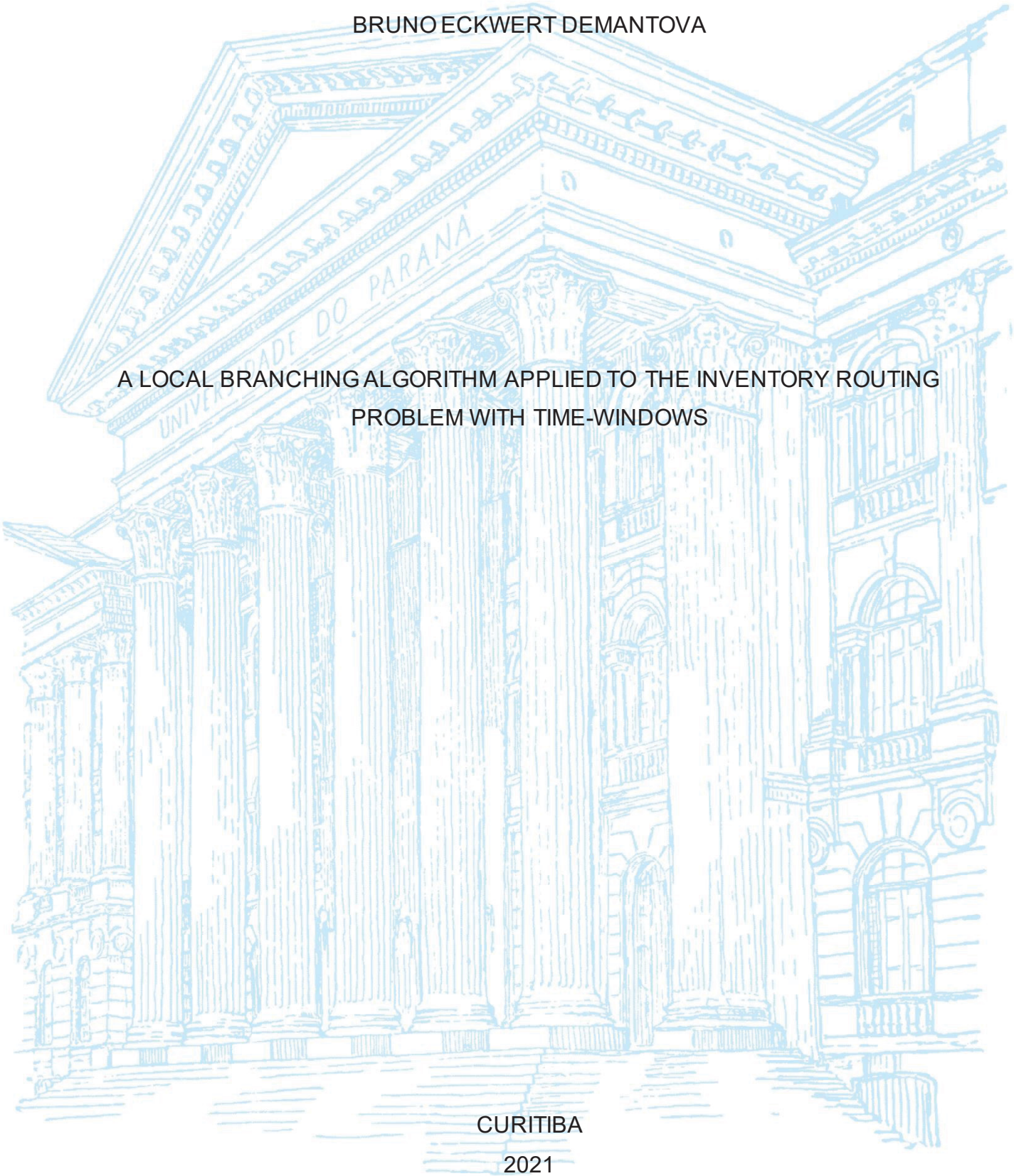
UNIVERSIDADE FEDERAL DO PARANÁ

BRUNO ECKWERT DEMANTOVA



A LOCAL BRANCHING ALGORITHM APPLIED TO THE INVENTORY ROUTING PROBLEM WITH TIME-WINDOWS

CURITIBA

2021

BRUNO ECKWERT DEMANTOVA

A LOCAL BRANCHING ALGORITHM APPLIED TO THE INVENTORY ROUTING PROBLEM WITH TIME-WINDOWS

Dissertação de mestrado apresentada ao curso de Pós-Graduação em Gestão de Organizações, Liderança e Decisão, Setor de Ciências Sociais Aplicadas, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Gestão de Organizações, Liderança e Decisão.

Orientador: Prof. Dr. Cassius Tadeu Scarpin

CURITIBA

2021

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DE CIÊNCIAS SOCIAIS APLICADAS – SIBI/UFPR COM DADOS FORNECIDOS PELO(A) AUTOR(A)
Bibliotecária: Maria Lidiane Herculano Graciosa – CRB 9/2018

Demantova, Bruno Eckwert
A local branching algorithm applied to the inventory routing problem with time-windows / Bruno Eckwert Demantova. - 2021.
61 p.

Dissertação (Mestrado) - Universidade Federal do Paraná. Programa de Pós-Graduação em Gestão de Organizações, Liderança e Decisão, do Setor de Ciências Sociais Aplicadas.

Orientador: Cassius Tadeu Scarpin.

Defesa: Curitiba, 2021.

1. Controle de estoque. 2. Roteamento. 3. Processos de ramificação. 4. Programação (Matemática). 5. Modelos matemáticos. I. Universidade Federal do Paraná. Setor de Ciências Sociais Aplicadas. Programa de Pós-Graduação em Gestão de Organizações, Liderança e Decisão. II. Scarpin, Cassius Tadeu. III. Título.

CDD 355.62132

MINISTÉRIO DA EDUCAÇÃO
SETOR DE CIÊNCIAS SOCIAIS E APLICADAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO GESTÃO DE ORGANIZAÇÕES, LIDERANÇA E DECISÃO - 40001016172P9

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação GESTÃO DE ORGANIZAÇÕES, LIDERANÇA E DECISÃO da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **BRUNO ECKWERT DEMANTOVA** intituiada: **A LOCAL BRANCHING ALGORITHM APPLIED TO THE INVENTORY ROUTING PROBLEM WITH TIME-WINDOWS**, sob orientação do Prof. Dr. CASSIUS TADEU SCARPIN, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.
A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 26 de Novembro de 2021.

Assinatura Eletrônica
29/11/2021 12:08:32.0
CASSIUS TADEU SCARPIN
Presidente da Banca Examinadora

Assinatura Eletrônica
29/11/2021 14:08:21.0
LEANDRO MAGATÃO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
29/11/2021 18:06:13.0
CLEDER MARCOS SCHENEKEMBERG
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
29/11/2021 12:17:09.0
THIAGO ANDRÉ GUIMARÃES
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Dedico este trabalho à todas as pessoas que passaram de forma significativa em minha vida e acima de tudo aos meus pais, Paulo e Vera.

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas que de uma forma ou de outra me influenciaram, tanto positivamente quanto negativamente, e que me trouxeram até onde cheguei. Agradeço aos meus amigos verdadeiros e acima de tudo minha família, pelo apoio condicional constante, inclusive nos momentos mais difíceis.

É como se diz: conhecendo o inimigo e conhecendo a si mesmo, não será preciso temer o resultado de uma centena de batalhas. Conhecendo a si mesmo, mas não o inimigo, a cada vitória corresponderá uma derrota.

(Sun Tzu. **A Arte da Guerra**. 770-476 a.C., p. 58)

RESUMO

O ininterrupto desenvolvimento de novas tecnologias e ferramentas para o controle e simulação de processos, aliado à constante busca por modelos matemáticos mais precisos e representativos da realidade, tem possibilitado uma aproximação entre teoria e prática inédita na operação de cadeias de suprimentos. Planejamentos táticos e operacionais de alta acurácia são essenciais para determinados tipos de operações, como por exemplo em empresas de entrega de bens perecíveis e de distribuição de combustíveis. Além da preocupação com o controle eficiente de seus estoques e de suas frotas veiculares, seus clientes devem ser atendidos dentro de intervalos de tempo determinados, de modo a atingir níveis de serviço estabelecidos e até mesmo garantir a viabilidade de seus produtos. Mesmo com os avanços expressivos na área da modelagem de sistemas de roteamento de veículos, alguns desafios na resolução destes problemas ainda persistem. Este trabalho propõe um modelo matemático de Programação Linear Inteira Mista (PLIM) para o Problema de Roteamento de Estoque com Janelas de Tempo (*Inventory-Routing Problem with Time-Windows* - IRPTW). Um modelo exato é elaborado, sendo testado seu desempenho computacional sob o auxílio de dois conjuntos de desigualdades válidas desenvolvidas para o Problema de Roteamento de Estoque (*Inventory-Routing Problem* - IRP), variadas técnicas de pré-processamento, heurísticas de melhoria de solução, e um algoritmo de *Local branching*. Uma configuração utilizando desigualdades válidas referentes a limites melhorados proporciona os melhores resultados dentre todas as avaliadas. Esta configuração é usada como base para o algoritmo de *Local Branching*, que apresenta modificações específicas para a exploração agressiva e rápida de vizinhanças reduzidas do espaço de busca do problema. Os resultados obtidos são comparados com um grupo de instâncias desenvolvido para o problema, apresentando ganhos consistentes quando comparado aos resultados existentes. Diversas novas melhores soluções são encontradas para o conjunto avaliado e estabelecem-se limites superiores e inferiores (*gaps*) para diversas outras instâncias. Este trabalho, até onde sabemos, é o primeiro a integrar todas essas ferramentas de otimização para a resolução do IRPTW, e é o primeiro a comparar resultados com um conjunto de instâncias exclusivamente desenvolvido para o IRPTW, ao mesmo tempo que expande este grupo com instâncias ainda mais complexas. A estratégia focada em exploração parcial de vizinhanças do *Local Branching* também é uma contribuição, podendo ser ainda mais aprofundada e melhorada em trabalhos futuros.

Palavras-chave: Roteamento de estoque. Roteamento de veículos. Janelas de tempo. *Local branching*.

ABSTRACT

The continuous development of new technologies and tools for better process control and simulation, combined with the strive for better and more representative mathematical models, has allowed supply chain models to reach levels of accuracy never seen. Tactical and operational planning are essential to the operation of many logistic chains, such as perishable products delivery and fuel distribution. Not only these companies have to efficiently manage their inventories and vehicle fleets to achieve predetermined levels of service, they must also fulfill their customers' needs in restricted time-windows and guarantee their product's viability during the entire delivery process. Even though many improvements were made in the field of vehicle routing, some challenges remain. This dissertation proposes a mixed-integer programming (MIP) model for the Inventory-Routing Problem with Time-Windows (IRPTW). An exact model is proposed and has its performance, alongside two groups of valid inequalities developed for the Inventory-Routing Problem (IRP), different pre-processing techniques, solution improvement heuristics, and a Local Branching algorithm, analyzed. A configuration with inventory control valid inequalities presented the best results between all analyzed configurations. This configuration is used as a basis for the Local Branching algorithm, which is specifically adapted to explore reduced neighborhoods of the problem's search space quickly and aggressively. The model is tested using a benchmark instance set and is shown to be superior in comparison to the existing results. Several new best-known solutions are determined for the instance set, just as new upper and lower bounds (gaps) are determined for several other instances. The developments presented here are, as far as we know, the first ones to integrate all these tools under one optimization framework for the IRPTW. This dissertation is also the first one to compare results with a benchmark instance set developed specifically for the problem, while also expanding said instance set. The partial neighborhood exploration used by the Local Branching algorithm is also a contribution to the literature since it enables a quick and efficient exploration of the method's tree. This integration of optimization tools can be worked on future papers, having its approach refined to provide even better results.

Keywords: Inventory-routing. Vehicle-routing. Time-windows. Local branching

SUMMARY

1 INITIAL REMARKS

The global post World War 2 scenario was a great driver for the development of concepts associated with entrepreneurial logistics and supply chain. The technological revolution that led to new industrial equipment and an expansion of operations scope, imposed a need for better material input/output control, and product transportation (MACHLINE, 2011). Operations Research had its origins during this period, proposing quantitative techniques for this advanced system control using algorithms based on linear programming (LP), mixed-integer linear programming (MILP), heuristics, and process simulation (MACHLINE, 2011).

Because of these developments, the concept of supply chain was created. This concept can be synthetized as the efforts around the production and delivery of a product, starting at the supplier of the supplier, until it reaches the customer of the customer (MACHLINE, 2011). One of the basic principles of a supply chain is that the synchronized operations between departments tend to reduce the overall costs, and to increase the aggregate value of the product generated by then. (SCAVARDA E HAMACHER, 2001).

The ongoing technological evolution of control systems, both in terms of hardware and software, allows companies to improve their operation's management computationally and systematically. In parallel to these technological advancements, the scope and complexity of supply chains has also increased significantly. These two aspects result in a continuous feedback loop that drives the technological advancement of supply chains.

Within the field of Operations Research, there are several successful real-life examples of mathematical modeling with the goal of optimizing supply chains. The following items are considered vital for mathematical modeling (LACHTERMACHER, 2016):

Input:

- Decision Variables.
- Parameters.

Output:

- Performance.
- Consequences.

The consequences of these models can be observed in three different hierarchical tiers:

- Strategical: decisions taken by the higher management of a company, such as investment decisions and market-share targeting. Generally associated with long-term initiatives (years).
- Tactical: decisions taken by regular management, such as determination of commercial partners or suppliers. Generally associated with medium-term initiatives (months, weeks).
- Operational: decisions taken by managers and supervisors, such as shift planning and maintenance routines. Generally associated with short-term initiatives (days, hours).

In the tactical and operational tiers, some of the most important optimization aspects are those related to inventory and vehicle routing. Both have a vast literature in terms of individual control methods, but considering today's conception of a supply chain, it is preferred a conjoint optimization, which is proven to yield higher financial and operational gains.

The denomination Inventory-Routing Problem (IRP) was established in the 1980's, and aims to gather optimization techniques for systems that consider different (GUIMARÃES et al., 2019):

- Fleet homogeneity (homogenous, heterogenous).
- Resupply policies (order-up-to-level, maximum-level).
- Supply chain structures (one-echelon, two-echelon, three-echelon).
- Planning horizon (days, weeks, months).

Time-windows can play an important role in industrial and commercial operations. These can be represented mathematically as time intervals in which suppliers/customers are able to service or be serviced. The Inventory-Routing Problem with Time-Windows (IRPTW) is the result of the addition of these time-windows to the IRP. Some practical examples are:

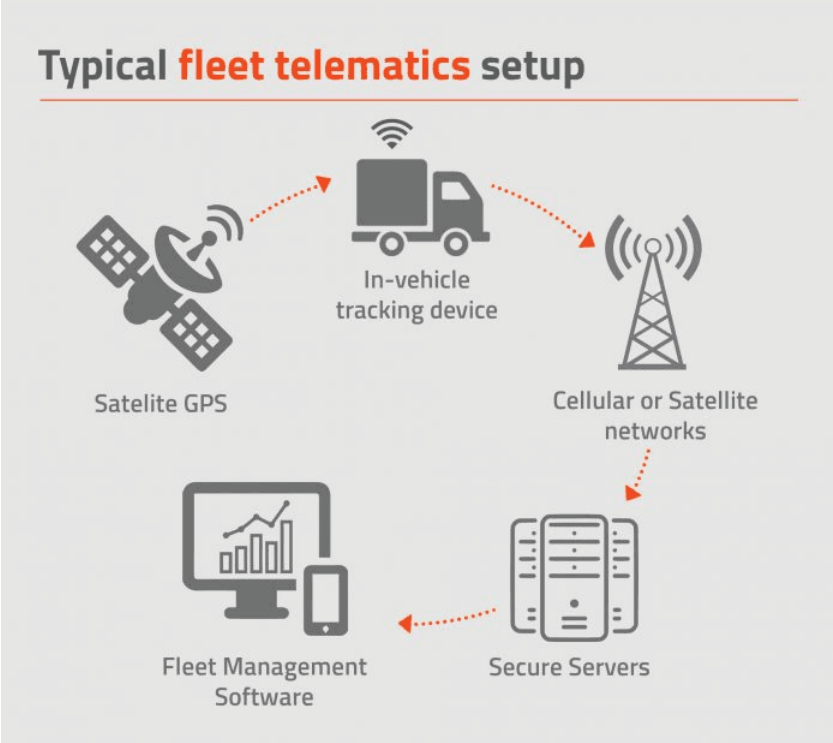
- Delivery deadlines imposed by suppliers.
- Preferential delivery hours requested by customers.
- Time restrictions imposed by legal/geographical reasons.

Companies that act as online marketplaces usually establish pre-determined delivery deadlines for their sales, to offer greater reliability to their customers. Violations of these deadlines can lead to decrease in service quality indicators, or even financial losses in the form of fines or refunds.

Passenger/food delivery services also have time-windows ingrained into their operations. Apart from the already mentioned negative impacts that violating said time-windows can cause to their business, factors such as food perishability and arrival/departure delays must also be considered.

Fleet management software with different degrees of complexity can be found in the market. They often rely on the efficient communication between a complex architecture of services sustained by a telecommunication network. These networks might be based on satellites, cellphones, or the internet, and often depend on physical servers maintained by these companies. This infrastructure is integrated with applications (either mobile or desktop) that allow to automatize tasks, visualize real-time statistics, reduce operational costs, and provide more security to drivers, as illustrated in Figure 1. (ALTEXSOFT, 2019a).

FIGURE 1 –TYPICAL SETUP OF A FLEET MANAGEMENT SYSTEM



SOURCE: Altexsoft (2019a)

Among the base aspects contemplated within these software are (ALTEXSOFT, 2019a):

- Routing: improved routing decisions through the usage of GPS systems, estimation of time losses due to traffic, and calculation of average movement speed on streets/highways.
- Fuel: tracking of fuel consumption and hazardous emissions.
- Vehicular maintenance: diagnosis, scheduling, general alerts related to routine inspections.
- Fleet management: service level analysis using integrated database systems.
- Freight control: expense reports, license control, real-time tracking of products/vehicles.
- Security: integrated systems aid drivers during work, reducing risks both to personnel and company property.

Examples of commercial fleet management software are shown in Panel 1 (ALTEXSOFT, 2019a):

PANEL 1 – EXAMPLES OF FLEET MANAGEMENT SYSTEMS

Characteristics	Geotab	Verizon	Teletrac	Fleetio
Strong points	Direct feedback for drivers Security functionalities Custom mapping	Route replay Proactive alerts	Non-verbal check-in Dynamic dashboards Vehicular inspection alerts	Detailed maintenance management Freight control Fuel management system
Provisions hardware?	YES	YES	YES	NO
Price	MODERATE	MODERATE	HIGH	LOW
Usability	EASY	MODERATE	MODERATE	EASY
Mobile applications	1	3	1	3
Technical support	MODERATE	MINIMAL	MODERATE	GOOD

SOURCE: Adapted form Altexsoft (2019b)

New technologies centered on Internet of Things (IoT) and cloud computing enable such applications to be used in manufacturing equipment, or to be embedded into vehicular fleets. (ALTEXSOFT, 2019a).

Another important aspect present on these applications is the optimization methodology behind them. While commercial solutions generally have specialized teams working towards the development of models and optimization algorithms, crafting highly specialized solutions, open-source initiatives allow small organizations that do not possess these same resources to employ tools on a par with the ones used by larger companies.

Open-source projects also deflect in some manner the responsibility of developing and maintaining software, which can lead to cost reductions, and therefore financial gain. Such solutions are usually designed around code libraries and APIs, some of which can be used for optimization purposes.

Some examples of popular open-source, or free to use software, are:

- Google OR-Tools: encapsulates many commercial solvers (Gurobi, CPLEX, COIN, CBC, SBIC, CP-SAT) and provides several mathematical interfaces for optimization. Has embedded algorithms for

routing problems and some of their variations, such as problems with multiple vehicles, pickup-and-delivery, and time-windows.

- Concorde TSP solver: optimization library written in C for the Travelling Salesman Problem (TSP). Many papers reference this library, since it is considered one of the benchmark sources for instances and results.
- TSPLIB: open-source library for TSP and some of its variants. Contains algorithms for the Sequential Ordering Problem (SOP) and the Capacitated Vehicle-Routing Problem (CVRP).

Aside from the resources mentioned above, famous commercial solvers also publish freeware versions of their libraries for academic research.

New data extraction methods and database technologies also contribute towards the development of mathematical modeling. Recent studies focus on the effect of time dependency in vehicle routing problems. Time dependency is commonly represented as variable time travel times/costs along periods. This aspect is often considered when dealing with routing problems under time-windows constraints.

Kok et al. (2012) study strategies to avoid urban traffic by simulating a real urban network. Work time reduction of 87% is achieved by incorporating models that consider time dependency in the Shortest Path Problem (SPP).

Alvarez et al. (2018) analyze the importance of considering urban traffic in VRPs. Data from the province of Catalonia is used for testing, which shows that mathematical models yield results around 11% better when these effects are considered.

Belhassine et al. (2018) present results for the TDVRP (Time-Dependent Vehicle-Routing Problem) by using data extracted from Québec city's urban network. Reductions of 22% in travel times are achieved, while also saving around 43 tons of CO_2 emissions.

Heni et al. (2018) study new fuel consumption estimation methods for time-dependent systems. Factors such as downtime, average movement speed, and cargo weight, are considered in their analysis.

All these studies map urban networks and show the impacts of time-dependent aspects in the problems. This shows that these parameters can yield greater accuracy to optimization models when appropriately employed.

For the optimization of such complex models, several techniques and algorithms have been proposed in the literature. Coelho, Cordeau and Laporte (2014) present some of these techniques:

- Branch-and-bound (B&B): exact procedure used for the resolution of most mixed-integer programming (MIP) problems, where a tree exploration is used and sequential problems are solved, each of them considering different branching options for integer variables (0 or 1). This algorithm can be extended to insert various specialized constraints during its execution (Branch-and-cut – B&C), or even use column generation to solve relaxed versions of the tree's nodes (Branch-and-price - B&P).
- Valid inequalities: constraints that are not necessary to the mathematical formulation of problems, but whose addition can lead to improved initial bounds for the B&B algorithm.
- Heuristics: algorithms that are not exact in nature but usually obtain high quality solutions in fast times. Such algorithms can be based on local search, node interchange, assignment, etc.
- Metaheuristics: specialized versions of heuristics that often integrate multiple different heuristics into a single optimization framework.
- Matheuristics: integration of heuristics/metaheuristics with mathematical programming. Given this intricate combination of tools, many can determine optimality through their usage.

Not only algorithms help the optimization of integrated logistics problems. Managerial concepts such as Vendor-managed inventory (VMI) are key to understand why conjoint decision-making is so valuable. Archetti and Speranza (2016) show that employing a replenishment policy where the supplier controls the inventory levels of its customers can achieve operational costs up to 10% cheaper than it would if decisions were made individually – and this cost reduction includes their customers' operational costs. The scenario where each actor in the supply chain take decisions for their sole benefit is known as Retail-managed inventory (RMI).

The replenishment policies that VMI uses can vary between supply chains. Some of the most known ones are order-up-to-level (OU) and maximum-level (ML). While the first always guarantee that the customer's inventory is full after a delivery, the second is free to choose the amount of product to be sent to a given customer. The choice between them will depend on factors such as the ratio between inventory holding costs and travel costs. Extensions of these policies also exist, for instance, the optimized-target level (OTL) policy aims to determine a custom, constant, value to be delivered for each customer in a determined time horizon. (COELHO AND LAPORTE, 2014a).

A next step would be to incorporate inventory costs and multi-period planning into IRPTW models. The resulting complexity will probably be high, but the potential gains to be obtained drive the development of more efficient resolution methods for these problems. The IRPTW is one of the problems that must deal with an increased complexity level. According to the study conducted in this project, few models integrate vehicle and inventory control simultaneously into their formulations, usually preferring to focus on the vehicular aspect of the problem.

This project proposes the application of a Local branching algorithm together with other optimization tools for an efficient optimization of the IRPTW. A model concept is elaborated, and auxiliary tools that can aid the optimization process are described. A thorough optimization framework is tested against benchmark instances, and the obtained results are analyzed for different configurations and scenarios.

1.1 MOTIVATION

In the inventory control and vehicle routing field, accurate representation of phenomena associated with periodicity/imprevisibility, such as travel and service times, is a great challenge. Such aspects can turn a continuous operation very complex, requiring the optimization system to have appropriate tools and considerations to deal with them.

This project proposes integrating the most recent routing models with the inventory aspect, while also representing decision-making processes associated with periodicity (when to deliver), quantity (how much to deliver), and rentability of the supply chain as whole (mutual benefit to customers and suppliers). Impacts caused by

the integration of time-windows into the model, which renders the model closer to real-world applications, are highlighted in our analysis.

1.2 OBJECTIVES

1.2.1 Main objective

The main objective of this project is to develop an exact mathematical model for the IRPTW, and to integrate it with a solution framework that combines valid inequalities, pre-processing techniques, solution improvement heuristics, and a Local branching algorithm.

1.2.2 Specific objectives

The following items can be highlighted as the specific objectives of this project:

- Evaluate problems that are either similar or precursor to the IRPTW and find approaches/tools that might be useful for an exact mathematical model.
- Determine the computational efficiency of different valid inequality groups designed around routing decisions, symmetry break, and improved variable bounds.
- Check the performance of pre-processing techniques that allow drastic reductions in the total number of variables and constraints used in the mathematical model.
- Design a Local Branching algorithm focused on the quick exploration of reduced neighborhoods of the search space, so improvements to the objective function are made consistently and in a quick manner.
- Develop an initial solution heuristic capable of providing solutions that are good and obtainable in fast times, to trigger the execution of the Local branching algorithm as quick as possible.
- Compare the results obtained by our solution framework with benchmark results from the literature.

1.3 LIMITATIONS

One meaningful limitation for project is the extensive variety of approaches and interpretations given to the IRPTW in the literature. This leads to papers considering different objective function compositions, constraints, and even variables between models. This restricts the benchmarking that can be made between models, since few of them share enough similarities to warrant numerical comparisons

1.4 STRUCTURE

This dissertation is structured as follows.

Chapter 2 contains an adaptation of the published version of the paper entitled *A local branching algorithm for the inventory-routing problem*, which was submitted and accepted by the International Journal of Production Research (IJPR), and that is available at https://doi.org/10.1080/00207543.2021.1998696. This paper presents a methodology to solve the IRPTW exactly using a mathematical model that is integrated with tools such as different valid inequality groups, pre-processing techniques, solution improvement heuristics, and a Local branching algorithm.

Chapter 3 presents the conclusion and final remarks about the matheuristic algorithm proposed by this project.

2 METHODOLOGY

This chapter presents the resulting scientific paper of this research project. It is titled *A local branching algorithm for the inventory-routing problem with time Windows*, and it was accepted by the International Journal of Production Engineering (IJPR).

This adapted version of the paper was originally written by:

- Bruno Eckwert Demantova, UFPR (PPGOLD). Conceptualization, software implementation, validation, and writing.
- Cassius Tadeu Scarpin, UFPR (PPGOLD). Conceptualization, validation, and writing.
- Leandro Callegari Coelho, Université Laval (CIRRELT). Conceptualization, validation, and writing.
- Maryam Darvish, Université Laval (CIRRELT). Conceptualization, validation, and writing.

2.1 ABSTRACT

The Inventory-Routing Problem (IRP) deals with the joint optimization of inventory and the associated routing decisions. The IRP with time windows (IRPTW) considers time windows for the deliveries at the customers. Due to its importance and several real-world applications, in this paper, we develop an intricate solution algorithm for this problem. A mix of tools ranging from established groups of valid inequalities, pre-processing techniques, local search procedures, and a local branching algorithm is utilized, in order to efficiently solve the IRPTW. We compare the performance of our algorithms over a benchmark set of instances and show how our solution algorithm provides very promising results. Moreover, the results of our study provide an overview of the performance of several already proposed techniques and their integration in the literature

2.2 INTRODUCTION

Introduced in the seminal paper of Bell et al. (1983), the Inventory Routing Problem (IRP) is an important optimization problem in which the inventory control vehicle routing decisions are integrated. Therefore, the main objective of the IRP is to minimize the total inventory and distribution costs. To date, several extensions of the IRP have been proposed in the literature (COELHO et al., 2014), including variations of the number of periods, number of suppliers and customers in the supply chain, type of routing considered, and vehicle fleet composition (ANDERSSON et al., 2010). Furthermore, with the recent interest in sustainable supply chains, growing attention has also been given to sustainability concerns in the IRP (SOYSAL et al., 2019). These studies include several aspects of sustainability such as in perishable products (SHAABANI AND KAMALABADI, 2016), reverse logistics (SOYSAL, 2016), emission reduction (DARVISH et al., 2019), among others.

A typical application of the IRP and its variants is in the city logistics (BERTAZZI et al., 2019). Application of this problem to several real-world city logistics situation imposes additional constraints of delivery time windows, e.g., in retail delivery every retailer has a preferred time interval to be visited (REPOUSSIS AND TARANTILIS, 2010). This gives rise to an important variant of the IRP, which is the IRP with Time Windows (IRPTW). Unlike the Vehicle Routing Problem with Time windows (VRPTW) which has been thoroughly studied (PARASKEVOPOULOS et al., 2008) and for which several problem instances are available (see TOTH AND VIGO (2014)), there are a few studies proposing exact methods for the IRPTW.

This paper proposes an exact algorithm for the IRPTW. We consider a single product and a homogeneous multi-vehicle fleet setting. Given the importance of greenhouse gas emission reduction in cities, the objective of our problem is to minimize the distances traveled, which eventually leads to less fuel cost and emissions. Moreover, we study the effectiveness of groups of valid inequalities, pre-processing techniques, and an initial solution heuristic. In order to improve the solution quality for bigger and more complex instances, a local branching algorithm is proposed. All tests are conducted on benchmark instances from the literature.

The paper is organized as follows: Section 2.3 overviews the literature on the IRPTW and local branching. Section 2.4 presents our problem definition. Section 2.5 presents the formal mathematical model of the problem along with several valid inequalities and symmetry breaking considerations. Section 2.6 presents the solution

algorithm. Section 2.7 contains the results of extensive computational experiments, and Section 2.8 provides the conclusions and final remarks.

2.3 LITERATURE REVIEW

We now review the relevant literature for the IRPTW in Section 2.3.1 and for the local branching algorithm in Section 2.3.2.

2.3.1 Inventory-routing problem with time windows

Several heuristics approaches are proposed in the literature to solve the IRP and IRPTW (KHEIRI, 2020), here we focus to the exact methods and modeling efforts.

Liu and Lee (2011) develop an enhanced model for the IRPTW, where a supplier must serve a set of customers under soft time windows. The concept of order cycle time to determine the periodicity of deliveries is used. A two-stage meta-heuristic is proposed, obtaining an initial solution and then improving it by means of a local search algorithm combined with a tabu search procedure. The authors use adaptations of the instances proposed by Solomon (1987), providing exact solutions for smaller instances and heuristic solutions for the larger ones.

Li et al. (2014) use a mixed integer programming model with a tabu search algorithm and lagrangean relaxation techniques to solve an IRP model with the objective of minimizing total travel time. Exact solutions for small instances are presented, while larger ones are solved using both relaxation techniques and a meta-heuristic.

Lappas, Kritikos, and Ioannou (2017) solve a model containing a set of customers with hard time windows, a single product, a homogeneous fleet, and an order-up-to-level (OU) policy for deliveries. A two-phase meta-heuristic is proposed, which determines the periodicity and quantities of product delivered, so a local search algorithm can be employed to determine the best routing. The results are evaluated on a set of instances created by the authors.

The time-constrained inventory routing problem (TCIRP) is a studied in Lefever et al. (2019) where the travel time on each arc is uncertain. The authors develop a Benders decomposition-based heuristic and describe several valid inequalities for the IRP. They test the efficiency of their approach on a set of benchmark instances.

A two-echelon inventory routing problem is modeled in Farias et al. (2020) and solved using a two-step matheuristic. The authors create a new set of benchmark instances. They show the efficiency of the proposed approach and the introduced valid inequalities on the created instances.

2.3.2 Local branching

As presented by Fischetti and Lodi (2003), the local branching method explores restricted regions from the problem's search space. The goal is to find improvements in the objective function while guaranteeing that previous explored spaces are not revisited.

Hansen et al. (2006) propose an extension of the local branching with the incorporation of a variable neighborhood search (VNS). Considering even smaller size neighborhoods than the ones proposed by Fischetti and Lodi (2003), They use the VNS as a local search tool and also utilize several neighborhoods to promote the diversification of the current one.

The application of local branching to the capacitated fixed-charge network design problem studied in Rodríguez-Martín and Salazar-González (2010) clearly outperforms other heuristics proposed in the literature.

Yu et al. (2016) solve a robust gate assignment problem using a local branching framework, comparing its performance with three other exact solution algorithms, including diving and relaxation induced neighborhood search.

To solve the open pit mine production scheduling problem, Samavati et al. (2017) combine local branching with an adaptive branching scheme. Tests conducted on the benchmark instances from the literature show that the method outperforms both the Branch-and-Cut and Lagrangian relaxation techniques.

Hernandez et al. (2019) propose a local branching matheuristic to solve a VRP with stochastic demands. This matheuristic employs an intensification procedure at each node of the local branching tree, which is embedded in a multi-descent scheme.

2.4 PROBLEM DESCRIPTION

The problem is defined as follows. Consider graph $G = (V, A)$, where $V = \{0,..., N + 1\}$ is the vertex set and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the arc set. Vertices 0 and $N + 1$

represent the depot location, being 0 the starting point of vehicle routes and N + 1 their ending point. Set V is partitioned as follows: V' = {0, ..., N}, V'' = {1, ..., N + 1}, and C = {1, ..., N}. For each period t $\in \tau$ = {1, ..., T}, each customer must satisfy a known demand
d_i^t while holding a maximum inventory capacity of U_i, such that potentially $d_i^t > U_i$. The inventory is controlled by an order-up-to-level (OU) policy, i.e., whenever a delivery occurs, the quantity must be enough to maximize the customer's inventory at the end of the period. Each arc has a travel cost c_{ij} and a travel time t_{ij}, both equal to the Euclidean distance between points *i* and *j*. Whenever a customer is served by a vehicle, the delivery must start within a specific time window, from E_i to L_i. Service duration is denoted s_i. Vehicles from a set κ = {1, ..., K} may visit each customer only once per period and can carry at most Q units.

In order to ensure cyclic operations and to avoid the end-of-horizon effect, all customers start with full inventories at the first period and must have their inventories replenished during the last one.

2.5 MATHEMATICAL MODEL

The variables used in our model are as follows. Binary variable y_{ij}^{kt} is equal to 1 if arc (i, j) is traversed by vehicle *k* during period t, 0 otherwise. Binary variable z_i^{kt} is equal to 1 if vertex *i* is visited by vehicle *k* during period t, 0 otherwise. Variable q_i^{kt} represents the quantity delivered to customer *i* by vehicle *k* during period t. Variable I_i^t is equal to the inventory level of customer *i* at the end of period t. Variable u_i^{kt} denotes the time that vertex *i* starts being served by vehicle *k* during period t. The model presented here is based on that of Lappas, Kritikos, and Ioannou (2017). Table 1 provides a summary of all parameters and variables used.

TABLE 1 - NOTATION USED IN THE MODEL

Sets	
V = {0,...,N + 1}	Set of vertices
C = {1,...,N}	Set of customers
V' = {0,...,N}	Set of vertices excluding the supplier's return node
V'' = {1,...,N + 1}	Set of vertices excluding the supplier's origin node
κ = {1,...,K}	Set of available vehicles
τ = {1,...,T}	Set of periods
Parameters	
c_{ij}	Travel cost of arc originating at *i* and ending at *j*;
t_{ij}	Travel time of arc originating at *i* and ending at *j*;
I_i^0	Initial inventory of supplier/customer *i*;
U_i	Inventory capacity of customer *i*;
Q	Vehicle maximum capacity;
d_i^t	Demand of customer *i* during *t*;
s_i	Service time of customer *i*;
E_i	Opening time of customer *i*;
L_i	Closing time of customer *i*.
M_{ij}	Big M parameter calculated for each *i* and *j* combination
Variables	
y_{ij}^{kt}	Binary variable equal to 1 if arc *ij* is traversed by vehicle *k* during period *t*, 0 otherwise;
z_i^{kt}	Binary variable equal to 1 if vertex *i* is serviced by vehicle *k* during period *t*, 0 otherwise;
x_i^{kt}	Quantity of product delivered to customer *i* by vehicle *k* during period *t*;
I_i^t	Inventory level of supplier/customer *i* at the end of period *t*;
u_i^{kt}	Time at which service starts for customer *i* by vehicle *k* during period *t*.

SOURCE: The Author (2021)

2.5.1 Mixed-integer linear programming formulation

The IRPTW model is described as follows.

$$\min W = \sum_{t \in \tau} \sum_{(i,j) \in A} \sum_{k \in \kappa, k \leq i,j} c_{ij} y_{ij}^{kt} \tag{1}$$

Subject to:

$$I_i^t = I_i^{t-1} + \sum_{k \in \kappa, k \leq i} q_i^{kt} - d_i^t, \quad i \in C, t \in \tau \tag{2}$$

$$\sum_{k \in \kappa, k \leq i} q_i^{kt} \leq U_i - I_i^{t-1} + d_i^t, \quad i \in C, t \in \tau \tag{3}$$

$$(U_i + d_i^t) z_i^{kt} - I_i^{t-1} \leq q_i^{kt} \leq (U_i + d_i^t) z_i^{kt}, \quad i \in C, k \in \kappa, k \leq i, t \in \tau \tag{4}$$

$$\sum_{i \in C} q_i^{kt} \leq Q z_0^{kt}, \quad k \in \kappa, k \leq i, t \in \tau \tag{5}$$

$$\sum_{j \in C} y_{0j}^{kt} = z_0^{kt}, \quad k \in \kappa, k \leq j, t \in \tau \tag{6}$$

$$\sum_{i \in C} y_{iN+1}^{kt} = z_{N+1}^{kt}, k \in \kappa, k \leq i, t \in \tau \tag{7}$$

$$\sum_{j \in V''} y_{ij}^{kt} = z_i^{kt}, \quad i \in C, k \in \kappa, k \leq i, j, t \in \tau \tag{8}$$

$$\sum_{j \in V''} y_{ij}^{kt} = \sum_{j \in V'} y_{ji}^{kt}, i \in C, k \in \kappa, k \leq i, j, t \in \tau \tag{9}$$

$$u_i^{kt} + s_i + t_{ij} \leq u_j^{kt} + M_{ij}\left(1 - y_{ij}^{kt}\right), \quad i, j \in V, k \in \kappa, k \leq i, j, \forall t \in \tau \tag{10}$$

$$E_i z_i^{kt} \leq u_i^{kt} \leq L_i z_i^{kt}, i \in C, k \in \kappa, k \leq i, t \in \tau \tag{11}$$

$$\sum_{k \in \kappa, k \leq i} z_i^{kt} \leq 1, \quad i \in C, t \in \tau \tag{12}$$

$$I_i^T = U_i, i \in C \tag{13}$$

$$q_i^{kt} \in \mathbb{Z}^+, \quad i \in C, k \in \kappa, k \leq i, t \in \tau \tag{14}$$

$$u_i^{kt} \in \mathbb{R}^+, \quad i \in V, k \in \kappa, k \leq i, t \in \tau \tag{15}$$

$$I_i^t \in \mathbb{Z}^+, i \in C, t \in \tau \tag{16}$$

$$z_i^{kt} \in \{0,1\}, \quad i \in V, k \in \kappa, k \leq i, t \in \tau \tag{17}$$

$$y_{ij}^{kt} \in \{0,1\}, \quad i \in V', j \in V'', k \in \kappa, k \leq i, j, t \in \tau \tag{18}$$

The objective function (1) minimizes the total distance traveled by the vehicles. Constraints (2) ensure inventory conservation for the supplier and the customers. For the initial period, $I_i^0 = U_i$. Constraints (3) - (4) impose a maximum inventory level to each customer and dictate an OU policy. In this case, it is necessary to add the demand into these constraints, so that $d_i^t \geq U_i$ can be held. Constraints (5) limit delivered quantities to respect the capacity of the vehicles. Constraints (6) - (9) are vehicle routing constraints, establishing the required relationships between routing and assignment variables. Constraints (10) prevent subtours while imposing that all visiting times must be consistent with the respective travel times between arcs and service times at customers. Constraints (11) forbid any customer to be visited outside their time windows. Constraints (12) forbid split deliveries. Constraints (13) ensure that all

customers are serviced during the last period. Constraints (14) - (18) impose the domain and nature of the variables.

Note that symmetry breaking considerations are embedded in all applicable constraint sets. These considerations ensure that vehicle $k+1$ will only visit a customer if vehicle k is also used in the period. This formulation helps reduce the number of variables and constraints generated when compared against the model of Lappas, Kritikos, and Ioannou (2017), especially when dealing with large instances.

2.5.2 Symmetry breaking and valid inequalities

The first group of valid inequalities is related to the IRP with multiple vehicles, and is adapted from Coelho and Laporte (2014a,b):

$$y_{0j}^{kt} \leq z_j^{kt}, \quad j \in C, k \in \kappa, k \leq j, t \in \tau \tag{19}$$

$$y_{ij}^{kt} \leq z_i^{kt}, \quad i,j \in C, \forall k \in \kappa, k \leq i,j, t \in \tau \tag{20}$$

$$z_i^{kt} \leq z_0^{kt}, \quad i \in C \cup \{N+1\}, k \in \kappa, k \leq i, t \in \tau \tag{21}$$

$$z_0^{kt} = z_{N+1}^{kt}, \quad k \in \kappa, t \in \tau \tag{22}$$

$$z_0^{kt} \leq z_0^{k-1t}, \quad k \in \kappa \backslash \{1\}, t \in \tau \tag{23}$$

$$z_i^{kt} \leq \sum_{j=1}^{i} z_j^{k-1t}, i \in C \backslash \{1\}, k \in \kappa \backslash \{1\}, k \leq i, j, t \in \tau \tag{24}$$

Valid inequalities (19) enforce that if the supplier is the immediate successor of a customer, then both, the supplier and the customer, must be visited by the same vehicle. Inequalities (20) establish relationships between arc usage and delivery assignment variables. Inequalities (21) state that a customer can only be served by a vehicle if it has left the depot. Inequalities (22) reinforce the relationship between the two depot designation variables. Inequalities (23) and (24) are symmetry breaking constraints. They ensure that a vehicle k cannot leave the depot if vehicle $k-1$ is not yet used. This is extended to customer vertices, so that if customer i is assigned to vehicle k in period t, then vehicle $k-1$ must serve a customer with an index smaller than i in the same period.

An extra group of symmetry breaking inequalities are added, stemming from Coelho and Laporte (2014b), constraints (25) state that a customer with a higher index

is assigned to a vehicle only if all the previous customers are already assigned to lower index vehicles. The extreme case is that each customer is assigned to one vehicle.

$$k(z_i^{k+1t}) \leq \sum_{j=1}^{i} z_j^{k-1t}, i \in C\backslash\{1\}, k \in \kappa\backslash\{1, K\}, k \leq i, j, t \in \tau \quad (25)$$

As a result, customer *i* can never be visited by a vehicle with an index greater than *i*. Thus, we can significantly reduce the number of variables used in the model. Another group of valid inequalities is derived from Lefever (2018) and are used to estimate delivery occurrence and to strengthen the bounds of inventory control variables. Two new parameters must be introduced, known as residual inventory ($\bar{I}_i^{0,t}$) and residual demand ($\bar{d}_i^t$) (DESAULNIERS et al., 2016). The residual inventory is the remaining of initial inventory (I_i^0) at each customer in period *t*, and the residual demand is the amount of demand that exceeds each customer's initial inventory. As shown in equations (26) – (28), both parameters vary over time. While the residual inventory decreases until it reaches zero, the residual demand increases to a maximum of d_i^t units per period.

$$\bar{I}_i^{0,t} = \max\left\{0, I_i^0 - \sum_{s=0}^{t} d_i^s\right\}, \ i \in C, t \in \tau \quad (26)$$

$$\bar{d}_i^1 \leq \max\{0, d_i^1 - I_i^0\}, i \in C \quad (27)$$

$$\bar{d}_i^t = \max\{0, d_i^t - \bar{I}_i^{0,t-1}\}, i \in C, t \in \tau\backslash\{1\} \quad (28)$$

Inequalities (29) determine a lower bound for the number of vehicles to be used in order to satisfy the residual demand of a given period.

$$\sum_{s=1}^{t} \sum_{k \in \kappa} z_0^{ks} \geq \left\lceil \frac{\sum_{i \in C} \sum_{s=1}^{t} \overline{d_i^t}}{Q} \right\rceil, t \in \tau \quad (29)$$

Desaulniers et al. (2016) also employ a set of inequalities that revolves around the minimum number of sub-deliveries per period. For each customer and period, the authors separate the remaining periods in two sets: periods in which a delivery is

enough to satisfy the demand of period *t* (τ_{it}^{+}), and periods that do not satisfy this condition (τ_{it}^{-}). Let us consider θ_i^{ts}, indicating if at period *t*, the inventory level at customer *i* after the delivery is made at period *s* is positive. If a given customer has a residual demand higher than zero in *t*, the set τ_{it}^{+} is non-empty. The elements of this set can be determined by analyzing which periods $s \leq t$ can receive a sub-delivery to guarantee a residual demand equal to zero in *t*. If a sub-delivery results in a non-positive residual demand, parameter θ_i^{ts} is equal to 1. This parameter is then used in constraints (30) and, every time the left-hand side of the equation forms a non-empty set, the corresponding inequality is added to the model.

$$\sum_{s=1}^{t} \sum_{k \in \kappa, k \leq i} z_i^{ks} \theta_i^{ts} \geq 1, i \in C, t \in \tau \tag{30}$$

Since customer inventories might not be empty during the initial periods, constraints (31) and (32) can be used to tighten the bounds of the inventory control variables. An adaptation is done to constraints (32), so they take into account demands higher than customer's holding capacities.

$$I_i^t \geq \bar{I}_i^{0,t}, i \in C, t \in \tau \tag{31}$$

$$\sum_{k \in \kappa, k \leq i} q \leq U_i - \bar{I}_i^{0,t} + \bar{d}_i^t, i \in C, t \in \tau \tag{32}$$

Considering time windows for each customer leads to several paths in the original graph to become infeasible. Having this information in advance enables us to pre-process the graph and eliminate these infeasible paths a priori. Ascheuer et al. (2001) describe a technique which allows tightening time window parameters and by eliminating any infeasible arcs. We apply it to the IRPTW as follows.

Given a reference vertex *i*, in cases where the earliest service time leaving from other vertices is later than E_i, or the minimization of waiting times in *i* can be achieved by shifting the original E_i, the tightening of the vertex opening time window is possible. If the earliest service time possible for vertex *i* when leaving from other vertices is earlier than L_i, or the latest possible arrival time at a successor vertex of *i* allows departures earlier than L_i, the tightening of the vertex closing time is viable.

Algorithm 1 describes the calculation steps used for the pre-processing of E_h and L_h. First, all arcs leading to customer h are evaluated in order to check if the earliest arrival time possible is higher than the original E_h, which leads to an increase in E_h. Then, all arcs leaving from h are evaluated in order to see if waiting times on the vertex can be minimized, if possible, resulting in a further increase of E_h. Similarly, all arcs leading to h are analyzed in order to identify if the latest possible arrival times from these vertices are lower than L_h. Next, the latest possible arrival times in *j* leaving from h are considered and, if a valid value lower than L_h is found, the current parameter has its value updated. If changes are made to any of the parameters, another execution is triggered, recursively tightening the time windows of the vertices.

ALGORITHM 1 – Time-windows Tightening (TWT) calculation steps

```
1  repeat ← true;
2  while repeat = true do
3      repeat = false;
4      for h in C do
5          t1 ← Lh, t2 ← Lh, t3 ← 0;
6          for i in C ∪ {0} do
7              if Ei + si + tih ≤ t1 then
8                  t1 = Ei + si + tih;
9              end
10         end
11         Eh = max(Eh, t1);
12         for j in C ∪ {N + 1} do
13             if Ej − thj ≤ t2 then
14                 t2 = Ej − thj;
15             end
16         end
17         t3 = min(Lh, t2);
18         Eh = max(Eh, t3);
19         t1 ← 0, t2 ← 0, t3 ← Eh;
20         for i in C ∪ {0} do
21             if Li + si + tih ≥ t1 then
22                 t1 = Ei + si + tih;
23             end
24         end
25         t2 = max(Eh, t1);
26         Lh = min(Lh, t2);
27         for j in C ∪ {N + 1} do
28             if Lj − thj ≤ t3 then
29                 t3 = Lj − thj;
30             end
31         end
32         Lh = min(Lh, t3);
33         if Eh or Lh was changed then
34             repeat = true;
35         end
36     end
37 end
```

SOURCE: The Author (2021)

As a final step, any arc that violates time windows sequencing rules is removed from the model. This is done by checking, for all valid y_{ij}^{kt} in the problem, if $E_i + s_i + t_{ij} \leq L_j$. In other words, if a vehicle cannot reach *j* on time to respect the time windows constraints, then, the corresponding arc is not feasible, and is, therefore, removed from the problem.

Algorithm 2 contains the final steps of the procedure, which dictates the removal of any identified infeasible arcs in the problem. The calculation is done by simply analyzing if the destination's L_j is lower than the earliest departure time possible from *i*. If that is true, then the arc is set as infeasible and removed from the problem's graph.

ALGORITHM 2 – Unfeasible arc removal steps

```
endTWT ← false;
while endTWT ≠ true do
    endTWT = true;
    while All arcs haven't been evaluated do
        if E_i + s_i + t_ij > L_j then
            Arc (i, j) is infeasible and therefore removed from the model;
        end
    end
    if Any arc was removed then
        endTWT = false;
    end
end
```

SOURCE: The Author (2021)

It is important to note that the time-windows tightening and arc removal (TWT/AR) can be repeated until no more changes are possible, creating each time a smaller and tighter model.

2.6 SOLUTION ALGORITHM

Our algorithm contains three main steps. First, an initial solution procedure creates the first feasible solution, described in Section 2.6.1. Then, a local branching framework is used to control and diversify the search procedure, as presented in Section 2.6.2. A solution improvement procedure based on local search is used to polish feasible solutions, as presented in Section 2.6.3. Finally, in Section 2.6.4, we describe how these building blocks are combined to create a comprehensive algorithm.

2.6.1 Initial Solution Heuristic (ISH)

In order to find an initial solution for the problem, we propose a heuristic: initial solution heuristic (ISH). The goal is to determine the customers to be visited at each period and their associated routes. The heuristic algorithm consists of two phases. The first phase determines a basic delivery schedule for the customers and the associated routes. In this step, a feasible solution is created. The second phase improves the routing decisions by solving individual VRPTW sub-problems for each period.

In the first phase, all customers are sorted in an increasing E_i order, and, for each period, deliveries are assigned to customers facing stock-out, i.e., whenever $I_i^{t-1} - d_i^t < 0$. A cheapest insertion procedure is applied to determine the vehicle routes (for a maximum number of *K* routes). This step is done concurrently with the evaluation of possible time windows and vehicle capacity violations.

Another reason to use this heuristic is to estimate the required number of vehicles. The number determined by the first phase (n_k) is used to reduce the size of some models in our procedure. It causes sub-problems to become less complex and enables more nodes to be explored in a limited time.

Although this heuristic can quickly provide feasible solutions, their quality can be refined using other tools, motivating a second phase to improve the solutions. It consists of solving a VRPTW for each period, further improving routing decisions previously made by the best insertion criterion. Given the original sets *V* and *C*, two new sets V_t and C_t can be generated by considering only the nodes included in the sub-problem per period. Sets κ_t can also be established, containing only the vehicles used by the first phase of the ISH for each corresponding period. A VRPTW is then solved for each period.

2.6.2 Local branching

The exact framework presented here is based on Fischetti and Lodi (2003). In order to show how the local branching method is structured, consider an optimization model as follows:

$$\min c^{\gamma} x_j \tag{33}$$

$$Ax \geq b \tag{34}$$

$$x_j \in \{0,1\}, \forall j \in \beta \neq \emptyset \tag{35}$$

This model showcases a minimization problem containing a set of binary variables x_j. Given a reference solution $\bar{x}$ and a binary support group $\bar{S}$ = {j ∈ β : x_j = 1}, an α−opt neighborhood can be established with the use of a branching constraint:

$$\Delta(x,\bar{x}) = \sum_{j\in\bar{S}} (1 - x_j) + \sum_{j\in\beta\setminus\bar{S}} x_j \leq \alpha \tag{36}$$

Constraint (36) restricts the search space to an α-sized neighborhood, allowing a maximum of α changes in the binary variable set. An optimization problem can be generated from this reduced search space, possibly leading to new upper bound (UB) improvements in much less time. After a given neighborhood is fully explored and a new optimal solution is found, constraint (36) can be turned into (37), removing the previously explored search space from future branching steps.

$$\Delta(x,\bar{x}) \geq \alpha + 1 \tag{37}$$

After a restricted neighborhood is thoroughly explored in local branching, the previous left-hand side branching constraint is transformed into a right-hand side constraint, effectively changing the solution space considered during the creation of the next branching step.

For our IRPTW, an algorithm based on the classic framework proposed by Fischetti and Lodi (2003) is developed. The mathematical model considered by the local branching algorithm is the same described in Section 2.5 with the following differences. In order to better take advantage of problem-specific properties, only delivery assignment binary variables (z_i^{kt}) are fixed in the branching constraints. Therefore, it leaves variables related to arc usage (y_{ij}^{kt}) free during optimization. Given an initial solution and an initial upper bound (x_{init},UB_{init}), a first α−opt neighborhood, corresponding to a left-node of the local branching tree, can be generated and optimized, providing a new current incumbent solution and a new UB for the problem (x_{curr}, UB_{curr}). If this new incumbent solution is better than all previously obtained

solutions, it is deemed as (x_{best}, UB_{best}). If an optimal solution is found for the given search space, the leftnode constraint can be reversed into a right-node one. These steps can be repeated indefinitely until a pre-determined ending condition is met.

In our implementation, left-nodes are optimized with a solution limit (s_l), which implies that execution is aborted after several incumbent solutions are found. Since this leads to the α−opt neighborhood not being fully explored, the reversal of the left node constraint into a right-node one is not exact. Therefore, the previously generated left-node is transformed into constraint (38), which excludes the previous solution from the next sub-problems while still considering any previously unexplored regions.

$$\Delta(x, \bar{x}) \geq 1 \tag{38}$$

Details of our implementation are shown in Algorithm 3. An initial solution x_{init}, associated to an initial upper bound (UB_{init}), and the pre-determined number of vehicles n_k are fed to the algorithm, which allows left-nodes with an initial neighborhood size of α_{ini} to be optimized until a time limit of tl_{left} is reached or a new incumbent solution is found. It should be noted that even if the conventional goal of the method is to quickly reach optimality at left-nodes, in this framework, reaching optimality means that no improved solutions can be found. Hence, the current neighborhood is altered for further optimizations, having its size permanently increased by a value of α_{aux}. This size change procedure is also exploited when the left-node time limit is reached, dynamically changing the left-node time limit and parameter α_{aux}. Parameters *time* and $time_{in}$ track the overall time spent in the local branching algorithm and the time spent in the inner loop, respectively, while parameter node tracks how many left-nodes are executed.

ALGORITHM 3 – Local branching framework optimization

```
Local_branching(tl, tl_left, tl_in, nl_in, α_ini, α_aux, x_init, n_K, div);
x_best ← x_curr ← x_init;
UB_best ← UB_curr ← UB_init;
time ← 0, time_in ← 0, node ← 0, diversify ← 0 ;
while time_in < tl_in, node < nl_in and diversify < div do
    create left branching constraint using x_curr, n_K and α;
    optimize left-node for tl_in − time_in;
    if termination due to solution limit reached then
        reverse last branching constraint into Δ(x, x̄) ≥ 1;
        update (x_curr, UB_curr);
        if new best solution found then
            update (x_best, UB_best);
        end
    end
    if termination due to optimality then
        remove the last branching constraint;
        α = α + α_aux;
    end
    if termination due to time limit then
        remove the last branching constraint;
        fix all variables from the considered binary set and refine the solution;
        α = α + α_aux;
        α_aux = 2α_aux;
        tl_left = 2tl_left;
        diversify = diversify + 1;
        if diversify = div then
            node = nl_in;
        end
    end
    update time and time_in;
    node = node + 1;
end
remove all previously added branching constraints;
optimize right-node for tl − time;
update time;
if termination due to optimality then
    add all vehicles back to the model's formulation;
    optimize right-node for tl − time;
end
return (x_best, UB_best);
```

SOURCE: The Author (2021)

In summary, our local branching algorithm takes the base ideas of the original approach proposed by Fischetti and Lodi (2003) and implements an aggressive node exploration strategy, which results from the solution limit imposed on each sub-problem. This idea leads to more left-nodes explored per run and steeper UB improvements.

Finally, it should be noted that the algorithm proposed here is exact, just as the original one. What ensures optimality is optimizing the right-branch of the local branching tree, which corresponds to the complementary search space of the problem. As long as this final sub-problem is fully optimized, optimality is guaranteed. The fact that we do not explore entire left-node branches due to the imposed solution limit does

not affect the method's functionality, since constraint (38) only removes the current solution from the next sub-problems, and not any additional feasible ones.

2.6.3 Solution improvement algorithm

A solution improvement procedure (SI) improves any new incumbent solution identified by the model. Based on the algorithm presented by Archetti et al. (2012) and further developed in Guimaraes et al. (2019), a MIP sub-problem is solved using customer insertion and removal costs for all routes present in the solution, but without using arc and assignment variables. While variables I_i^t and x_i^{kt} remain unchanged, routing variables are replaced by variables f_i^{kt} and g_i^{kt}, representing whether customer *i* is removed from or inserted into route *k* in period *t*, respectively. The maximum number of changes that a route can go through is fixed to one, which leads to exact removal/insertion costs, represented by parameters a_i^{kt} and b_i^{kt}, respectively. This sub-problem is always generated upon a reference solution that can be recursively optimized, generating new reference solutions, until no more changes can be made to the neighborhood. The parameters and variables used by the model are described in Table 2.

TABLE 2 – NOTATION USED IN THE SOLUTION IMPROVEMENT MODEL

	Parameters
a_i^{kt}	Removal cost of customer *i* from route *k* in period *t*;
b_i^{kt}	Insertion cost of customer *i* in route *k* in period *t*;
r_i^{kt}	Binary parameter equal to 1 if customer *i* is in route *k* in period *t* in the original solution, 0 otherwise;
	Variables
f_i^{kt}	Binary variable equal to 1 if customer *i* is removed from route *k* in period *t*, 0 otherwise;
g_i^{kt}	Binary variable equal to 1 if customer *i* is inserted to route *k* in period *t*, 0 otherwise;

SOURCE: The Author (2021)

$$\min Z = -\sum_{t \in \tau} \sum_{k \in \kappa} \sum_{i \in C} a_i^{kt} f_i^{kt} + \sum_{t \in \tau} \sum_{k \in \kappa} \sum_{i \in C} b_i^{kt} g_i^{kt} \tag{39}$$

Subject to (4) and to:

$$q_i^{kt} \geq (U_i + d_i^t)\left(r_i^{kt} - f_i^{kt} + g_i^{kt}\right), i \in C, k \in \kappa, k \leq i, t \in \tau \quad (40)$$

$$q_i^{kt} \leq (U_i + d_i^t)\left(r_i^{kt} - f_i^{kt} + g_i^{kt}\right), i \in C, k \in \kappa, k \leq i, t \in \tau \quad (41)$$

$$q_i^{kt} \leq U_i \left(r_i^{kt} - f_i^{kt} + g_i^{kt}\right), i \in C, k \in \kappa, k \leq i, t \in \tau \quad (42)$$

$$g_i^{kt} \leq 1 - r_i^{kt}, i \in C, k \in \kappa, k \leq i, t \in \tau \quad (43)$$

$$f_i^{kt} \leq r_i^{kt}, i \in C, k \in \kappa, k \leq i, t \in \tau \quad (44)$$

$$\sum_{i \in C} f_i^{kt} + \sum_{i \in C} g_i^{kt} \leq 1, k \in \kappa, k \leq i, t \in \tau \quad (45)$$

$$\sum_{i \in C} q_i^{kt} \leq Q, k \in \kappa, k \leq i, t \in \tau \quad (46)$$

$$\sum_{i \in C} g_i^{kt} \leq 1, i \in C, t \in \tau \quad (47)$$

$$f_i^{kt} \in \{0,1\}, i \in C, k \in \kappa, k \leq i, t \in \tau \quad (48)$$

$$g_i^{kt} \in \{0,1\}, i \in C, k \in \kappa, k \leq i, t \in \tau \quad (49)$$

$$q_i^{kt} \in Z^+, i \in C, k \in \kappa, k \leq i, t \in \tau \quad (50)$$

The objective function (39) minimizes routing costs. Constraints (40) – (42) impose the OU replenishment policy. Constraints (43) guarantee that only customers who are not visited can be added to routes. Constraints (44) state that only customers who are visited can be removed from their respective routes. Constraints (45) limit the maximum number of changes per route, guaranteeing that all insertion and removal costs are taken into account. Constraints (46) ensure that vehicles' capacities are not violated. Constraints (47) prevent a customer from being added to multiple routes. Constraints (48) – (50) are domain and nature related.

The main differences between the formulation of the SI algorithm described here and the original formulation are as follows. Firstly, the symmetry breaking considerations are adapted to the problem, represented by the limited ranges of index *k* in the constraint sets. Secondly, to avoid infeasible and worse solutions throughout optimization, callback functions are employed to analyze all incumbent solutions, optimizing traveling salesman problem with time windows models for each route, consequently identifying and removing infeasible solutions

2.6.4 Solution framework

The optimization tools presented in this paper are organized and combined as follows. The TWT/AR procedures are executed to adjust customer time windows to tighter intervals and remove infeasible arcs. The ISH is used to provide an initial solution. These steps yield a polished initial solution, which is then fed to the local branching algorithm. The SI algorithm is applied, and the number of vehicles n_k is verified. Each time a new incumbent solution is found by local branching, the SI algorithm is executed recursively, further improving the solution. Whenever one of the preset diversification parameters is reached (inner loop time limit, inner loop node limit, or diversify parameter), a right node is executed. If an optimal solution is obtained from this right node, the number of vehicles returns to its original value, and a problem, equivalent to the original one, is optimized. Any lower bound values obtained during this step will be valid for the original problem since this version of the right-node is equivalent to the original IRPTW.

2.7 COMPUTATIONAL EXPERIMENTS

This section describes how the proposed improved local branching algorithm performs. All tests are run on an Intel Core i7-4790 CPU with 3.60GHz cores and 32 GB RAM. The algorithm is implemented in C++, and Gurobi is used as the MIP solver. A limit of four threads is set, and both tests, with and without local branching, have a maximum time limit of 7200s. First, tests are performed considering only the original model described in Section 2.4. These tests aimed to determine the computational gains of applying the valid IRP inequalities and the TWT/AR procedures. The results of these initial tests are used to tune the local branching parameters and the overall algorithm framework.

2.7.1 Test Instances

Instances created by Lappas, Kritikos, and Ioannou (2017) are based on the benchmark VRPTW instances of Solomon (1987) with the addition of IRP-related parameters, such as demands for each period, customers' holding capacities, and service times. The Solomon (1987) problem classes differ with respect to the disposition of the customers and the supplier: 'C' contains clusters of vertices, 'R' distributes them randomly, and 'RC' uses a combination of the two. The first number

after each letter (‘1’ or ‘2’) indicates the instance sub-class, and the next two ones indicate the time length of the customer time windows: in this case, the value ‘01’ indicates small time windows considered in all the original instance sets. Three different supply chain sizes are considered: 25, 50, and 100 customers. An instance example is ‘IRPTW C101_n25_p6’, which considers 25 customers separated in clusters under small-sized time windows. All instances have six time periods.

Lappas, Kritikos, and Ioannou (2017) consider an OU replenishment policy and demand values that can be higher than a customer’s inventory capacity. All customers start with full inventories at the first period and must have their inventories replenished during the last one to create a closed order cycle. The supplier’s inventory flow is also not considered in their formulation, meaning that the supplier’s initial inventory is sufficient to serve all customers during the planning horizon.

To ensure feasibility is guaranteed, the number of vehicles must be equal to the number of customers so that in the worst-case scenario, each customer is served by one vehicle, i.e., K = N, which leads to a very large model.

To test the limits of our methods, we also generate larger instances with 9 and 12 periods following the same patterns used in Lappas, Kritikos, and Ioannou (2017). These instances and detailed results are available upon request.

2.7.2 Results and analysis

Here, we first show the effectiveness of the methods proposed in the literature to decrease the computational time to solve the problem in Sections 2.7.2.1 and 2.7.2.2 and then analyze the results obtained by our proposed local branching algorithm in Section 2.6.2.

2.7.2.1 Original model with TWT/AR

First, we analyze the gains provided by the TWT/AR procedures described in Section 2.6. Table 3 shows a comparison between the number of arcs before and after the model’s pre-processing. Here, we show the number of iterations for the recursive algorithm and the number of changes made to the opening or closing times of the customers. The results between instances vary, and in some cases, no changes are made. Since these instance sub-classes (‘01’ and ‘02’) consider the smallest size time

windows, not many overlaps occur between them. When changes occur, they are usually related to those customers who have early opening times due to the supplier's operation hours. This finding indicates the importance of exploiting a more powerful solution algorithm.

TABLE 3 – CHANGES MADE TO THE MODEL BY THE TWT/AR PROCEDURES

Instance	TWT		AR		
	Cycles Executed	Changes	Original	Final	%
IRPTW_C101_n25_p6	2	2	109350	50250	45,95%
IRPTW_C201_n25_p6	1	0	109350	57150	52,26%
IRPTW_R101_n25_p6	2	1	109350	37650	34,43%
IRPTW_R201_n25_p6	2	1	109350	63750	58,30%
IRPTW_RC101_n25_p6	2	1	109350	45600	41,70%
IRPTW_RC201_n25_p6	2	1	109350	64350	58,85%
IRPTW_C101_n50_p6	2	4	811200	365700	45,08%
IRPTW_C201_n50_p6	1	0	811200	417600	51,48%
IRPTW_R101_n50_p6	2	2	811200	258000	31,80%
IRPTW_R201_n50_p6	2	5	811200	471300	58,10%
IRPTW_RC101_n50_p6	2	5	811200	265500	32,73%
IRPTW_RC201_n50_p6	2	1	811200	464400	57,25%
IRPTW_C101_n100_p6	2	8	6242400	2707200	43,37%
IRPTW_C201_n100_p6	2	2	6242400	3193800	51,16%
IRPTW_R101_n100_p6	2	4	6242400	1998600	32,02%
IRPTW_R201_n100_p6	2	9	6242400	3612000	57,86%
IRPTW_RC101_n100_p6	2	9	6242400	2243400	35,94%
IRPTW_RC201_n100_p6	2	8	6242400	3610200	57,83%
Average	1,89	3,50	2387649,96	1107025	47,01%

SOURCE: The Author (2021)

The last three columns show the number of arcs present in the original model, the number of removed arcs during pre-processing, and the percentage of remaining arcs at the end of the process, respectively. The number of original arcs is always the same for each instance size since all instances contain the same number of vertices (N + 2), vehicles (K = N), and periods (T). This step of the algorithm produces more evident gains, with an average arc removal rate of 53%. This step substantially reduces the model size and the solver's pre-processing time, which is essential to optimize a larger number of nodes of the local branching tree.

It can then be confirmed that these pre-processing tools result in an enhanced model, which is used in all further test configurations.

2.7.2.2 Enhanced model with valid inequalities

First, we used the mathematical model with no valid inequalities, pre-processing techniques, and initial solution heuristic, for which we obtained low quality solutions. Moreover, no solutions can even be obtained within the time limit for several medium and large-size instances.

Tests are executed with different valid inequality configurations to analyze the computational gains of applying them together with symmetry breaking constraints, the pre-processing techniques, and the ISH. In total, three different configurations are used: Coelho and Laporte (2015) + Coelho and Laporte (2014) valid inequalities (19) – (25), Lefever (2018) valid inequalities (26) – (32), and all valid inequalities. In order to facilitate the attainment of the first lower bound, with which the solver can work and start exploring the branch-and-bound (B&B) tree, all valid inequalities are added to the model after the root node is solved. Therefore, the first lower bound is estimated only by applying the pre-processing and ISH procedures, resulting in similar root nodes throughout all configurations. Even though better root node gaps can be achieved by adding the inequalities mentioned earlier directly to the model, this can significantly slow down the optimization process and even hinder the solver from obtaining the first lower bound. The results for instances in C100, C200 and R100 classes are presented in Table A1 and for instances in R200, RC100 and RC200 classes in Table A2 in the Appendix.

A summary of these results is presented in Table 4, where column Best UB shows how many times the configuration reached the best overall UB obtained by all three configurations, and column $Best^*$ evaluates configurations taking into account UB ties, setting as tiebreakers (1) the gap and (2) the runtime. Column Deviation is calculated as the average of $UB - UB_{best}$, where *UB* is the value obtained by the configuration and UB_{best} is the best overall value obtained. This calculation allows to determine which configuration had the best average performance, showing that the use of only valid inequalities (26) – (32) present the best average results. It is important to note that these valid inequalities act on the routing part of the problem, which is the hardest part of distribution problems with time windows.

TABLE 4 – COMPARISON BETWEEN TESTED CONFIGURATIONS

Config	Best UB	$Best^*$	Gap (%)	Runtime (s)	Root Gap (%)	Root Runtime (s)	Deviation
(19) – (25)	12	9	21,20	5.696,92	60,30	1.124,87	8,67
(26) – (32)	17	7	21,02	5.701,26	60,30	1.138,34	2,41
All	8	2	22,07	5.722,57	60,31	1.150,47	23,25

SOURCE: The Author (2021)

Valid inequalities (26) – (32) present superior results in terms of best upper bounds, final gaps, and, most importantly *Deviation*. Valid inequalities (19) – (25) perform best with smaller instances. This configuration presents the highest value of parameter $Best^*$, however its average performance is inferior to that of using inequalities (26) – (32), as the Deviation parameter shows. The configuration containing all inequalities presents the worst overall performance, both in terms of best solutions and of *Deviation*, as the model becomes too large with too many constraints.

For the next step of tests, it is chosen to proceed with configuration (26) – (32), since it presented the best average performance and because it reached the best overall upper bounds in 17 out of 18 instances.

Table 5 compares the best results obtained by the three enhanced model configurations presented so far with the method used in Lappas, Kritikos, and Ioannou (2017). All obtained results for instances with 25 and 50 customers are significantly better. Parameter ΔUB shows the UB improvement and is calculated as $((UB - UB_{Lappas})/ UB_{Lappas}) \times 100$. Among all new improvements, four new optimal solutions are obtained, which are highlighted in bold. The enhanced model, however, has difficulties dealing with some 100-customer instances from classes R201 and RC101.

TABLE 5 – COMPARISON BETWEEN THE RESULTS OF THE ENHANCED MODEL AND LAPPAS, KRITIKOS, AND IOANNOU (2017)

Instance	Lappas, Kritikos, and Ioannou (2017)	Exact Model		
	UB	UB	Gap (%)	ΔUB
IRPTW_C101_n25_p6	1.115,40	913,73	22,72	-18,08
IRPTW_C201_n25_p6	883,80	665,71	0,00	-24,68
IRPTW_R101_n25_p6	1.880,09	1.682,85	0,00	-10,49
IRPTW_R201_n25_p6	1.629,61	1.407,57	0,00	-13,63
IRPTW_RC101_n25_p6	1.892,24	1.628,87	11,70	-13,92
IRPTW_RC201_n25_p6	1.521,46	1.083,72	0,00	-28,77
IRPTW_C101_n50_p6	2.077,38	1.943,49	32,44	-6,44
IRPTW_C201_n50_p6	1.614,12	1.233,11	10,20	-23,60
IRPTW_R101_n50_p6	3.462,52	3.069,58	0,54	-11,35
IRPTW_R201_n50_p6	2.626,25	2.313,34	2,27	-11,91
IRPTW_RC101_n50_p6	3.907,75	3.582,81	24,18	-8,32
IRPTW_RC201_n50_p6	2.878,27	2.187,04	24,53	-24,02
IRPTW_C101_n100_p6	5.067,13	4.920,21	43,26	-2,90
IRPTW_C201_n100_p6	2.812,41	2.390,79	38,60	-14,99
IRPTW_R101_n100_p6	5.566,81	5.021,40	8,62	-9,80
IRPTW_R201_n100_p6	3.837,75	3.968,80	48,13	3,41
IRPTW_RC101_n100_p6	6.145,41	6.628,08	48,63	7,85
IRPTW_RC201_n100_p6	5.053,30	4.710,53	58,80	-6,78

SOURCE: The Author (2021)

2.7.2.3 Local branching algorithm

Parameter tuning of the local branching algorithm is a critical step in obtaining quality solutions. Some of these parameters include the α−opt neighborhood size used for each left-node, the αaux parameter, and both left-node (inner-loop) and overall time limits.

Aiming to improve the previously obtained UBs, we use only one inner-loop (a nonstop sequence of left-nodes, represented by lines 5 – 31 of Algorithm 3), and a right-node that can be triggered during the execution. The right-node of the branching is executed at the end of the local branching procedure, mainly being used to prove optimality.

To improve the lower bounds, a limit of 50 left-nodes is used together with the inner-loop time limit and the diversify parameter, triggering a right-node, if either of the associated limits is reached. As for the value of α, we set it to 50 after preliminary tests. A summary of the values used for the parameters in our implementation is shown in Table 6.

TABLE 6 – COMPARISON BETWEEN THE RESULTS OF THE ENHANCED MODEL AND LAPPAS, KRITIKOS, AND IOANNOU (2017)

Parameter	Value	Description
s_l	2	Solution limit
α	50	Initial α-opt neighborhood size
α_{aux}	65	Auxiliary neighborhood parameter
tl	7200s	Overall time-limit
tl_{left}	650s	Left-node time-limit
tl_{in}	∞	Inner-loop time-limit
nl_{in}	50	Inner-loop node limit
div	2	*diversify* parameter

SOURCE: The Author (2021)

Table 7 presents a comparison of the results obtained by Lappas, Kritikos, and Ioannou (2017), the enhanced model, and the proposed local branching framework. Our local branching algorithm improves the results of the enhanced model for 15 out of 18 instances and significantly improves the results of Lappas, Kritikos, and Ioannou (2017). The average improvement over the mathematical model is 4.66%; for large instances containing 100 customers, this improvement reaches an average value of 10.69%. The average improvement over the original benchmark is equal to 16.64%. It should be noted that, compared to the enhanced model, we observe that the local branching algorithm proves optimality for one extra instance, even though it reaches the same optimal upper bounds for all other instances.

TABLE 7 – RESULTS OBTAINED BY THE LOCAL BRANCHING ALGORITHM

Instance	Lappas, Kritikos, and Ioannou (2017)		Enhanced Model			Local Branching			
	UB	Runtime (s)	UB	Gap (%)	Runtime (s)	UB	Runtime (s)	ΔUB (%)	ΔSol (%)
IRPTW_C101_n25_p6	1.115,40	142,51	913,73	22,72	7.200,00	905,00	7.200,00	-0,96	-18,86
IRPTW_C201_n25_p6	883,80	165,43	665,71	0,00	330,47	665,71	7.200,00	0,00	-24,68
IRPTW_R101_n25_p6	1.880,09	68,96	1.682,85	0,00	3,37	1.682,85	165,50	0,00	-10,49
IRPTW_R201_n25_p6	1.629,61	135,56	1.407,57	0,00	510,37	1.407,57	7.200,00	0,00	-13,63
IRPTW_RC101_n25_p6	1.892,24	82,06	1.628,87	11,70	7.200,00	1.644,38	7.200,00	0,95	-13,10
IRPTW_RC201_n25_p6	1.521,46	213,60	1.083,72	0,00	862,55	1.083,72	7.200,00	0,00	-28,77
IRPTW_C101_n50_p6	2.077,38	1.082,70	1.943,49	32,44	7.200,00	1.792,81	7.200,00	-7,75	-13,70
IRPTW_C201_n50_p6	1.614,12	2.119,30	1.233,11	10,20	7.200,00	1.241,82	7.200,00	0,71	-23,06
IRPTW_R101_n50_p6	3.462,52	461,92	3.069,58	0,54	7.200,00	3.069,58	7.200,00	0,00	-11,35
IRPTW_R201_n50_p6	2.626,25	1.537,70	2.313,34	2,27	7.200,00	2.314,75	7.200,00	0,06	-11,86
IRPTW_RC101_n50_p6	3.907,75	556,18	3.582,81	24,18	7.200,00	3.322,96	7.200,00	-7,25	-14,96
IRPTW_RC201_n50_p6	2.878,27	2.241,00	2.187,04	24,53	7.200,00	2.066,25	7.200,00	-5,52	-28,21
IRPTW_C101_n100_p6	5.067,13	7.657,80	4.920,21	43,26	7.200,00	4.354,79	7.200,00	-11,49	-14,06
IRPTW_C201_n100_p6	2.812,41	39.252,00	2.390,79	38,60	7.200,00	2.242,83	7.200,00	-6,19	-20,25
IRPTW_R101_n100_p6	5.566,81	4.602,60	5.021,40	8,62	7.200,00	4.957,33	7.200,00	-1,28	-10,95
IRPTW_R201_n100_p6	3.837,75	15.402,00	3.968,80	48,13	7.200,00	3.384,14	7.200,00	-14,73	-11,82
IRPTW_RC101_n100_p6	6.145,41	5.214,00	6.628,08	48,63	7.200,00	5.554,28	7.200,00	-16,20	-9,62
IRPTW_RC201_n100_p6	5.053,30	12.655,00	4.710,53	58,80	7.200,00	4.039,57	7.200,00	-14,24	-20,06
Average	2.998,43	5.199,46	2.741,76	20,81	5.694,82	2.540,57	6.809,19	-4,66	-16,64

SOURCE: The Author (2021)

In Table 8, we present a performance comparison between the method used in Lappas, Kritikos, and Ioannou (2017) and our proposed optimization frameworks. While our enhanced model has the advantage of determining optimal solutions faster for smaller instances, the average performance of the local branching algorithm is superior, as indicated by the *Deviation* parameter. We have also generated new and larger instances as follows. Following the same procedure of Lappas, Kritikos, and Ioannou (2017), we created 12 instances with nine periods and 12 instances with 12 periods. These are by far the largest instances available for IRPs, even without time windows. We have six instances with 25 customers for each of these two sets and six instances with 50 customers. They are based on the Solomon instances as before, and for completeness, we also use all three configurations (C, R, and RC). These instances and the detailed results described next are available upon request.

TABLE 8 – SUMMARY OF THE PERFORMANCE COMPARISON

Config	UB_{best}	$Best^*$	***Deviation***
Lappas, Kritikos, and Ioannou (2017)	0	0	473,19
Enhanced Model	8	3	216,52
Local Branching	15	15	1,42

SOURCE: The Author (2021)

Table 9 presents the results for the instances with nine periods. We have run our enhanced model and our local branching algorithm for 7200s. Three of these instances have been solved to optimality, and average results indicate that our local branching algorithm provides solutions almost 4% better, with individual cases of improvements of more than 17%. In no case has our local branching provided worse solutions.

TABLE 9 – RESULTS FOR INSTANCES WITH NINE PERIODS

Instance	Enhanced Model			Local Branching		
	UB	Gap (%)	Runtime (s)	UB	Runtime (s)	ΔUB (%)
IRPTW_C101_n25_p9	1.374,60	23,80	7.200,00	1.335,31	7.200,00	-2,86
IRPTW_C201_n25_p9	**997,62**	**0,00**	**3.273,42**	**997,62**	**7.200,00**	**0,00**
IRPTW_R101_n25_p9	**2.358,82**	**0,00**	**2,86**	**2.358,82**	**187,36**	**0,00**
IRPTW_R201_n25_p9	**2.024,78**	**0,00**	**1.733,07**	**2.024,78**	**7.200,00**	**0,00**
IRPTW_RC101_n25_p9	2.434,00	13,03	7.200,00	2.413,96	7.200,00	-0,82
IRPTW_RC201_n25_p9	1.654,77	2,22	7.200,00	1.654,77	7.200,00	0,00
IRPTW_C101_n50_p9	2.831,74	29,70	7.200,00	2.702,84	7.200,00	-4,55
IRPTW_C201_n50_p9	2.062,97	18,82	7.200,00	1.861,52	7.200,00	-9,76
IRPTW_R101_n50_p9	4.468,35	1,16	7.200,00	4.466,88	7.200,00	-0,03
IRPTW_R201_n50_p9	3.461,84	8,93	7.200,00	3.401,72	7.200,00	-1,74
IRPTW_RC101_n50_p9	5.289,00	24,42	7.200,00	4.850,69	7.200,00	-8,29
IRPTW_RC201_n50_p9	3.855,41	45,93	7.200,00	3.184,03	7.200,00	-17,41
Average	2.734,49	14,00	5.817,45	2.604,41	6.615,61	-3,79

SOURCE: The Author (2021)

The most challenging instances with 12 periods have their results shown in Table 10. Here, two instances are solved to optimality, and despite very similar runtimes between our local branching algorithm and that of the enhanced model, the improvements provided by our method are even more significant on smaller instances. The average improvement now is more than 4.5%, with individual improvements of more than 20%. These results indicate that our method can handle very large instances and that it does so in a much more efficient way than the (already enhanced) mathematical programming model.

TABLE 10 – RESULTS FOR INSTANCES WITH 12 PERIODS

Instance	Enhanced Model			Local Branching		
	UB	Gap (%)	Runtime (s)	UB	Runtime (s)	ΔUB (%)
IRPTW_C101_n25_p12	1.803,07	23,46	7.200,00	1.801,67	7.200,00	-0,08
IRPTW_C201_n25_p12	1.331,41	3,10	7.200,00	1.331,41	7.200,00	0,00
IRPTW_R101_n25_p12	**3.152,22**	**0,00**	**10,99**	**3.152,22**	**435,51**	**0,00**
IRPTW_R201_n25_p12	**2.657,20**	**0,00**	**6.147,14**	**2.657,20**	**7.200,00**	**0,00**
IRPTW_RC101_n25_p12	3.265,51	14,79	7.200,00	3.161,14	7.200,00	-3,20
IRPTW_RC201_n25_p12	2.199,75	4,14	7.200,00	2.173,88	7.200,00	-1,18
IRPTW_C101_n50_p12	3.840,54	32,57	7.200,00	3.528,66	7.200,00	-8,12
IRPTW_C201_n50_p12	2.685,23	19,61	7.200,00	2.507,96	7.200,00	-6,60
IRPTW_R101_n50_p12	5.921,98	3,21	7.200,00	5.949,81	7.200,00	0,47
IRPTW_R201_n50_p12	4.703,78	11,86	7.200,00	4.504,39	7.200,00	-4,24
IRPTW_RC101_n50_p12	7.186,48	27,38	7.200,00	6.345,65	7.200,00	-11,70
IRPTW_RC201_n50_p12	5.285,92	53,14	7.200,00	4.211,64	7.200,00	-20,32
Average	3.669,42	16,11	6.513,18	3.443,80	6.636,29	-4,58

SOURCE: The Author (2021)

2.8 CONCLUSION

In this paper, we have proposed an exact optimization approach to solve the IRPTW. Our method combines different valid inequalities, pre-processing techniques, a heuristic initial solution procedure, a solution improvement procedure, and a local branching algorithm. Our method was tested on a recent benchmark set from the literature, specifically introduced for this problem. We have also introduced larger instances containing up to 12 periods, much larger than any other available IRP instance. We have obtained significant improvements and provided new best-known solutions for all instances in this dataset, besides proving optimal solutions for some of them. We have demonstrated through detailed experiments that the many different aspects of our algorithm contribute to its success.

Aside from the contributions in terms of performance, this paper is also the first, to the best of our knowledge, to integrate symmetry breaking considerations and TWT/AR into the IRPTW. These pre-processing techniques significantly reduce the complexity of the computational model, resulting in better bounds for the problem. Additionally, the newly proposed local branching algorithm, which focuses on the noncomplete exploration of sub-problems, showed high efficiency and potential for further exploration in future studies.

An important suggestion for further research is applying such high-performance solution algorithms to real industrial cases. It might also prove beneficial to assess businesses from an economic perspective and assess environmental gains such as reduced greenhouse gas emissions, less fuel consumption, and the usage of fewer trucks that contribute to congestion and pollution.

2.9 DATA AVAILABILITY STATEMENT

The authors confirm that the data supporting the findings of this study are available within the article and/or its supplementary materials as indicated in the paper, further inquiries can be directed to the corresponding author.

2.10 ACKNOWLEDGEMENTS

This project was partly funded by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2020-00401 and 2019-00094. This support is greatly acknowledged. We thank Calcul Quebec for providing high performance computing resources. We thank the editor and three anonymous referees for their valuable suggestions on an earlier version of this paper.

2.11 REFERENCES

ANDERSSON, H., A. HOFF, M. CHRISTIANSEN, G. HASLE, AND A. LØKKETANGEN. 2010. **Industrial aspects and literature survey: Combined inventory management and routing**. Computers & Operations Research 37 (9): 1515–1536.

ARCHETTI, C., L. BERTAZZI, A. HERTZ, AND M.G. SPERANZA. 2012. **A hybrid heuristic for an inventory routing problem**. INFORMS Journal on Computing 24 (1): 101–116.

ASCHEUER, N., M. FISCHETTI, AND M. GROTSCHEL. 2001. **Solving the asymmetric travelling salesman problem with time windows by branch-and-cut**. Mathematical Programming 90: 475– 506.

BELL, W.J., L.M. DALBERTO, AND M.L. FISHER. 1983. **Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer**. Interfaces 13 (6): 4–23.

BERTAZZI, L., L.C. COELHO, A. DE MAIO, AND D. LAGANA. 2019. **A matheuristic algorithm for the multi-depot inventory routing problem**. Transportation Research Part E: Logistics and Transportation Review 122: 524–544.

COELHO, L.C., J.-F. CORDEAU, AND G. LAPORTE. 2014. **Thirty years of inventory routing**. Transportation Science 48 (1): 1–19.

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

COELHO, L.C., AND G. LAPORTE. 2015. **An optimised target-level inventory replenishment policy for vendor-managed inventory systems**. International Journal of Production Research 53 (12): 3651–3660.

DARVISH, M., C. ARCHETTI, AND L.C. COELHO. 2019. **Trade-offs between environmental and economic performance in production and inventory-routing problems**. International Journal of Production Economics 217: 269–280.

DESAULNIERS, G., J.G. RAKKE, AND L.C. COELHO. 2016. **A branch-price-and-cut algorithm for the inventory-routing problem**. Transportation Science 50 (3): 1060–1076.

FARIAS, K., K. HADJ-HAMOU, AND C. YUGMA. 2021. **Model and exact solution for a two-echelon inventory routing problem**. International Journal of Production Research 59 (10): 3109– 3132.

FISCHETTI, M., AND A. LODI. 2003. **Local branching**. Mathematical Programming 98 (1-3): 23–47.

GUIMARAES, T.A., L.C. COELHO, C.M. SCHENEKEMBERG, AND C.T. SCARPIN. 2019. **The two-echelon multi-depot inventory-routing problem**. Computers & Operations Research 101 (1): 220– 223.

GUIMARAES, T. A., C. M. SCHENEKEMBERG, LEANDRO C. COELHO, C. SCARPIN, J. E. PECORA, AND C. ARCHETTI. 2020. **Mechanisms for feasibility and improvement for inventory-routing problems**. Technical Report CIRRELT-2020-12. Montreal, Canada.

HANSEN, P., N. MLADENOVIC, AND D. UROSEVIC. 2006. **Variable neighborhood search and local branching**. Computers & Operations Research 33 (10): 3034–3045.

HERNANDEZ, F., M. GENDREAU, O. JABALI, AND W. REI. 2019. **A local branching matheuristic for the multi-vehicle routing problem with stochastic demands**. Journal of Heuristics 25: 215–245.

KHEIRI, A. 2020. **Heuristic sequence selection for inventory routing problem**. Transportation Science 54 (2): 302–312.

LAPPAS, P.Z., M.N. KRITIKOS, AND G.D. IOANNOU. 2017. **A two-phase solution algorithm for the inventory routing problem with time windows**. Journal of Mathematics and System Science 7 (9).

LEFEVER, W. 2018. **Stochastic and Robust Optimization Algorithms for the Inventory-Routing Problem and its Extensions**. PhD diss., Ghent University.

LEFEVER, W., F. A. TOUZOUT, KH. HADJ-HAMOU, AND E.-H. AGHEZZAF. 2019. **Benders' decomposition for robust travel time-constrained inventory routing problem**. International Journal of Production Research 1–25.

LI, K., B. CHEN, A.I. SIVAKUMAR, AND Y. WU. 2014. **An inventory-routing problem with the objective of travel time minimization**. European Journal of Operational Research 236 (3): 936–945.

LI, P., J. HE, D. ZHENG, Y. HUANG, AND C. FAN. 2015. **Vehicle routing problem with soft time windows based on improved genetic algorithm for fruits and vegetables distribution**. Discrete Dynamics in Nature and Society 2015: 18.

LI, Y., F. CHU, J.-F. COTE, L. C. COELHO, AND C. CHU. 2020. **The multi-plant perishable food production routing with packaging consideration**. International Journal of Production Economics 221: 107472.

LIU, S.C., AND W.T. LEE. 2011. **A heuristic method for the inventory routing problem with time windows**. Expert Systems with Applications 38 (10): 13223–13231.

MANOUSAKIS, E., P. REPOUSSIS, E. ZACHARIADIS, AND C. TARANTILIS. 2021. **Improved branch and-cut for the inventory routing problem based on a two-commodity flow formulation**. European Journal of Operational Research 290 (3): 870–885.

OSVALD, A., AND L. STIRN. 2008. **A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food**. Journal of Food Engineering 85: 285–295.

PARASKEVOPOULOS, D.C., P.P. REPOUSSIS, C.D. TARANTILIS, G. IOANNOU, AND G.P. PRASTACOS. 2008. **A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows**. Journal of Heuristics 14 (5): 425–455.

REPOUSSIS, P.P., AND C.D. TARANTILIS. 2010. **Solving the fleet size and mix vehicle routing problem with time windows via adaptive memory programming**. Transportation Research Part C: Emerging Technologies 18 (5): 695–712.

RODRIGUEZ-MARTIN, I., AND J. J. SALAZAR-GONZALEZ. 2010. **A local branching heuristic for the capacitated fixed-charge network design problem**. Computers & Operations Research 37 (3): 575–581.

SAMAVATI, M., D. ESSAM, M. NEHRING, AND R. SARKER. 2017. **A local branching heuristic for the open pit mine production scheduling problem**. European Journal of Operational Research 257 (1): 261–271.

SHAABANI, H., AND I. N. KAMALABADI. 2016. **An efficient population-based simulated annealing algorithm for the multi-product multi-retailer perishable inventory routing problem**. Computers & Industrial Engineering 99: 189–201.

SOLOMON, M.M. 1987. **Algorithms for the vehicle routing and scheduling problems with time window constraints**. Operations Research 35 (2): 254–265.

SOYSAL, M. 2016. **Closed-loop inventory routing problem for returnable transport items**. Transportation Research Part D: Transport and Environment 48: 31–45.

SOYSAL, M., M. C, IMEN, S. BELBAG, AND E. TOGRUL. 2019. **A review on sustainable inventory routing**. Computers & Industrial Engineering 132: 395–411.

2.12 APPENDIX A. RESULTS

TABLE A1 – RESULTS FOR THE ENHANCED MODEL FOR INSTANCES CLASSES C100, C200 AND R100

Test	Configuration	UB	LB	Gap (%)	Runtime (s)	ISH UB	ISH LB	Root Gap (%)	Root Runtime (s)
IRPTW_C101_n25_p6	(19) - (25)	931,29	708,76	23,89	7.200,00	1.141,56	16,68	98,54	0,34
	(26) - (32)	**913,73**	**706,15**	**22,72**	**7.200,00**	**1.141,56**	**16,68**	**98,54**	**0,34**
	All	933,83	708,49	24,13	7.200,00	1.141,56	16,68	98,54	0,36
IRPTW_C101_n50_p6	(19) - (25)	1.979,52	1.309,27	33,86	7.200,00	2.069,20	1.100,83	46,80	57,71
	(26) - (32)	**1.943,49**	**1.312,93**	**32,44**	**7.200,00**	**2.069,20**	**1.100,83**	**46,80**	**59,04**
	All	1.956,46	1.313,72	32,85	7.200,00	2.069,20	1.100,83	46,80	69,23
IRPTW_C101_n100_p6	(19) - (25)	4.920,21	2.791,69	43,26	7.200,00	5.007,58	2.573,13	48,62	2.746,74
	(26) - (32)	4.920,21	2.791,69	43,26	7.200,00	5.007,58	2.573,13	48,62	2.777,07
	All	**4.920,21**	**2.791,69**	**43,26**	**7.200,00**	**5.007,58**	**2.573,13**	**48,62**	**2.618,27**
IRPTW_C201_n25_p6	**(19) - (25)**	**665,71**	**665,71**	**0,00**	**330,47**	**834,94**	**495,11**	**40,70**	**3,71**
	(26) - (32)	665,71	665,71	0,00	444,34	834,94	495,11	40,70	3,72
	All	665,71	665,71	0,00	388,50	834,94	495,11	40,70	3,61
IRPTW_C201_n50_p6	(19) - (25)	1.261,65	1.109,59	12,05	7.200,00	1.519,41	818,14	46,15	37,61
	(26) - (32)	1.276,46	1.098,50	13,94	7.200,00	1.519,41	818,14	46,15	38,71
	All	**1.233,11**	**1.107,39**	**10,20**	**7.200,00**	**1.519,41**	**818,14**	**46,15**	**41,29**
IRPTW_C201_n100_p6	**(19) - (25)**	**2.390,79**	**1.468,05**	**38,60**	**7.200,00**	**2.641,07**	**1.468,05**	**44,41**	**5.643,42**
	(26) - (32)	2.390,79	1.468,05	38,60	7.200,00	2.641,07	1.468,05	44,41	5.730,40
	All	2.444,49	1.468,05	39,94	7.200,00	2.641,07	1.468,05	44,41	5.659,38
IRPTW_R101_n25_p6	(19) - (25)	1.682,85	1.682,85	0,00	3,38	1.852,85	6,66	99,64	0,20
	(26) - (32)	**1.682,85**	**1.682,85**	**0,00**	**3,37**	**1.852,85**	**6,66**	**99,64**	**0,19**
	All	1.682,85	1.682,85	0,00	4,10	1.852,85	6,66	99,64	0,19
IRPTW_R101_n50_p6	**(19) - (25)**	**3.069,58**	**3.052,92**	**0,54**	**7.200,00**	**3.488,38**	**45,65**	**98,69**	**1,94**
	(26) - (32)	3.069,58	3.051,48	0,59	7.200,00	3.488,38	45,65	98,69	2,00
	All	3.069,58	3.052,57	0,55	7.200,00	3.488,38	45,65	98,69	1,67
IRPTW_R101_n100_p6	**(19) - (25)**	**5.021,40**	**4.588,61**	**8,62**	**7.200,00**	**5.524,63**	**3.327,72**	**39,77**	**587,47**
	(26) - (32)	5.021,40	4.588,61	8,62	7.200,00	5.524,63	3.327,72	39,77	595,95
	All	5.052,13	4.535,30	10,23	7.200,00	5.524,63	3.327,72	39,77	766,51

SOURCE: The Author (2021)

TABLE A2 – RESULTS FOR THE ENHANCED MODEL FOR INSTANCES CLASSES R200, RC100 AND RC200

Test	Configuration	UB	LB	Gap (%)	Runtime (s)	ISH UB	ISH LB	Root Gap (%)	Root Runtime (s)
IRPTW_R201_n25_p6	(19) - (25)	1.407,57	1.407,57	0,00	548,17	1.658,26	868,57	47,62	3,60
	(26) - (32)	**1.407,57**	**1.407,57**	**0,00**	**510,37**	**1.658,26**	**868,57**	**47,62**	**3,58**
	All	1.407,57	1.407,57	0,00	831,46	1.658,26	868,57	47,62	3,70
IRPTW_R201_n50_p6	(19) - (25)	2.326,83	2.248,14	3,38	7.200,00	2.897,28	1.485,17	48,74	45,84
	(26) - (32)	**2.313,34**	**2.260,79**	**2,27**	**7.200,00**	**2.897,28**	**1.485,17**	**48,74**	**46,44**
	All	2.325,98	2.231,74	4,05	7.200,00	2.897,28	1.485,17	48,74	47,00
IRPTW_R201_n100_p6	**(19) - (25)**	**3.968,80**	**2.058,51**	**48,13**	**7.200,00**	**4.379,45**	**2.058,51**	**53,00**	**5.489,75**
	(26) - (32)	3.968,80	2.058,51	48,13	7.200,00	4.379,45	2.058,51	53,00	5.533,60
	All	4.143,87	2.058,51	50,32	7.200,00	4.379,45	2.058,51	53,00	5.989,50
IRPTW_RC101_n25_p6	(19) - (25)	1.629,35	1.440,60	11,58	7.200,00	1.864,24	1.267,93	31,99	5,69
	(26) - (32)	**1.628,87**	**1.438,24**	**11,70**	**7.200,00**	**1.864,24**	**1.267,93**	**31,99**	**5,69**
	All	1.641,88	1.439,18	12,35	7.200,00	1.864,24	1.267,93	31,99	5,48
IRPTW_RC101_n50_p6	**(19) - (25)**	**3.582,81**	**2.716,61**	**24,18**	**7.200,00**	**3.748,99**	**7,07**	**99,81**	**3,19**
	(26) - (32)	3.582,81	2.716,12	24,19	7.200,00	3.748,99	7,07	99,81	3,19
	All	3.583,53	2.718,19	24,15	7.200,00	3.748,99	7,07	99,81	3,14
IRPTW_RC101_n100_p6	**(19) - (25)**	**6.628,08**	**3.404,61**	**48,63**	**7.200,00**	**6.708,90**	**2.855,10**	**57,44**	**1.482,08**
	(26) - (32)	6.628,08	3.404,61	48,63	7.200,00	6.708,90	2.855,10	57,44	1.497,63
	All	6.667,76	3.418,42	48,73	7.200,00	6.723,89	2.855,10	57,54	1.342,25
IRPTW_RC201_n25_p6	**(19) - (25)**	**1.083,72**	**1.083,72**	**0,00**	**862,55**	**1.377,83**	**579,98**	**57,91**	**5,82**
	(26) - (32)	1.083,72	1.083,72	0,00	864,56	1.377,83	579,98	57,91	5,81
	All	1.083,72	1.083,72	0,00	982,15	1.377,83	579,98	57,91	6,24
IRPTW_RC201_n50_p6	(19) - (25)	2.246,99	1.662,47	26,01	7.200,00	2.686,14	939,80	65,01	128,33
	(26) - (32)	**2.187,04**	**1.650,57**	**24,53**	**7.200,00**	**2.686,14**	**939,80**	**65,01**	**130,74**
	All	2.246,99	1.667,71	25,78	7.200,00	2.686,14	939,80	65,01	129,18
IRPTW_RC201_n100_p6	**(19) - (25)**	**4.710,53**	**1.940,91**	**58,80**	**7.200,00**	**4.928,66**	**1.940,91**	**60,62**	**4.004,26**
	(26) - (32)	4.710,53	1.940,91	58,80	7.200,00	4.928,66	1.940,91	60,62	4.056,04
	All	4.710,53	1.940,91	58,80	7.200,00	4.928,66	1.940,91	60,62	4.021,50

SOURCE: The Author (2021)

3 FINAL REMARKS

This dissertation proposes an exact resolution approach for the IRPTW. Different groups of valid inequalities developed for the IRP, pre-processing techniques, solution improvement heuristics, and a Local branching algorithm, are employed to solve the problem. The optimization model can efficiently explore reduced neighborhoods and obtain consistent improvements for the objective function of the problem.

To our best knowledge, this is also the first project to compare results with benchmark instances developed exclusively for the IRPTW. The interaction between all the tools employed alongside the model can also be considered a contribution to the literature. Valid inequalities associated with inventory control and symmetry break have their trade-offs analyzed in our testing scenarios. It is observed that using only the valid inequalities proposed by Lefever (2018) provide better results, since the interaction between all groups at the same time result in a greater model complexity, but no proportional gains in terms of results. One reason that might explain this is the fact that symmetry break is applied as a pre-processing technique in our model, which leads to many inequalities becoming redundant in the system.

The pre-processing techniques used by Ascheuer et al. (2001), especially those designed to eliminate unfeasible arcs, generate huge computational gains to the performance of the mathematical model, both in terms of memory consumed as in terms of optimization time. The consequences of this more efficient model and the valid inequalities result in the root node being determined faster, which in turn speeds up the initialization of the Local branching algorithm.

Another important aspect of the optimization framework developed are the methods used to obtain improved initial solutions. The ISH and the SI model help to find a feasible solution in near instant time, and to quickly refine it into a good starting point for the model.

Even though the original Local branching algorithm was presented more than 10 years ago, it still lacks studies where it is applied to routing problems. By restricting the number of binary variables considered by this algorithm and applying a custom strategy for the exploration of the algorithm's tree, an aggressive and effective exploration of the search space is achieved. This strategy, which is also a contribution from this research project, is proven to obtain improved results over the base

mathematical model, and the optimization heuristic proposed by Lappas, Kritikos, and Ioannou (2017).

Regarding the tests executed, the optimization framework can consistently beat the existing times for the benchmark instances studied, providing gains objective function gains up to 20% depending on the instance. Plus, 24 new instances were developed for the IRPTW, these considering even larger period horizons.

Real world scenarios could profit from the developments of this research. The fact that the algorithm developed here is exact in nature, it can provide gaps and upper/lower limits for its calculations, and it could also be further extended by other optimization tools. The constant technological evolution mentioned in Chapter 1 will eventually increase the complexity of the IRPTW. Resolution approaches that target the root problem, such as this, helps tackling these more complex variants, since they can be treated as a subproblem of a much more complex one.

Finally, the Local branching method has a high adaptability, and can be used in pretty much any problem from the Operations Research field of study. The improvements suggested to this algorithm by this research can potentially be employed in many other optimization problems, allowing complex operations to be better controlled by supply chain managers.

3.1 SUGGESTIONS FOR FUTURE RESEARCH

Two main groups of valid inequalities were used in this project. These groups are closely related to the IRP, but that does not mean that valid inequalities must be related to the IRP to provide positive performance gains to the optimization model. Inequalities designed for problems such as the VRP or VRPTW could be implemented into the model and potentially boost performance even further.

The ISH has simple and efficient mechanisms to determine a first feasible solution. More robust heuristics originated from the IRP or VRP could be adapted into the model to provide more substantial gains to the initial upper bound of the problem. Other heuristics could also be incorporated into the aggressive exploration strategy of our Local branching algorithm.

To correct one flaw of this aggressive strategy, calculations could be developed to determine how much of the current neighborhood was effectively explored, so that a constraint could be added to the model to remove it from future

iterations of the algorithm. Lastly, the base B&B used to solve the resulting mathematical models could be replaced with more robust methods, such as the branch-and-price and cut (BP&C) model.

BIBLIOGRAPHY

ARCHETTI, C.; SPERANZA, M. G. **The inventory routing problem: the value of integration: The inventory routing problem: the value of integration**. International Transactions in Operational Research, v. 23, n. 3, p. 393–407, 2016.

ALTEXSOFT. **Fleet Management Software: Key Functions, Solutions, and Innovations,** 2019. Disponível em <https://www.altexsoft.com/blog/business/fleet-management-software-key-functions-solutions-and-innovations/>. Acesso: 11/2/2021.

ALTEXSOFT. **How to Solve Vehicle Routing Problems: Route Optimization Software and Their APIs**, 2019. Disponível em: <https://www.altexsoft.com/blog/business/how-to-solve-vehicle-routing-problems-route-optimization-software-and-their-apis/>. Acesso em: 11/2/2021.

ALVAREZ, P.; LERGA, I.; SERRANO-HERNANDEZ, A.; FAULIN, J. **The impact of traffic congestion when optimising delivery routes in real time. A case study in Spain.** International Journal of Logistics Research and Applications, v. 21, n. 5, p. 529–541, 2018.

ASCHEUER, N., M. FISCHETTI, AND M. GROTSCHEL. 2001. **Solving the asymmetric travelling salesman problem with time windows by branch-and-cut**. Mathematical Programming 90: 475– 506.

BELHASSINE, K.; COELHO, L. C.; RENAUD, J.; GAGLIARDI, J.-P. **Improved Home Deliveries in Congested Areas Using Geospatial Technology**. fev. 2018.

COELHO, L. C.; CORDEAU, J.-F.; LAPORTE, G. **Thirty Years of Inventory Routing.** Transportation Science, v. 48, n. 1, p. 1–19, 2014.

GUIMARAES, T.A., L.C. COELHO, C.M. SCHENEKEMBERG, AND C.T. SCARPIN. 2019. **The two-echelon multi-depot inventory-routing problem**. Computers & Operations Research 101 (1): 220– 223.

HENI, H.; DIOP, S. A.; COELHO, L. C.; RENAUD, J. **Measuring Fuel Consumption in Vehicle Routing: New Estimation Models Using Supervised Learning.** Mar. 2018.

LACHTERMACHER, G. **Pesquisa Operacional Na Tomada de Decisões.** 5º ed. Rio de Janeiro: LTC, 2016.

KOK, A. L.; HANS, E. W.; SCHUTTEN, J. M. J. **Vehicle routing under time-dependent travel times: The impact of congestion avoidance.** Computers & Operations Research, v. 39, n. 5, p. 910–918, 2012.

LAPPAS, P.Z., M.N. KRITIKOS, AND G.D. IOANNOU. 2017. **A two-phase solution algorithm for the inventory routing problem with time windows**. Journal of Mathematics and System Science 7 (9).

LEFEVER, W. 2018. **Stochastic and Robust Optimization Algorithms for the Inventory-Routing Problem and its Extensions**. PhD diss., Ghent University.

SCAVARDA, L. F. R.; HAMACHER, S. **Evolução da cadeia de suprimentos da indústria automobilística no Brasil.** Revista de Administração Contemporânea, v. 5, n. 2, p. 201–219, 2001.