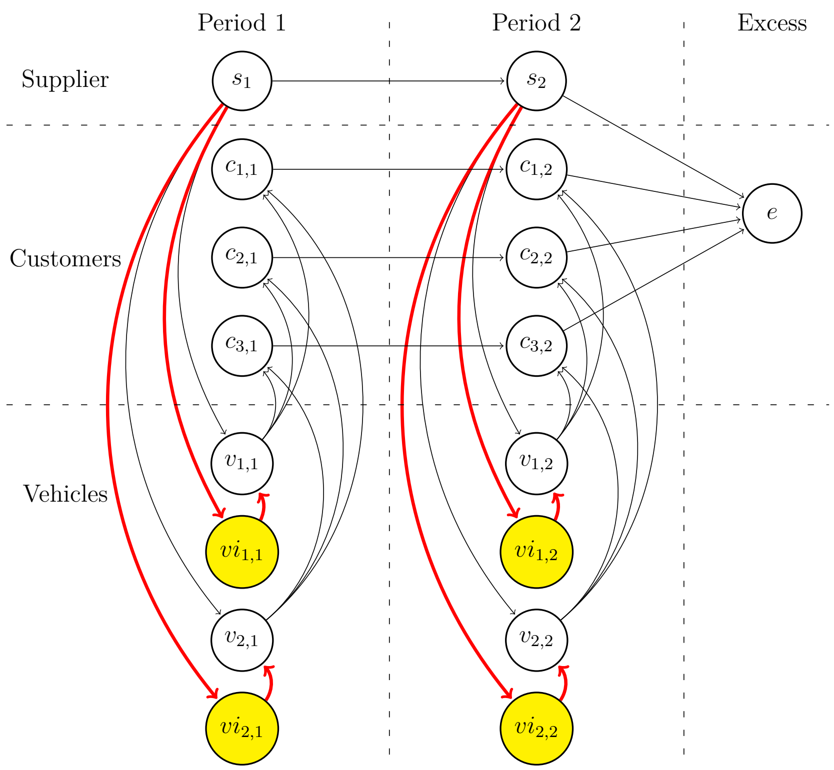

Figure 4.5: This figure portrays the proposed Network Flow Model for a system encompassing 3 customers, 2 vehicles, and 2 periods. Additional vehicle capacity is denoted by the yellow nodes, providing a visual representation of the system's enhanced transport potential.

4.5.3 Maximum degradation (OP3)

The motivation behind using the cheaper heuristic before running the FFNS in the IRP local search is to save computation time potentially. As seen in Section 3.1, the IRP local search decodes a solution routing κ into the optimal inventory solution δ by solving an MCFP using the FFNS algorithm. This exact algorithm has been found to perform well, according to the work of Diniz et al. (2020), however, running this algorithm for every local search can be time-consuming. Thus, if we can determine an upper bound on the inventory cost beforehand using the cheaper heuristic, we may avoid unnecessary calculations and improve the overall efficiency of the algorithm.

Algorithm 9 shows how to avoid running the exact method during the local search. A heuristic, *Heuristic Degradation*, is used to calculate the heuristic inventory cost improvement, denoted as Δ^{δ^H}. Suppose the sum of the routing cost improvement and the heuristic inventory cost improvement is greater than zero. In that case, the function returns a very large hypothetical cost, represented by ∞, indicating that this solution will probably have a degradation in its solution cost. Hence, it is not worth calculating the inventory cost improvement using FFNS.

Algorithm 9: Outline of Local Search Evaluation – with adaptations highlighted

// **Routing Cost Improvement**
$\kappa \leftarrow NeighborhoodOperator(s^i.\kappa, params)$;
$\Delta^\kappa \leftarrow RountingCost(\kappa) - RountingCost(s^i.\kappa)$;

// **Heuristic Inventory Cost Improvement**
$\Delta^{\delta^H} \leftarrow HeuristicDegradation(s^i.\kappa, params)$;
if $\Delta^\kappa + \Delta^{\delta^H} > 0$ **then**
    return ∞;
end

// **Inventory Cost Improvement**
$\delta \leftarrow FFNS(s^i.\kappa, params)$;
$\Delta^\delta \leftarrow InventoryCost(\delta) - InventoryCost(s^i.\delta)$;

return $\Delta^\kappa + \Delta^\delta$;

Algorithm 10 calculates the heuristic upper bound on the inventory cost for neighborhoods that affect a single client like *Insert*, *Remove*, and *Relocate*. It compares the inventory cost before and after using the FFNS method for a single client. The comparison is made by first calculating the inventory cost that only affects the related client and then calculating the optimal inventory cost for the client using the FFNS method. The maximum inventory cost improvement is the difference between the current value and its optimal. The algorithm is based on the network flow model represented in Figure 4.6. For neighborhoods that affect two clients, like the *Swap* heuristic, we sum the calculation for both clients.

Algorithm 10: HeuristicDegradation

$n \leftarrow params.n;$

$before \leftarrow InventoryCostForClient(s^i, n);$

$after \leftarrow MinCostFlowForClient(s^i.\kappa, params, n);$

return $after - before$

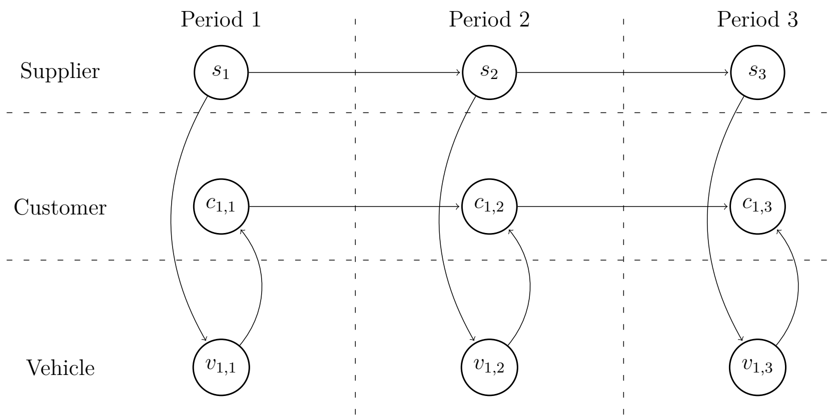

Figure 4.6: This figure presents the Network Flow model specifically tailored for the MinCostFlowForClient heuristic.

5
Computational Experiments and Analysis

To assess the proposed methodology, this study conducted three sets of experiments: the evaluation of Optimization Proposals (OP) discussed in Section 5.1, the comparison with existing literature in Section 5.2, and the comparison with the 12$^{\text{th}}$ DIMACS Implementation Challenge in Section 5.3. As a result of these experiments, 79 new Best Known Solution (BKS) were obtained, positioning this work at the top of the competition.

The code was developed using C++ and compiled with GCC, utilizing the -O3 optimization flag. It was written on top of the open source Hybrid Genetic Search for the CVRP (HGS-CVRP) *v.2.0.0* available at `https://github.com/vidalt/HGS-CVRP/releases/tag/v2.0.0`. All computational experiments were performed using a single thread on an Intel Xeon Platinum 8275L @ 3GHz with 192 GB RAM. A single thread *PassMark®* of 2,386 MOps/Sec. The algorithm runs for a number *nbIter* of iterations without improvements, or to a maximum CPU time limit T_{MAX} of 1509 seconds, which comes first. This follows the 12$^{\text{th}}$ DIMACS Implementation Challenge rules, which dictate a maximum execution time limit proportional to a *PassMark®* result of 2000, allowing up to 1800 seconds.

The parameters related to Hybrid Genetic Search (HGS) remained consistent with and are described in Table 5.1. The Network Simplex IRP (NSIRP) algorithm includes parameters only pertaining to the Iterated Local Search (ILS) framework and, as such, is not directly applicable to this study.

Table 5.1: *HGSIRP* parameters

Parameter		**Value**
nbIter	Number of iterations without improvements	20000
T_{MAX}	The CPU time limit until termination (in seconds)	1509
μ	Population size	25
λ	Generation Size	40
n^{ELITE}	Number of elite solutions considered in the fitness calculations	4
n^{CLOSE}	Number of close solutions considered in the diversity-contribution measure	5
Γ	Granular search parameter	20
ξ	Target proportion of feasible individuals for penalty adaption	0.2

5.1 Contribution of Optimization Proposals

This section presents a sensitivity analysis of each implementation improvement discussed in Section 4.5. All instances from the classical benchmark with an identifier equal to 1 were selected, resulting in 160 instances from the Small subset and 30 instances from the Large subset. For each experiment, ten rounds of independent executions were conducted, and the average result was used for comparison.

To evaluate the contribution of the Optimization Proposals, this study proposed five test scenarios, where each improvement was individually disabled. All these scenarios, which we refer to as $HGSIRP^S$, where $S \subseteq \{\overline{OP1}, \overline{OP2}, \overline{OP3}\}$ indicates the set of Optimization Proposals that were disabled in each case, compared to the algorithm with all features turned on, namely $HGSIRP$.

The **CVRP local search (OP1)** was disabled by simply ignoring the method call. While disabling the **Infeasible vehicle capacity (OP2)** was done by using the same model as proposed by Diniz et al. (2020). The **Maximum degradation (OP3)** was disabled by allowing the execution of the network simplex for every local search evaluation.

It is important to note that OP3 is necessary for solving large instances. Without it, none of the large instances can progress past the population initialization. For this reason, the experiments with the large instances were all conducted with this feature enabled.

Two metrics are employed to assess the impact of deactivating the Optimization Processes (OPs). The first metric, termed **Primal Gap**, represents the disparity in the primal result obtained by each implementation:

$$\frac{HGSIRP^S_{UB} - HGSIRP_{UB}}{HGSIRP_{UB}}$$

The second metric, denoted as **Time Gap**, measures the difference in the time taken for the algorithm's completion, factoring in the stopping criteria. The algorithm concludes either after $nbIter$ iterations without enhancements, or when reaching a maximum CPU time limit of T_{MAX}, whichever comes first:

$$\frac{HGSIRP^S_{Time} - HGSIRP_{Time}}{HGSIRP_{Time}}$$

The impact of disabling **CVRP local search (OP1)** showed a slight improvement of 0.01% on the Primal Gap for the Small Set, but a degradation of 0.74% for the Large set. The deterioration in the Time Gap was significant, with an increase of 48.52% for the Small Set and 32.24% for the Large Set. The benefit of running the CVRP local search is to provide the IRP local search with a better routing solution, allowing it to waste less time in its local search. However, it may also reduce the diversity of the solution. The results show that the quality of solutions for the Small set was minimally affected, while the time spent on the IRP local search for the Large set allowed more CPU execution, leading to better results.

The content of Section 5.1 illustrates five distinct scenarios that have been examined. The initial three scenarios elucidate the effects of individually deactivating each OP. The fourth scenario, on the other hand, delves into the consequences of deactivating all procedures simultaneously. This can only be accomplished with smaller instances, as the presence of the **Maximum degradation (OP3)** is indispensable for the larger dataset, as highlighted in the fifth scenario.

The impact of disabling **Infeasible vehicle capacity (OP2)** had a deterioration of 0.06% and 0.13% on the Primal Gap for the Small and Large sets, respectively. However, the Time Gap was positively impacted, with a 24.79% decrease for the Small set and a 6.26% decrease for the Large set. This decrease in time can be explained by the fact that the adapted Minimum Cost Flow Problem (MCFP) model proposed in Section 4.5.2 contains more nodes and arcs to allow extra space for each vehicle.

The impact of **Maximum degradation (OP3)** had a limited impact on the Primal Gap but worsened the Time Gap by 534.21%. This deterioration in time can avoid finding better solutions for instances that are limited by time, such as the Large set.

When all three OPs were removed, the deterioration in the Primal Gap was 0.07% and 0.92% for the Small and Large sets, respectively. The Time Gap also deteriorated by 434.60% for the Small set and 17.87% for the Large set.

In summary, removing the Optimization Proposals has negatively impacted the performance of the $HGSIRP$ algorithm in finding better solutions for the IRP. The results suggest that combining all three improvements is recommended to achieve the best performance.

5.2 Comparison with literature methods

In this section, we compare our proposed algorithm, $HGSIRP$, with existing literature on 1098 instances of the classical IRP benchmark. We performed ten independent runs, selecting the average. The complete list of results can be found on Appendix C.

Table 5.2: The Effect of Disabling each Optimization Proposal.
OP1: CVRP Local Search, OP2: Infeasible vehicle capacity, OP3: Maximum degradation

	Primal Gap		**Time Gap**	
	Small	**Large**	**Small**	**Large**
$HGSIRP^{\overline{OP1}}$	–0.01%	+0.74%	+48.52%	+32.24%
$HGSIRP^{\overline{OP2}}$	+0.06%	+0.13%	–24.27%	–6.26%
$HGSIRP^{\overline{OP3}}$	0.00%	NA	+534.21%	NA
$HGSIRP^{\overline{OP1},\overline{OP2},\overline{OP3}}$	+0.07%	NA	+434.60%	NA
$HGSIRP^{\overline{OP1},\overline{OP2}}$	NA	+0.92%	NA	+17.87%

5.2.1 Examination of data sources

We gathered the outcomes from 21 studies, drawing upon data provided by a subset of these works. For instance, in the case of Schenekemberg et al. (2023), we acquired the results directly from their website at `https://www.leandro-coelho.com/three-front-parallel`. In the cases of Archetti et al. (2021), Coelho and Laporte (2013), and Alvarez et al. (2018), we sourced the results from `https://or-brescia.unibs.it/instances#h.p_ID_48`. The remaining 17 studies' results were obtained from the Axiom Research Project, which is responsible for the publication of works such as Vadseth et al. (2021), Skålnes et al. (2022), Vadseth et al. (2023), and Skålnes et al. (2023b). These results are available at `http://axiomresearchproject.com/publications`.

Out of the 21 literature methods reviewed, not all reported results for all 1098 instances. Some of these methods were published before the introduction of newer instances, while others did not provide comprehensive results for every category. Adulyasak et al. (2014) concentrated on Small Multi-Vehicle instances, considering scenarios with 5 to 25 clients and either 2 or 3 vehicles. For instances involving 30 to 50 clients, they limited the vehicles to 3 or 4. Avella et al. (2018) also explored small Multi-Vehicle instances, but specifically for 15 to 35 clients over 6 periods, and for 50-client scenarios over 3 periods. Manousakis et al. (2021) provided results for all Small Multi-Vehicle instances, excluding the Large instances with 200 clients. Skålnes et al. (2022) detailed results for small scenarios with 15 to 30 clients over a span of 6 periods. Lastly, Achamrah et al. (2022) only focused on large instances, considering vehicle counts of 1, 2, or 3. Table 5.3 summarizes the information regarding these studies and the specific instances for which they reported results.

Another noteworthy aspect concerning these results is the variation in processors and the differing number of available threads employed by

Table 5.3: Summary of literature methods addressing IRP classical benchmark instances, categorized by single and multi-vehicle cases. The checkmark (✓) indicates that the paper released results for all instances within its groups, while the asterisk (∗) indicates that only parts of the group were released.

	Single-Vehicle		Multi-Vehicle	
	Small	Large	Small	Large
Archetti et al. (2012)	✓	✓		
Coelho and Laporte (2013)			✓	
Coelho and Laporte (2014)			✓	
Adulyasak et al. (2014)			∗	
Desaulniers et al. (2016)			✓	
Archetti et al. (2017)			✓	✓
Alvarez et al. (2018)			✓	✓
Avella et al. (2018)			∗	
Chitsaz et al. (2019)	✓	✓	✓	✓
Alvarez et al. (2020)	✓		✓	
Diniz et al. (2020)	✓		✓	
Guimarães et al. (2020)	✓	✓	✓	✓
Manousakis et al. (2021)		∗	✓	∗
Archetti et al. (2021)			✓	✓
Vadseth et al. (2021)	✓	✓	✓	✓
Skålnes et al. (2022)	∗		∗	
Solyalı and Süral (2022)			✓	✓
Achamrah et al. (2022)		✓		∗
Schenekemberg et al. (2023)	✓	✓	✓	✓
Vadseth et al. (2023)				✓
Skålnes et al. (2023b)			✓	✓

each study. In Table 5.4, we present a comprehensive list of the works along with their respective processors and thread counts. The *PassMark®* CPU Score was acquired from `https://www.cpubenchmark.net/`. In cases where obtaining the CPU score proved unattainable, Coelho and Laporte (2013) did not provide the CPU model, and both Archetti et al. (2012) and Achamrah et al. (2022) featured an elusive CPU model designation. For instances that utilize a single thread, we utilized the **Single Thread** score, while for algorithms employing multiple threads, we referenced the **Average CPU Mark** results. It's important to note that the Average CPU Mark for multi-threaded algorithms assumes the utilization of all available threads, yet some works, such as Archetti et al. (2017), Guimarães et al. (2020), and Schenekemberg et al. (2023), employed fewer threads. In such cases, we normalized the value to align with the correct number of threads.

For a fair comparison between methods, it's crucial that they utilize the same amount of CPU resources during experiments. However, can be challenging when considering variations in the CPU models. Following the guidelines of the DIMACS (2022) Challenge, we established a hypothetical machine with a *PassMark®* CPU Score of 2000 points as a baseline. We then calculated a Time Adjust Factor (TAF) by comparing the *PassMark®* CPU Score of each literature method to this hypothetical machine. This factor allows us to standardize the time taken by each method to solve instances.

Table 5.4: Hardware and Solver Configurations for Literature Methods

Refference	**CPU**	**Thread**	***PassMark®***	**TAF**
Archetti et al. (2012)	Intel Core2 Duo E6300 @ 1.86GHz	-	-	-
Coelho and Laporte (2013)	-	-	-	-
Coelho and Laporte (2014)	Core i7-2600 3.4 GHz	1	1739	0.870
Adulyasak et al. (2014)	Intel Xeon 2.67 Ghz	8	5701	2.851
Desaulniers et al. (2016)	Core i7-2600 3.4 GHz	1	1739	0.870
Archetti et al. (2017)	Xeon W3680, 3.33 GHz	8	4630	2.315
Alvarez et al. (2018)	Core i7-2600 3.4 GHz	1	8695	4.348
Avella et al. (2018)	Core i7-2620, 2.70 GHz	1	1466	0.733
Chitsaz et al. (2019)	Xeon X5650 2.67 GHz	1	1298	0.649
Alvarez et al. (2020)	Xeon X5650 2.67 GHz	1	1298	0.649
Diniz et al. (2020)	Intel Core i7-8700K 3.7 GHz	1	2747	1.374
Guimarães et al. (2020)	Xeon E5-2630 v2 2.60 GHz	6	3742	1.871
Manousakis et al. (2021)	Intel Core i7-7700 CPU 3.60 GHz	8	8658	4.329
Archetti et al. (2021)	Xeon E5-1620 v3 3.50 GHz	1	2017	1.009
Vadseth et al. (2021)	Xeon Gold 6144 3.5 GHz	1	2520	1.260
Skålnes et al. (2022)	Intel E5-2670v3 2.3GHz	1	1702	0.851
Solyalı and Süral (2022)	Xeon X5650 2.67 GHz	1	1298	0.649
Achamrah et al. (2022)	Quad-core Intel Core i7 3.3 GH	-	-	-
Schenekemberg et al. (2023)	AMD EPYC 7532 2.4 GHz	24	20547	10.274
Vadseth et al. (2023)	Xeon Gold 6144 3.5 GHz	1	2520	1.260
Skålnes et al. (2023b)	Intel E5-2670v3 2.3GHz	1	1702	0.851
This Work	Intel Xeon Platinum 8275L @ 3GHz	1	2386	1.193

5.2.2 Contribution to classical benchmark instances

In this section, we assess the performance of $HGSIRP$ on classical benchmark instances and compare it with results obtained from existing methods in the literature. The outcomes are summarized in Table 5.5. Out of a total of 1098 instances, our work identified 657 instances (60%) achieving the BKS, and for cases where the optimal solution remains unknown, we achieved better results for 79 instances (20%). Notably, the overall average Duality Gap was improved from 0.80% to 0.79%, signifying a reduction of 0.01%. The most significant enhancement was observed in the Large Multi-Vehicle instance set, widely acknowledged as the most challenging. Among the 240 instances in this set, we achieved improved solutions for 66 instances (28% and a reduction of 0.06% for the average Duality Gap.

Table 5.5: Performance Enhancements on Classical Benchmark Instances

		Single-Vehicle		Multi-Vehicle	
		Small	**Large**	**Small**	**Large**
Instances	**1098**	160	60	638	240
BKS Found	**657(60%)**	127(79%)	1(2%)	462(72%)	67(28%)
Opened Instances	**401**	0	37	124	240
New BKS	**79(20%)**	**0**	**1(3%)**	**12(10%)**	**66(28%)**
Before - Avg. Duality Gap	**0.80%**	0%	0.80%	0.16%	3.03%
After - Avg. Duality Gap	**0.79%**	0%	0.79%	0.16%	2.97%
Difference - Avg. Duality Gap	**-0.01%**	**0%**	**-0.01%**	**0%**	**-0.06%**

5.2.3
Literature methods performance in benchmark perspective

Within this section, we have integrated $HGSIRP$ into the realm of top-performing outcomes within the context of classical benchmark instances. This integration involved the addition of 79 new results. Our focus then shifted towards an extensive comparison of its overall performance against various methods documented in the literature. As elaborated upon in Section 5.2.1, it's worth noting that not all studies presented outcomes for every instance. To ensure a fairer assessment, we have segregated the results for distinct categories: Small Single-Vehicle, Large Single-Vehicle, Small Multi-Vehicle, and Large Multi-Vehicle sets.

To facilitate these comparisons, four key metrics take center stage. The initial metric is **BKS**, which quantifies the instances where the methods successfully make identifications within the realm of classical benchmark instances. A related metric, **BKS Unique**, signifies the instances where our method stood as the sole identifier among all others. Following this, we delve into the **Average Primal Gap** , a metric calculated using the ensuing formula:

$$\frac{METHOD_{UB} - LIT_{UB}}{LIT_{UB}}$$

Lastly, the **Average STime** emerges as the average scaled-time, obtained by incorporating the time reported by the method after applying the TAF. This standardization is particularly valuable in mitigating the impact of varying CPU models, rendering the comparative analysis more equitable.

To offer a unified perspective on the average Primal Gap and average STime, we present a complementary graph. In this graph, the X-axis depicts the average Primal Gap, while the Y-axis represents the average STime. Interpreting the graph, we observe that algorithms with superior performance tend to be positioned closer to the origin.

5.2.3.1
Small Single-Vehicle Set

The Table 5.6 displays the results obtained from the Small Single-Vehicle set of instances, which was originally introduced by Archetti et al. (2007). Skålnes et al. (2022) had to be excluded from the comparison due to incomplete results. Among a total of 160 instances, both Guimarães et al. (2020) and Schenekemberg et al. (2023) successfully identified all the BKS. Subsequently, Diniz et al. (2020) achieved results for 140 instances, while the $HGSIRP$ algorithm reached a count of 127 instances.

When considering the average Primal Gap, the $HGSIRP$ algorithm secured the second position with a gap of only 0.04%. For a more visual representation of these findings, refer to the graph presented in Figure 5.1, where the positioning, in order, aligns $HGSIRP$ (Castro2023) and Vadseth et al. (2021) closer to the origin. Notably, both Archetti et al. (2012) and Alvarez et al. (2020) omitted the reporting of execution times and, consequently, were not included in this graphical representation.

Table 5.6: Performance Comparison on Small Single-Vehicle Instances

Paper	**BKS (%)**	**BKS Unique**	**Avg. Primal Gap**	**Avg. STime**
Archetti et al. (2012)	121(76%)	0	0.06%	
Chitsaz et al. (2019)	30(19%)	0	1.76%	31.21
Alvarez et al. (2020)	65(1%)	0	1.79%	
Diniz et al. (2020)	140(88%)	0	0.07%	44.46
Guimarães et al. (2020)	160(100%)	0	0.00%	112.23
Vadseth et al. (2021)	109(68%)	0	0.41%	5.83
Schenekemberg et al. (2023)	160(100%)	0	0.00%	50.44
HGSIRP	127(79%)	0	0.04%	21.82

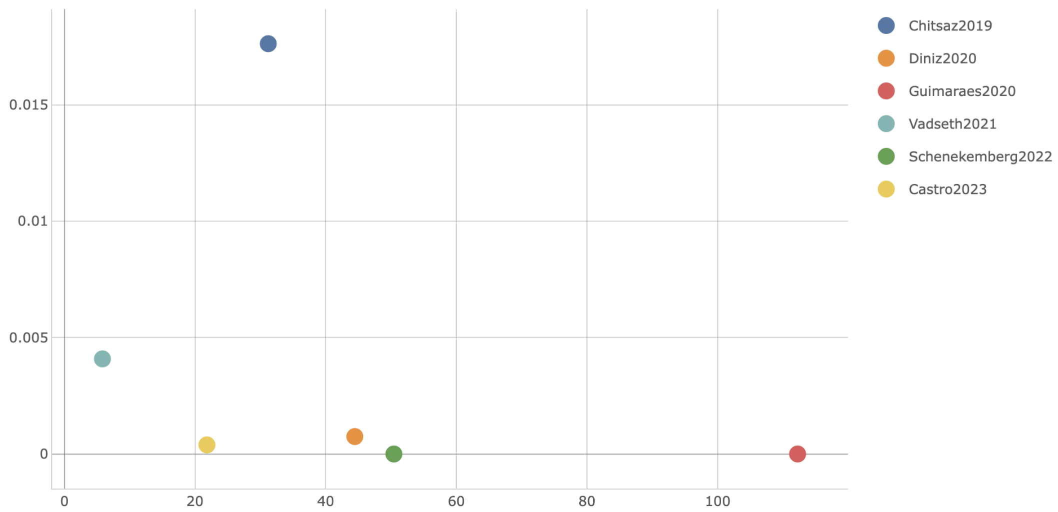

Figure 5.1: Graphical Representation of Performance on Small Single-Vehicle Instances

5.2.3.2 Large Single-Vehicle Set

The Table 5.7 presents results from the Large Single-Vehicle set, introduced by Archetti et al. (2012). Regrettably, Manousakis et al. (2021) could not be included in the comparison as they did not provide results for instances with 200 customers. Among the total of 60 instances, the most impressive performance was achieved by Schenekemberg et al. (2023), who identified 49 (81%) instances as BKS, with 26 of them being unique, resulting in an average Primal Gap of 0.03%. The second most notable contribution comes from Guimarães et al. (2020), who identified 32 (53%) BKS instances. In third place, Achamrah et al. (2022) found 8 (13%) BKS instances. Following closely are three other works, each identifying only 1 (2%) of the BKS instances: $HGSIRP$, Archetti et al. (2012), and Vadseth et al. (2021). Despite Achamrah et al. (2022) identifying more BKS instances compared to our work, we achieved a lower average Primal Gap of 0.55%.

The relationships between these results are visually represented in Figure 5.2, demonstrating that $HGSIRP$ (Castro2023) is the method closest to the origin, followed by Guimarães et al. (2020). Notably, both Archetti et al. (2012) and Vadseth et al. (2021) did not provide information regarding the execution times of their results.

Table 5.7: Performance Comparison on Large Single-Vehicle Instances

Paper	BKS (%)	BKS Unique	Avg. Primal Gap	Avg. STime
Archetti et al. (2012)	1(2%)	0	0.58%	
Chitsaz et al. (2019)	0	0	4%	4327.41
Guimarães et al. (2020)	32(53%)	8	0.41%	10406.54
Vadseth et al. (2021)	1(2%)	1	1.91%	137.60
Achamrah et al. (2022)	8(13%)	0	1.99%	
Schenekemberg et al. (2023)	49(82%)	26	0.03%	50142.26
HGSIRP	1(2%)	1	0.55%	926.32

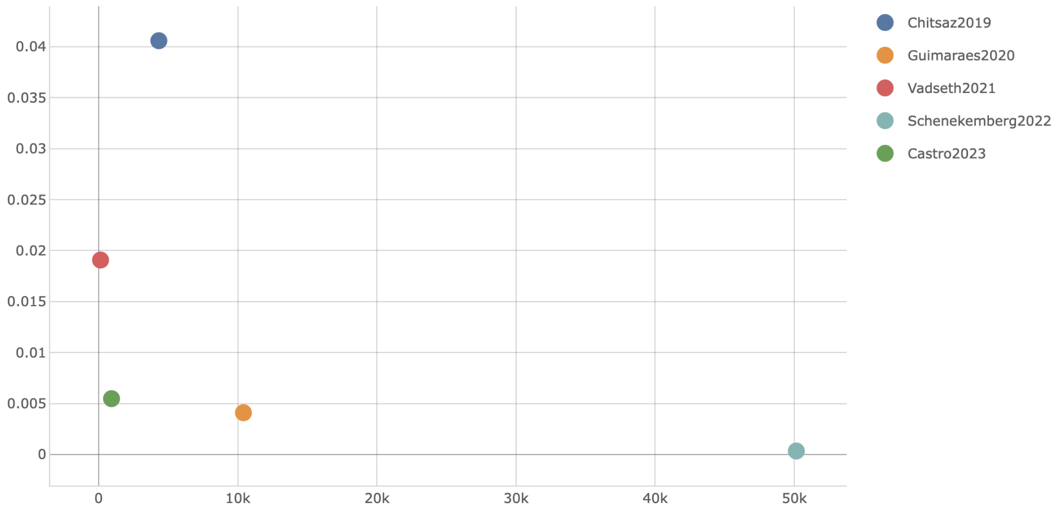

Figure 5.2: Graphical Representation of Performance on Large Single-Vehicle Instances

5.2.3.3 Small Multi-Vehicle Set

The Table 5.8 displays the outcomes obtained from the Small Multi-Vehicle set, a set introduced by Coelho and Laporte (2013). Notably, Adulyasak et al. (2014), Skålnes et al. (2022), and Avella et al. (2018) were excluded from the comparison due to their incomplete results for the entire dataset. Among the 638 instances analyzed, Schenekemberg et al. (2023) achieved the majority of the BKS instances, identifying 621 (97%), of which 55 are unique. Similarly, Skålnes et al. (2023b) and Manousakis et al. (2021) identified 513 (80%) and 487 (76%) BKS instances, respectively, with each discovering a single new unique BKS. *HGSIRP* secured the fourth position, identifying 462 (72%) instances as BKS, including 12 unique cases. In addition to its fourth-place ranking in terms of BKS instances found, this approach also exhibited superior average Primal Gap results, second only to the outcomes of Schenekemberg et al. (2023).

A visual representation can be found in Figure 5.3, where *HGSIRP*

(Castro2023) is the method positioned closest to the origin within the graph. Despite Schenekemberg et al. (2023) and Skålnes et al. (2022) achieving better results, it's worth noting that these approaches incurred higher CPU consumption. Noteworthy is the absence of execution time data for the results of Coelho and Laporte (2013) and Alvarez et al. (2020), which led to their exclusion from this graphical analysis.

Table 5.8: Performance Comparison on Small Multi-Vehicle Instances

Paper	**BKS (%)**	**BKS Unique**	**Avg. Primal Gap**	**Avg. STime**
Coelho and Laporte (2013)	407(64%)	0	4.90%	
Coelho and Laporte (2014)	342(54%)	0	2.79%	3403.06
Desaulniers et al. (2016)	356(56%)	0	19.40%	3432.50
Archetti et al. (2017)	218(34%)	0	1.51%	2639.46
Alvarez et al. (2018)_ILS	104(16%)	0	1.92%	130.43
Alvarez et al. (2018)_SA	245(38%)	0	1.42%	120.87
Chitsaz et al. (2019)	77(12%)	2	3.21%	44.92
Alvarez et al. (2020)	101(16%)	0	3.00%	
Diniz et al. (2020)	290(45%)	0	0.53%	252.87
Guimarães et al. (2020)	446(70%)	0	0.43%	5951.52
Manousakis et al. (2021)	487(76%)	1	0.10%	13962.02
Archetti et al. (2021)	233(37%)	0	0.78%	417.30
Vadseth et al. (2021)	190(30%)	0	1.28%	79.73
Schenekemberg et al. (2023)	621(97%)	55	0.002%	20159.21
Solyalı and Süral (2022)	155(24%)	0	0.57%	299.69
Skålnes et al. (2023b)	513(80%)	1	0.08%	3303.70
HGSIRP	462(72%)	12	0.03%	97.75

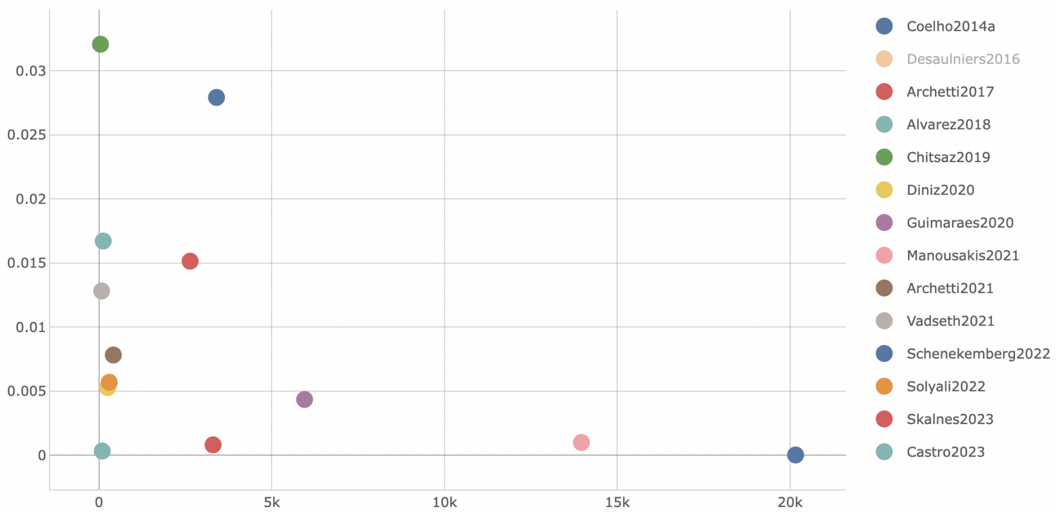

Figure 5.3: Graphical Representation of Performance on Small Multi-Vehicle Instances

5.2.3.4 Large Multi-Vehicle Set

Finally, the Table 5.9 displays the outcomes from the Large Multi-Vehicle set, also initially proposed by Coelho and Laporte (2013). This analysis excludes Manousakis et al. (2021) and Achamrah et al. (2022) due to their lack of results for the complete dataset. Among the 240 instances considered, Skålnes et al. (2023b) stands out as the most prolific contributor, identifying 97 (40%) BKS results, with 96 of those being unique. Following closely, Schenekemberg et al. (2023) secures the second position by finding 77 (32%) BKS results, 73 of which are unique. Subsequently, $HGSIRP$ identifies 67 (28%) BKS instances, of which 66 are unique. When analyzing the average Primal Gap, $HGSIRP$ ranks in the second place with a gap of 0.34%, just slightly behind Skålnes et al. (2023b) with a 0.25% gap.

A visual examination of Figure 5.4 reinforces the position of $HGSIRP$ (Castro2023) as the method situated nearest to the origin on the graph, signifying a more favorable equilibrium between cost and CPU consumption. Notably, Vadseth et al. (2023) is not included in this graph analysis due to the absence of execution time data for their results.

Table 5.9: Performance Comparison on Large Multi-Vehicle Instances

Paper	BKS (%)	BKS Unique	Avg. Primal Gap	Avg. STime
Archetti et al. (2017)	0(0%)	0	7.74%	10001.60
Alvarez et al. (2018)_ILS	0(0%)	0	5.76%	261.20
Alvarez et al. (2018)_SA	0(0%)	0	6.87%	262.53
Chitsaz et al. (2019)	0(0%)	0	3.34%	3330.23
Guimarães et al. (2020)	0(0%)	0	18.10%	13471.20
Archetti et al. (2021)	0(0%)	0	7.06%	2097.47
Vadseth et al. (2021)	0(0%)	0	1.43%	959.81
Schenekemberg et al. (2023)	77(32%)	73	0.44%	74170.23
Solyalı and Süral (2022)	3(1%)	3	1.77%	2542.05
Skålnes et al. (2023b)	97(40%)	93	0.25%	6126.61
Vadseth et al. (2023)	0(0%)	0	1.63%	
HGSIRP	67(28%)	66	0.34%	1571.53

5.2.4 In-Depth Analysis

In this section, we delve into a comprehensive analysis that involves grouping instances based on the number of customers. The findings presented in Table 5.10 highlight an examination of instances categorized according to customer count. Our $HGSIRP$ algorithm successfully identified 657 instances that achieve the BKS (Best Known Solution) status, accounting for 60%

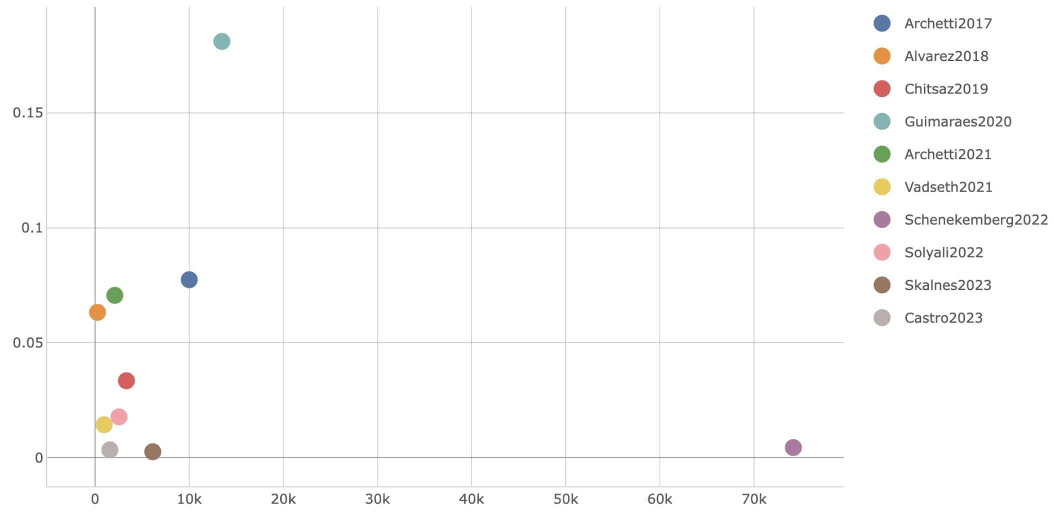

Figure 5.4: Graphical Representation of Performance on Large Multi-Vehicle Instances

of the total. Notably, 79 instances were newly discovered as BKS. Across all instances, the average Primal Gap stands at 0.11%, indicative of the algorithm's impressive performance.

Of particular significance is the algorithm's efficacy when dealing with large instances. Notably, for instances with 100 clients, the results were exceptional, with the algorithm unveiling 42 new BKS. However, the algorithm's performance dipped when faced with instances featuring 200 clients, managing to find only 1 (1%) new BKS. This outcome potentially underscores room for improvement. Our analysis suggests that our method encounters challenges as the number of customers increases, primarily due to the expanding complexity of the *Relocate and Swap neighborhoods*, as discussed in 3.1.2. These neighborhoods exhibit a computational complexity of $\mathcal{O}(T^2 * K^2 * N^2)$, implying a quadratic increase in the time required to execute the Fast Flow Network Simplex (FFNS) algorithm.

As discussed in the previous section, it is worth noting that $HGSIRP$ requires significantly less CPU time compared to other top methods. Consequently, dedicating additional computational resources to run this category of instances for an extended duration could potentially lead to more competitive results.

5.3 Comparison with 12$^{\text{th}}$ DIMACS Implementation Challenge

In 2022, the 12$^{\text{th}}$ DIMACS Implementation Challenge selected multi-vehicle instances for their competition. This instance set introduced two significant changes in comparison to the instances proposed by Coelho et al. (2012a).

Table 5.10: Distribution of Instances by Customer Count and Associated Best Known Solutions (BKS)

Customers	**BKS**	**BKS(%)**	**Avg. Primal Gap**
5	90	92%	0.02%
10	86	86%	0.01%
15	66	66%	0.02%
20	65(3)	65%	0.02%
25	53(3)	53%	0.07%
30	49(6)	49%	0.09%
35	42	84%	0.03%
40	47	94%	0.03%
45	46	92%	0.00%
50	69(24)	46%	0.10%
100	43(42)	43%	0.10%
200	1(1)	1%	0.74%
1098	**657(79)**	**60%**	**0.11%**

Firstly, 160 new instances were added, completing the small instances for clients from 35 to 50 with the missing six-periods, resulting in a total of 1040 instances.

Secondly, a significant difference was made in the vehicle capacity calculation. The vehicle capacity was rounded to the nearest lower integer, unlike the approach proposed by Coelho et al. (2012a), where the vehicle capacity was rounded to the nearest integer. This rounding method resulted in 297 different instances, 227 for the small set and 70 new instances for the large set. The details can be found in Appendix A.

The rules of the competition impose a CPU limit of a single thread execution in at most 1800 seconds in a machine of a *PassMark®* result of 2000. Different machines could be used but the time should be linearized. For calculating the ranking, the best solution across all participants gets a 10, the second 8, then 6, 5, 4, 3, 2, 1. In the case of ties, the points at play are evenly split among the solvers involved. More detail can be found at `http://dimacs.rutgers.edu/programs/challenge/vrp/irp/`.

During the competition, five teams submitted their results, but only the top four were invited to present their work. The winning team was *NTNU AXIOM*, whose solver, *MrOptimal*, employed a Branch and Cut (B&C) method and an efficient matheuristic for warm-starting. The team *GSCC*, with the solver *2FHBC*, also employed the B&C method and placed second. The third-place team, *SmartLab*, used a three-stage matheuristic with the methods *Relax-and-Fix*, *Local Search*, and *Tabu-Search* in their solver *TSMHA*. The fourth-place team was *PUC-Rio* with the solver *IRP-PUC*, an improved version of the

NSIRP discussed in Section 3.1. The complete papers and presentations can be found at `http://dimacs.rutgers.edu/programs/challenge/vrp/papers-videos/`. The results are summarized in Table 5.11.

Table 5.11: Ranking of 12th DIMACS Implementation Challenge

Team	**Solver**	**Avg. Point**
NTNU AXIOM	MrOptimal	7.574
GSCC	2FHBC	7.354
SmartLab	TSMHA	6.691
PUC-Rio	IRPUC++	6.639
sb	plasir	4.742

To compare $HGSIRP$ with 12th DIMACS Implementation Challenge, we followed the same rules. We ran the code on a single-threaded machine with a *PassMark®* score of 2386, respecting the CPU time limit of 1509 seconds. However, in this comparison, we deviated from the previous approach. Here, we ran all instances for the entire CPU limit, ignoring the usual termination criterion of $nbIter$ iterations without improvements. The evaluation used the same set of 1040 instances, including both the 160 new instances and the 297 different instances.

As shown in Table 5.12, if the proposed $HGSIRP$ had participated in the competition, it would have achieved first place. Comparing the results with the best results of all five solvers, the $HGSIRP$ would have found 332 better results with an average Primal Gap of -0.055%. The detailed results are presented in Table 5.13.

Table 5.12: Ranking of 12th DIMACS Implementation Challenge considering This Work

Team	**Solver**	**Avg. Point**
-	**HGSIRP**	**7.890**
NTNU AXIOM	MrOptimal	6.589
GSCC	2FHBC	6.335
PUC-Rio	IRPUC++	5.694
SmartLab	TSMHA	5.621
sb	plasir	3.837

Table 5.13: Comparison of Best Solutions from 12^{th} DIMACS Implementation Challenge and This Work

Customers	**Better**	**Equals**	**Worst**	**Avg. Primal Gap**
5	0	76	1	0.002%
10	0	79	1	0.000%
15	2	76	2	0.001%
20	12	61	7	-0.017%
25	22	54	4	-0.040%
30	26	46	8	-0.065%
35	26	50	4	-0.090%
40	31	46	3	-0.119%
45	32	44	4	-0.133%
50	110	40	10	-0.276%
100	66	0	14	-0.398%
200	5	0	75	0.703%
Total	332	572	133	-0.055%

6
Conclusions

In conclusion, this study proposed a novel solution to the IRP within the context of Vender-Managed Inventory (VMI). The proposed solution combined the Hybrid Genetic Search (HGS) framework with the Network Simplex IRP (NSIRP) local search strategy, resulting in an efficient and competitive approach that resulted in finding 79 new Best Known Solution (BKS) from the classical benchmark.

Additionally, this study proposed three Optimization Proposals that increase its efficiency and improve its ability to find better solutions. The proposed modifications included adding a Capacitated Vehicle Routing Problem (CVRP) local search, allowing for exploring infeasible vehicle capacity solutions, and incorporating a heuristic to calculate the maximum inventory cost degradation.

However, our experimentation revealed that the proposed algorithm's performance diminishes for instances involving a higher number of clients, indicating the need for further investigation and improvements in these scenarios.

There are two potential optimizations that we did not pursue but could enhance the algorithm's performance. These possibilities could be explored in future research:

– In the context of the crossover operation (Section 4.3), an alternative approach could involve solving a Traveling Salesperson Problem (TSP) for each time period based on the scheduling, rather than constructing a new $chrom\boldsymbol{T}$ keep the parents' visit order. This approach has the potential to generate superior solutions as inputs for the local search, thereby enhancing Intensification. However, it might lead to a reduction in Diversity since it would exclude the creation of slightly suboptimal solutions.
– Regarding the implementation enhancement that addresses infeasible vehicle capacities (Section 4.5.2), a more efficient strategy could be considered. Instead of introducing a new auxiliary vehicle for each existing vehicle, a more streamlined approach would be to use a single auxiliary vehicle that accounts for the entire additional vehicle capacity.

In future directions, applying this method to other variations of the IRP, such as multi-product IRP, and related problems like Production Routing Problem (PRP) can provide a more comprehensive evaluation of the proposed algorithm's versatility and effectiveness in different settings.

Another prospective avenue involves adapting the algorithm to optimize the logistic ratio as the objective function. This second objective function, which divides the total travel cost by inventory cost, offers greater realism in specific logistics contexts.

Furthermore, an interesting avenue for experimentation would be to run the proposed algorithm on newly proposed benchmarks by Skålnes et al. (2023a), which would enable us to assess the algorithm's performance in comparison to new instances.

Overall, exploring these future directions can help us gain a deeper understanding of the algorithm's strengths and limitations, as well as its potential for real-world applications.

Bibliography

Achamrah, F. E., Riane, F., Martinelly, C. D., and Limbourg, S. (2022). A matheuristic for solving inventory sharing problems. *Computers and Operations Research*, 138.

Adulyasak, Y., Cordeau, J. F., and Jans, R. (2014). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, 26:103–120.

Alvarez, A., Cordeau, J.-F., Jans, R., Munari, P., and Morabito, R. (2020). Formulations, branch-and-cut and a hybrid heuristic algorithm for an inventory routing problem with perishable products. *European Journal of Operational Research*, 283:511–529.

Alvarez, A., Munari, P., and Morabito, R. (2018). Iterated local search and simulated annealing algorithms for the inventory routing problem. *International Transactions in Operational Research*, 25:1785–1809.

Andersson, H., Hoff, A., Christiansen, M., Hasle, G., and Løkketangen, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers and Operations Research*, 37:1515–1536.

Archetti, C., Bertazzi, L., Hertz, A., and Speranza, M. G. (2012). A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24:101–116.

Archetti, C., Bertazzi, L., Laporte, G., and Speranza, M. G. (2007). A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, 41:382–391.

Archetti, C., Boland, N., and Speranza, M. G. (2017). A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing*, 29:377–387.

Archetti, C., Guastaroba, G., Huerta-Muñoz, D. L., and Speranza, M. G. (2021). A kernel search heuristic for the multivehicle inventory routing problem. *International Transactions in Operational Research*, 28:2984–3013.

Avella, P., Boccia, M., and Wolsey, L. A. (2015). Single-item reformulations for a vendor managed inventory routing problem: Computational experience with benchmark instances. *Networks*, 65:129–138.

Avella, P., Boccia, M., and Wolsey, L. A. (2018). Single-period cutting planes for inventory routing problems. *Transportation Science*, 52:497–508.

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., and Prutzman, P. J. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces (Providence, Rhode Island)*, 13:4–23.

Bertazzi, L. and Speranza, M. G. (2012). Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1:307–326.

Bertazzi, L. and Speranza, M. G. (2013). Inventory routing problems with multiple customers. *EURO Journal on Transportation and Logistics*, 2:255–275.

Blumenfeld, D. E., Burns, L. D., Diltz, J., and Daganzo, C. F. (1985). Analyzing trade-offs between transportation, inventory and production costs on freight networks. *Transportation Research Part B: Methodological*, 19:361–380.

Chitsaz, M., Cordeau, J. F., and Jans, R. (2019). A unified decomposition matheuristic for assembly, production, and inventory routing. *INFORMS Journal on Computing*, 31:134–152.

Coelho, L. C., Cordeau, J. F., and Laporte, G. (2012a). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24:270–287.

Coelho, L. C., Cordeau, J. F., and Laporte, G. (2012b). The inventory-routing problem with transshipment. *Computers and Operations Research*, 39:2537–2548.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, 48:1–19.

Coelho, L. C. and Laporte, G. (2013). The exact solution of several classes of inventory-routing problems. *Computers and Operations Research*, 40:558–565.

Coelho, L. C. and Laporte, G. (2014). Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, 155:391–397.

Desaulniers, G., Rakke, J. G., and Coelho, L. C. (2016). A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, 50:1060–1076.

DIMACS (2022). 12th dimacs implementation challenge. `http://dimacs.rutgers.edu/programs/challenge/vrp/irp/` Accessed: 2023-04-07.

Diniz, P., Martinelli, R., and Poggi, M. (2020). An efficient matheuristic for the inventory routing problem. In Baïou, M., Gendron, B., Günlük, O., and Mahjoub, A. R., editors, *Combinatorial Optimization*, pages 273–285, Cham. Springer International Publishing.

Dror, M. and Ball, M. (1987). Inventory/routing: Reduction from an annual to a short-period problem. *Naval Research Logistics*, 34:891–905.

Federgruen, A. and Zipkin, P. (1984). A combined vehicle routing and inventory allocation problem. *Operations Research*, 32:1019–1037.

Glover, F. and Hao, J.-K. (2011). The case for strategic oscillation. *Annals of Operations Research*, 183:163–173.

Guimarães, T. A., Schenekemberg, C. M., Coelho, L. C., Scarpin, C. T., and Pécora, J. E. (2020). Mechanisms for feasibility and improvement for inventory-routing problems. Technical report.

Manousakis, E., Repoussis, P., Zachariadis, E., and Tarantilis, C. (2021). Improved branch-and-cut for the inventory routing problem based on a two-commodity flow formulation. *European Journal of Operational Research*, 290:870–885.

Orlin, J. B. (1983). A polynomial-time parametric simplex algorithm for the minimum cost network flow problem. Working papers 1484-83., Massachusetts Institute of Technology (MIT), Sloan School of Management.

Roldán, R. F., Basagoiti, R., and Coelho, L. C. (2017). A survey on the inventory-routing problem with stochastic lead times and demands. *Journal of Applied Logic*, 24:15–24.

Sakhri, M. S. A., Tlili, M., and Korbaa, O. (2022). A memetic algorithm for the inventory routing problem. *Journal of Heuristics*, 28:351–375.

Santos, E., Ochi, L. S., Simonetti, L., and González, P. H. (2016). A hybrid heuristic based on iterated local search for multivehicle inventory routing problem. *Electronic Notes in Discrete Mathematics*, 52:197–204.

Schenekemberg, C. M., Guimarães, T. A., Chaves, A. A., and Coelho, L. C. (2023). A three-front parallel branch-and-cut algorithm for production and inventory routing problems. *Transportation Science*.

Skålnes, J., Andersson, H., Desaulniers, G., and Stålhane, M. (2022). An improved formulation for the inventory routing problem with time-varying demands. *European Journal of Operational Research*, 302:1189–1201.

Skålnes, J., Ben Ahmed, M., Hvattum, L. M., and Stålhane, M. (2023a). New benchmark instances for the inventory routing problem. *European Journal of Operational Research*.

Skålnes, J., Vadseth, S., Andersson, H., and Stålhane, M. (2023b). A branch-and-cut embedded matheuristic for the inventory routing problem. Technical report, NTNU.

Solyalı, O. and Süral, H. (2022). An effective matheuristic for the multivehicle inventory routing problem. *Transportation Science*, 56:1044–1057.

Subramanian, A. (2012). *Heuristic, Exact and Hybrid Approaches for Vehicle Routing*. PhD thesis, UFF.

Vadseth, S. T., Andersson, H., and Stålhane, M. (2021). An iterative matheuristic for the inventory routing problem. *Computers & Operations Research*, 131:105262.

Vadseth, S. T., Andersson, H., Stålhane, M., and Chitsaz, M. (2023). A multi-start route improving matheuristic for the production routeing problem. *International Journal of Production Research*, pages 1–22.

Vidal, T. (2016). Technical note: Split algorithm in o(n) for the capacitated vehicle routing problem. *Computers and Operations Research*, 69:40–47.

Vidal, T. (2022). Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60:611–624.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2015). Time-window relaxations in vehicle routing heuristics. *Journal of Heuristics*, 21:329–358.

Ǒguz Solyali and Süral, H. (2011). A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem. *Transportation Science*, 45:335–345.

A
Vehicle Capacity Comparison

Table A.1 illustrates the variations in vehicle capacity between the multi-vehicle instances presented in Coelho et al. (2012a) and those within the 12^{th} DIMACS Implementation Challengedataset.

Instance	Coelho et al. (2012a)	DIMACS (2022)
S_abs5n10_4_H6	240	239
S_abs4n25_2_H3	1015	1014
S_abs2n35_4_L3	653	652
S_abs1n20_4_L6	413	412
S_abs4n25_5_H3	406	405
S_abs5n35_2_L3	1270	1269
S_abs1n5_5_H3	58	57
S_abs1n40_5_H3	628	627
S_abs2n10_2_H6	328	327
S_abs5n35_5_L3	508	507
S_abs1n40_2_H3	1570	1569
S_abs1n5_2_H3	145	144
S_abs3n25_4_H3	530	529
S_abs1n30_4_H6	691	690
S_abs4n35_5_L3	487	486
S_abs4n35_2_L3	1217	1216
S_abs5n25_5_H3	470	469
S_abs1n50_2_L3	1823	1822
S_abs1n15_2_L3	620	619
S_abs5n25_2_H3	1175	1174
S_abs1n15_5_L3	248	247
S_abs4n10_4_H6	192	191
S_abs3n50_5_L3	767	766
S_abs5n5_4_H3	88	87
S_abs5n40_4_H3	822	821
S_abs3n15_2_L3	634	633
S_abs4n30_5_H6	444	443

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022) (*cont...*)

Instance	Coelho et al. (2012a)	DIMACS (2022)
S_abs1n35_4_L3	692	691
S_abs4n50_4_L3	1017	1016
S_abs5n20_2_L6	863	862
S_abs3n30_4_H6	712	711
S_abs1n10_2_H6	436	435
S_abs2n5_2_H3	119	118
S_abs2n40_2_H3	1369	1368
S_abs3n40_2_H3	1642	1641
S_abs2n30_4_H6	634	633
S_abs3n40_5_H3	657	656
S_abs5n50_4_L3	1007	1006
S_abs2n15_2_L3	593	592
S_abs2n50_2_L3	1870	1869
S_abs2n50_5_L3	748	747
S_abs1n30_2_H6	1382	1381
S_abs1n30_5_H6	553	552
S_abs2n25_5_H3	414	413
S_abs1n15_4_L3	310	309
S_abs3n35_5_L3	653	652
S_abs1n20_2_L6	826	825
S_abs5n10_5_H6	192	191
S_abs3n25_5_H3	424	423
S_abs5n35_4_L3	635	634
S_abs3n25_2_H3	1060	1059
S_abs1n40_4_H3	785	784
S_abs2n10_4_H6	164	163
S_abs2n30_2_H6	1268	1267
S_abs1n25_2_H3	943	942
S_abs3n40_4_H3	821	820
S_abs5n15_2_L3	511	510
S_abs4n20_4_L6	393	392
S_abs2n50_4_L3	935	934
S_abs4n5_5_H3	54	53
S_abs5n30_4_H6	566	565
S_abs4n40_2_H3	1373	1372
S_abs2n20_5_L6	334	333
S_abs5n5_2_H3	176	175

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022) (*cont...*)

Instance	Coelho et al. (2012a)	DIMACS (2022)
S_abs4n30_4_H6	555	554
S_abs3n15_4_L3	317	316
S_abs3n50_4_L3	959	958
S_abs4n15_2_L3	539	538
S_abs3n30_2_H6	1424	1423
S_abs4n5_2_L6	236	235
S_abs2n15_4_H6	323	322
S_abs1n45_4_H3	866	865
S_abs4n20_4_H3	375	374
S_abs3n5_4_L6	110	109
S_abs1n25_2_L6	856	855
S_abs3n30_2_L3	1421	1420
S_abs5n20_4_H3	434	433
S_abs4n15_2_H6	578	577
S_abs5n5_5_L6	74	73
S_abs5n5_2_L6	185	184
S_abs2n20_5_H3	313	312
S_abs5n25_4_L6	539	538
S_abs2n45_4_H3	793	792
S_abs2n25_5_L6	371	370
S_abs5n45_2_H3	1763	1762
S_abs3n10_4_L3	172	171
S_abs1n5_4_L6	127	126
S_abs3n25_5_L6	401	400
S_abs1n20_2_H3	800	799
S_abs4n25_4_L6	469	468
S_abs1n20_5_H3	320	319
S_abs2n5_2_L6	203	202
S_abs5n20_5_H3	347	346
S_abs4n15_4_H6	289	288
S_abs4n5_4_L6	118	117
S_abs2n15_2_H6	646	645
S_abs3n5_5_L6	88	87
S_abs5n15_4_H6	272	271
S_abs4n20_5_H3	300	299
S_abs1n25_4_L6	428	427
S_abs2n30_4_L3	678	677

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022) (*cont...*)

Instance	Coelho et al. (2012a)	DIMACS (2022)
S_abs2n10_2_L3	409	408
S_abs1n5_2_L6	254	253
S_abs4n45_4_H3	852	851
S_abs1n20_4_H3	400	399
S_abs3n45_5_H3	713	712
S_abs4n25_2_L6	938	937
S_abs5n25_2_L6	1078	1077
S_abs4n10_4_L3	206	205
S_abs2n45_2_H3	1586	1585
S_abs2n25_4_L6	464	463
S_abs3n10_2_L3	344	343
S_abs1n30_4_L3	713	712
S_abs5n5_4_L3	88	87
S_abs5n40_4_L3	822	821
S_abs3n50_5_H3	767	766
S_abs4n30_5_L6	444	443
S_abs1n35_4_H3	692	691
S_abs3n15_2_H3	634	633
S_abs5n20_2_H6	863	862
S_abs4n50_4_H3	1017	1016
S_abs2n5_2_L3	119	118
S_abs2n40_2_L3	1369	1368
S_abs1n10_2_L6	436	435
S_abs3n30_4_L6	712	711
S_abs2n30_4_L6	634	633
S_abs3n40_2_L3	1642	1641
S_abs5n50_4_H3	1007	1006
S_abs3n40_5_L3	657	656
S_abs2n15_2_H3	593	592
S_abs2n50_2_H3	1870	1869
S_abs2n50_5_H3	748	747
S_abs4n25_2_L3	1015	1014
S_abs5n10_4_L6	240	239
S_abs1n20_4_H6	413	412
S_abs4n25_5_L3	406	405
S_abs2n35_4_H3	653	652
S_abs1n5_5_L3	58	57

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022) (*cont...*)

Instance	Coelho et al. (2012a)	DIMACS (2022)
S_abs1n40_5_L3	628	627
S_abs5n35_2_H3	1270	1269
S_abs3n25_4_L3	530	529
S_abs1n40_2_L3	1570	1569
S_abs1n5_2_L3	145	144
S_abs5n35_5_H3	508	507
S_abs2n10_2_L6	328	327
S_abs1n30_4_L6	691	690
S_abs4n35_5_H3	487	486
S_abs4n35_2_H3	1217	1216
S_abs1n50_2_H3	1823	1822
S_abs1n15_2_H3	620	619
S_abs5n25_5_L3	470	469
S_abs1n15_5_H3	248	247
S_abs4n10_4_L6	192	191
S_abs5n25_2_L3	1175	1174
S_abs3n40_4_L3	821	820
S_abs1n25_2_L3	943	942
S_abs2n30_2_L6	1268	1267
S_abs4n20_4_H6	393	392
S_abs5n15_2_H3	511	510
S_abs4n5_5_L3	54	53
S_abs2n50_4_H3	935	934
S_abs4n40_2_L3	1373	1372
S_abs5n30_4_L6	566	565
S_abs4n30_4_L6	555	554
S_abs5n5_2_L3	176	175
S_abs2n20_5_H6	334	333
S_abs3n15_4_H3	317	316
S_abs3n50_4_H3	959	958
S_abs4n15_2_H3	539	538
S_abs3n30_2_L6	1424	1423
S_abs1n10_4_L6	218	217
S_abs1n30_2_L6	1382	1381
S_abs2n25_5_L3	414	413
S_abs1n30_5_L6	553	552
S_abs1n15_4_H3	310	309

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022) (*cont...*)

Instance	Coelho et al. (2012a)	DIMACS (2022)
S_abs3n35_5_H3	653	652
S_abs5n10_5_L6	192	191
S_abs1n20_2_H6	826	825
S_abs5n35_4_H3	635	634
S_abs3n25_5_L3	424	423
S_abs2n10_4_L6	164	163
S_abs1n40_4_L3	785	784
S_abs3n25_2_L3	1060	1059
S_abs5n25_4_H6	539	538
S_abs2n45_4_L3	793	792
S_abs5n45_2_L3	1763	1762
S_abs2n25_5_H6	371	370
S_abs3n10_4_H3	172	171
S_abs1n5_4_H6	127	126
S_abs3n25_5_H6	401	400
S_abs1n20_2_L3	800	799
S_abs4n25_4_H6	469	468
S_abs1n20_5_L3	320	319
S_abs4n5_2_H6	236	235
S_abs1n45_4_L3	866	865
S_abs2n15_4_L6	323	322
S_abs4n20_4_L3	375	374
S_abs1n25_2_H6	856	855
S_abs3n5_4_H6	110	109
S_abs3n30_2_H3	1421	1420
S_abs4n15_2_L6	578	577
S_abs5n20_4_L3	434	433
S_abs5n5_5_H6	74	73
S_abs2n20_5_L3	313	312
S_abs5n5_2_H6	185	184
S_abs1n5_2_H6	254	253
S_abs2n10_2_H3	409	408
S_abs4n45_4_L3	852	851
S_abs1n20_4_L3	400	399
S_abs4n25_2_H6	938	937
S_abs3n45_5_L3	713	712
S_abs4n10_4_H3	206	205

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022) (*cont...*)

Instance	Coelho et al. (2012a)	DIMACS (2022)
S_abs5n25_2_H6	1078	1077
S_abs2n45_2_L3	1586	1585
S_abs1n30_4_H3	713	712
S_abs3n10_2_H3	344	343
S_abs2n25_4_H6	464	463
S_abs5n20_5_L3	347	346
S_abs2n5_2_H6	203	202
S_abs4n15_4_L6	289	288
S_abs4n5_4_H6	118	117
S_abs2n15_2_L6	646	645
S_abs5n15_4_L6	272	271
S_abs3n5_5_H6	88	87
S_abs2n30_4_H3	678	677
S_abs4n20_5_L3	300	299
S_abs1n25_4_H6	428	427
L_abs2n200_2_H	8794	8793
L_abs6n100_2_L	4024	4023
L_abs5n200_2_L	8534	8533
L_abs8n50_2_H	1663	1662
L_abs3n200_2_H	8260	8259
L_abs5n50_2_L	1949	1948
L_abs8n200_2_H	8075	8074
L_abs5n100_2_L	4316	4315
L_abs4n100_2_L	3934	3933
L_abs9n200_2_H	8083	8082
L_abs5n50_2_H	1949	1948
L_abs6n100_2_H	4024	4023
L_abs2n200_2_L	8794	8793
L_abs5n200_2_H	8534	8533
L_abs3n200_2_L	8260	8259
L_abs8n200_2_L	8075	8074
L_abs5n100_2_H	4316	4315
L_abs8n50_2_L	1663	1662
L_abs4n100_2_H	3934	3933
L_abs9n200_2_L	8083	8082
L_abs8n100_4_H	2058	2057
L_abs2n200_4_H	4397	4396

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022) (*cont...*)

Instance	Coelho et al. (2012a)	DIMACS (2022)
L_abs6n100_4_L	2012	2011
L_abs1n100_4_H	2102	2101
L_abs5n200_4_L	4267	4266
L_abs3n50_4_L	980	979
L_abs4n200_4_L	4266	4265
L_abs10n50_4_L	1028	1027
L_abs1n50_4_L	986	985
L_abs6n50_4_H	1107	1106
L_abs9n100_4_H	2117	2116
L_abs3n200_4_H	4130	4129
L_abs7n100_4_L	2001	2000
L_abs2n50_4_H	983	982
L_abs6n200_4_L	4304	4303
L_abs5n100_4_L	2158	2157
L_abs4n100_4_L	1967	1966
L_abs7n50_4_L	969	968
L_abs2n50_4_L	983	982
L_abs8n100_4_L	2058	2057
L_abs6n100_4_H	2012	2011
L_abs2n200_4_L	4397	4396
L_abs5n200_4_H	4267	4266
L_abs1n100_4_L	2102	2101
L_abs4n200_4_H	4266	4265
L_abs10n50_4_H	1028	1027
L_abs7n50_4_H	969	968
L_abs9n100_4_L	2117	2116
L_abs7n100_4_H	2001	2000
L_abs3n200_4_L	4130	4129
L_abs6n200_4_H	4304	4303
L_abs10n100_4_H	2037	2036
L_abs3n50_4_H	980	979
L_abs5n100_4_H	2158	2157
L_abs10n100_4_L	2037	2036
L_abs4n100_4_H	1967	1966
L_abs6n50_4_L	1107	1106
L_abs1n50_4_H	986	985
L_abs4n50_5_H	809	808

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022) (*cont...*)

Instance	Coelho et al. (2012a)	DIMACS (2022)
L_abs3n50_5_L	784	783
L_abs3n200_5_H	3304	3303
L_abs6n200_5_L	3443	3442
L_abs8n200_5_H	3230	3229
L_abs7n50_5_L	775	774
L_abs7n50_5_H	775	774
L_abs3n200_5_L	3304	3303
L_abs6n200_5_H	3443	3442
L_abs8n200_5_L	3230	3229
L_abs3n50_5_H	784	783
L_abs4n50_5_L	809	808

Table A.1: Vehicle capacity comparison between Coelho et al. (2012a) and DIMACS (2022)

B
Invalid Literature Solutions

Table B.1 displays a compilation of results that have not been considered in this study due to their manifestation of a Lower Bound (LB) that surpasses an Upper Bound (UB) established by a different work. The table encompasses information about the instances and the corresponding works that were disregarded, along with the work that identified a smaller UB.

Instance	LB Paper	UB Paper	LB	UB	Difference	Duality Gap
S_abs4n25_2_H6	Coelho and Laporte (2014)	Vadseth et al. (2021)	18293.67	15289.77	-3003.9	-16.42%
S_abs5n25_2_H6	Coelho and Laporte (2014)	Diniz et al. (2020)	21582.22	18141.4	-3440.82	-15.94%
S_abs2n30_2_H6	Coelho and Laporte (2014)	Schenekemberg et al. (2023)	21456.63	18715.71	-2740.92	-12.77%
S_abs1n25_2_H6	Coelho and Laporte (2014)	Skålnes et al. (2023b)	16045.99	14621.29	-1424.7	-8.88%
S_abs2n25_2_H6	Coelho and Laporte (2014)	Schenekemberg et al. (2023)	17533.01	16086.07	-1446.94	-8.25%
S_abs4n35_5_L3	Coelho and Laporte (2013)	Diniz et al. (2020)	5087.1	4667.55	-419.55	-8.25%
S_abs4n30_2_H6	Coelho and Laporte (2014)	Coelho and Laporte (2013)	17577.2	16697.1	-880.1	-5.01%
L_abs10n50_5_L	Manousakis et al. (2021)	Skålnes et al. (2023b)	16119.02	15520.79	-598.23	-3.71%
S_abs5n30_5_L6	Manousakis et al. (2021)	Schenekemberg et al. (2023)	11264.5	11195.44	-69.06	-0.61%
S_abs2n15_4_H6	Manousakis et al. (2021)	Schenekemberg et al. (2023)	13500.63	13499.78	-0.85	-0.01%
S_abs2n30_3_H3	Coelho and Laporte (2014)	Chitsaz et al. (2019)	9584.51	9584.21	-0.3	-0.00%
S_abs2n50_3_H3	Schenekemberg et al. (2023)	Chitsaz et al. (2019)	13118.66	13118.32	-0.34	-0.00%
S_abs2n15_5_H6	Manousakis et al. (2021)	Schenekemberg et al. (2023)	14463.35	14462.98	-0.37	-0.00%

Table B.1: Invalid Literature Solutions

C
Detailed Computational Results

Table C.1 presents the average results after ten individual executions for every literature instance, showing the obtained final cost and the Primal Gap compared to BKS from literature.

Instance	UB	Time(s)	Primal Gap
S_abs1n5_1_L3	1213	1	0.0000
S_abs1n5_1_L6	3147.74	4	0.0000
S_abs1n5_2_L3	1373.92	3	0.0371
S_abs1n5_2_L6	3736.12	14	0.0000
S_abs1n5_3_L3	1407.59	3	0.0000
S_abs1n5_3_L6	4617.59	24	0.0000
S_abs1n5_4_L3	1578.65	5	0.0000
S_abs1n5_4_L6	5466.63	36	0.0000
S_abs1n5_5_L3	1687.42	7	0.0000
S_abs1n5_5_L6	6406.12	51	0.0000
S_abs2n5_1_L3	967.04	1	0.0000
S_abs2n5_1_L6	2529.32	5	0.0000
S_abs2n5_2_L3	1155.87	2	0.0000
S_abs2n5_2_L6	3148.59	13	0.0000
S_abs2n5_3_L3	1561.07	5	0.0000
S_abs2n5_3_L6	4184.4	24	0.0000
S_abs2n5_4_L3	1791.03	4	0.0000
S_abs2n5_4_L6	5089.26	51	0.0006
S_abs2n5_5_L3	1997.96	6	0.0000
S_abs2n5_5_L6	5972.8	48	0.0000
S_abs3n5_1_L3	1721.33	2	0.0000
S_abs3n5_1_L6	4453.72	5	0.0000
S_abs3n5_2_L3	2401.33	3	0.0000
S_abs3n5_2_L6	5926.65	16	0.0000
S_abs3n5_3_L3	2960.75	4	0.0000
S_abs3n5_3_L6	7667.42	28	0.0000
S_abs3n5_4_L3	3567.05	9	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs3n5_4_L6	9284.7	45	0.0009
S_abs3n5_5_L3	3942.69	6	0.3446
S_abs3n5_5_L6	11246.58	48	0.0000
S_abs4n5_1_L3	1381.71	2	0.0000
S_abs4n5_1_L6	3144.91	5	0.0000
S_abs4n5_2_L3	1701.71	4	0.0000
S_abs4n5_2_L6	3776.54	11	0.0000
S_abs4n5_3_L3	2275.59	5	0.0000
S_abs4n5_3_L6	4465.87	26	0.0000
S_abs4n5_4_L3	2616.03	8	0.0011
S_abs4n5_4_L6	5056.37	35	0.0000
S_abs4n5_5_L3	3303.39	11	0.0000
S_abs4n5_5_L6	6255.56	55	0.0000
S_abs5n5_1_L3	963.95	1	0.0000
S_abs5n5_1_L6	2230.31	3	0.0000
S_abs5n5_2_L3	1186.95	2	0.1865
S_abs5n5_2_L6	2855.06	11	0.0000
S_abs5n5_3_L3	1478.29	3	0.0000
S_abs5n5_3_L6	3842.3	28	0.0000
S_abs5n5_4_L3	1640.93	3	0.0000
S_abs5n5_4_L6	4876.62	34	0.0000
S_abs5n5_5_L3	1973.07	6	0.0005
S_abs1n5_1_H3	1870.88	1	0.0000
S_abs1n5_1_H6	5382.66	3	0.0000
S_abs1n5_2_H3	2028.95	3	0.0592
S_abs1n5_2_H6	5972.87	16	0.0000
S_abs1n5_3_H3	2061.27	3	0.0000
S_abs1n5_3_H6	6852.36	24	0.0000
S_abs1n5_4_H3	2234.65	5	0.0000
S_abs1n5_4_H6	7704.62	42	0.0000
S_abs1n5_5_H3	2340.08	7	0.0000
S_abs1n5_5_H6	8636.29	68	0.0000
S_abs2n5_1_H3	1553.82	1	0.0000
S_abs2n5_1_H6	4524.84	4	0.0000
S_abs2n5_2_H3	1756.07	2	0.0000
S_abs2n5_2_H6	5139.71	14	0.0000
S_abs2n5_3_H3	2156.69	5	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs2n5_3_H6	6171.42	30	0.0000
S_abs2n5_4_H3	2386.4	4	0.0000
S_abs2n5_4_H6	7052.71	49	0.0044
S_abs2n5_5_H3	2594.02	6	0.0000
S_abs2n5_5_H6	7939.93	49	0.0000
S_abs3n5_1_H3	2610.7	2	0.0000
S_abs3n5_1_H6	6281.62	5	0.0000
S_abs3n5_2_H3	3290.7	3	0.0000
S_abs3n5_2_H6	7746.36	16	0.0000
S_abs3n5_3_H3	3828.96	4	0.0000
S_abs3n5_3_H6	9501.22	29	0.0000
S_abs3n5_4_H3	4445.22	10	0.0000
S_abs3n5_4_H6	11113.6	41	0.0000
S_abs3n5_5_H3	4810.52	5	0.2486
S_abs3n5_5_H6	13037.46	52	0.0002
S_abs4n5_1_H3	1823.15	2	0.0000
S_abs4n5_1_H6	4778.41	5	0.0000
S_abs4n5_2_H3	2143.15	4	0.0000
S_abs4n5_2_H6	5419.55	12	0.0000
S_abs4n5_3_H3	2716.21	5	0.0000
S_abs4n5_3_H6	6107.27	28	0.0000
S_abs4n5_4_H3	3052.49	8	0.0046
S_abs4n5_4_H6	6686.93	34	0.0000
S_abs4n5_5_H3	3741.83	11	0.0000
S_abs4n5_5_H6	7881.11	60	0.0000
S_abs5n5_1_H3	1821.42	1	0.0000
S_abs5n5_1_H6	4015.55	3	0.0000
S_abs5n5_2_H3	2044.42	2	1.0219
S_abs5n5_2_H6	4637.08	12	0.0000
S_abs5n5_3_H3	2315.04	3	0.0000
S_abs5n5_3_H6	5610.62	29	0.0000
S_abs5n5_4_H3	2476.72	3	0.0339
S_abs5n5_4_H6	6634.2	35	0.0000
S_abs5n5_5_H3	2818.21	6	0.0025
S_abs1n10_1_L3	1666.67	2	0.0000
S_abs1n10_1_L6	4073.19	14	0.0005
S_abs1n10_2_L3	2186.79	6	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs1n10_2_L6	5599.07	27	0.0000
S_abs1n10_3_L3	2656.21	7	0.0000
S_abs1n10_3_L6	7050.9	60	0.0000
S_abs1n10_4_L3	3185.54	12	0.0000
S_abs1n10_4_L6	8353.54	82	0.0000
S_abs1n10_5_L3	3652.38	22	0.0000
S_abs1n10_5_L6	9753.7	257	0.0062
S_abs2n10_1_L3	2163.62	3	0.0000
S_abs2n10_1_L6	4990.79	8	0.0000
S_abs2n10_2_L3	2744.23	7	0.0000
S_abs2n10_2_L6	6458.59	23	0.0000
S_abs2n10_3_L3	3404.52	9	0.0000
S_abs2n10_3_L6	8221.94	47	0.0000
S_abs2n10_4_L3	4206.37	16	0.0233
S_abs2n10_4_L6	9820.37	78	0.0000
S_abs2n10_5_L3	4615.71	19	0.0000
S_abs2n10_5_L6	11577.48	168	0.0002
S_abs3n10_1_L3	1809.16	2	0.0000
S_abs3n10_1_L6	4445.56	11	0.0398
S_abs3n10_2_L3	2158.48	4	0.0000
S_abs3n10_2_L6	5197.4	20	0.0000
S_abs3n10_3_L3	2587.4	7	0.0046
S_abs3n10_3_L6	6123.03	44	0.0127
S_abs3n10_4_L3	2907.34	11	0.0200
S_abs3n10_4_L6	7192.53	83	0.0000
S_abs3n10_5_L3	3339.2	17	0.0000
S_abs3n10_5_L6	8176.62	168	0.0005
S_abs4n10_1_L3	1712.82	3	0.0000
S_abs4n10_1_L6	4764.15	10	0.0000
S_abs4n10_2_L3	2421.88	7	0.0000
S_abs4n10_2_L6	6139.11	21	0.0000
S_abs4n10_3_L3	3123.18	10	0.0000
S_abs4n10_3_L6	7423.9	47	0.0000
S_abs4n10_4_L3	3622.99	14	0.0000
S_abs4n10_4_L6	8585.83	58	0.0013
S_abs4n10_5_L3	4096.78	15	0.0000
S_abs4n10_5_L6	10021.98	101	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs5n10_1_L3	1855.4	3	0.0000
S_abs5n10_1_L6	4463.61	12	0.0000
S_abs5n10_2_L3	2076.4	5	0.0000
S_abs5n10_2_L6	5055.19	21	0.0008
S_abs5n10_3_L3	2378.33	7	0.0000
S_abs5n10_3_L6	5741.73	40	0.0002
S_abs5n10_4_L3	2789.31	12	0.0000
S_abs5n10_4_L6	6522.53	89	0.0000
S_abs5n10_5_L3	2911.11	6	0.0000
S_abs5n10_5_L6	7132.12	79	0.0000
S_abs1n10_1_H3	3726.94	3	0.0000
S_abs1n10_1_H6	7783.16	15	0.0000
S_abs1n10_2_H3	4248.38	5	0.0000
S_abs1n10_2_H6	9299.95	31	0.0000
S_abs1n10_3_H3	4722.42	8	0.0000
S_abs1n10_3_H6	10743.86	67	0.0000
S_abs1n10_4_H3	5237.42	12	0.0000
S_abs1n10_4_H6	12089.75	92	0.3150
S_abs1n10_5_H3	5714.66	21	0.0292
S_abs1n10_5_H6	13449.08	195	0.0000
S_abs2n10_1_H3	3861.85	3	0.0000
S_abs2n10_1_H6	7813.82	8	0.0000
S_abs2n10_2_H3	4437.78	6	0.0000
S_abs2n10_2_H6	9280.42	24	0.0000
S_abs2n10_3_H3	5100.49	10	0.0004
S_abs2n10_3_H6	11046.82	50	0.0004
S_abs2n10_4_H3	5896.57	19	0.0003
S_abs2n10_4_H6	12648.29	76	0.0001
S_abs2n10_5_H3	6318.13	24	0.0000
S_abs2n10_5_H6	14399.44	187	0.0000
S_abs3n10_1_H3	3414.59	2	0.0000
S_abs3n10_1_H6	7720.98	11	0.3228
S_abs3n10_2_H3	3763.18	4	0.2144
S_abs3n10_2_H6	8445.51	22	0.0000
S_abs3n10_3_H3	4193.44	7	0.0523
S_abs3n10_3_H6	9357.14	37	0.0000
S_abs3n10_4_H3	4515.54	8	0.1195

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs3n10_4_H6	10442.39	71	0.0003
S_abs3n10_5_H3	4940.37	15	0.0000
S_abs3n10_5_H6	11428.86	181	0.0009
S_abs4n10_1_H3	3342.05	3	0.0000
S_abs4n10_1_H6	7894.27	11	0.0000
S_abs4n10_2_H3	4051.83	7	0.0000
S_abs4n10_2_H6	9284.36	23	0.0000
S_abs4n10_3_H3	4743.88	11	0.0000
S_abs4n10_3_H6	10588.51	66	0.0000
S_abs4n10_4_H3	5243.27	17	0.0000
S_abs4n10_4_H6	11743.92	73	0.0001
S_abs4n10_5_H3	5717.36	18	0.0000
S_abs4n10_5_H6	13174.95	127	0.0004
S_abs5n10_1_H3	3892.44	3	0.0000
S_abs5n10_1_H6	8568.48	12	0.0000
S_abs5n10_2_H3	4113.44	5	0.0000
S_abs5n10_2_H6	9200.85	25	0.0000
S_abs5n10_3_H3	4407.1	7	0.0000
S_abs5n10_3_H6	9886.04	34	0.0153
S_abs5n10_4_H3	4827.04	13	0.0000
S_abs5n10_4_H6	10656.94	69	0.0283
S_abs5n10_5_H3	4954.12	6	0.0000
S_abs5n10_5_H6	11251.4	77	0.0041
S_abs1n15_1_L3	2037.35	5	0.0000
S_abs1n15_1_L6	5287.24	20	0.0197
S_abs1n15_2_L3	2203.33	10	0.0000
S_abs1n15_2_L6	5885.76	45	0.0211
S_abs1n15_3_L3	2690.08	13	0.0000
S_abs1n15_3_L6	6758.19	74	0.0000
S_abs1n15_4_L3	3073.1	20	0.0114
S_abs1n15_4_L6	7661.59	134	0.0000
S_abs1n15_5_L3	3487.12	23	0.0000
S_abs1n15_5_L6	8658.01	274	0.0256
S_abs2n15_1_L3	2039.33	4	0.0000
S_abs2n15_1_L6	5316.69	15	0.0000
S_abs2n15_2_L3	2461.85	10	0.0000
S_abs2n15_2_L6	6052.46	36	0.0223

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs2n15_3_L3	2665.57	9	0.0000
S_abs2n15_3_L6	7008.11	76	0.0303
S_abs2n15_4_L3	3304.41	16	0.0000
S_abs2n15_4_L6	7971.77	95	0.0000
S_abs2n15_5_L3	3797.18	26	0.0000
S_abs2n15_5_L6	8966.64	248	0.0051
S_abs3n15_1_L3	2355.35	3	0.0000
S_abs3n15_1_L6	5806.34	14	0.4959
S_abs3n15_2_L3	2691.67	7	0.0000
S_abs3n15_2_L6	6891.21	47	0.0000
S_abs3n15_3_L3	2964.51	8	0.0000
S_abs3n15_3_L6	8038.43	81	0.0000
S_abs3n15_4_L3	3649.03	20	0.0000
S_abs3n15_4_L6	9163.67	113	0.0000
S_abs3n15_5_L3	4025.27	21	0.0000
S_abs3n15_5_L6	10312.89	216	0.0076
S_abs4n15_1_L3	2075.95	4	0.0000
S_abs4n15_1_L6	5257.04	21	0.0230
S_abs4n15_2_L3	2437.88	11	0.0074
S_abs4n15_2_L6	6047.53	55	0.0000
S_abs4n15_3_L3	2810.33	13	0.0000
S_abs4n15_3_L6	7082.09	90	0.0518
S_abs4n15_4_L3	3124.19	13	0.0000
S_abs4n15_4_L6	8237.47	106	0.0000
S_abs4n15_5_L3	3496.54	22	0.0000
S_abs4n15_5_L6	9314.02	179	0.0785
S_abs5n15_1_L3	2079.78	6	0.0000
S_abs5n15_1_L6	4967.54	24	0.0000
S_abs5n15_2_L3	2531.53	8	0.0692
S_abs5n15_2_L6	6280.02	41	0.0000
S_abs5n15_3_L3	3182.83	10	0.0959
S_abs5n15_3_L6	7520.76	49	0.0000
S_abs5n15_4_L3	3605.62	28	0.4278
S_abs5n15_4_L6	8868.3	118	0.0000
S_abs5n15_5_L3	4180.09	22	0.1138
S_abs5n15_5_L6	10353.38	186	0.0247
S_abs1n15_1_H3	4636.33	5	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs1n15_1_H6	11007.19	31	0.0020
S_abs1n15_2_H3	4802.17	7	0.0000
S_abs1n15_2_H6	11579.09	65	0.0000
S_abs1n15_3_H3	5289.53	12	0.0000
S_abs1n15_3_H6	12473.34	113	0.0107
S_abs1n15_4_H3	5657.94	22	0.0000
S_abs1n15_4_H6	13365.21	179	0.0000
S_abs1n15_5_H3	6070.3	31	0.0000
S_abs1n15_5_H6	14376.27	264	0.0000
S_abs2n15_1_H3	4522.63	6	0.0000
S_abs2n15_1_H6	10809.83	18	0.0112
S_abs2n15_2_H3	4932.66	9	0.0000
S_abs2n15_2_H6	11553.49	50	0.0000
S_abs2n15_3_H3	5150.55	9	0.0000
S_abs2n15_3_H6	12503.81	92	0.0482
S_abs2n15_4_H3	5785.16	13	0.0000
S_abs2n15_4_H6	13499.78	125	0.0000
S_abs2n15_5_H3	6276.48	27	0.0461
S_abs2n15_5_H6	14474.51	233	0.0797
S_abs3n15_1_H3	5211.67	3	0.0000
S_abs3n15_1_H6	12096.84	13	0.0001
S_abs3n15_2_H3	5557.43	6	0.0000
S_abs3n15_2_H6	13248.45	52	0.0548
S_abs3n15_3_H3	5836.99	7	0.0000
S_abs3n15_3_H6	14380.04	91	0.0049
S_abs3n15_4_H3	6518.49	21	0.0000
S_abs3n15_4_H6	15499.6	143	0.0000
S_abs3n15_5_H3	6892.58	22	0.0017
S_abs3n15_5_H6	16635.66	258	0.0111
S_abs4n15_1_H3	4216.68	5	0.0000
S_abs4n15_1_H6	9702.03	17	0.1530
S_abs4n15_2_H3	4588.9	11	0.2499
S_abs4n15_2_H6	10478.47	55	0.0273
S_abs4n15_3_H3	4944.26	13	0.0000
S_abs4n15_3_H6	11505.37	98	0.0130
S_abs4n15_4_H3	5256.56	14	0.0000
S_abs4n15_4_H6	12672.81	107	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs4n15_5_H3	5627.24	23	0.0000
S_abs4n15_5_H6	13748.98	243	0.0041
S_abs5n15_1_H3	4072.03	7	0.0000
S_abs5n15_1_H6	9216.16	24	0.0000
S_abs5n15_2_H3	4522.03	9	0.0000
S_abs5n15_2_H6	10536.45	54	0.0000
S_abs5n15_3_H3	5171.75	8	0.0000
S_abs5n15_3_H6	11782.44	65	0.0821
S_abs5n15_4_H3	5581.89	24	0.0358
S_abs5n15_4_H6	13110.18	122	0.0197
S_abs5n15_5_H3	6168.06	23	0.0000
S_abs5n15_5_H6	14606.85	229	0.0193
S_abs1n20_1_L3	2141.98	4	0.0000
S_abs1n20_1_L6	5997.18	36	0.2020
S_abs1n20_2_L3	2791.96	11	0.0000
S_abs1n20_2_L6	7259.85	59	0.0000
S_abs1n20_3_L3	3480.38	21	0.0000
S_abs1n20_3_L6	8654.07	90	0.0087
S_abs1n20_4_L3	4022.66	24	0.0000
S_abs1n20_4_L6	10137.72	139	0.0435
S_abs1n20_5_L3	4279.43	24	0.0000
S_abs1n20_5_L6	11656.01	364	-0.0175
S_abs2n20_1_L3	2367.96	4	0.0000
S_abs2n20_1_L6	5821.44	26	0.0161
S_abs2n20_2_L3	2535.04	10	0.0000
S_abs2n20_2_L6	6239.34	50	0.2811
S_abs2n20_3_L3	2778.54	11	0.0000
S_abs2n20_3_L6	6791.26	56	0.0000
S_abs2n20_4_L3	3002.57	13	0.1488
S_abs2n20_4_L6	7457.15	135	-0.0059
S_abs2n20_5_L3	3215.92	23	0.0544
S_abs2n20_5_L6	8151.49	155	0.0998
S_abs3n20_1_L3	2453.79	4	0.0000
S_abs3n20_1_L6	6663.49	30	0.0297
S_abs3n20_2_L3	2681.93	10	0.0000
S_abs3n20_2_L6	7351.06	71	0.0000
S_abs3n20_3_L3	2928.09	10	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs3n20_3_L6	8264.7	90	0.0000
S_abs3n20_4_L3	3508.79	13	0.0000
S_abs3n20_4_L6	9264.49	157	0.0421
S_abs3n20_5_L3	3879.79	16	0.0000
S_abs3n20_5_L6	10366.64	238	0.0369
S_abs4n20_1_L3	3017.56	11	0.0000
S_abs4n20_1_L6	7201.08	39	0.0282
S_abs4n20_2_L3	3455.87	14	0.0000
S_abs4n20_2_L6	8200.31	87	0.0000
S_abs4n20_3_L3	3984.13	22	0.0000
S_abs4n20_3_L6	9854	145	0.0029
S_abs4n20_4_L3	4668.98	28	0.0000
S_abs4n20_4_L6	11452.79	236	0.0268
S_abs4n20_5_L3	5111.03	31	0.0000
S_abs4n20_5_L6	13076.12	390	0.0438
S_abs5n20_1_L3	2705.89	7	0.0000
S_abs5n20_1_L6	6820.71	28	0.0000
S_abs5n20_2_L3	3273.17	16	0.0000
S_abs5n20_2_L6	8368.75	80	0.0000
S_abs5n20_3_L3	3980.23	20	0.0000
S_abs5n20_3_L6	10222.62	150	0.0183
S_abs5n20_4_L3	4620.77	21	0.0000
S_abs5n20_4_L6	12271.12	249	0.0165
S_abs5n20_5_L3	5362.01	37	0.0000
S_abs5n20_5_L6	14210.4	425	0.1099
S_abs1n20_1_H3	5593.01	4	0.0000
S_abs1n20_1_H6	12965.48	45	0.0151
S_abs1n20_2_H3	6242.75	11	0.0000
S_abs1n20_2_H6	14237.37	61	0.0004
S_abs1n20_3_H3	6906.53	20	0.1032
S_abs1n20_3_H6	15632.9	106	0.0204
S_abs1n20_4_H3	7451.82	24	0.0000
S_abs1n20_4_H6	17145.56	164	0.0230
S_abs1n20_5_H3	7699.2	24	0.0000
S_abs1n20_5_H6	18680.65	339	0.1448
S_abs2n20_1_H3	5812.35	4	0.0000
S_abs2n20_1_H6	13128.37	21	0.0003

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs2n20_2_H3	5979.59	10	0.0000
S_abs2n20_2_H6	13599.65	60	0.0650
S_abs2n20_3_H3	6224.11	12	0.0000
S_abs2n20_3_H6	14141.85	81	0.0795
S_abs2n20_4_H3	6450.37	17	0.2308
S_abs2n20_4_H6	14803.1	137	0.0059
S_abs2n20_5_H3	6636.51	27	0.0211
S_abs2n20_5_H6	15496.93	167	0.0375
S_abs3n20_1_H3	6001.53	4	0.0000
S_abs3n20_1_H6	13110.51	32	0.0275
S_abs3n20_2_H3	6238.99	12	0.0000
S_abs3n20_2_H6	13774.43	67	0.0001
S_abs3n20_3_H3	6487.39	10	0.0000
S_abs3n20_3_H6	14724.73	119	0.0226
S_abs3n20_4_H3	7060.93	14	0.0000
S_abs3n20_4_H6	15699.78	123	0.0000
S_abs3n20_5_H3	7434.33	19	0.0003
S_abs3n20_5_H6	16820.49	290	0.1026
S_abs4n20_1_H3	5907.68	11	0.0000
S_abs4n20_1_H6	13285.28	24	0.0070
S_abs4n20_2_H3	6335.48	16	0.0000
S_abs4n20_2_H6	14317.13	108	0.0003
S_abs4n20_3_H3	6873.98	26	0.0000
S_abs4n20_3_H6	15976.18	196	0.0000
S_abs4n20_4_H3	7544.17	34	0.0000
S_abs4n20_4_H6	17584.58	275	-0.0110
S_abs4n20_5_H3	8002.23	38	0.0000
S_abs4n20_5_H6	19188.11	427	0.0583
S_abs5n20_1_H3	6436.13	7	0.0000
S_abs5n20_1_H6	14148.34	31	0.0000
S_abs5n20_2_H3	7003.41	17	0.0000
S_abs5n20_2_H6	15688.12	100	0.0000
S_abs5n20_3_H3	7711.03	21	0.0000
S_abs5n20_3_H6	17549.13	117	0.0000
S_abs5n20_4_H3	8344.42	31	0.0000
S_abs5n20_4_H6	19610.93	310	0.0337
S_abs5n20_5_H3	9085.7	39	0.0003

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs5n20_5_H6	21551.71	578	0.1096
S_abs1n25_1_L3	2695.11	6	0.0000
S_abs1n25_1_L6	6940.59	45	0.2968
S_abs1n25_2_L3	2987.47	14	0.0000
S_abs1n25_2_L6	7331.85	69	0.0315
S_abs1n25_3_L3	3357.57	19	0.0000
S_abs1n25_3_L6	8329.72	125	0.6566
S_abs1n25_4_L3	3803.92	21	0.0000
S_abs1n25_4_L6	9118.22	156	0.2892
S_abs1n25_5_L3	3949.39	23	0.0000
S_abs1n25_5_L6	10160.02	329	0.0783
S_abs2n25_1_L3	2854.01	11	0.0000
S_abs2n25_1_L6	7148.45	51	0.9131
S_abs2n25_2_L3	3340.88	18	0.0087
S_abs2n25_2_L6	8162.33	83	0.0075
S_abs2n25_3_L3	3791.53	25	0.0000
S_abs2n25_3_L6	9510.16	184	0.1028
S_abs2n25_4_L3	4340.93	33	0.0032
S_abs2n25_4_L6	10856.64	233	0.0166
S_abs2n25_5_L3	4849.87	49	0.0136
S_abs2n25_5_L6	12400.2	504	0.3263
S_abs3n25_1_L3	2871.43	6	0.0000
S_abs3n25_1_L6	7346.48	43	0.0000
S_abs3n25_2_L3	3292.85	13	0.0000
S_abs3n25_2_L6	8522.12	110	0.0000
S_abs3n25_3_L3	3889.69	19	0.0000
S_abs3n25_3_L6	10168.37	162	0.1538
S_abs3n25_4_L3	4508.65	23	0.0000
S_abs3n25_4_L6	11867.67	310	0.1914
S_abs3n25_5_L3	5050.35	28	0.0000
S_abs3n25_5_L6	13684.48	422	-0.0098
S_abs4n25_1_L3	2930.73	13	0.0000
S_abs4n25_1_L6	7324.08	45	0.0019
S_abs4n25_2_L3	3099.67	21	0.0000
S_abs4n25_2_L6	7746.19	98	0.0000
S_abs4n25_3_L3	3513.84	29	0.0752
S_abs4n25_3_L6	8669.85	181	0.0252

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs4n25_4_L3	3897.74	31	0.0218
S_abs4n25_4_L6	9669.72	264	0.1952
S_abs4n25_5_L3	4230.81	52	0.0000
S_abs4n25_5_L6	10565.62	250	0.1020
S_abs5n25_1_L3	2753	6	0.0000
S_abs5n25_1_L6	6880.98	40	0.2615
S_abs5n25_2_L3	3304.74	14	0.0000
S_abs5n25_2_L6	8322.11	119	0.0702
S_abs5n25_3_L3	3918.3	16	0.0000
S_abs5n25_3_L6	10048.22	145	0.2291
S_abs5n25_4_L3	4470.86	19	0.0000
S_abs5n25_4_L6	11988.78	322	0.1227
S_abs5n25_5_L3	5112.16	27	0.0000
S_abs5n25_5_L6	13870.28	537	0.1545
S_abs1n25_1_H3	6758.89	6	0.0000
S_abs1n25_1_H6	14171.87	38	0.1252
S_abs1n25_2_H3	7052.55	14	0.0000
S_abs1n25_2_H6	14634.35	85	0.0893
S_abs1n25_3_H3	7424.85	19	0.0000
S_abs1n25_3_H6	15583.56	155	0.1514
S_abs1n25_4_H3	7818.93	17	0.0000
S_abs1n25_4_H6	16402.12	160	0.1226
S_abs1n25_5_H3	8004.24	23	0.0000
S_abs1n25_5_H6	17460.07	317	0.0912
S_abs2n25_1_H3	7154.75	11	0.0000
S_abs2n25_1_H6	15049.86	38	0.2577
S_abs2n25_2_H3	7647.13	17	0.0000
S_abs2n25_2_H6	16092.18	99	0.0380
S_abs2n25_3_H3	8097.37	25	0.0000
S_abs2n25_3_H6	17459.55	235	0.0687
S_abs2n25_4_H3	8648.13	33	0.0000
S_abs2n25_4_H6	18794.83	327	-0.0498
S_abs2n25_5_H3	9152.05	49	0.0582
S_abs2n25_5_H6	20336.19	744	0.1947
S_abs3n25_1_H3	7607.39	6	0.0000
S_abs3n25_1_H6	16172.38	49	0.0003
S_abs3n25_2_H3	8029.89	14	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs3n25_2_H6	17364.93	130	0.0094
S_abs3n25_3_H3	8629.85	21	0.0000
S_abs3n25_3_H6	19013.47	196	0.0141
S_abs3n25_4_H3	9251.79	25	0.0002
S_abs3n25_4_H6	20704.33	307	0.0277
S_abs3n25_5_H3	9801.31	29	0.0004
S_abs3n25_5_H6	22552.94	467	-0.0206
S_abs4n25_1_H3	6981.14	14	0.0000
S_abs4n25_1_H6	14821.94	40	0.0088
S_abs4n25_2_H3	7159.54	20	0.0000
S_abs4n25_2_H6	15292.92	87	0.0206
S_abs4n25_3_H3	7577.28	34	0.0000
S_abs4n25_3_H6	16222.86	154	0.0178
S_abs4n25_4_H3	7968.55	39	0.0447
S_abs4n25_4_H6	17204.06	205	0.0223
S_abs4n25_5_H3	8247.54	48	0.0000
S_abs4n25_5_H6	18117.12	222	0.0261
S_abs5n25_1_H3	8058.61	6	0.0002
S_abs5n25_1_H6	16796.62	51	0.6357
S_abs5n25_2_H3	8610.25	15	0.0000
S_abs5n25_2_H6	18158.19	159	0.0926
S_abs5n25_3_H3	9228.11	17	0.0002
S_abs5n25_3_H6	19855.08	205	0.1396
S_abs5n25_4_H3	9783.17	21	0.0000
S_abs5n25_4_H6	21784.52	366	0.0280
S_abs5n25_5_H3	10428.37	27	0.0000
S_abs5n25_5_H6	23696.41	724	0.0901
S_abs1n30_1_L3	3189.6	8	0.0000
S_abs1n30_1_L6	7822.68	45	0.0385
S_abs1n30_2_L3	3565.6	19	0.0000
S_abs1n30_2_L6	8842.82	138	0.2160
S_abs1n30_3_L3	4013.46	24	0.0000
S_abs1n30_3_L6	10166.16	230	0.2106
S_abs1n30_4_L3	4482.4	36	0.0000
S_abs1n30_4_L6	11646.31	690	0.0293
S_abs1n30_5_L3	5076.56	51	0.0000
S_abs1n30_5_L6	13184.65	740	-0.1952

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs2n30_1_L3	3119.16	17	0.0000
S_abs2n30_1_L6	7442.69	50	0.1161
S_abs2n30_2_L3	3435.42	25	0.0000
S_abs2n30_2_L6	8210.54	99	0.0659
S_abs2n30_3_L3	3892.59	41	0.0000
S_abs2n30_3_L6	9222.97	125	0.0203
S_abs2n30_4_L3	4219.9	32	0.0045
S_abs2n30_4_L6	10361.43	142	0.2681
S_abs2n30_5_L3	4669.58	54	0.0000
S_abs2n30_5_L6	11550.97	401	0.1014
S_abs3n30_1_L3	3224.88	8	0.0000
S_abs3n30_1_L6	7938.67	63	0.5940
S_abs3n30_2_L3	3369.2	17	0.0000
S_abs3n30_2_L6	8203.85	84	0.0712
S_abs3n30_3_L3	3573.68	22	0.0000
S_abs3n30_3_L6	9060.89	115	0.1223
S_abs3n30_4_L3	3895.14	25	0.0005
S_abs3n30_4_L6	9993.92	155	0.2752
S_abs3n30_5_L3	4211.96	25	0.0005
S_abs3n30_5_L6	11050.1	313	0.2441
S_abs4n30_1_L3	3145.84	21	0.0045
S_abs4n30_1_L6	7344.57	45	0.0305
S_abs4n30_2_L3	3422.3	39	0.0000
S_abs4n30_2_L6	8076.54	154	0.1322
S_abs4n30_3_L3	3783.19	56	0.0103
S_abs4n30_3_L6	9084.68	161	0.1920
S_abs4n30_4_L3	4249.42	48	0.2011
S_abs4n30_4_L6	10250.97	195	0.2251
S_abs4n30_5_L3	4741.89	53	0.7466
S_abs4n30_5_L6	11447.75	300	-0.0148
S_abs5n30_1_L3	2719.35	8	0.0000
S_abs5n30_1_L6	7054.72	83	0.1873
S_abs5n30_2_L3	3020.61	20	0.0000
S_abs5n30_2_L6	7816.22	160	0.0073
S_abs5n30_3_L3	3393.39	23	0.0000
S_abs5n30_3_L6	8859.53	225	0.4285
S_abs5n30_4_L3	3796.45	30	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs5n30_4_L6	9972.4	475	-0.0392
S_abs5n30_5_L3	4156.91	43	0.0002
S_abs5n30_5_L6	11159.98	794	-0.3167
S_abs1n30_1_H3	9669.76	9	0.0000
S_abs1n30_1_H6	20584.48	51	0.4372
S_abs1n30_2_H3	10052.78	22	0.0000
S_abs1n30_2_H6	21525.76	191	0.0604
S_abs1n30_3_H3	10511.8	26	0.0000
S_abs1n30_3_H6	22819.57	275	0.0330
S_abs1n30_4_H3	10996.1	32	0.0000
S_abs1n30_4_H6	24318.66	709	0.0977
S_abs1n30_5_H3	11593.4	55	0.0000
S_abs1n30_5_H6	25879.26	656	0.0749
S_abs2n30_1_H3	8839.33	18	0.0000
S_abs2n30_1_H6	17928.03	54	0.0639
S_abs2n30_2_H3	9156.11	26	0.0000
S_abs2n30_2_H6	18751.08	112	0.1890
S_abs2n30_3_H3	9584.47	34	0.0027
S_abs2n30_3_H6	19745.43	172	0.0176
S_abs2n30_4_H3	9931.93	38	0.0608
S_abs2n30_4_H6	20862.56	191	0.1508
S_abs2n30_5_H3	10373.61	56	0.1040
S_abs2n30_5_H6	22083.74	357	0.1070
S_abs3n30_1_H3	9671.61	8	0.0000
S_abs3n30_1_H6	20774.64	40	0.5061
S_abs3n30_2_H3	9826.31	19	0.0000
S_abs3n30_2_H6	21079.37	103	0.4262
S_abs3n30_3_H3	10037.21	24	0.0001
S_abs3n30_3_H6	21927.57	167	0.2847
S_abs3n30_4_H3	10372.87	29	0.0021
S_abs3n30_4_H6	22821.37	237	0.1340
S_abs3n30_5_H3	10678.31	26	0.0015
S_abs3n30_5_H6	23829.92	288	0.1426
S_abs4n30_1_H3	7947.52	22	0.0143
S_abs4n30_1_H6	15952.99	52	0.1336
S_abs4n30_2_H3	8223.26	34	0.0000
S_abs4n30_2_H6	16710.5	134	0.0803

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs4n30_3_H3	8546.37	45	0.0000
S_abs4n30_3_H6	17716.48	222	0.2759
S_abs4n30_4_H3	9035.54	62	0.0000
S_abs4n30_4_H6	18890.6	233	0.1934
S_abs4n30_5_H3	9542.94	45	0.5273
S_abs4n30_5_H6	20088.36	367	-0.0109
S_abs5n30_1_H3	7826.27	8	0.0000
S_abs5n30_1_H6	16904.58	59	0.2346
S_abs5n30_2_H3	8115.83	21	0.0000
S_abs5n30_2_H6	17680.26	198	0.0050
S_abs5n30_3_H3	8502.79	25	0.0014
S_abs5n30_3_H6	18718.67	295	0.2531
S_abs5n30_4_H3	8910.03	32	0.0000
S_abs5n30_4_H6	19816.54	443	0.0239
S_abs5n30_5_H3	9272.27	46	0.0000
S_abs5n30_5_H6	21020.5	896	-0.0482
S_abs1n35_1_L3	3122.83	10	0.0000
S_abs1n35_2_L3	3374.61	31	0.0000
S_abs1n35_3_L3	3857.31	31	0.0000
S_abs1n35_4_L3	4145.67	34	0.0000
S_abs1n35_5_L3	4541.69	47	0.0000
S_abs2n35_1_L3	3349.09	21	0.0000
S_abs2n35_2_L3	3664.9	25	0.0797
S_abs2n35_3_L3	4031.38	36	0.0563
S_abs2n35_4_L3	4446.31	60	0.3324
S_abs2n35_5_L3	4816.58	48	0.0546
S_abs3n35_1_L3	3532.48	10	0.0000
S_abs3n35_2_L3	3904.76	23	0.0000
S_abs3n35_3_L3	4446.56	29	0.0000
S_abs3n35_4_L3	5002.64	36	0.0000
S_abs3n35_5_L3	5638.84	36	0.0000
S_abs4n35_1_L3	3032.23	11	0.0000
S_abs4n35_2_L3	3244.21	27	0.0000
S_abs4n35_3_L3	3814.17	35	0.0063
S_abs4n35_4_L3	4305.25	38	0.0005
S_abs4n35_5_L3	4667.55	46	0.0000
S_abs5n35_1_L3	3104.54	11	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs5n35_2_L3	3382.92	24	0.0000
S_abs5n35_3_L3	3823.98	36	0.0000
S_abs5n35_4_L3	4224.46	40	0.0000
S_abs5n35_5_L3	4624.38	46	0.0000
S_abs1n35_1_H3	9385.82	11	0.0000
S_abs1n35_2_H3	9648.24	28	0.0000
S_abs1n35_3_H3	10121.9	30	0.0000
S_abs1n35_4_H3	10424.24	33	0.0000
S_abs1n35_5_H3	10831.94	49	0.0004
S_abs2n35_1_H3	8546.09	24	0.0000
S_abs2n35_2_H3	8864.09	26	0.2171
S_abs2n35_3_H3	9218.81	44	0.1548
S_abs2n35_4_H3	9628.19	60	0.1925
S_abs2n35_5_H3	10018.38	68	0.1656
S_abs3n35_1_H3	10963.77	11	0.0000
S_abs3n35_2_H3	11334.79	23	0.0000
S_abs3n35_3_H3	11881.77	28	0.0000
S_abs3n35_4_H3	12462.95	36	0.0003
S_abs3n35_5_H3	13075.99	41	0.0000
S_abs4n35_1_H3	8357.92	10	0.0000
S_abs4n35_2_H3	8572.64	25	0.0002
S_abs4n35_3_H3	9143.44	37	0.0004
S_abs4n35_4_H3	9632.54	42	0.0023
S_abs4n35_5_H3	10002.08	49	0.0000
S_abs5n35_1_H3	8733.79	11	0.0002
S_abs5n35_2_H3	9013.01	25	0.0004
S_abs5n35_3_H3	9454.31	35	0.0004
S_abs5n35_4_H3	9856.11	42	0.0004
S_abs5n35_5_H3	10256.37	47	0.0000
S_abs1n40_1_L3	3435.06	13	0.0000
S_abs1n40_2_L3	3725.38	28	0.0000
S_abs1n40_3_L3	4265.76	40	0.0000
S_abs1n40_4_L3	4766.5	48	0.0000
S_abs1n40_5_L3	5287.04	73	0.0000
S_abs2n40_1_L3	3619.71	32	0.0000
S_abs2n40_2_L3	3967.69	59	0.0000
S_abs2n40_3_L3	4331.99	47	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs2n40_4_L3	4851.57	51	0.0000
S_abs2n40_5_L3	5188.33	83	0.8867
S_abs3n40_1_L3	3598.72	13	0.0000
S_abs3n40_2_L3	3829.68	26	0.0000
S_abs3n40_3_L3	4034.34	33	0.0000
S_abs3n40_4_L3	4366.4	40	0.0000
S_abs3n40_5_L3	4641.06	47	0.0000
S_abs4n40_1_L3	3318.31	14	0.0000
S_abs4n40_2_L3	3520.29	27	0.0000
S_abs4n40_3_L3	3777.89	36	0.0000
S_abs4n40_4_L3	4256.29	50	0.0000
S_abs4n40_5_L3	4561.39	52	0.0000
S_abs5n40_1_L3	3310.77	21	0.0000
S_abs5n40_2_L3	3630.77	35	0.0000
S_abs5n40_3_L3	4074.85	41	0.0000
S_abs5n40_4_L3	4486.05	85	0.0000
S_abs5n40_5_L3	4781.31	59	0.0000
S_abs1n40_1_H3	10657.28	13	0.0002
S_abs1n40_2_H3	10952.76	30	0.0000
S_abs1n40_3_H3	11516.48	41	0.0002
S_abs1n40_4_H3	12014.08	48	0.0002
S_abs1n40_5_H3	12546.9	77	0.0003
S_abs2n40_1_H3	9202.74	35	0.0000
S_abs2n40_2_H3	9550.62	53	0.0003
S_abs2n40_3_H3	9900.51	46	0.0002
S_abs2n40_4_H3	10444.53	68	0.1321
S_abs2n40_5_H3	10770.99	96	0.5330
S_abs3n40_1_H3	10855.53	14	0.0004
S_abs3n40_2_H3	11086.69	29	0.0000
S_abs3n40_3_H3	11296.95	34	0.0023
S_abs3n40_4_H3	11629.51	44	0.0002
S_abs3n40_5_H3	11909.45	52	0.0000
S_abs4n40_1_H3	9208.3	26	0.0000
S_abs4n40_2_H3	9445.33	27	0.0002
S_abs4n40_3_H3	9703.89	37	0.0000
S_abs4n40_4_H3	10184.19	53	0.0000
S_abs4n40_5_H3	10491.43	56	0.0002

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs5n40_1_H3	10403.09	21	0.0000
S_abs5n40_2_H3	10723.27	33	0.0000
S_abs5n40_3_H3	11168.05	57	0.0004
S_abs5n40_4_H3	11566.06	51	0.0004
S_abs5n40_5_H3	11876.21	54	0.0000
S_abs1n45_1_L3	3667.63	16	0.0000
S_abs1n45_2_L3	3794.63	33	0.0000
S_abs1n45_3_L3	4254.07	48	0.0000
S_abs1n45_4_L3	4706.25	69	0.0000
S_abs1n45_5_L3	5186.87	75	0.0010
S_abs2n45_1_L3	3438.38	16	0.0000
S_abs2n45_2_L3	3919.34	38	0.0000
S_abs2n45_3_L3	4480.62	61	0.0000
S_abs2n45_4_L3	5057.04	61	0.0000
S_abs2n45_5_L3	5707.6	67	0.0000
S_abs3n45_1_L3	3661.2	17	0.0000
S_abs3n45_2_L3	3814.88	36	0.0000
S_abs3n45_3_L3	3960.58	37	0.0000
S_abs3n45_4_L3	4207.66	51	0.0000
S_abs3n45_5_L3	4500.78	66	0.0000
S_abs4n45_1_L3	3730.42	25	0.0000
S_abs4n45_2_L3	4189.42	42	0.0000
S_abs4n45_3_L3	4653.32	51	0.0000
S_abs4n45_4_L3	5206.32	57	0.0000
S_abs4n45_5_L3	5801.84	97	0.0000
S_abs5n45_1_L3	3444.23	28	0.0168
S_abs5n45_2_L3	3638.23	40	0.0038
S_abs5n45_3_L3	3866.85	62	0.0000
S_abs5n45_4_L3	4149.73	93	0.0082
S_abs5n45_5_L3	4340.75	88	0.0041
S_abs1n45_1_H3	11319.47	16	0.0000
S_abs1n45_2_H3	11447.71	32	0.0003
S_abs1n45_3_H3	11911.35	46	0.0000
S_abs1n45_4_H3	12362.25	68	0.0000
S_abs1n45_5_H3	12844.95	70	0.0037
S_abs2n45_1_H3	10513.61	17	0.0002
S_abs2n45_2_H3	11012.27	34	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs2n45_3_H3	11582.39	63	0.0000
S_abs2n45_4_H3	12152.01	58	0.0002
S_abs2n45_5_H3	12805.13	69	0.0003
S_abs3n45_1_H3	11762.18	17	0.0000
S_abs3n45_2_H3	11926.76	35	0.0000
S_abs3n45_3_H3	12088.54	37	0.0003
S_abs3n45_4_H3	12370.24	50	0.0003
S_abs3n45_5_H3	12648.14	53	0.0000
S_abs4n45_1_H3	10936.09	25	0.0000
S_abs4n45_2_H3	11395.09	35	0.0000
S_abs4n45_3_H3	11838.33	45	0.0020
S_abs4n45_4_H3	12391.57	60	0.0000
S_abs4n45_5_H3	12985.45	113	0.0000
S_abs5n45_1_H3	10829.11	32	0.0000
S_abs5n45_2_H3	11034.21	58	0.0000
S_abs5n45_3_H3	11254.05	71	0.0000
S_abs5n45_4_H3	11538.27	81	0.0055
S_abs5n45_5_H3	11728.25	106	0.0000
S_abs1n50_1_L3	3752.25	39	0.0000
S_abs1n50_2_L3	4272.23	86	0.0000
S_abs1n50_3_L3	4907.25	84	0.0000
S_abs1n50_4_L3	5558.51	90	0.0000
S_abs1n50_5_L3	6190.25	137	0.0000
S_abs2n50_1_L3	4220.32	41	0.0000
S_abs2n50_2_L3	4550.28	84	0.0000
S_abs2n50_3_L3	5237.83	104	0.0000
S_abs2n50_4_L3	5873.24	163	0.0317
S_abs2n50_5_L3	6417.69	172	0.0000
S_abs3n50_1_L3	4147.66	47	0.0000
S_abs3n50_2_L3	4401.96	51	0.0000
S_abs3n50_3_L3	4809.53	76	0.0081
S_abs3n50_4_L3	5200	91	0.0000
S_abs3n50_5_L3	5709.54	108	0.0231
S_abs4n50_1_L3	4062.84	19	0.0000
S_abs4n50_2_L3	4319.84	45	0.0000
S_abs4n50_3_L3	4943.44	57	0.0000
S_abs4n50_4_L3	5610.62	91	0.0000

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
S_abs4n50_5_L3	6311.76	132	0.0000
S_abs5n50_1_L3	3897.11	41	0.0000
S_abs5n50_2_L3	4246.17	46	0.0000
S_abs5n50_3_L3	4767	74	0.0000
S_abs5n50_4_L3	5411.59	99	0.0000
S_abs5n50_5_L3	5912.51	105	0.0003
S_abs1n50_1_H3	11607.98	39	0.0000
S_abs1n50_2_H3	12127.78	57	0.0000
S_abs1n50_3_H3	12763.28	84	0.0000
S_abs1n50_4_H3	13397.98	105	0.0000
S_abs1n50_5_H3	14048.2	134	0.0000
S_abs2n50_1_H3	12092.36	43	0.0000
S_abs2n50_2_H3	12433.66	70	0.0003
S_abs2n50_3_H3	13118.66	86	0.0026
S_abs2n50_4_H3	13731.62	144	0.0000
S_abs2n50_5_H3	14298.34	149	0.0001
S_abs3n50_1_H3	12240.8	46	0.0000
S_abs3n50_2_H3	12482.26	66	0.0004
S_abs3n50_3_H3	12902.06	84	0.0004
S_abs3n50_4_H3	13280.3	82	0.0000
S_abs3n50_5_H3	13796.8	115	0.0000
S_abs4n50_1_H3	13126.73	19	0.0000
S_abs4n50_2_H3	13384.85	48	0.0001
S_abs4n50_3_H3	14011.35	63	0.0016
S_abs4n50_4_H3	14678.79	89	0.0000
S_abs4n50_5_H3	15414.22	143	0.1780
S_abs5n50_1_H3	12469.12	46	0.0000
S_abs5n50_2_H3	12818.18	48	0.0002
S_abs5n50_3_H3	13327.57	80	0.0002
S_abs5n50_4_H3	13984.44	113	0.0000
S_abs5n50_5_H3	14486.24	128	0.0013
L_abs10n50_1_L	9377.16	195	0.7366
L_abs10n50_2_L	10507.27	597	0.4339
L_abs10n50_3_L	12007.56	565	-0.0366
L_abs10n50_4_L	13777.09	1158	-0.3155
L_abs10n50_5_L	15514.27	1265	-0.0420
L_abs1n50_1_L	9750.95	205	1.1143

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
L_abs1n50_2_L	10968.26	504	0.1456
L_abs1n50_3_L	12617.44	593	-0.1195
L_abs1n50_4_L	14555.65	1023	0.0901
L_abs1n50_5_L	16577.49	1467	-0.1717
L_abs2n50_1_L	10355.52	146	0.2980
L_abs2n50_2_L	11383.14	702	0.0374
L_abs2n50_3_L	13166.51	612	-0.3033
L_abs2n50_4_L	14976.14	1047	0.0490
L_abs2n50_5_L	17052.2	1479	0.1224
L_abs3n50_1_L	10209.14	115	0.1130
L_abs3n50_2_L	10738.69	379	0.1445
L_abs3n50_3_L	11860.65	642	0.5632
L_abs3n50_4_L	13131.83	1195	0.3316
L_abs3n50_5_L	14510.78	1305	0.1201
L_abs4n50_1_L	10176.86	118	0.0604
L_abs4n50_2_L	10844.36	338	0.4098
L_abs4n50_3_L	12677.97	822	0.1701
L_abs4n50_4_L	14684.79	1269	-0.1938
L_abs4n50_5_L	16627.7	1454	0.0741
L_abs5n50_1_L	9819.63	176	0.1595
L_abs5n50_2_L	10524.79	421	0.0075
L_abs5n50_3_L	12092.04	929	-0.0688
L_abs5n50_4_L	13764.79	1107	0.0353
L_abs5n50_5_L	15428.55	1388	0.1394
L_abs6n50_1_L	9820.33	172	0.1873
L_abs6n50_2_L	11052.86	554	0.5244
L_abs6n50_3_L	12932.13	669	0.4549
L_abs6n50_4_L	14721.44	1304	-0.1161
L_abs6n50_5_L	16650.6	1395	0.1880
L_abs7n50_1_L	9834.04	156	1.6827
L_abs7n50_2_L	11120.19	475	0.1142
L_abs7n50_3_L	12953.76	606	-0.0465
L_abs7n50_4_L	14858.09	1150	0.3906
L_abs7n50_5_L	16853.69	1421	-0.2104
L_abs8n50_1_L	10126.47	135	0.7647
L_abs8n50_2_L	11710.84	910	0.0423
L_abs8n50_3_L	13960.57	964	-0.1274

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
L_abs8n50_4_L	16290.7	1332	-0.1057
L_abs8n50_5_L	18625.53	1489	-0.1250
L_abs9n50_1_L	9698.21	187	0.0549
L_abs9n50_2_L	10806.51	438	0.5069
L_abs9n50_3_L	12240.92	572	-0.0879
L_abs9n50_4_L	13752.97	1009	-0.2012
L_abs9n50_5_L	15426.67	1455	0.2533
L_abs10n50_1_H	28055.22	103	0.7563
L_abs10n50_2_H	29029.74	555	0.0591
L_abs10n50_3_H	30547.72	659	0.0902
L_abs10n50_4_H	32316.17	1005	0.0696
L_abs10n50_5_H	34046.25	1159	-0.0771
L_abs1n50_1_H	27114.62	108	0.6668
L_abs1n50_2_H	28184.75	366	0.0286
L_abs1n50_3_H	29909.22	602	0.1532
L_abs1n50_4_H	31800.75	1416	0.0476
L_abs1n50_5_H	33809.32	1509	0.0474
L_abs2n50_1_H	26773.28	130	0.1249
L_abs2n50_2_H	27839.85	552	0.0910
L_abs2n50_3_H	29635.36	947	-0.1231
L_abs2n50_4_H	31425.86	1185	0.0251
L_abs2n50_5_H	33451.61	1509	-0.1444
L_abs3n50_1_H	26684.01	102	0.0871
L_abs3n50_2_H	27288.12	453	0.2257
L_abs3n50_3_H	28372.95	700	0.1560
L_abs3n50_4_H	29656.01	1076	0.0126
L_abs3n50_5_H	31054.14	1480	0.0904
L_abs4n50_1_H	28162.64	201	0.1099
L_abs4n50_2_H	28919.15	289	0.1685
L_abs4n50_3_H	30722.52	1065	0.1336
L_abs4n50_4_H	32774.36	1422	0.1427
L_abs4n50_5_H	34705.04	1493	0.3013
L_abs5n50_1_H	26156.37	129	0.0140
L_abs5n50_2_H	26952.86	432	0.0212
L_abs5n50_3_H	28550.71	711	0.1166
L_abs5n50_4_H	30224.22	1204	0.0107
L_abs5n50_5_H	31837.87	1508	-0.2540

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
L_abs6n50_1_H	28625.32	126	0.7103
L_abs6n50_2_H	29736.64	715	0.3273
L_abs6n50_3_H	31707.11	1010	0.5120
L_abs6n50_4_H	33502.18	1221	0.1827
L_abs6n50_5_H	35368.88	1483	0.1462
L_abs7n50_1_H	26782.48	117	0.6676
L_abs7n50_2_H	28041.01	452	0.0207
L_abs7n50_3_H	29902.7	634	-0.1378
L_abs7n50_4_H	31856.15	1454	0.1289
L_abs7n50_5_H	33824.93	1509	0.1097
L_abs8n50_1_H	23458.38	151	0.1350
L_abs8n50_2_H	25155.51	767	0.1222
L_abs8n50_3_H	27406.82	1158	0.0056
L_abs8n50_4_H	29723.63	1451	-0.0418
L_abs8n50_5_H	32055.55	1509	-0.1702
L_abs9n50_1_H	27042.54	236	0.0739
L_abs9n50_2_H	28054.5	373	0.1525
L_abs9n50_3_H	29530.51	659	-0.0998
L_abs9n50_4_H	31080.02	900	0.2116
L_abs9n50_5_H	32703.7	1385	0.1242
L_abs10n100_1_L	14846.24	539	0.2933
L_abs10n100_2_L	15038.89	1201	0.6112
L_abs10n100_3_L	15736.18	1439	-0.4416
L_abs10n100_4_L	16577.94	1509	0.1590
L_abs10n100_5_L	17624.9	1509	-0.2128
L_abs1n100_1_L	15116.13	795	0.9353
L_abs1n100_2_L	15453.69	1212	0.8402
L_abs1n100_3_L	16274.97	1509	-0.6000
L_abs1n100_4_L	17252.21	1509	-0.3284
L_abs1n100_5_L	18482.25	1509	-0.5198
L_abs2n100_1_L	14069.02	616	0.9612
L_abs2n100_2_L	14412.57	1472	0.6895
L_abs2n100_3_L	15120.33	1446	0.0197
L_abs2n100_4_L	16084.8	1497	-0.4403
L_abs2n100_5_L	17089.3	1509	-0.0608
L_abs3n100_1_L	14954.45	610	0.7219
L_abs3n100_2_L	15394.12	1214	0.4697

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
L_abs3n100_3_L	16506.35	1454	-0.6417
L_abs3n100_4_L	18075.71	1509	-0.7411
L_abs3n100_5_L	19689.31	1509	-0.7186
L_abs4n100_1_L	14163.24	742	0.8456
L_abs4n100_2_L	14421.96	1350	0.7969
L_abs4n100_3_L	14985.16	1392	0.4777
L_abs4n100_4_L	15710.16	1494	-0.3889
L_abs4n100_5_L	16653.93	1504	-0.3901
L_abs5n100_1_L	14637.52	611	0.6943
L_abs5n100_2_L	15014.21	1356	0.1238
L_abs5n100_3_L	15677.77	1493	0.2796
L_abs5n100_4_L	16565.77	1509	-0.6336
L_abs5n100_5_L	17562.72	1509	-0.6658
L_abs6n100_1_L	14711.54	449	0.8684
L_abs6n100_2_L	15192.42	1322	-0.0374
L_abs6n100_3_L	16619.74	1509	-0.4792
L_abs6n100_4_L	18184	1509	-0.3097
L_abs6n100_5_L	19954.69	1509	-0.1640
L_abs7n100_1_L	14808.08	475	0.9084
L_abs7n100_2_L	15099.67	953	0.5119
L_abs7n100_3_L	15943.06	1466	-0.4627
L_abs7n100_4_L	17084.74	1509	0.2951
L_abs7n100_5_L	18368.9	1509	-0.0009
L_abs8n100_1_L	14441.7	522	0.4572
L_abs8n100_2_L	15239.27	1493	-0.1817
L_abs8n100_3_L	16573.62	1509	-0.4062
L_abs8n100_4_L	18281.52	1509	0.4260
L_abs8n100_5_L	20032.38	1509	0.2929
L_abs9n100_1_L	14968.9	636	0.6979
L_abs9n100_2_L	15516.69	989	0.5102
L_abs9n100_3_L	17096.16	1509	-0.2532
L_abs9n100_4_L	18852.32	1509	0.2888
L_abs9n100_5_L	20762.76	1509	-0.7261
L_abs10n100_1_H	49935.7	893	0.4249
L_abs10n100_2_H	50176.1	1502	0.3526
L_abs10n100_3_H	50904.67	1509	-0.0159
L_abs10n100_4_H	51743.09	1509	-0.0993

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
L_abs10n100_5_H	52780.31	1509	-0.3441
L_abs1n100_1_H	50867.13	871	0.3594
L_abs1n100_2_H	51316.57	1500	0.2353
L_abs1n100_3_H	52161.89	1509	-0.0888
L_abs1n100_4_H	53201.1	1509	0.0934
L_abs1n100_5_H	54442.18	1509	0.0010
L_abs2n100_1_H	47342.32	564	0.4889
L_abs2n100_2_H	47700.71	1459	0.1987
L_abs2n100_3_H	48408.54	1509	-0.0025
L_abs2n100_4_H	49359.9	1509	-0.0094
L_abs2n100_5_H	50433.59	1509	-0.0205
L_abs3n100_1_H	51649.11	768	0.3414
L_abs3n100_2_H	52097.5	1504	0.2015
L_abs3n100_3_H	53255.98	1509	-0.1542
L_abs3n100_4_H	54866.1	1509	0.0324
L_abs3n100_5_H	56556.23	1509	-0.0899
L_abs4n100_1_H	45951.6	748	0.6924
L_abs4n100_2_H	46127.06	1424	0.1769
L_abs4n100_3_H	46759.13	1447	0.3108
L_abs4n100_4_H	47476.71	1502	-0.0794
L_abs4n100_5_H	48383.58	1509	-0.0286
L_abs5n100_1_H	51323.91	747	0.4702
L_abs5n100_2_H	51606.25	1509	0.0560
L_abs5n100_3_H	52337.1	1493	-0.0225
L_abs5n100_4_H	53232.69	1509	-0.0170
L_abs5n100_5_H	54290.72	1509	0.0125
L_abs6n100_1_H	48861.45	754	0.5100
L_abs6n100_2_H	49527.21	1509	0.1567
L_abs6n100_3_H	50921.73	1509	-0.0380
L_abs6n100_4_H	52494.66	1509	-0.1905
L_abs6n100_5_H	54299.43	1509	-0.0358
L_abs7n100_1_H	49634.86	633	0.2045
L_abs7n100_2_H	49955.36	1414	-0.0420
L_abs7n100_3_H	50909.65	1504	0.2886
L_abs7n100_4_H	52135.39	1509	0.3021
L_abs7n100_5_H	53415.96	1509	0.0864
L_abs8n100_1_H	48796.59	742	0.3802

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
L_abs8n100_2_H	49595.03	1509	0.1010
L_abs8n100_3_H	50962.8	1509	0.0266
L_abs8n100_4_H	52713.51	1509	0.0515
L_abs8n100_5_H	54630.83	1509	0.3278
L_abs9n100_1_H	51657.72	685	0.0809
L_abs9n100_2_H	52377.78	1503	0.2282
L_abs9n100_3_H	54037.05	1509	0.0601
L_abs9n100_4_H	55817.33	1509	-0.0335
L_abs9n100_5_H	57678.44	1509	-0.0535
L_abs10n200_1_L	22215.3	1509	0.9297
L_abs10n200_2_L	22558.06	1509	0.6332
L_abs10n200_3_L	23325.55	1509	0.5368
L_abs10n200_4_L	24710.69	1509	0.4324
L_abs10n200_5_L	26534.67	1509	0.5074
L_abs1n200_1_L	22971.4	1509	0.6306
L_abs1n200_2_L	23266.47	1509	1.2826
L_abs1n200_3_L	23763.23	1509	0.7184
L_abs1n200_4_L	24412.55	1509	1.0251
L_abs1n200_5_L	25325.97	1509	0.9782
L_abs2n200_1_L	23182.44	1509	1.3721
L_abs2n200_2_L	23425.82	1509	0.5269
L_abs2n200_3_L	23935.46	1509	0.0896
L_abs2n200_4_L	24712.5	1509	0.2213
L_abs2n200_5_L	25849.42	1509	0.9659
L_abs3n200_1_L	22636.35	1509	0.7237
L_abs3n200_2_L	23119.04	1509	1.1059
L_abs3n200_3_L	23861.17	1509	0.2323
L_abs3n200_4_L	25497.51	1509	1.1723
L_abs3n200_5_L	27228.86	1509	2.0463
L_abs4n200_1_L	22953.52	1509	0.3833
L_abs4n200_2_L	23221	1509	0.5606
L_abs4n200_3_L	23690.24	1509	0.2840
L_abs4n200_4_L	24355.39	1509	0.6128
L_abs4n200_5_L	25129.67	1509	0.4024
L_abs5n200_1_L	22808.95	1509	0.1528
L_abs5n200_2_L	23128.71	1509	0.3578
L_abs5n200_3_L	23862.3	1509	0.1651

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
L_abs5n200_4_L	25398.33	1509	1.9610
L_abs5n200_5_L	26506.63	1509	1.1007
L_abs6n200_1_L	22325.33	1509	0.6392
L_abs6n200_2_L	22843.55	1509	0.6660
L_abs6n200_3_L	23698.16	1509	0.9553
L_abs6n200_4_L	24848.48	1509	1.1276
L_abs6n200_5_L	26232.53	1509	1.2573
L_abs7n200_1_L	22299.5	1509	-0.1932
L_abs7n200_2_L	22751.56	1509	1.6773
L_abs7n200_3_L	23391.97	1509	1.1604
L_abs7n200_4_L	24401.15	1509	1.0013
L_abs7n200_5_L	25611.29	1509	1.2534
L_abs8n200_1_L	22246.1	1509	1.9558
L_abs8n200_2_L	22343.26	1509	1.0489
L_abs8n200_3_L	22986.71	1509	0.8864
L_abs8n200_4_L	23735.48	1509	0.7160
L_abs8n200_5_L	24740.34	1509	1.2673
L_abs9n200_1_L	22581.87	1509	1.0998
L_abs9n200_2_L	23076.43	1509	0.8155
L_abs9n200_3_L	23764.74	1509	0.2873
L_abs9n200_4_L	25103.88	1509	0.3237
L_abs9n200_5_L	26512.15	1509	0.6262
L_abs10n200_1_H	95485.52	1509	0.6259
L_abs10n200_2_H	95985.23	1509	0.6834
L_abs10n200_3_H	96905.63	1509	0.5948
L_abs10n200_4_H	98576.92	1509	0.9930
L_abs10n200_5_H	100345.05	1509	1.1536
L_abs1n200_1_H	97325.79	1509	0.4991
L_abs1n200_2_H	97732.29	1509	0.8476
L_abs1n200_3_H	98055.15	1509	0.5892
L_abs1n200_4_H	98756.59	1509	0.5690
L_abs1n200_5_H	99828.93	1509	0.5825
L_abs2n200_1_H	98545.43	1509	0.5502
L_abs2n200_2_H	98926.72	1509	0.4581
L_abs2n200_3_H	99642.05	1509	0.4824
L_abs2n200_4_H	100356.17	1509	0.4339
L_abs2n200_5_H	101672.76	1509	0.8342

Table C.1: Detailed results for all IRP instances (*cont...*)

Instance	UB	Time(s)	Primal Gap
L_abs3n200_1_H	94431.02	1509	0.6598
L_abs3n200_2_H	94958.4	1509	0.5745
L_abs3n200_3_H	96057.22	1509	0.7814
L_abs3n200_4_H	97650.87	1509	0.9286
L_abs3n200_5_H	99563.17	1509	1.3165
L_abs4n200_1_H	95495.51	1509	0.2718
L_abs4n200_2_H	95859.49	1509	0.4873
L_abs4n200_3_H	96243.81	1509	0.1928
L_abs4n200_4_H	97058.98	1509	0.4589
L_abs4n200_5_H	98048.75	1509	0.4191
L_abs5n200_1_H	95510.56	1509	0.2985
L_abs5n200_2_H	96074.91	1509	0.7642
L_abs5n200_3_H	96582.51	1509	0.3926
L_abs5n200_4_H	98025.19	1509	0.6220
L_abs5n200_5_H	99809.65	1509	1.1701
L_abs6n200_1_H	95687.93	1509	0.8300
L_abs6n200_2_H	96231.57	1509	0.7999
L_abs6n200_3_H	97120.95	1509	0.7793
L_abs6n200_4_H	98426.29	1509	0.9212
L_abs6n200_5_H	100103.39	1509	1.0707
L_abs7n200_1_H	85880.01	1509	0.1924
L_abs7n200_2_H	86321.53	1509	0.6229
L_abs7n200_3_H	86995.29	1509	0.5455
L_abs7n200_4_H	88196.43	1509	0.6649
L_abs7n200_5_H	89756	1509	1.0678
L_abs8n200_1_H	89734.82	1509	0.6504
L_abs8n200_2_H	90260.5	1509	0.9195
L_abs8n200_3_H	90669.73	1509	0.5396
L_abs8n200_4_H	91584.25	1509	0.7404
L_abs8n200_5_H	92627.21	1509	0.8175
L_abs9n200_1_H	91861.39	1509	0.5428
L_abs9n200_2_H	92604.73	1509	0.7207
L_abs9n200_3_H	93108.56	1509	0.3887
L_abs9n200_4_H	94655.08	1509	0.7150
L_abs9n200_5_H	96201.49	1509	0.5281

Table C.1: Detailed results for all IRP instances