Thesis presented to the *Instituto Tecnológico de Aeronáutica*, in partial fulfillment of the requirements for the degree of Doctor of Science in the Graduate Program of *Engenharia Eletrônica e Computação, Field of Informática (EEC-I).*

Antonio Celio Pereira de Mesquita

AIR CARGO LOAD AND ROUTE PLANNING IN PICKUP-DELIVERY OPERATIONS

Thesis approved in its final version by the signatory below:

[illegible]

Prof. Dr. Carlos Alberto Alonso Sanches

Advisor

Campo Montenegro
São José dos Campos, SP - Brazil
2024

Cataloging-in Publication Data
Documentation and Information Division

Mesquita, Antonio Celio Pereira de
Air Cargo Load and Route Planning in Pickup-Delivery Operations / Antonio Celio Pereira de Mesquita.
São José dos Campos, 2024.
110f.

Thesis of Doctor of Science – Course of *Engenharia Eletrônica e Computação. Area of Informática (EEC-I) – Instituto Tecnológico de Aeronáutica, 2024. Advisor: Prof. Dr. Carlos Alberto Alonso Sanches.*

1. Air Cargo Load Planning. 2. OR in Airlines. 3. Vehicle Routing. 4. Weight and Balance. 5. Pickup and Delivery. 6. Mixed-Integer Programming. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

MESQUITA, Antonio Celio Pereira de. **Air Cargo Load and Route Planning in Pickup-Delivery Operations**. 2024. 110f. Thesis of Doctor of Science – *Instituto Tecnológico de Aeronáutica, São José dos Campos.*

CESSION OF RIGHTS

AUTHOR'S NAME: Antonio Celio Pereira de Mesquita
PUBLICATION TITLE: Air Cargo Load and Route Planning in Pickup-Delivery Operations.
PUBLICATION KIND/YEAR: Thesis / 2024

It is granted to *Instituto Tecnológico de Aeronáutica* permission to reproduce copies of this thesis and to only loan or to sell copies for academic and scientific purposes. The author reserves other publication rights and no part of this thesis can be reproduced without the authorization of the author.

Antonio Celio Pereira de Mesquita
celio@ita.br
Instituto Tecnológico de Aeronáutica - AeroLogLab
Praça Mal. Eduardo Gomes, 50
12.228-900 – São José dos Campos – SP – Brazil

AIR CARGO LOAD AND ROUTE PLANNING IN PICKUP-DELIVERY OPERATIONS

Antonio Celio Pereira de Mesquita

Thesis Committee Composition:

President:	Prof. Dr.	Mariá C. V. Nascimento Rosset	ITA
Advisor:	Prof. Dr.	Carlos Alberto Alonso Sanches	ITA
Internal member:	Prof. Dr.	Leonardo Ramos Rodrigues	IAE/DCTA
External members:	Dr.	Pedro Augusto Munari Junior	UFSCar
	Dr.	Flávio Keidi Miyazawa	UNICAMP

ITA

I dedicate this work to my wife, *Givani de Barros Mesquita*. Without her support and love this work would never be possible.

Acknowledgments

First, I would like to thank the Almighty God for providing me with continuous courage, strength, and support at each step in my whole life.

I would like to thank my advisor, Prof. Dr. Carlos Alberto Alonso Sanches, for his excellent guidance, helpful comments, and generous assistance and support during our research period. Without his guidance, I could not have written this piece of work as it is now, nor could the results have exceeded the previously expected level of quality. His insightful comments on the mathematical formulation and the complexity analysis significantly improved the scientific rigor of my research approach.

It is also essential to highlight the contributions of professors Mischel Carmen Neyra Belderrain and Leonardo Ramos Rodrigues for their many insights during the qualification exam.

To Professors Flávio Keidi Miyazawa and Pedro Augusto Munari Junior (external members) and Prof. Mariá Nascimento Rosset (President of the Committee), I would like to express my sincere gratitude for their exceptional contributions during my thesis defense. Their insightful questions and valuable suggestions provided the final touch that was lacking for the scientific rigor and quality required for a Doctor of Science Thesis.

This research was partially supported by *São Paulo Research Foundation* (FAPESP, grant 2022/05803-3).

Success is born of wanting, determination,
and persistence in reaching a goal.
Even if they do not reach the target,
those who seek and overcome obstacles
will at least do admirable things.
— José de Alencar, a Brazilian lawyer

Abstract

This thesis addresses the critical challenge of planning and executing aerial pickup and delivery of goods across Brazil's vast territory, a complex process often jeopardized by rapid loading requirements and the risks of cargo unbalancing and misdelivery. Faced with the urgency of rapid takeoffs and the complexities of distributing goods over Brazil's extensive territory, this research tackles the unexplored problem of optimizing air cargo load planning with routing, pickup, and delivery, balancing utility scores, cargo weight, and fuel consumption against the constraints of aircraft capacity and center of gravity. We introduce a comprehensive model that integrates air palletization, weight and balance, pickup and delivery, and vehicle routing into a unified framework. Through rigorous mathematical modeling and the development of innovative sequential and parallel heuristics, our approach minimizes both ground handling times and fuel consumption, directly contributing to reduced carbon emissions. Our methods were validated through extensive testing with both commercial solvers and metaheuristics, using data reflective of real-world scenarios from the Brazilian Air Force. These data contain a variety of items, like aircraft components and supplies, medication, and other supplies for remote population needs in isolated regions in the Brazilian territory, as well as for resupplying the defense forces near the Brazilian border. Despite issues such as incomplete data and historical records, our findings demonstrate the practical applicability and adaptability of our solutions to a broad range of logistical and optimization challenges. This research not only advances the field of aerial logistics but also offers adaptable tools for tackling diverse optimization problems across military and civilian contexts, promising significant improvements in efficiency and sustainability. Future work may explore the integration of real-time data analytics to further refine loading and routing efficiency.

List of Acronyms and Special Terms

ACLP+RPDP	Air Cargo Load Planning with Routing, Pickup, and Delivery Problem
ACO	Ant Colony Optimization
ACP	Aircraft Configuration Problem
APP	Air Cargo Palletization Problem
BF	Best-Fit, a technique for bin-packing solving
BSP	Build-up Scheduling Problem
CG	Center of Gravity
DEAP	Distributed Evolutionary Algorithms
FFD	First-Fit Decreasing, a technique for bin-packing solving
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
IP	Integer Programming
LP	Linear Programming
MIP	Mixed-Integer Programming
MMKP	Multidimensional Multiple Knapsack Problem
MLU	minimization of loading and unloading costs
Multiprocessing	a Python package that implements process-based parallelism
mpShims	a special parallel heuristic created in this work
NM	Noising Methods
Packed	a set of items packed on a flat pallet
PDP	Pickup and Delivery Problem
PH	Parallel Heuristics
Shims	a special heuristic created by (MESQUITA; SANCHES, 2024)
TS	Tabu Search
TSP	Traveling Salesman Problem
ULD	Unit Load Devices
WBP	Weight and Balance Problem

Contents

1 Introduction

We explore a common problem in transport aviation, as there are risks of cargo unbalancing, cost-inefficient tours, and incorrect deliveries due to the urgency required on loading for rapid take-off and mission accomplishment. This work addresses the problem of planning the loading and routing of an aircraft according to weight and balance principles, fuel economy, agility in assembling pallets under safety requirements, and also serving multiple nodes in an efficient tour, attending simultaneous pickup and delivery demands at intermediate airfields, until the return to the base. This problem requires a comprehensive solution that takes into account various factors such as minimal possible fuel consumption, efficient use of resources, and adherence to safety protocols. Additionally, the solution should aim to minimize turnaround time at intermediate airfields to ensure timely delivery and avoid any disruptions in the mission's accomplishment.

We identified the limitations of existing approaches and developed a novel heuristic, Shims, with low search times, ideal for multi-leg airlift planning. Shims considers factors like weight and balance, fuel efficiency, and efficient pallet assembly. Shims is particularly well-suited for real-world scenarios with time constraints due to its exceptionally low search times.

The *Air Cargo Load Planning Problem* (ACLPP) was recently defined by (BRANDT; NICKEL, 2019, p. 71) as the composition of four subproblems: *Aircraft Configuration Problem* (ACP), *Build-up Scheduling Problem* (BSP), *Air Cargo Palletization Problem* (APP), and *Weight and Balance Problem* (WBP). The authors meticulously analyzed the air freight load planning problem, identifying its subproblems and areas for further research. They also discussed the implications of their findings for air cargo load planning practices, suggesting ways to enhance operations and outlining the potential benefits of implementing their model and algorithm in real-world scenarios.

(MESQUITA; SANCHES, 2024) proposed a rapid solution method for this problem and introduced the first mathematical model of ACLP+RPDP, a complex system comprising four well-known NP-hard problems. Additionally, they developed a comprehensive process for solving these connected problems. However, their work lacked parallel algorithms for enhanced computational efficiency and did not model 3-D packing solutions. In this thesis, we delve deeper into their approaches, introducing advancements aimed at improving performance and incorporating new functionalities by considering more detailed real-world intricacies.

In this work, we define *Palletization* as a logistical procedure that entails arranging goods on a pallet to secure and consolidate the load, facilitating its transit, storage, handling, and distribution. We refer to palletized items as *Packed* items or *carga consolidada* in Brazilian Portuguese.

As the problem of this research extends the complexities of ACLPP, (MESQUITA; SANCHES, 2024) named it *Air Cargo Load Planning with Routing, Pickup and Delivery Problem* (ACLP+RPDP).

In this chapter, we emphasize the importance of acquiring efficient solutions. We present an overview of the problem structure in both civil and military contexts and provide a glimpse into real operational scenarios. These include the network cost structure, operational premises, and the two aircraft sizes under study. Additionally, we discuss considerations and calculations related to cost increases from Center of Gravity (CG) displacement.

In the next chapter, we integrate these elements with the mathematical model of the problem. The mathematical model allows us to analyze and optimize various aspects of the problem, such as the allocation of resources, scheduling, and decision-making processes. We can generate insights and recommendations by incorporating real-world data and constraints into the model, which can lead to more efficient and cost-effective solutions. Additionally, we discuss the potential implications and benefits of implementing these solutions.

1.1 Problem relevance

Solving ACLP+RPDP is critical because it enhances the loaded items' strategic score (profit or other client-specific parameters). This optimization not only saves time and effort by eliminating trial-and-error in the loading process, but also ensures safety through overall load balance. Additionally, it ensures the correct pickup and delivery of items to their intended destinations, while identifying the most fuel-efficient routes, taking into account potential imbalances in cargo distribution.

The current APP process often relies on outdated methods, such as pen-and-paper or spreadsheet entries, combined with trial-and-error during pallet construction. This approach is labor-intensive, costly, and typically fails to yield optimal results. It may also pose safety risks due to the ongoing pressure to quickly resolve complex issues under time constraints.

Safety and Efficiency in Air Cargo Transport

Regarding WBP, we can draw attention to a significant study by G.W.H. van Es on safety incidents between 1970 and 2004:

> *The risk of having a weight and balance-related accident with cargo flights is 8.5 times higher than with passenger flights; and that there were various factors involved in weight and balance accidents/incidents such as errors in the load sheet, shifting of cargo, incorrect loading, etc. (ES, 2007, p. 22).*

Therefore, it becomes clear the importance of a decision support system, that is, an embedded algorithm to quickly and efficiently generate safe loading plans for immediate take-off and mission accomplishment.

Another important issue is managing fuel consumption. To get an idea of the influence of CG displacement on fuel consumption, (MONGEAU; BES, 2003, p. 140) report that:

> *A displacement of the CG of less than 75 cm in a long-range aircraft yields, over a 10,000 km flight, a saving of 4,000 kg of fuel.*

This reinforces the importance of balancing cargo.

Challenges and Technological Solutions in Logistics Management

The purchasing department of an organization often acquires supplies from major capitals domestically and internationally to meet the company's needs. The requesting organizations must receive these materials within the stipulated deadlines. Meanwhile, another department responsible for component maintenance may need to expedite items to supporting contractors, necessitating efficient transportation solutions.

The organization's logistic management system typically documents transportation demands. This system provides excellent visibility into transport needs, aiding in resource planning and allocation. It enables real-time tracking of shipments to ensure that materials reach their destinations on time and in optimal condition. The system also facilitates access to transportation demand data, allowing the company to identify and address any bottlenecks or inefficiencies in the logistics process.

However, the sheer volume of information that cargo terminal managers must process makes it challenging for humans to develop effective and secure transportation plans without technological assistance. As a result, technology plays a critical role by enabling operators at each location to input existing loads with their destinations and other parameters, ensuring that logistical operations are both efficient and secure.

Economic and Safety Implications of Current Practices

Inefficient transport plans significantly drive up costs by necessitating extra routes, increasing travel distances, and sometimes resulting in deliveries to incorrect destinations. Furthermore, unbalanced cargo can lead to increased fuel consumption because higher pitch angles cause increased drag.

Despite available technology, many airlines still rely on manual aircraft loading and balancing procedures. For CG computation and constraint testing, loadmasters primarily use simple computer-aided tools, but often resort to time-consuming, experience-based planning. Since load plans are typically finalized about an hour before takeoff, this manual approach can lead to delays.

Such practices pose significant safety risks. Improperly balanced cargo can severely impact flight safety. A forward-shifted CG can result in a nose-down attitude, compromising safe takeoff, whereas a rearward CG can cause a nose-high attitude, making it difficult to recover from a stall and land safely. Maintaining the CG within safe operational limits is therefore crucial.

To mitigate these economic and safety risks, it is essential to adopt a more rational decision-making process that leverages advanced technology for planning and execution. This is particularly important given the high costs of fuel, lubricants, and fleet maintenance, as well as the potential costs associated with outsourced transport and the risks to operational integrity from failing to deliver critical materials or accidents caused by unbalanced loads.

1.2 Thesis objective

The primary goal of this thesis is to develop a smart decision-making tool that enables safe and efficient transportation planning. This tool will feature a solution process with minimal search time, be suitable for planning multi-leg airlift operations, and be adaptable for integration into a cloud-based or desktop-hosted decision support system.

Additionally, the project aims to optimize the distribution of loads on pallets within the cargo bay. By doing so, it seeks to maintain aircraft balance, maximize total score, and minimize fuel consumption during airlifts. We design the resulting tour, pallet building, and arrangement plans to ensure flight safety, enhance ground operations efficiency, and guarantee the correct delivery of each item to its intended destination.

This objective builds on the foundations laid in (MESQUITA; SANCHES, 2024), where initial problems were addressed by this author and his advisor. The current work expands on those findings by incorporating 3-D packing solutions and parallel processing techniques into the developed algorithms, addressing gaps identified in the previous research.

1.3 Some premises on the problem

(ROESENER; BARNES, 2016) argue that commercial airlift operations primarily aim to balance efficiency (generating revenue) and effectiveness (maintaining customer satisfaction), with a clear focus on profit. Conversely, military airlift operations prioritize effectiveness (ensuring timely delivery of necessary goods and equipment) while also striving to spend government funds wisely.

Distinct operational goals necessitate differing approaches to problem formulation and resolution in commercial versus military airlift scenarios. This distinction, emphasized in their work, highlights the unique objectives of profit in commercial operations versus effectiveness in military logistics.

In military operations, *Unit Load Devices* (ULDs) are often standardized, simplifying the *Aircraft Configuration Problem* (ACP) as pallet positions can be predetermined before loading. The *Build-up Scheduling Problem (BSP)* in military settings typically features pre-established schedules, with items prepared for palletization in advance.

In commercial settings, ULDs may vary significantly in shape, complicating the task of minimizing CG displacement. Additionally, aircraft with multiple cargo doors can facilitate more efficient cargo sequencing.

Economically, the optimal choice often involves selecting the most efficient route-not necessarily the shortest-that maximizes the benefit-cost ratio. The core challenge addressed in this study is the integration of three classic problems: the *Air Cargo Palletization Problem* (allocation and packing of items to maximize scoring), the *Weight and Balance Problem* (ensuring cargo balance constraints are met), and the *Simultaneous Delivery and Pickup Problem* (managing cargo at each node for both delivery and pickup). This integrative approach aims to create an efficient transportation plan that addresses these varied challenges.

As it is an extremely complex problem, we decided to simplify a few aspects as approached by (LURKIN; SCHYNS, 2015):

a. We considered a unique ULD type with an amount and predefined positions in the aircraft, although this model may be generalized for other scenarios. The items to be allocated to these ULDs in each leg of the flight plan may be characterized by weight, dimensions, scores, and previously known destinations.

b. We considered that a pallet which has not yet gotten to its destination may receive more items, although it is known that these operations of removing restraining nets may increase handling time.

c. Finally, we also disregarded *hazardous products*, which eventually could be treated as high-scoring items in a future work.

Pallet builder technicians are highly skilled at assembling flat pallets, maintaining a centered and low center of gravity, and securing loads with specialized tight nets. Despite their invaluable expertise and strict adherence to safety regulations, we must explore technological advancements to improve these processes. Therefore, we aim to develop a 3-dimensional packing algorithm that meets the operational demands for runtime performance.

By ensuring more efficient pallet packing, maximizing space utilization, and enhancing stability, this algorithm will complement the technicians' skills. It will also standardize the pallet assembly process, minimizing the potential for human error and ensuring consistency across all operations.

Throughout this text, we refer to a *Packed* as a set of items destined for the same location, stacked on a pallet, and secured with a restraining net. Each *Packed* is treated as a unique entity, possessing the combined attributes of its components, such as the sum of individual scores, weights, and volumes. See Figure 1.1.

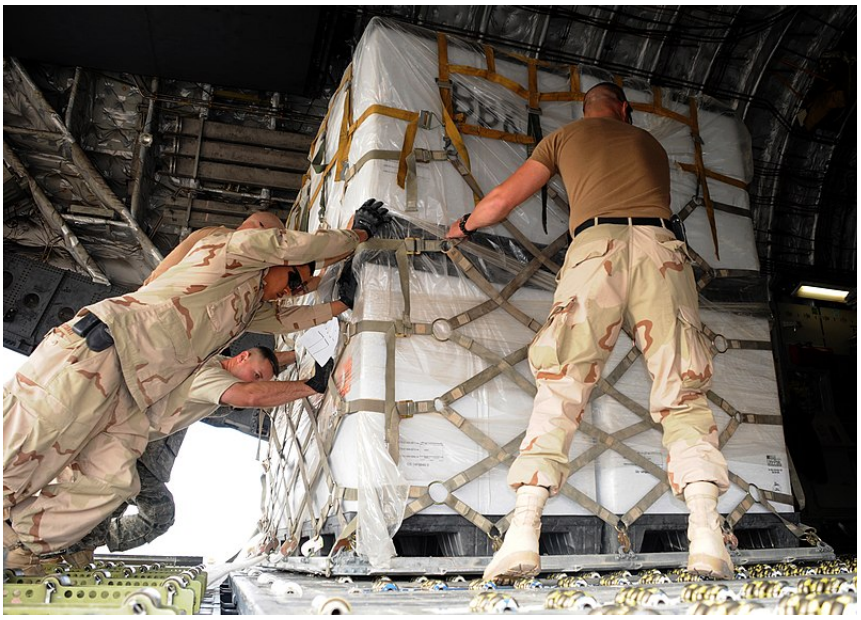


Figure 1.1 – *Packed* on a pallet inside a *Boeing C-17*

Source: From Wikimedia Commons, the free media repository

It is important to highlight that the *Packed* must stay on board until they reach their destination to maintain accuracy in pickup and delivery operations.

Informally, ACLP+RPDP can be summarized as follows:

max	(items' score sum) / (tour cost) of picked up and delivered items at each node on a tour.
s.t.	$\rightarrow$ For each leg along a route, the set of unvisited nodes is updated.
	$\rightarrow$ *Packed* are composed at each node and remain on board until their destinations.
	$\rightarrow$ Only items destined for the remaining flight legs will be loaded.
	$\rightarrow$ Weights and volumes correspond to all the packed items on a pallet with the same destination.
	$\rightarrow$ Double-lane-decked larger aircraft have their lateral torque limited to the operational range.
	$\rightarrow$ The longitudinal torque operational range is applied to any aircraft size.
	$\rightarrow$ Larger aircraft have payloads typically larger than the overall pallet weight capacities.
	$\rightarrow$ Weight and volume limitations (pallets or aircraft) must be respected.
	$\rightarrow$ In a node, an item may be included in at most one pallet.
	$\rightarrow$ In a node, *Packed* must be included in one pallet if the destinations are the same.
	$\rightarrow$ Pallets destined for the next node should be put as near as possible to the ramp door.
	$\rightarrow$ Items allocated to the pallet must fit on the pallet before the packing operation.

There is not, to our knowledge, any work in the literature that solves this complex problem.

1.4 Scenarios envisaged for the problem

The ULD (or pallet) on which this work is based is the *463L Master Pallet*, a common size platform for bundling and moving air cargo, and serves as the primary air cargo pallet for more than 70 air forces and transportation companies flying mainly on the *Lockheed C-130 Hercules* and other types of cargo lifters (*Boeing CH-47 Chinook*, *Casa CN-235*, *Embraer KC-390*, *Lockheed C-17 Globemaster*, *Boeing C-767*, *Douglas DC-10*, *Boeing 747*, *Leonardo C-27 Spartan*, *Airbus A330 MRTT*, and *A400M*). This ULD has a capacity of $4500kg$, is equipped for locking pallets into cargo aircraft rail systems, and includes tie-down rings to secure nets and cargo loads, which in total weighs $140kg$. For more information, see `www.463lpallet.com`.

Another important issue is that the airplane has some pallets that will remain on board for the next leg in a sequence of delivery and pickup points in a tour. This means that pallets eventually have to be reallocated, guaranteeing a feasible solution regarding cargo balancing.

(LURKIN; SCHYNS, 2015) produced an innovative work: they were the first to model the ACLPP, which is simultaneously a *Weight and Balance Problem* (WBP) and a *Sequencing Problem* (SP), as well as two flight legs, including pickups and deliveries at the intermediate airport. Some of their suppositions are:

(i) Only items destined for the same node are loaded in each *Unit Load Device* (ULD); and

(ii) ULDs that have not yet reached their destinations are not altered or moved in the intermediate legs of the flight plan.

However, their approach has a limitation. Since the flight plan is predetermined, it might not consider the most efficient routes. Instead, their method focuses solely on positioning Unit Load Devices (ULDs) to minimize fuel consumption.

The model of the problem in this work could be viewed as a generalization of their work, as it deals with more real-world constraints.

Nodes network

Brazil is Latin America's largest economy, and the country's vast size necessitates the use of aviation to connect its many regions, from rural towns to cosmopolitan state capitals. This is why the country has the continent's largest air transportation market by far. As a country with a growing economy, it knows how important airport development is to reaching its goals of more tourism and international trade.

According to the recent report of (IATA..., 2017), the aviation sector contributes with US$ 18.8 billion to Brazil's GDP and generates more than 800,000 jobs. With demand set to double over the next 20 years, the economic contribution of aviation to the Brazilian economy could increase to more than US$ 8.8 billion per year with more than 1.4 million jobs.

Brazil has 2,499 airports registered by ANAC (National Civil Aviation Agency - *Agência Nacional de Aviação Civil*) of which 1,911 are private and 588 are public. Although it is an immense distribution network, the Brazilian Air Force missions have always considered 3-5 nodes per planned flight. It is important to emphasize that this data is not an imposed limitation but a historical fact that we will explore in our method.

Based on the most recent data available from ANAC, there are 73 airports in Brazil with dedicated cargo terminals, of which 30 are international airports and 43 are domestic airports. These airports are located in major cities throughout Brazil, including São Paulo, Rio de Janeiro, Campinas, Belo Horizonte, Manaus, and Curitiba (see Table 1.1).

Table 1.1 – Top 10 airports with the most cargo handled in Brazil in 2022

Airport	**IATA code**	**Cargo handled (tons)**
Guarulhos	GRU	632,000
Viracopos	VCP	415,000
Confins	CNF	190,000
Galeão	GIG	167,000
Recife	REC	82,000
Fortaleza	FOR	74,000
Brasilia	BSB	69,000
Curitiba	CWB	68,000
Porto Alegre	POA	65,000
Salvador	SSA	57,000

The air cargo sector in Brazil is expected to continue to grow in the coming years, driven by the country's growing economy and increasing demand for e-commerce. As a result, there is likely to be a need for more air cargo terminals in Brazil in the future.

Throughout this work, we have focused on routes involving up to 7 nodes but have developed a solution capable of efficiently handling up to 15 nodes in less than 20 minutes, as demonstrated in Table 1.2, Figure 1.2, and Figure 7.1. This capability not only highlights the solution's scalability and performance but also ensures its applicability to larger network scenarios if needed.

The Brazilian Air Force selected these seven nodes due to their high demand and the urgency of transport deadlines. Private companies recognize these nodes as the most demanding air cargo hubs. We prioritized these nodes to optimize resources and response times in high-stakes situations. Alternative transportation methods such as cabotage, rail, or road more economically serve airports with lower demand. This strategic approach aligns with the broader goal of utilizing resources where they are most impactful.

Table 1.2 presents a general example of nodes run by the Brazilian Air Force in its logistics missions, which are part of the testing scenarios of this work.

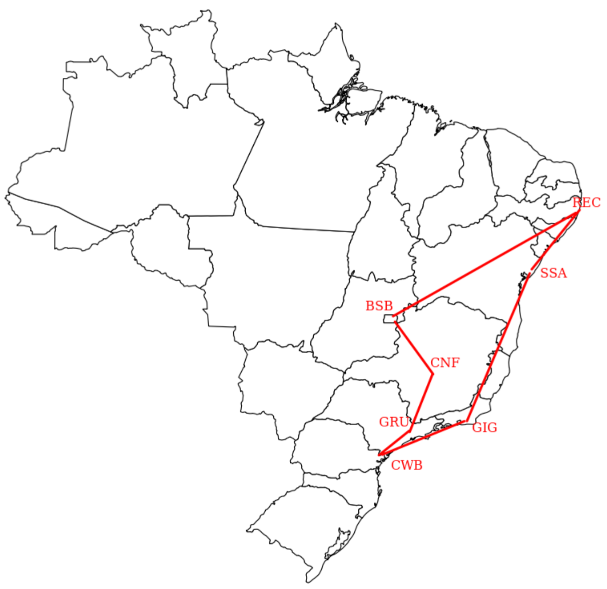


Figure 1.2 – Nodes network for testing

Source: the author

Table 1.2 – Distances between some airports (km)

Node ID Airport IATA*	0 (base) Guarulhos GRU	1 Galeão GIG	2 Salvador SSA	3 Confins CNF	4 Curitiba CWB	5 Brasilia BSB	6 Recife REC
GRU	0	343	1439	504	358	866	2114
GIG	343	0	1218	371	677	935	1876
SSA	1439	1218	0	938	1788	1062	676
CNF	504	371	938	0	851	606	1613
CWB	358	677	1788	851	0	1084	2462
BSB	866	935	1062	606	1084	0	1658
REC	2114	1876	676	1613	2462	1658	0

*International Air Transport Association (IATA) airport code

Source: the Author, calculated with a tool on `www.airportdistancecalculator.com`

Operational premises

The word *Packed*, cited early, is a set of items stacked on a pallet and covered with a restraining net. It is considered unique for having the same attributes as its components, whose values are the sum of individual scores, weights, and volumes, and having the same destination. *Packed* are commonly used in the logistics and transportation industries to optimize storage space and ensure secure transportation. This method allows for efficient handling and tracking of multiple items as a single unit, reducing the risk of damage or loss during transit.

To keep the cargo safely balanced, all pallet assembly and cargo operations must deal with the loaded aircraft torque limitations by keeping the CG within its safe range. Failure to maintain the center of gravity within a safe range can result in unstable flight conditions and a potential loss of control. This requires careful planning and the precise distribution of weight throughout the aircraft to ensure safe balance and stability during flight.

To guarantee precision in the pickup and delivery operations, the en route *Packed* must be guaranteed to remain on board before its delivery point, and all empty pallets will be assembled with items having the same destinations.

Another challenge is the single cargo door (ramp door). Ideally, pallets closest to their final destination would be loaded near the door to minimize handling time during unloading. However, with only one door, all cargo is loaded and unloaded in a stack, pushing or pulling everything through the same opening. This makes it difficult to minimize handling time because pallets further from the door may need to be moved multiple times to access others.

To address this limitation, this work focuses on positioning pallets for the next destination as close to the ramp door as possible. This helps to minimize re-handling during the unloading process.

In aircraft with two docking lanes, it may be possible, in some situations, to exchange pallet positions inside the aircraft in an attempt to minimize handling times. But not in aircraft with a single cargo lane. In such a case, the exchange of pallet positions may require the use of external loaders and may require more handling time.

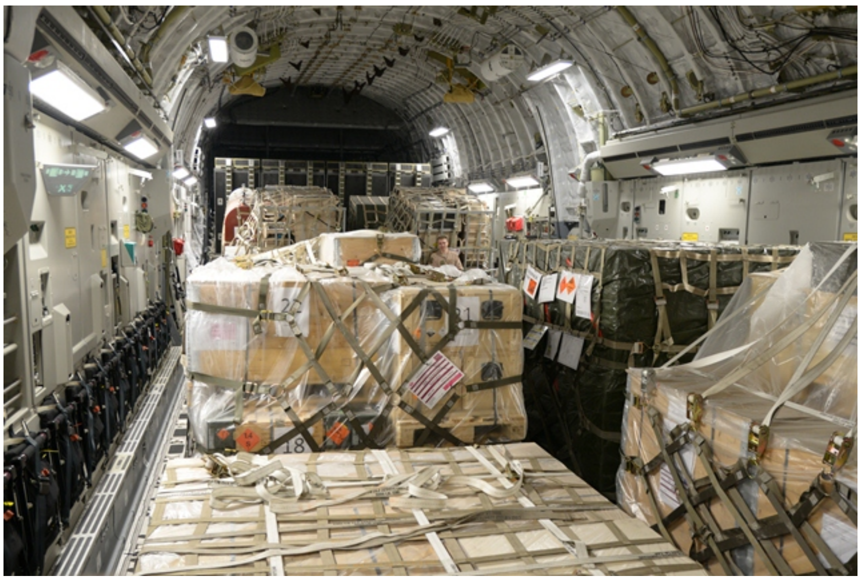


Figure 1.3 – *Packed* in a two-docking-lanes aircraft

Source: https://www.sacprogram.org/en/Pages/Boeing-C-17-Globemaster-III.aspx

We do not consider oversized cargo in this work. Only cargo that fits on 463L pallets.

This project focuses on finding the best route for a single aircraft, considering both benefits and costs. In the future, we might explore challenges of planning routes for multiple aircraft, either with similar capacities or a mix of different sizes.

Aircraft of this work

We consider many real scenarios with a smaller or larger aircraft with capacities (or payloads) of $26000kg$ or $75000kg$ respectively. In both cases, the torque applied to the aircraft must keep its CG in the operational range, which corresponds to a percentage of the *Mean Aerodynamic Chord*.

Chord is the distance between the leading and trailing edges of the wing, measured parallel to the normal airflow over the wing (HOUGHTON; CARPENTER, 2003, p.18).

The CG operational range around the CG point is: $\pm 0.556m$ in the smaller aircraft and $\pm 1.17m$ in the larger one (see Figure 1.5).

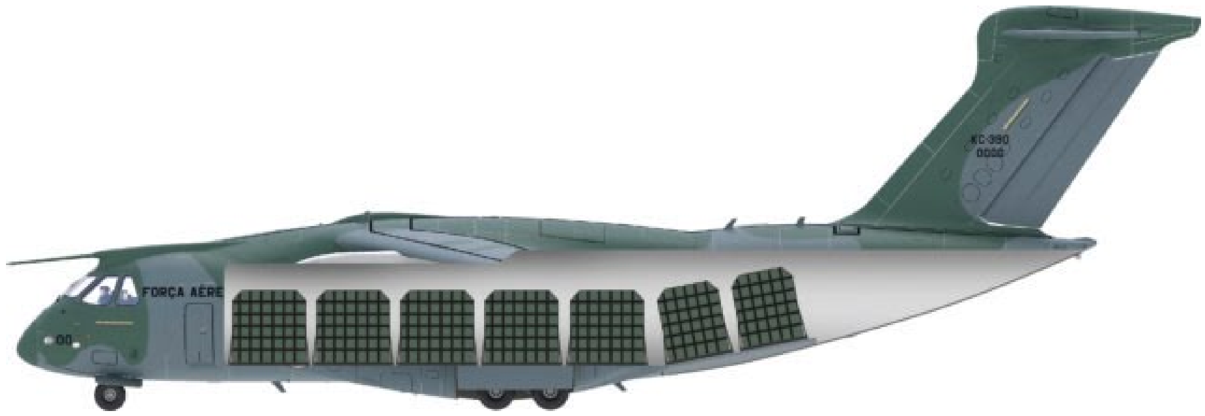


Figure 1.4 – An aircraft longitudinal cut illustrating the positions of the pallets.

Source: "The KC 390". Embraer Defense & Security. Retrieved 13 July 2016.

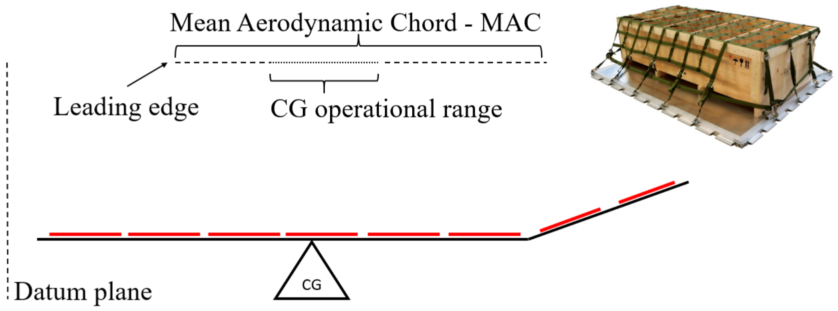


Figure 1.5 – A longitudinal cut of the aircraft in Figure 1.4.

Source: the author

Both pallets layouts are represented in Figures 1.6 (smaller aircraft) and 1.7 (larger aircraft), where pallets are identified by p_i. Tables 1.3 and 1.4 show, in both cases, the payloads, the CG-limits, and the load limit allowed on each pallet. In these tables, *distances to CG* refer to the distances of pallets centroids (in meters) in relation to the CG of aircraft along the longitudinal axis, which may be negative (fwd) or positive (aft).

In both aircraft, as the ramps are considered to have an inclination of 25 degrees, we made the necessary correction in the *distances to CG* of the corresponding pallets.

Another important parameter for travel cost calculations is the aircraft fuel consumption rate. As it is out of the scope of this work to establish adequate ranges per aircraft type, we estimated 6 kg/km for the larger and 1.2 kg/km for the smaller aircraft. We also considered the fuel price as 0.82 US$/kg. These values are used to calculate the cost matrix based on Table 1.2, the distances to be traveled by both aircraft.

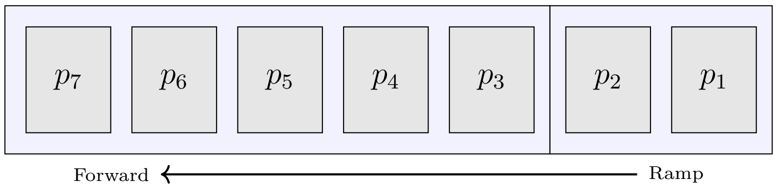


Figure 1.6 – Smaller aircraft layout

Table 1.3 – Smaller aircraft parameters

Limits	*Payload*: 26,000kg			$limit^{CG}_{long}$: 0.556m			
$\boldsymbol{p_i}$	p_7	p_6	p_5	p_4	p_3	p_2	p_1
$\boldsymbol{D^{long}_i}$ (m)	-5.10	-2.70	-0.30	2.10	4.50	6.25	8.39
$\boldsymbol{W_i}$ (kg)	4,500	4,500	4,500	4,500	4,500	4,000	3,500
$\boldsymbol{V_i}$ (m^3)	13.7	13.7	13.7	13.7	13.7	8.9	6.9
Fuel cost per kilometer	c_{km} = US$ 1.10/km						
Maximum weight	W_{max} = 26,000kg						

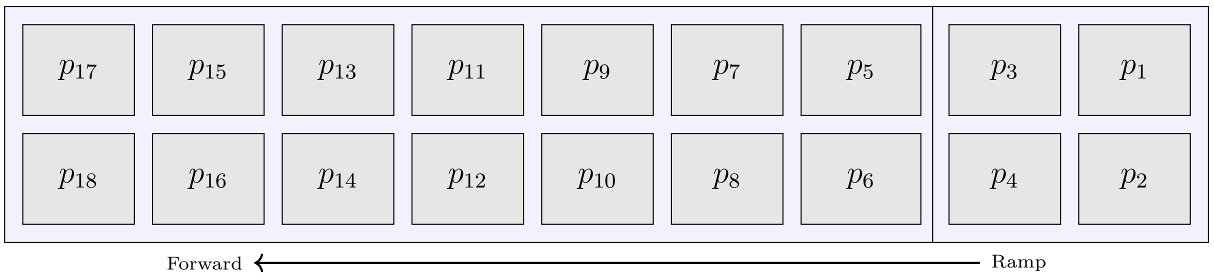


Figure 1.7 – Larger aircraft layout

Although pallets weight and volume capacities are identical, in the aircraft, these capacities are limited in weight according to their position along the cargo bay. Pallets on ramp door are more limited in volume due to their proximity to the aircraft ceiling.

Table 1.4 – Larger aircraft parameters

Limits	*Payload*: 75,000kg			$limit^{CG}_{long}$: 1.170m			$limit^{CG}_{lat}$: 0.19m		
$\boldsymbol{p_i}$	p_{17} p_{18}	p_{15} p_{16}	p_{13} p_{14}	p_{11} p_{12}	p_9 p_{10}	p_7 p_8	p_5 p_6	p_3 p_4	p_1 p_2
$\boldsymbol{D_i^{long}}$ (m)	-17.57 -17.57	-13.17 -13.17	-8.77 -8.77	-4.40 -4.40	0 0	4.40 4.40	8.77 8.77	11.47 11.47	14.89 14.89
$\boldsymbol{D_i^{lat}}$ (m)	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32
$\boldsymbol{W_i}$ (kg) $\boldsymbol{V_i}$ (m^3)	4,500 14.8	4,500 14.8	4,500 14.8	4,500 14.8	4,500 14.8	4,500 14.8	4,500 14.8	3,000 10.0	3,000 7.0
Fuel cost per kilometer	c_{km} = US\$ 4.90$/km$								
Maximum weight	W_{max} = 75,000kg								

CG displacement costs

In `www.flightdeckfriend.com` (accessed in: January 5, 2017), we found some relevant information regarding CG displacement costs:

The four engines of the Boeing 747 Jumbo Jet burn approximately 10 to 11 tonnes of fuel an hour when on cruise.

A Jumbo Jet (Boeing 747-400) flying from London to New York burns approximately 70,000 kilograms of fuel. Jet fuel has a specific gravity of about 0.85, which means it takes up about 82,353 liters.

This would roughly align with a 6-7 hour flight at the mentioned cruise burn rates. Of course, this rate can vary depending on a variety of factors, including altitude, speed, and load.

As the straight distance from London to New York is 5,571km and the fuel consumption is about 70,000kg, 10,000 km would consume 126,000kg of fuel. As 10,000km with an unbalanced aircraft would consume 4,000kg of fuel (MONGEAU; BES, 2003, p. 140), this represents an increase of 3.17% fuel burn.

Let's consider that the larger aircraft of this research have the same fuel consumption rate as a Boeing 747-400, and fuel burn is linear with CG displacement, so these percentages would be as stated in Table 1.5.

Table 1.5 – Increased fuel consumption due to CG displacement.

CG displacement	c_g (%)	**Aircraft**
75cm	3.17	Boeing 747-400
55.6cm	2.37	smaller aircraft, Table 1.3, $limit^{CG}_{long}$
117cm	4.95	larger aircraft, Table 1.4, $limit^{CG}_{long}$

c_g is the cost increase considered in the mathematical modeling.

The percentages in Table 1.5 are approximate, requiring a specific study for the aircraft chosen, which is out of the scope of this work.

1.5 Thesis organization

Overall, our goal was to provide a comprehensive understanding of the Air Cargo Load Planning with Routing, Pickup, and Delivery Problem (ACLP+RPDP) and its practical implications. By showcasing real operational scenarios and network dimensions, we aimed to highlight the relevance of efficient solutions in optimizing cargo aircraft operations. Our analysis also considered the impact of CG displacement on cost increases and introduced ground handling burdens to further emphasize the complexity of the problem.

This thesis is organized into seven additional chapters: In Chapter 2, we present the mathematical model, laying the foundational framework for the ACLP+RPDP. Chapter 3 reviews the state of the art in solving problems similar to or related to the ACLP+RPDP, providing a context for the research. In Chapter 4, we introduce the solution approaches and a new heuristic developed specifically for the ACLP+RPDP. Chapter 5 details the integration of computer parallelism and 3-D packing techniques, enhancing the efficiency and scope of our solutions. In Chapter 6, we describe testing scenarios and item instances generation, including preparation procedures before applying any solution method, and search for optimal results using integer programming. Chapter 7 compares and discusses the results, addressing the issues encountered with each method. The conclusions and future research directions are presented in Chapter 8.

2 Mathematical model

Given the assumptions and parameters described in the previous chapter, we are ready to present the integer programming model. ACLP+RPDP has the objective function (2.1), and the calculus equations (2.2) to (2.9) subject to constraints (2.12) to (2.21), which will be described below.

2.1 Objective function

Let $\Pi_K = \{\pi : \{1, \ldots, K\} \to \{1, \ldots, K\}\}$ be the set of $K!$ permutations, which correspond to all possible tours that have node 0 as origin and end, passing through the other K nodes.

Let π_k be k^{th} node of the tour π, $1 \leq k \leq K$. In this way, the tour π is described as $\{0, \pi_1, \ldots, \pi_K, 0\}$. For ease of notation, we can define $\pi_0 = \pi_{K+1} = 0$.

Prioritizing the most relevant items is crucial for any company. Whether the goal is to maximize profit or any other parameter, it is equally critical to minimize the overall cost. So, we chose a simple objective function that can simultaneously attend to both objectives.

The objective of ACLP+RPDP is to find the permutation $\pi \in \Pi_K$, with the corresponding allocation of items on the pallets at each node, that maximizes the function in equation 2.1, where $\tilde{s}_\pi$ is the total score of transported items, and $\tilde{c}_\pi$ is the total cost of fuel consumed.

$$\max_{\pi \in \Pi_K} f(\pi) = \tilde{s}_\pi / \tilde{c}_\pi \tag{2.1}$$

2.2 Problem structure

In these descriptions, we use the values assigned in tables 1.2, 1.3, and 1.4.

Problem structure description

Notation	**Description**
$L = \{0, \pi_1, \pi_2, \ldots, \pi_K\}$	a set with the $K+1$ nodes
$L_k = \{\pi_{k+1}, \ldots, 0\}$	the set of remaining nodes when the aircraft is in the position k.
$d(a,b)$	the distance from node a to node b, where $0 \leq a, b \leq K$. By definition, $d(a,a) = 0, \forall a$.
$C = [c_{a,b}]$	the cost matrix of flights, where $c_{a,b} = c_{km} * d(a,b)$, being c_{km} the cost per kilometer for the considered aircraft.
$M = \{1, 2, \ldots, m\}$	set of m empty pallets assigned to specific positions within the aircraft. m may be equal to 7 or 18, depending on the size of the aircraft. Each pallet i, $1 \leq i \leq m$, has weight capacity W_i, volume capacity V_i, longitudinal distance to the CG of aircraft D_i^{long}, lateral distance to the centerline of aircraft D_i^{lat}, and a destination $T_i^{\pi_k} \in L_k$.
$N^{\pi_k} = \{1, \ldots, n^{\pi_k}\}$	set of n^{π_k} items available for loading at node π_k, $1 \leq j \leq n^{\pi_k}$, $0 \leq k \leq K$. Each item has the following attributes: score s_j, weight w_j, volume v_j, and destination $to_j \in L_k$.
$N = \bigcup_{0 \leq k \leq K} N^{\pi_k}$	set of items in all nodes along a tour.
$Q^{\pi_k} = \{1, \ldots, m^{\pi_k}\}$	set of $m^{\pi_k} \leq m$ *Packed* $a_q^{\pi_k}$ that remain on board at node π_k, $1 \leq q \leq m^{\pi_k}$, $0 \leq k \leq K$. $a_q^{\pi_k}$ has the same attributes of the items. By definition, $m_0 = 0$, and therefore $Q_0 = \varnothing$. *Packed* that were destined to node π_k are unloaded when the aircraft arrives at this node, that is, they are not considered in Q^{π_k}.

2.3 Decision variables

Let $X_{ij}^{\pi_k}$ and $Y_{iq}^{\pi_k}$ be binary variables, where $1 \leq i \leq m$, $1 \leq j \leq n^{\pi_k}$, $1 \leq q \leq m^{\pi_k}$ and $0 \leq k \leq K$.

$\rightarrow$ $X_{ij}^{\pi_k} = 1$ if the item j at node π_k is assigned to the pallet i, and 0 otherwise. Other pallets may receive this item, or it may not be assigned at all, reserving it for other transport missions.

$\rightarrow$ $Y_{iq}^{\pi_k} = 1$ if the *Packed* q at node π_k is assigned to pallet i, and 0 otherwise. The eventual relocation of the *Packed* that remain on board for cargo balancing purposes makes this variable crucial.

2.4 Allocation graph

Allocations of items or *Packed* to the pallets in node π_k can be seen as a bipartite graph $G^{\pi_k}(V^{\pi_k}, E^{\pi_k})$, where:

$\rightarrow V^{\pi_k} = M \cup N^{\pi_k} \cup Q^{\pi_k}$

$\rightarrow E^{\pi_k} = E_{N^{\pi_k}} \cup E_{Q^{\pi_k}}$

$\rightarrow (i, j) \in E_{N^{\pi_k}}$ if $X_{ij}^{\pi_k} = 1$, where i is a pallet and j is a item at node π_k

$\rightarrow (i, q) \in E_{Q^{\pi_k}}$ if $Y_{iq}^{\pi_k} = 1$, where i is a pallet and q is a *Packed* at node π_k

2.5 Calculus equations

Calculus equations descriptions

Symbol	**Equation**	**Description**
$\tilde{s}_\pi$	2.2	the sum of the scores of the items loaded on the aircraft throughout the tour π.
τ^{π_k}	2.3	the longitudinal torque applied by the loaded pallets at the node π_k, in proportion relative to the highest torque supported by the aircraft.
$\tilde{c}_\pi$	2.4	the cost of the total fuel consumed in the tour π due to the distances travelled and the CG longitudinal deviations. c_g is the percentage cost increase due to the CG deviation. In our experiments, we found that the magnitude of the lateral torque was always very small, so we decided to ignore it in the fuel consumption calculation in each pallet.
L_0	2.5	all nodes to be visited.
L^{π_k}	2.6	the set of not visited nodes when staying in π_k.
m_0	2.7	At the beginning of tour π, there are no packed contents.
$\epsilon_t^{\pi_k}$	2.8	the lateral torque applied by the loaded pallets at the node π_k, in proportion relative to the highest torque supported by the aircraft.
$\epsilon_a^{\pi_k}$	2.9	the *Packed* lateral torque.
W_{max}	2.10	The maximum weight is the minimum between the sum of pallets weights capacities and the aircraft payload.

We have the following calculation equations:

$$\tilde{s}_\pi = \sum_{k=0}^{K} \sum_{i=1}^{m} \sum_{j=1}^{n^{\pi_k}} X_{ij}^{\pi_k} \times s_j \tag{2.2}$$

$$\tau^{\pi_k} = \sum_{i=1}^{m} \left[D_i^{long} \times \Big(\sum_{j=1}^{n^{\pi_k}} X_{ij}^{\pi_k} \times w_j + \sum_{q=1}^{m^{\pi_k}} Y_{iq}^{\pi_k} \times w_q \Big) \right] \Big/ W_{max} \times limit_{long}^{CG};\ k \in \{0, \dots, K\} \tag{2.3}$$

$$\tilde{c}_\pi = \sum_{k=0}^{K} \left[c_{\pi_k, \pi_{k+1}} \times (1 + c_g \times |\tau^{\pi_k}|) \right] \tag{2.4}$$

$$L_0 = L \tag{2.5}$$

$$L^{\pi_k} = L^{\pi(k-1)} - \{\pi_k\};\ k \in \{1, \dots, K\} \tag{2.6}$$

$$m_0 = 0 \tag{2.7}$$

$$\epsilon_t^{\pi_k} = \sum_{i=1}^{m} \left[D_i^{lat} \times \sum_{j=1}^{n^{\pi_k}} \Big(X_{ij}^{\pi_k} \times w_j \times (i\%2) - X_{ij}^{\pi_k} \times w_j \times (i+1)\%2 \Big) \right] \Big/ W_{max} \times limit_{lat}^{CG} \tag{2.8}$$

$$\epsilon_a^{\pi_k} = \sum_{i=1}^{m} \left[D_i^{lat} \times \sum_{q=1}^{m^{\pi_k}} \Big(Y_{iq}^{\pi_k} \times w_q \times (i\%2) - Y_{iq}^{\pi_k} \times w_q \times (i+1)\%2 \Big) \right] \Big/ W_{max} \times limit_{lat}^{CG} \tag{2.9}$$

$$W_{max} = \min(Payload, \sum_{i=1}^{m} W_i) \tag{2.10}$$

2.6 Constraints

Finally, we can consider the constraints at each node π_k:

s.t.:

$\rightarrow$ The longitudinal (2.11) and the lateral (2.12) torques must be within the limits of the aircraft.

$$|\tau^{\pi_k}| \leq 1;\ k \in \{0, \dots, K\} \tag{2.11}$$

$$|\epsilon_t^{\pi_k} + \epsilon_a^{\pi_k}| \leq 1;\ k \in \{0, \dots, K\} \tag{2.12}$$

$\rightarrow$ $\epsilon_t^{\pi_k}$ denotes the lateral torque caused by the items (2.8) and $\epsilon_a^{\pi_k}$ the lateral torque produced by the *Packed* (2.9).

→ The items allocated to each pallet cannot exceed its weight (2.13) and volume (2.14) limits.

$$\sum_{j=1}^{n^{\pi k}} X_{ij}^{\pi_k} \times w_j + \sum_{q=1}^{m^{\pi k}} Y_{iq}^{\pi_k} \times w_q \leq W_i;\ i \in \{1, \ldots, m\} \tag{2.13}$$

$$\sum_{j=1}^{n^{\pi k}} X_{ij}^{\pi_k} \times v_j + \sum_{q=1}^{m^{\pi k}} Y_{iq}^{\pi_k} \times v_q \leq\ V_i;\ i \in \{1, \ldots, m\} \tag{2.14}$$

→ The sum of the pallets weight capacities must be less or equal the aircraft's payload (2.10 and 2.15).

$$\sum_{i=1}^{m} (\sum_{j=1}^{n^{\pi k}} X_{ij}^{\pi_k} \times w_j^{\pi_k} + \sum_{q=1}^{m^{\pi k}} Y_{iq}^{\pi_k} \times w_q^{\pi_k}) \leq W_{max};\ k \in \{0, 1, \ldots, K\} \tag{2.15}$$

→ An item must be assigned to any pallet at most once or may not be assigned.

$$\sum_{i=1}^{m} X_{ij}^{\pi_k} \leq 1;\ j \in \{1, \ldots, n^{\pi_k}\} \tag{2.16}$$

→ *Packed* that have not yet reached their destination must remain on board (2.17). Some balancing procedures may cause the *Packed* to be moved to another pallet.

$$\sum_{i=1}^{m} Y_{iq}^{\pi_k} = 1;\ to_q \in L^{\pi_k};\ q \in \{1, \ldots, m^{\pi_k}\} \tag{2.17}$$

→ Items on the same pallet must have the same destination as the pallet (2.18, 2.19).

$$X_{ij}^{\pi_k} \leq X_{ij}^{\pi_k} \times (T_i^{\pi_k} - to_j + 1);\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n^{\pi_k}\} \tag{2.18}$$

$$X_{ij}^{\pi_k} \leq X_{ij}^{\pi_k} \times (to_j - T_i^{\pi_k} + 1);\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n^{\pi_k}\} \tag{2.19}$$

→ If there is a *Packed* on the pallet, they must have the same destination (2.20, 2.21).

$$Y_{iq}^{\pi_k} \leq Y_{iq}^{\pi_k} \times (T_i^{\pi_k} - to_q + 1);\ i \in \{1, \ldots, m\};\ q \in \{1, \ldots, m^{\pi_k}\} \tag{2.20}$$

$$Y_{iq}^{\pi_k} \leq Y_{iq}^{\pi_k} \times (to_q - T_i^{\pi_k} + 1);\ i \in \{1, \ldots, m\};\ q \in \{1, \ldots, m^{\pi_k}\} \tag{2.21}$$

3 Literature review

Considering air cargo transport, Table 3.1 lists the main works in the literature and the corresponding subproblems addressed, most of them dealing with **WBP**.

We also indicate whether the dimensions of the items were taken into account (**3-D** or **2-D**) and which solution method was used: heuristics (**H**), integer programming (**IP**), linear programming (**LP**), or parallel heuristics (**PH**). We also indicate if the minimization of loading and unloading costs (**MLU**) was included in the work scope.

Table 3.1 – Air cargo transport: literature, problems and features

Research	**APP**	**WBP**	**MLU**	**SPDP**	**TSP**	**2-D**	**3-D**	**H**	**IP**	**LP**	**PH**
(LARSEN; MIKKELSEN, 1979)	.	★	.	.	.	.	.	★	.	.	.
(BROSH, 1981)	.	★	.	.	.	.	.	.	.	★	.
(NG, 1992)	.	★	.	.	.	.	.	.	★	.	.
(HEIDELBERG *et al.*, 1998)	.	★	.	.	.	★	.	★	.	.	.
(MONGEAU; BES, 2003)	★	★	.	.	.	.	.	.	★	.	.
(FOK; CHUN, 2004)	.	★	.	.	.	.	.	.	★	.	.
(KALUZNY; SHAW, 2009)	.	★	.	.	.	★	.	.	★	.	.
(VERSTICHEL *et al.*, 2011)	.	★	.	.	.	.	.	.	★	.	.
(LIMBOURG *et al.*, 2012)	.	★	.	.	.	.	.	.	★	.	.
(ROESENER; HALL, 2014)	★	★	.	.	.	.	★	.	★	.	.
(VANCROONENBURG *et al.*, 2014)	★	★	.	.	.	.	.	.	★	.	.
(LURKIN; SCHYNS, 2015)	.	★	★	★	.	.	.	.	★	.	.
(ROESENER; BARNES, 2016)	.	★	.	.	.	.	.	★	.	.	.
(PAQUAY *et al.*, 2016)	★	.	.	.	.	.	★	.	★	★	.
(PAQUAY *et al.*, 2018b)	★	.	.	.	.	.	★	★	.	★	.
(CHENGUANG *et al.*, 2018)	.	★	.	.	.	★	.	★	.	.	.
(WONG; LING, 2020)	★	★	.	.	.	.	.	.	★	.	.
(WONG *et al.*, 2021)	★	★	.	.	.	.	.	.	★	.	.
(ZHAO *et al.*, 2021)	.	★	.	.	.	.	.	.	★	.	.
(ZHAO *et al.*, 2023)	★	★	.	.	.	.	.	★	.	.	.
(MIGUEL *et al.*, 2023)	.	★	.	.	.	.	.	.	★	.	.
(MESQUITA; SANCHES, 2024)	★	★	.	★	★	.	.	★	★	.	.
This work	★	★	★	★	★	.	★	★	★	.	★

As to the ★ symbol, it indicates that, except for the MIP solver, hybrid heuristics with IP procedures were used to compose the other solution methods (algorithms, heuristics, and metaheuristics).

As can be seen, so far (LURKIN; SCHYNS, 2015) is the only work that simultaneously addresses an air cargo (WBP) and a simultaneous pick-up and delivery problem (SPDP) subproblem. Although it is innovative, strong simplifications were imposed by the authors: in relation to loading, APP was ignored; with regard to routing, it is assumed a pre-defined flight plan is restricted to one destination. It is important to note that these authors consider an aircraft with two doors, and the minimization of loading and unloading costs (MLU) at the intermediate node was modeled through a container sequencing problem.

Referring directly to this work, (BRANDT; NICKEL, 2019, p. 409) comment: *However, not even these subproblems are acceptably solved for real-world problem sizes, or the models omit some practically relevant constraints.*

3.1 Solved WBP only

(LARSEN; MIKKELSEN, 1979) developed an interactive, computer-based procedure for solving the vehicle loading problem encountered when loading containers and pallets into an aircraft. They model with nonlinear programming and solve with heuristics, apply their solution method to a Boeing 747, handle seven different containers or pallets that must be reallocated to keep ground stability and balance, minimize handling time by changing pallet positions as little as possible, and consider two legs (one intermediate airport).

Their work is a foundational effort that our more contemporary and nuanced approach builds upon, especially with regard to computational methods, the variety of objectives considered, and potential real-world applications. The present author's approach not only handles ground stability but also flight stability and balance. Their problem is more about optimizing the existing space and weight distribution for a given aircraft, without the additional complexities of routing and delivery at multiple points within a network.

(BROSH, 1981) assumed a large number of similar cargo types, leading to a continuous optimization problem. He employed a more comprehensive model with linear constraints, taking into account the aircraft's center of gravity (CG) and its weight. Since the CG constraints became nonlinear due to weight distribution, he solved them as a series of simpler linear optimization problems.

This work's approach uses simpler constraints for the center of gravity, focusing on acceptable forward and backward limits. This simplifies the model by keeping the constraints linear, possibly making the problem easier to solve.

(NG, 1992) reports a multicriteria optimization approach to aircraft loading. Only pallets assembled by experienced loadmasters are employed in the optimization procedure. The model provides timely planning and improves airlift support for combat operations. It solves a WBP with pallets in fixed positions, maximizes volume, solves a cargo dropping sequence with 20 different items, handles multicriteria optimization with integer programming, and is applied to a C-130 aircraft by the Canadian Air Force.

(HEIDELBERG *et al.*, 1998) approached the airlift loading problem with WBP as a 2-dimensional bin packing problem (BPP), ignoring the height dimension and using the length and width of the cargo pallets and the aircraft's cargo bay. The author indicated that the classical methods of BPP are inadequate for aircraft loading because they ignore aircraft CG concerns and pallet volume. In this work's approach, pallets are fixed in number and position.

They address the airlift loading problem through a 2-dimensional bin packing approach, focusing on length and width dimensions while omitting height. This simplification underscores a challenge in classical bin packing methods, which typically do not consider the center of gravity (CG) and pallet volume-critical factors in aircraft loading. The approach assumes a fixed number and position of pallets, which streamlines the complexity of the problem to two dimensions.

(FOK; CHUN, 2004) developed a web-based application to first perform long-term forecasting based on an analysis of historical data, and then, secondarily, operational load planning with mathematical optimization. The container load planning is usually done roughly 2 hours before departure, when all the details of the cargo are expected to be present. They aim to make efficient use of space and load balancing, and they receive a set of filled containers to be conveniently allocated in the cargo bay.

(KALUZNY; SHAW, 2009) solve the problem of determining the arrangement of a set of items in a cargo hold that optimizes the load balance. Items are modeled as rectangles with specified dimensions, masses, and center of gravity offsets. The main decision variables determine the orientation and placement of a given set of items. The objective function can be chosen to minimize the deviation of the center of gravity from the target position or to maximize the function of the items loaded. Like (HEIDELBERG *et al.*, 1998), this approach does not palletize items. Boxes are arranged in the cargo bay, and a 2-D packing problem is solved.

(VERSTICHEL *et al.*, 2011) solve the WBP by selecting the most profitable subset of containers to be loaded into an aircraft. An integer programming approach to the WBP was introduced. Their aim was to increase the value of loaded cargo and decrease deviation from the optimal center of gravity. The same objective as (KALUZNY; SHAW, 2009), but containers are considered already filled, i.e., they do not palletize items.

(LIMBOURG *et al.*, 2012) developed a mixed-integer program designed for optimally loading a set of pallets into a compartmentalized cargo aircraft, specifically the Boeing

747. Their approach aimed to optimally position the CG by rearranging pallets. In this case, containers are distributed in diverse areas of the same cargo bay.

On the other hand, this work simply positions the CG into its permitted range with the sole objective of maximizing the cargo utility score. As long as the CG deviation of a workable solution is as low as it can go, there won't be any safety issues because fly-by-wire aircraft with automatic trimming take care of the longitudinal stability. This should also make the plane able to carry more cargo.

(ROESENER; BARNES, 2016) proposed a *Tabu Search* application to solve the *Dynamic Airlift Loading Problem* (DALP). Given a set of palletized cargo items that require transportation between two ports in a given time frame, DALP seeks to partition the pallets into aircraft loads, select an efficient and effective subset of aircraft, and assign the pallets to allowable positions on those aircraft. Their approach is concerned with more than one aircraft departing from the same node rather than with the simultaneous pickup and delivery of a single aircraft along a route.

In the work of (CHENGUANG *et al.*, 2018), various cargoes are loaded, in an appropriate manner, into various kinds of transport aircraft with constraints on volume, weight, and gravity center. A new hybrid genetic algorithm (GA) is proposed to solve the multi-constraint loading problem of transporting aircraft while consuming the least amount of fuel. An optimization algorithm is applied to optimize single-aircraft loading in GA decoding, and the procedure of hybrid GA is summarized for the multi-aircraft loading issues. In the case study, eight kinds of cargo are distributed among three different aircraft.

As a continuation of this work, a future article will address these two final approaches to using multiple aircraft.

(ZHAO *et al.*, 2021) addressed the WBP with CG envelope constraints for air cargo transportation and considered both the problem of maximizing profits by selecting ULD from a larger number of ULD for an aircraft and the problem of optimizing the CG by locating each ULD in an appropriate position within the aircraft's cargo bay. They provide an integer programming model for the WBP with features of the knapsack and assignment problems combined. In terms of WBP, their approach is similar to this work.

3.2 Solved APP only

(CHAN *et al.*, 2006) worked on a case study with heterogeneous pallets, and their method generated the loading plan for each pallet, aiming to minimize the total cost of shipment. Their work did not present any approach regarding the CG of aircraft or cargo balancing. They also did not deal with pallet arrangement in the cargo bay but rather pallet completion through a 3-D bin packing heuristic. Their solution is of great relevance in commercial and industrial applications, where cargo items tend to be less dense.

(PAQUAY *et al.*, 2016) discuss the problem of optimizing the loading of a set of strongly heterogeneous boxes into commercial aircraft containers with the goal of minimizing the unused volume within the container. They include pallet center of gravity considerations in their formulation but do not deal with simultaneous en route pickup and delivery.

(ROESENER; HALL, 2014) formulated a 3-D bin packing problem for mounting containers in specific positions within aircraft but also did not deal with simultaneous pickup and delivery en route.

3.3 Solved PDP only

(MESQUITA; CUNHA, 2011) present a proposal for a solution to a problem of the Brazilian Air Force (FAB), which consists of defining transport routes with simultaneous pick-up and delivery from a central distribution terminal located in Rio de Janeiro. A solution method based on the *Scatter Search* metaheuristic integrated with the *Variable Neighborhood Descent* is proposed as an improvement method. This work does not consider pallet loading, CG displacement costs, or cargo balancing.

(NACCACHE *et al.*, 2018) describe an extension of the pickup and delivery problem with multi-pickups, then propose a vehicle flow formulation with time windows and an adaptive large neighborhood search algorithm tailored for this problem. They formally describe, model, and solve this in the field of pickup and delivery vehicle routing. Likewise, they solve the problem via branch-and-bound and a heuristic hybrid adaptive large neighborhood search with improvement operations. Their method is compared to the exact one to show how well it works.

According to (ÖZTAs; TUs, 2022), the vehicle routing problem with pick-up and delivery is NP-hard, which means that exact methods fail to find optimal solutions for large-scale instances in a reasonable amount of time. Their study aims to solve it using a hybrid algorithm combining iterated local search, variable neighborhood descent, and threshold acceptance metaheuristics. The proposed algorithm's main framework is *Iterated Local Search* is the main framework of the proposed algorithm. Since vehicle routing problems with simultaneous pickup and delivery carry out both pick-up and delivery operations, the amount of load in the vehicle changes after each customer visit.

It is important to highlight that in (ÖZTAs; TUs, 2022) the fluctuation in load affects the feasibility of the routes. The distances between visited locations on a route impact the total cost. The experimental results indicate that the proposed algorithm reaches the best-known solution values in a reasonable time for most of the test instances used for benchmarking in the literature. Although the proposed algorithm seems particularly successful in small and medium-sized problem instances, this is a novel approach to solving the PDP.

3.4 Solved WBP and APP

(MONGEAU; BES, 2003) addressed APP and WBP in aircraft to minimize fuel consumption and satisfy stability and safety requirements. Given a list of containers with specific weights and volumes, a subset of these containers must be assigned to a finite number of possible locations. Two objectives are optimized: (1) as much weight as possible should be loaded, and (2) the resulting CG of the aircraft should be as far aft as possible to minimize fuel consumption but not behind a limit imposed by stability requirements.

Differently, in this work, the pallets are planned to be assembled with the previous knowledge of their positions in the cargo bay in a schedule that fosters the same benefits as theirs but takes advantage of fine-tuning, as more item-pallet combinations proportionately provide more opportunities for torque adjustment.

(VANCROOONENBURG *et al.*, 2014) accomplished a case study on the Boeing 747-400, where they solved the Aircraft Weight and Balance Optimization Problem, which corresponds to APP and WBP in our approach. Their work finds the most profitable selection from a set of Unit Load Devices (ULDs) to be loaded onto slots where several cargoes can be stored. As a secondary objective, it minimizes the deviation in the CG of aircraft. Based on historical data, the authors showed how frequent situations in this aircraft can be solved through a MIP solver within a 10-minute algorithm run (these authors used the *Gurobi* MIP solver).

While their work aims at profitable selection of ULD and minimizing deviation in the CG, this author's research has multiple objectives, including a utility score, fuel consumption, and minimizing costs associated with the drag caused by CG displacement.

(WONG; LING, 2020) provided a mathematical model and optimization tool to aid in ACLP for an airline. According to the authors, even small improvements in load planning, such as the optimal allocation and placement of unit load devices to be stowed in an airplane, can have a significant impact on its operational and financial performance. Under the constraints of aircraft configuration, dangerous product segregation, weight and balance, and flight safety, their research attempts to maximize cargo loading profit while also improving operation efficiency. Their work does not address multi-leg operations planning.

(WONG *et al.*, 2021) state that real-time visualization and loading optimization are becoming increasingly important due to dynamic considerations, including the segregation of dangerous goods and inherent lithium batteries, weight balancing, and oversize cargo handling. A closed-loop dynamic air cargo loading digital twin system, integrating a cargo load plan optimization simulation, a multidimensional immersive virtual reality system, telematics, and real-time sensors, is proposed for connecting, monitoring, and controlling the operations in physical and virtual space. A virtual reality system is used to visualize and experiment with loading procedures. The system uses a feedback loop during sensor data capture to facilitate the decision-making process on the optimal cargo load plan. Their work is very creative and will make a big difference in how well air cargo works.

(ZHAO *et al.*, 2023) presented three models that use integer programming for air cargo planning and weight balance optimization: bi-objective optimization (BOM), combinatorial optimization (COM), and enhanced combinatorial optimization (IOM). Considering a Boeing 777F in several scenarios, the tests revealed performance problems: BOM is fast but produces a large CG deviation; COM offers accurate optimization but with impractical run times; and IOM provides a balanced solution, improving speed over COM but requiring high computational demands in some cases. Although IOM stands out for its effectiveness, all models face trade-offs between speed, accuracy, and computational efficiency. This work alerted us to potential performance issues in solution methods.

3.5 Solved WBP and PDP

Currently, (LURKIN; SCHYNS, 2015) is the only work that simultaneously addresses an aircraft cargo problem and a flight itinerary problem. Although it is innovative, strong simplifications were imposed by the authors: in relation to the item load, only the WBP was considered; regarding the flights, it is assumed a travel plan was previously defined and restricted to two legs (one intermediate node). On the other hand, as these authors consider an aircraft with two doors, the minimization of loading and unloading costs at the intermediate node was modeled through a container sequencing problem.

Referring directly to the work of Lurkin and Schyns, Brandt and Nickel comment: *However, not even these subproblems are acceptably solved for real-world problem sizes or the models omit some practically relevant constraints* (BRANDT; NICKEL, 2019, p. 409).

3.6 Special cases of the Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is the task of finding a route through a given set of cities with the shortest possible length. The study of this problem has attracted many researchers from different fields, e.g., mathematics, operations research, physics, biology, or artificial intelligence (REINELT, 1994, p. 1).

The concept of TSP with Profits, (FEILLET *et al.*, 2005), extends the scope of the traditional TSP by allowing for the possibility of not visiting all vertices. A profit accompanies each vertex. The primary goal is to achieve a simultaneous optimization of both the accumulated profit and the travel expenses. Both of these optimization requirements are present, either in the objective function or as a constraint. These authors' study presents a suggested taxonomy of TSP that incorporates profits and conducts a comprehensive survey of the relevant existing literature. This study identifies and compares various classes of applications, modeling approaches, and solution methodologies, including both exact and heuristic methods.

Their work is similar to this research in terms of profits and costs, except that all nodes must be visited and the relation score/costs for the trip must be maximized.

(PANTUZA; SOUZA, 2022) state that the prize collection traveling salesman problem (PCTSP) is a generalization of the TSP in which a tour starting at a root node must visit a subset of nodes to collect a prescribed amount of the total prize, minimizing the summation of travel costs and penalties associated with non-visited nodes. It is also different from our routing problem in that all nodes must be visited.

(BOWEN *et al.*, 2024) work presents a learning method for the pickup-and-delivery TSP (PDTSP), focusing on finding the shortest tour along a sequence of one-to-one pickup-and-delivery nodes. Classic OR algorithms for PDTSP are challenging to scale to large problems. Reinforcement learning (RL) is used to restrict solution search within a feasible space. The method is compared to classic OR algorithms and existing learning methods, showing that it can find tours shorter than baselines.

The Traveling Salesman Problem with Pickup and Delivery (TSPPD or PDTSP) is defined on a graph containing pickup and delivery vertices between which there exists a one-to-one relationship. The problem involves figuring out a minimum cost that prioritizes each pickup vertex over its corresponding delivery vertex.

The present work also deals with a special case where the focus is on the most efficient route, not the shortest one. Similar to the PCTSP and PDTSP, it aims to maximize the ratio of the sum of scores, or the benefit, of items picked up and delivered, divided by the cost of a trial tour.

3.7 Some other relevant works

(OLJA *et al.*, 2010) present a relevant research in a number of ways. First, it emphasizes the importance of accurate weight and balance calculations in ensuring the safety of aircraft operations. Second, it highlights the potential risks of load sheet errors. Third, it provides insights into the factors that contribute to weight and balance accidents, which can be used to develop strategies for preventing these accidents.

(TANG, 2011) developed a method for solving air cargo loading problems under stochastic demands with a scenario decomposition genetic algorithm, minimizing handling costs. He does not solve a PDP or an APP. We assume that he does not consider the load-balancing part of the problem (WBP) because he does not mention it.

(BORTFELDT; WÄSCHER, 2013) present a state-of-the-art survey on practically relevant constraints in container loading. They also review the modeling approaches as well as exact and heuristic algorithms. It has been claimed, though, that the proposed approaches are of limited practical value since they do not pay enough attention to the constraints encountered in practice. They figure out what practical things need to be considered when dealing with problems with loading containers, and they look at whether and how these things are considered in methods for solving such problems. They do not describe the APP or the PDP.

3.8 Works more related to this work

The studies of (LIMBOURG *et al.*, 2012), (LURKIN; SCHYNS, 2015), (MONGEAU; BES, 2003), (VERSTICHEL *et al.*, 2011), (ZHAO *et al.*, 2021), and especially (LURKIN; SCHYNS, 2015), which we started from, relate most closely to this work. These five articles indeed deal with commercial cargo aircraft with predefined positions and standardized ULD, use exact methods, and consider the aircraft's center of gravity. We nonetheless significantly depart from these works.

Although the research in this section has a strong influence on this work, there are some aspects that differentiate it. None of the problems reported deal with pallets yet on board with multiple destinations, and none of them try to find the most efficient tour, considering *Score* (or other client-chosen parameter) maximization while minimizing cost increases due to CG displacement along the route. This last has a direct impact on carbon emissions.

3.9 Road-map

That way, in the following chapters, we accomplish some planned steps:

(1) Generate problem scenarios and instances for benchmarking.

(2) Develop a complete solution process for these scenarios and instances.

(3) Develop an approach for ACLP+RPDP relying on a general-purpose MIP solver.

(4) Solve these scenarios and instances with the solver implemented.

(5) If the solver performance for ACLP+RPDP is not acceptable for operational use or even does not manage to solve it:

   (a) Look for a suitable meta-heuristic to substitute the commercial solver.

   (b) In case none present an acceptable result, develop a solution method that would quickly provide a feasible and not necessarily optimal solution in any situation.

4 Solution approaches

To approach some solution alternatives, we consider a tour consisting of several delivery and pickup points and the return to the base. This drove us to customize the constraints of the Air Palletization Problem (APP), the Pickup and Delivery Problem (PDP), and the Weight and Balance Problem (WBP) in each node, to assess the performance of five well-known metaheuristics and to develop a fast heuristic for aircraft loading in multi-leg operations, as well as a Mixed-Integer Program based on the state-of-the-art *Gurobi* MIP solver.

Possible alternatives could be the Benders and Dantzig-Wolfe decomposition, as these methods are powerful tools for solving large-scale optimization problems by breaking them down into smaller, more manageable subproblems. However, due to the strong interconnectedness of the subproblems, we were unable to find a way to apply these methods to the ACLP-RPDP.

We implemented five metaheuristics and a special method to solve ACLP+RPDP, which are presented in this chapter. A permutation on L yields $K!$ possible tours to investigate, where K is the index of last node before returning to the base. Throughout this chapter, π_k is a node in position k of a route, and it will be represented in superscript.

The metaheuristics implemented solve a node at a time, following the tour sequence, considering only the items in the node (N^{π_k}), and the *Packed* already on board (Q^{π_k}), whose destinations are in L^{π_k}, the subset of possible destinations departing from π_k.

An important observation is that the algorithms here presented were designed based partly on the traditional ideas from the literature and partly on the experiences learned during initial implementation tests. This has resulted in slightly different algorithms from the research sources.

4.1 Metaheuristics selection

As ACLP+RPDP resembles the *Multidimensional Multiple Knapsack Problem* (MMKP) in each node, having pallets as multiple knapsacks to allocate items to be embarked, we researched for metaheuristics previously used to solve MMKP, as alternatives to solve the *Air Cargo Load Planning* (ACLP) part of this thesis' problem.

The implemented metaheuristics were all referred by (LAABADI *et al.*, 2018) to solve variants o the Multiple Knapsack Problem: *Ant Colony Optimization* (ACO), *Noising Methods* (NM), *Tabu Search* (TS), *Greedy Randomized Adaptive Search Procedure* (GRASP), and *Genetic Algorithm (GA).* As (FIDANOVA, 2006) solved the MMKP with ACO, we decided to implement it as a candidate to solve ACLP+RPDP.

(ZHAN *et al.*, 2020) showed advantages of NM to solve the *Multidimensional Knapsack Problem*, so we also decided to include NM as a candidate method. The books (HANDBOOK..., 2010) and (HANDBOOK..., 2003), both with the same title *Handbook of Metaheuristics*, provide comprehensive overviews of various optimization techniques, including a dedicated chapter on Noising Methods. They discuss the advantages in high-dimensional problems and presents applications in various fields.

As (NIAR; FREVILLE, 1997) presented a parallel TS for MMKP, reducing run-time and setting dynamically strategy parameters, we also decided to implement TS, but without parallelism. (ALONSO *et al.*, 2019) utilized GRASP, a heuristic that found high-quality solutions in short run times. So, we also decided to implement GRASP. (SHAH, 2020) used GA to solve a *0/1 Multidimensional Knapsack Problem*, it also seemed to be a good candidate to solve the ACLPP.

We considered several ideas from these authors, and we were careful to use the same data structures and procedures in all implementations to enforce fair results comparison, except for GA, where we used the DEAP (Distributed Evolutionary Algorithms in Python), an evolutionary computation framework. For more details, see (FORTIN *et al.*, 2012) and `github.com/deap/deap`, due to its ease of implementation and its wide applications in real-world problems.

Common elements used by the methods

To guarantee that performance comparisons are made correctly, our implementations use the same elements described below:

<u>Common to the MIP solver and to the metaheuristics:</u>

Solution printing: All methods use the same procedure to print solutions. This facilitates results comparisons as well as feasibility checks.

Pallets destinations: Algorithm 21 is applied before solving, to define pallets destinations based on volume demands for each node.

Best tour selection: each tour problem is solved, and the result is saved for later best-tour selection.

Data structure: items, pallets and edges (pallet-item possible allocations) are static typed arrays from the *multiprocessing* Python library, which are initialized with the same data loaded from text files.

Stopping criterion: all metaheuristics stop the search after 0.7-seconds per node, to keep the tour solution run time within acceptable operational range. With this run time limit of 0.7 seconds per node, solutions should be found in less than 10 seconds for scenarios 1 and 2, less than 30 seconds for scenario 3, 2 minutes for scenario 4, 10 minutes for scenario 5, and less than an hour for scenario 6.
(For other comparisons, the stopping criteria will be differently set). Later we changed the stopping criterion to be proportional to the volume to be loaded in each node.

Used by the metaheuristics only:

Feasibility: the procedure to evaluate each individual or solution vector.

Local search: the procedure that tries to insert edges to improve solution quality.

Edge attractiveness: The selection of edges for $E_N^{\pi_k}$ uses the *edge attractiveness* $\theta_{ij}^{\pi_k}$, Equation 4.3, which can be understood as the tendency to allocate the $item_j^{\pi_k}$ to the pallet p_i. It is directly proportional to the score, and inversely relative to the volume and torque of each item.

The higher the score and the lower the item volume as the pallet centroid $|D_i|$ approaches the aircraft CG, the better the edge attractiveness $\theta_{ij}^{\pi_k}$. For $i, q \in \{1, 2, \ldots, m\}$ and $j, s \in \{1, 2, \ldots, n^{\pi_k}\}$.

$$w_{max} \leftarrow \max\left\{ w_j^{\pi_k} \;\middle|\; for\ j \in \{1, 2, \ldots, n^{\pi_k}\} \right\} \tag{4.1}$$

$$d_{max} \leftarrow \max\left\{ |D_i| \;\middle|\; for\ i \in \{1, 2, \ldots, m\} \right\} \tag{4.2}$$

$$\theta_{ij}^{\pi_k} = \frac{s_j^{\pi_k}}{v_j^{\pi_k}} \cdot \left(1 - \frac{w_j^{\pi_k} \cdot |D_i|}{w_{max} \cdot d_{max}}\right) \tag{4.3}$$

The expression $w_{max} \cdot d_{max}$ in Equation 4.3 represents the worst torque caused by the heaviest item.

Finally, we enforced that the only difference among these methods was the search strategy.

Some considerations on the metaheuristics search space

In the Genetic Algorithm (GA), the search space is a population of solutions with a number of generations. In Ant Colony Optimization (ACO), many ants have a number of pheromone trails. The Noising Methods (NM) perform multiple iterations at each of the numerous noise levels. In the Tabu Search (TS), there are many iterations with a number of tries to update the Tabu Queue. In the Greedy Randomized Adaptive Search Procedure (GRASP), there are many iterations with a number of tries with the solutions from the Restrict Candidates List. Each of the metaheuristics employs a particular set of strategies to explore the solution space, which may require more computation resources than simple heuristics.

Here's a breakdown of some points, as well as some additional thoughts on the search space in metaheuristics:

- GA: The population size and number of generations define the search space. A larger population allows for more diverse solutions but requires more computation.
- ACO: The number of ants and pheromone trails influences exploration and exploitation. More ants can explore a wider area, but managing trails adds complexity.
- NM: The number of iterations and noise levels determine the search space. Noise can help escape local optima, but too much noise can hinder convergence.
- TS: Iterations and Tabu list size are crucial. A larger list prevents revisiting recent solutions, but a small list may get trapped in local minima.
- GRASP: Iterations and the size of the restricted candidate list both play a role. This list helps focus search but requires balancing exploration and exploitation.

There is a trade-off between exploration and exploitation. Metaheuristics navigate the search space by balancing exploration (finding new areas) and exploitation (refining promising areas). While a large search space in metaheuristics can be computationally expensive, simple heuristics are faster, but may become stuck in local optima.

Some additional considerations are that the design of the search space within a metaheuristic can be influenced by the specific problem under consideration. Tuning parameters like population size, number of iterations, or noise levels can significantly impact performance.

Overall, understanding the search space in metaheuristics is essential for effectively utilizing these powerful optimization tools.

4.2 The main process

Once the assumptions of this work and the mathematical modeling of the problem are presented, it is easy to see that ACLP+RPDP is NP-hard, as the subproblems are NP-hard.

Real cases are more complex as they have hundreds of different items in each node and involve three intractable sub-problems: APP, WBP and PDP. Through the mathematical modeling presented in the previous chapter, we verify that integer programming is not able to solve these cases in feasible time. Thus, it is necessary to adopt some strategy to find a viable solution, not necessarily optimal, that seeks to maximize the objective function $\boldsymbol{f}$.

Our strategy is based on the fact that, in real cases, K is usually small. Specifically, we consider $K \leq 6$ throughout this work, which is a higher value than usual in *Brazilian Air Force* missions. As a result, if there are fast node-by-node solutions that allow us to construct a complete tour, we will be able to test all possible $K!$ tours and thus select the one that provides the best sum for the $\boldsymbol{f}$ function, i.e., the best score sum to unbalance cost ratio.

Each node, except the base, inherits information from the previous nodes (pallets that must remain on board), and must be solved taking into account the remaining not visited nodes. One more reason for a node-by-node approach.

The tactic will be, at each shipping node, to predefine the destinations of the pallets at that node. In this way, a number of pallets will be reserved proportional to the volume demanded by each destination at the shipping node.

The present author could have used another criterion, but it was observed in the experiments that volume is more constrictive in airlift.

Once the destinations of the pallets are defined, our methods uses serial and multiprocessing heuristics to find the best possible node-by-node solutions. This strategy is summarized in Algorithm 1.

The argument *method* corresponds to one of the heuristics that will be described ahead in this thesis, or the MIP solver.

In this algorithm, there are six values for the *scenario* parameter, according to Table 6.1 (the scenarios), which defines K, the sets of nodes, the aircraft, the pallets and the costs from Tables 1.3 (smaller aircraft) or 1.4 (larger aircraft) that will be used (line 2).

Throughout this work, when we refer to *surplus*, we mean a volume surplus of 20%, 50% or 100% more than the aircraft capacity, to make the problem more difficult to solve ($surplus \in \{1.2, 1.5, 2.0\}$).

Algorithm 1 Main

```
1: procedure SOLVE_ACLP_RPDP($method,\ scenario,\ surplus,\ timeLim,\ S_k$)
2:     Let $L,\ C,\ M$ be according to $scenario$
3:     $N \leftarrow getItems(scenario,\ surplus)$
4:     Let $G(L, C, M, N)$ be an empty tour solution
5:     $G^* \leftarrow G$
6:     $f_{max} \leftarrow 0$
7:     $ntours \leftarrow |S_K|$
8:     for each $\pi \in S_K$ do
9:         $G^\pi,\ f(G^\pi) \leftarrow SolveTour(\pi,\ L,\ C,\ M,\ N,\ method,\ timeLim/ntours)$
10:        if $f(G^\pi) > f_{max}$ then
11:            $f_{max} \leftarrow f(G^\pi)$
12:            $G^* \leftarrow G^\pi$
13:    return $G^*(L,\ C,\ M,\ N,\ E_L,\ E_M^N)$
```

In line 3, the items are retrieved from the text files previously generated by Algorithm 19; in line 4, we have an empty solution with reference to the set of nodes L, the set of travel costs C, pallets M, and the items N in all nodes to be transported; and $timeLim/ntours$ in line 9 is the time limit for each node solution.

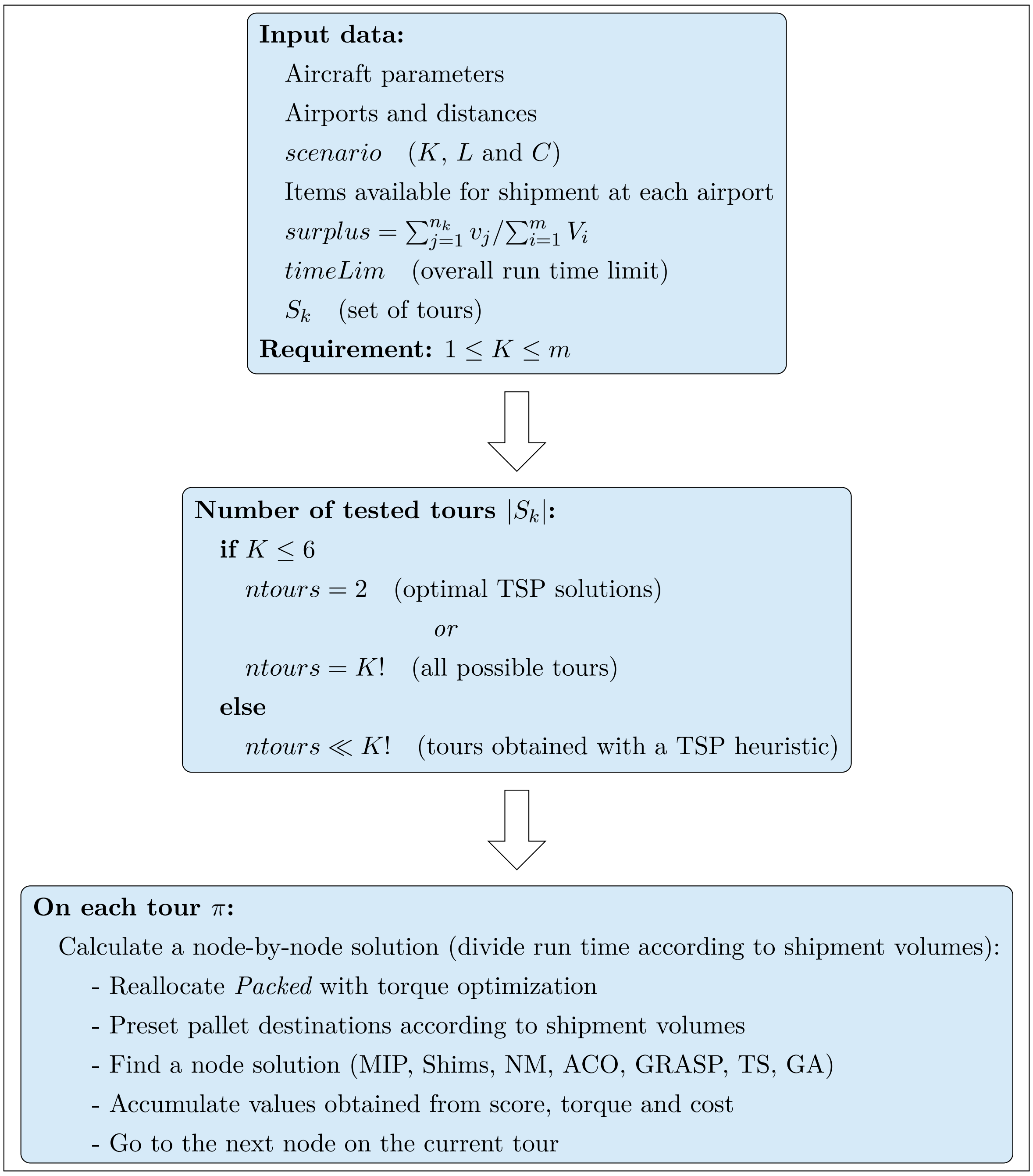
The loop of lines 8-12 goes through all permutations, where the node-by-node solutions are performed by $SolveTour()$, whose result is stored in $f(G^\pi)$.

The best outcome among all $K!$ tours will be the solution G^* for the *method*, *scenario*, and volume *surplus* chosen (line 13), where E_L contains the edges connecting the nodes of the best tour, and E_M^N the items loaded in all nodes onto the pallets.

In practical cases, we know that a common aircraft has $m = 18$ pallets, flight itineraries have $K \leq 6$ nodes plus the base, and each node has hundreds of items to be shipped. We also know that missions with fewer nodes are more frequent than longer ones. Under these circumstances, we can adopt some important strategies summarized in Figure 4.1:

1. We consider that the number of destinations is smaller than the number of pallets ($K < m$), and we avoid the trivial case where $K = 1$. With this premise, we can preset the destinations of the pallets at each shipping node, reserving a number of pallets proportional to the volume available for each destination. We could have used another criterion, but it was observed in the experiments that the volume is more constrictive in airlift.

2. An important parameter is the number *ntours* of tours tested. In practical cases where $K \leq 6$, we have the possibility to check all possible tours ($ntours = K!$). In this situation, as K is small, we can also specially analyse the two optimal solutions of the corresponding TSP ($ntours = 2$). Finally, in cases where $K > 6$, we will use a heuristic to select a number of distinct tours ($ntours \ll K!$), and search among them for the one that provides the best value for the objective function.
3. To compare the performance of each strategy, an overall run time limit *timeLim* is established and divided by *ntours* tours. In turn, the run time limit for each tour will be distributed among its nodes in proportion to the volume available for boarding.
4. At each node of a tour, the *Packed* that remain on board are reallocated on pallets to minimize torque on the aircraft. This calculation is done quickly using a MIP solver. Then, the destinations of the pallets are previously defined in proportion to the shipment volume. Finally, considering the run time limit of each node, we will use a MIP solver and five well-known meta-heuristics to find the best allocation of shipping items: *Ant Colony Optimization* (ACO), *Noising Methods* (NM), *Tabu Search* (TS), *Greedy Randomized Adaptive Search Procedure* (GRASP), and *Genetic Algorithm* (GA).
5. We will generate benchmarks using the *surplus* parameter, which is a value in $\{1.2, 1.5, 2.0\}$. It corresponds, at each node k, to the ratio between the sum of the volumes of the items and the load capacity of the pallets ($surplus = \sum_{j=1}^{n_k} v_j / \sum_{i=1}^{m} V_i$). This parameter allows us to verify the different behaviour of each method, according to *scenario* and the quantity of items available for shipment.
6. We will do tests by varying the number K of destinations, the set L of nodes, and the costs C. Each group of values tested is called *scenario*, according to Tables 1.2 and 6.1, where $1 < K \leq 6$. After finding the method with the best performance in node-by-node solution, we will test it in solving cases with $K > 6$.



Input data:
Aircraft parameters
Airports and distances
$scenario$ (K, L and C)
Items available for shipment at each airport
$surplus = \sum_{j=1}^{n_k} v_j / \sum_{i=1}^{m} V_i$
$timeLim$ (overall run time limit)
S_k (set of tours)
Requirement: $1 \leq K \leq m$

Number of tested tours $|S_k|$:
```
if $K \leq 6$
    $ntours = 2$   (optimal TSP solutions)
                   or
    $ntours = K!$   (all possible tours)
else
    $ntours \ll K!$   (tours obtained with a TSP heuristic)
```

On each tour π:
Calculate a node-by-node solution (divide run time according to shipment volumes):
- Reallocate *Packed* with torque optimization
- Preset pallet destinations according to shipment volumes
- Find a node solution (MIP, Shims, NM, ACO, GRASP, TS, GA)
- Accumulate values obtained from score, torque and cost
- Go to the next node on the current tour

Figure 4.1 – Solution process

4.3 The algorithm to solve a tour

In addition to the set of nodes, pallets, costs and items, $SolveTour$, described in Algorithm 2, receives the parameter *method* for solving the node-by-node subproblems and the parameter π, which is a permutation that defines the order of visits in tour π.

All tours start and end at π_0 (lines 2-3).

After initializing the score and cost values (lines 4-5), there is a loop for the $K + 1$

tours (lines 10-26).

Algorithm 2 Solves the sequence of nodes of tour π

```
 1: procedure $SolveTour(\pi, L, M, C, N, method, tourTime)$
 2:     $\pi_0 \leftarrow 0$
 3:     $\pi_{K+1} \leftarrow 0$
 4:     $score \leftarrow 0$
 5:     $cost \leftarrow 0$
 6:     $G^\pi \leftarrow \{\ \}$
 7:     $sumVol \leftarrow 0$
 8:     for $k \in \{1, 2, 3, ..., K\}$ do
 9:         $sumVol \leftarrow sumVol + \sum_{j=1}^{n^{\pi_k}} v_j$
10:     for $k \in \{1, 2, 3, ..., K\}$ do
11:         $nodeTime = tourTime \cdot (\sum_{j=1}^{n^{\pi_k}} v_j / sumVol)$
12:         for $i \in \{1, 2, 3, ..., m\}$ do
13:             $T_i^{\pi_k} \leftarrow -1$                          ▷ reset this pallet destination
14:         if $k = 0$ then                                  ▷ in the base - nothing on board
15:             $L^{\pi_0} \leftarrow L$
16:             $Q^{\pi_0} \leftarrow \varnothing$
17:         else                               ▷ there are some pallets that will remain on board
18:             $L^{\pi_k} \leftarrow L^{\pi_k} - \{\pi_k\}$
19:             $Q^{\pi_k} \leftarrow UpdatePacked(\pi_k)$
20:         $G_1 \leftarrow InitialSolution(M,\ N^{\pi_k},\ \pi_k,\ L^{\pi_k},\ Q^{\pi_k})$
21:         $G_2 \leftarrow SetPalletsDestinations(G_1)$                ▷ only for the empty pallets
22:         $G_3 \leftarrow SolveNode(method, \pi_k, G_2, nodeTime)$
23:         $s, \epsilon \leftarrow ScoreAndDeviation(\pi_k, G_3)$
24:         $score \leftarrow score + s$
25:         $cost \leftarrow cost + c_{\pi_k, \pi_{k+1}} * (1 + c_g * |\epsilon|)$
26:         $G^\pi \leftarrow G^\pi \cup \{G_3\}$
27:     $f(G^\pi) \leftarrow score/cost$
28:     return $G^\pi(L,\ C,\ M,\ N,\ E_L,\ E_M^N),\ f(G^\pi)$
```

In line 11 the node time limit is proportional to the shipment volume.

Initially we set pallets destination as -1 (line 13).

When the aircraft is at node π_0, the initial graph G_1 is empty because it has no *Packed* (line 14). Otherwise, the set L^{π_k} of remaining nodes when departing from π_k is updated (line 18), and *UpdatePacked* (line 19) returns the *Packed* that have not yet reached their destinations and remain on board, rearranging them on the pallets to minimize CG deviation. This allocation is stored in graph G_1 (line 20).

The initial solution referred to in line 20 is generated by Algorithm 3, which is required by all methods, and also minimizes the CG displacement. This initial solution represents the system state out of the base, with all delivered cargo disembarked, but the *Packed* destined for not-visited nodes still remain on board.

Algorithm 3 Initial solution

```
1: procedure $InitialSolution(M,\ \pi_k)$
2:     $M \leftarrow$ set of pallets with destinations defined by Algorithm 21.
3:     $Q^{\pi_k} \leftarrow$ *Packed* to be kept on board in node $\pi_k$
4:     $G_1 \leftarrow G(M \cup Q^{\pi_k})$
5:     Minimize CG displacement on $G_1$ according to Equation 6.1
6:     return an initial solution $G_1$
```

In the context of this work, we know that $m > K$, once the aircraft has 7 or 18 pallets and $K \leq 6$, allowing there to be at least one pallet for each node to be visited. *SetPalletsDestinations* (line 21) presets the destination of each pallet based on the volume demands caused by the items to be embarked in the current node, without changing the pallets destination pallets already with *Packed.*

Finally, *SolveNode* includes the edges corresponding to the items shipped at the current node, returning the graph G_3 (line 22). The score and the CG deviation of this graph are calculated (line 23) and accumulated (lines 24-25), allowing the final result of this tour (line 28).

The procedure *UpdatePacked* finds the best allocation for the *Packed* that remain on board. Initially, the set Q^{π_k} is created, with the *Packed* that did not reach their destination yet.

Then *MinCGDeviation* is run through a MIP solver to relocate the *Packed* on the pallets minimizing torque and ensuring that they all remain on board, one on each pallet. As there are few variables, the MIP solver returns an allocation $E_Q^{\pi_k}$ very quickly.

$ScoreAndDeviation(\pi_k, G_3)$ evaluates the allocation graph generated by *SolveNode*, returning the corresponding score and CG deviation. It consists of a loop that goes through all the pallets, accumulating the scores and the torques of the shipped items, allowing the final calculation of the CG deviation.

4.4 The Greedy algorithm for initial solutions

Almost all methods to be described ahead in this chapter (except GA, that starts from a random generated population) will need initial solutions. That's the purpose of this section to describe the greedy algorithm implemented to generate these initial solutions.

The Greedy algorithm (4) may receive a partial solution with only *Packed* on board and, with a greedy approach based on $\theta_{ij}^{\pi_k}$, builds a simple solution.

Arguments η_1 and $level_1$ are reserved for future use in this Thesis.

For now, let's consider the argument $level_1 = 1.0$ to make the solution limited by exactly the pallet volumetric capacity.

This method also updates the aircraft torque τ^{π_k} in node π_k, as well as argument η_1.

Line 2 represents the solution graph G with it's pallets (M), *Packed* (Q^{π_k}) and edges between pallets and *Packed* $(E_Q^{\pi_k})$.

Algorithm 4 Mount a greedy solution for node π_k.

```
1: procedure Greedy($\pi_k$, $G$, $\tau^{\pi_k}$, $level_1$)
2:     $G(M,\ Q^{\pi_k},\ E_Q^{\pi_k})$
3:     $volume \leftarrow \{0 \mid i \in \{1,2,3,...,m\}\}$
4:     $\eta_1 \leftarrow \{0 \mid i \in \{1,2,3,...,m\}\}$
5:     $E_N^{\pi_k} \leftarrow \varnothing$
6:     for $i \in \{1,2,3,...,m\}$ do
7:         for $q \in \{1,2,3,...,m\}$ do
8:             if $(p_i, a_q^{\pi_k}) \in E_Q^{\pi_k}$ then
9:                 $volume_i \leftarrow volume_i + v_q^{\pi_k}$        ▷ Packed contribution to pallet $i$ volume
10:                $\tau^{\pi_k} \leftarrow \tau^{\pi_k} + w_q^{\pi_k} \cdot D_i$        ▷ Packed contribution to aircraft torque
11:        for each $e_{ij}^{\pi_k}$ in non-ascending order of $\theta_{ij}^{\pi_k}$ do
12:            if $(E_N^{\pi_k} \cup \{e_{ij}^{\pi_k}\})$ is feasible and $(volume_i \leq V_i \cdot level_1)$ then
13:                $E_N^{\pi_k} \leftarrow E_N^{\pi_k} \cup \{e_{ij}^{\pi_k}\}$
14:                $volume_i \leftarrow volume_i + v_j^{\pi_k}$        ▷ item contribution to pallet $i$ volume
15:                $\tau^{\pi_k} \leftarrow \tau^{\pi_k} + w_j^{\pi_k} \cdot D_i$        ▷ item contribution to aircraft torque
16:                $\eta_1^i \leftarrow \eta_1^i + 1$
17:    return $G(M,\ Q^{\pi_k},\ N^{\pi_k},\ E_Q^{\pi_k} \cup E_N^{\pi_k}),\ \tau^{\pi_k},\ \eta_1$
```

This method returns a greedy solution (graph G) with it's pallets (M), *Packed* (Q^{π_k}), items (N^{π_k}) and edges connecting pallets with *Packed* or items $(E_Q^{\pi_k} \cup E_N^{\pi_k})$.

Algorithm 4 takes in the worst case $\mathcal{O}(m^2 + m \cdot n^{\pi_k})$ steps. However, if the number of items is much larger than the number of pallets $(n >> m)$, then the time complexity can be simplified to $\mathcal{O}(m \cdot n^{\pi_k})$.

4.5 Ant Colony

In ACO, introduced by (DORIGO, 1992), a population of A ants performs independent search, where each Ant_a finds its own solution G_a.

(JU *et al.*, 2021) explore troops' movement from one location to another using available means. To minimize damage from enemies, the military simultaneously uses reconnaissance and transportation units during troop movements. The authors propose a vehicle routing problem considering reconnaissance and transportation for wartime troop movements. They used an ACO metaheuristic, and computational experiments were conducted to compare the ACO algorithm's performance with that of the Integer programming model. According to these authors, the performance of the ACO algorithm was shown to yield excellent results, even for the real-size problem.

Each edge e_{ij} has an attribute $ga_{ij}^{\pi_k}$ (*general attractiveness*), Equation 4.4, whose value depends on the trail pheromone $\phi_{ij}^{\pi_k}$ and the edge heuristic attractiveness $\theta_{ij}^{\pi_k}$. If $e_{ij}^{\pi_k}$ is included in G_a, Ant_a increases solution trails pheromone $\phi_{ij}^{\pi_k}$.

$$ga_{ij}^{\pi_k} = (\phi_{ij}^{\pi_k})^{\alpha} \cdot (\theta_{ij}^{\pi_k})^{\beta} \tag{4.4}$$

α and β are ACO constants in Equation 4.4, and $\phi_{ij}^{\pi_k}$ and $\theta_{ij}^{\pi_k}$ are kept in the 0-1 range.

A greedy solution G^* is created from G^+ (line 3) and (Algorithm 4). The third argument with value 1.0 refers to a volume limitation not used by ACO. The returned parameter η_1 also is not used by ACO.

A per iteration number of ants was empirically set (line 5). The total number increases until the search does not reach stagnation, this yielded solutions with a good quality-to-time ratio.

The global $m \cdot n^{\pi_k}$ matrices $\phi_{ij}^{\pi_k}$ and $ga_{ij}^{\pi_k}$ (lines 10-11) are used and updated by all ants.

In line 18, edges are chosen by a proportional roulette wheel mechanism, based on matrix $ga_{ij}^{\pi_k}$ in E^{π_k} universe, except the already tested edges.

$$\Delta\phi_{ij}^{\pi_k} = \frac{ga_{ij}^{\pi_k}}{A} \tag{4.5}$$

Equation 4.5 states that the pheromone increments $(\Delta\phi_{ij}^{\pi_k})$ are calculated by dividing the current *general attractiveness* by the number of ants.

A ants are generated in each iteration (lines 14-22).

Algorithm 5 Ant Colony Optimization

```
1: procedure $ACO(\pi_k,\ G_1,\ nodeTime)$
2:     $\tau^{\pi_k} \leftarrow 0$
3:     $G^*,\ \tau^{\pi_k},\ \eta_1 \leftarrow Greedy(\pi_k,\ G_1,\ \tau^{\pi_k},\ 1.0)$      ▷ a initial greedy solution based on $\theta_{ij}^{\pi_k}$
4:     $E^n \leftarrow \exists e_{ij} \notin E^*$      ▷ set of edges not in the initial solution
5:     $A \leftarrow 8$      ▷ number of ants per iteration
6:     $Ants \leftarrow 0$
7:     $\alpha \leftarrow 1.0$
8:     $\beta \leftarrow 4.0$
9:     $\rho \leftarrow 0.2$
10:    $ga_{ij}^{\pi_k} \leftarrow 0$
11:    $\phi_{ij}^{\pi_k} \leftarrow 1$
12:    $stagnant \leftarrow 0$
13:    while $stagnant < 3$ and $runtime < nodeTime$ do
14:        for $a = 1$ to $A$ do
15:            $Ants \leftarrow Ants + 1$
16:            $G_a(E_a) \leftarrow G^+$
17:            while $\exists e_{ij} \in E^n$ not tested yet do
18:                $e_{ij} \leftarrow roulette(ga_{ij}^{\pi_k}, E_n)$
19:                if $E_a \cup \{e_{ij}\}$ is feasible then
20:                    $E_a \leftarrow E_a \cup \{e_{ij}\}$
21:            if $f(G_a) > f(G^+)$ then
22:                $G^+ \leftarrow G_a$
23:                $\phi_{ij}^{\pi_k} \leftarrow \phi_{ij}^{\pi_k} - \Delta\phi_{ij}^{\pi_k}$
24:                $ga_{ij}^{\pi_k} \leftarrow$ result from 4.4
25:        if $f(G^+) > f(G^*)$ then
26:            $G^* \leftarrow G^+$
27:            $stagnant \leftarrow 0$
28:            for each $e_{ij} \in E^*$ do
29:                $\phi_{ij}^{\pi_k} \leftarrow (1-\rho)\cdot\phi_{ij}^{\pi_k}$
30:                $\phi_{ij}^{\pi_k} \leftarrow \phi_{ij}^{\pi_k} + \Delta\phi_{ij}^{\pi_k}$
31:                $ga_{ij}^{\pi_k} \leftarrow$ result from 4.4
32:        else
33:            $stagnant \leftarrow stagnant + 1$
34:        for all $\tau_{ij}$ do
35:            $\phi_{ij}^{\pi_k} \leftarrow (1-\rho)\cdot\phi_{ij}^{\pi_k}$
36:    return the best solution found $G^*$
```

In lines 16-20, Ant_a elaborates its solution G_a, considering the current values of $ga_{ij}^{\pi_k}$.

In this work, α was set to 1.0 and β was defined as 4.0 (lines 7-8). For further explanation, see (DORIGO *et al.*, 1996).

In lines 23-24, the pheromone trail is decreased to encourage ants that are behind to choose other trails, reinforcing diversification and better exploration of the solution space.

The best solution of each generation is stored in G^+ (lines 21-22).

The update of G^*, the stopping criterion, and the global pheromone updates are performed on lines 25-31. In these lines, as improvements occur, pheromone ϕ_{ij} evaporates in line 29, enforcing diversification.

The evaporation rate ρ can be seen as a mechanism that delays premature convergence towards a sub-optimal solution. In line 35 pheromone is evaporated from all edges, to foster diversification.

4.6 Noising Methods

The features and variants of NM are detailed by (CHARON; HUDRY, 1993) and (CHARON; HUDRY, 2001), concluding that NM can be considered a generalization of *Simulated Annealing* or *Threshold Acceptance* when its components are properly chosen.

(ZHAN *et al.*, 2020) compared six variants of NM for the 0-1 knapsack problem, observing a negligible difference in performance among them. For this reason, we decided to implement an unique variant of NM, described in **Algorithm 6**.

Although (CHARON; HUDRY, 1993), (CHARON; HUDRY, 2001) and (ZHAN *et al.*, 2020) start their algorithms with a randomly generated initial solution, we generated an initial greedy solution G^* (line 3). This decision improved the overall results.

Line 6 gives the total number of trials, and line 5 gives the number of noised elementary transformations. These numbers are adjusted as a function of the problem size, as the larger the problem, the more trials and iterations are needed for the space search.

The initial noise level r_{init} (line 8) is of amplitude $2 \cdot r_{init}$, which can vary depending on whether new solutions are accepted or not. Initially, it is large to foster solution space exploration, and it is stepped down at each trial, narrowing the search space. The method for calculating r_{init} and adjusting the noise rate r over trials is crucial because it can determine the rate of convergence towards the optimal solution. See Figure 4.2.

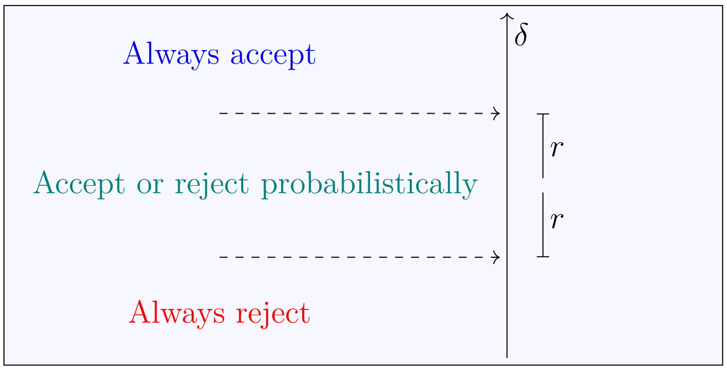


Figure 4.2 – Behavior of $\delta \leftarrow f(G'') - f(G') + u * r$

In Figure 4.2, considering the hill climbing, there is a level up where δ is always true and a level down where δ is always false. The intermediary range may return true or false randomly to help escape from local optima.

There are a number of elementary transformations at the same noise rate that are carried out at random. In each iteration within a trial, a remaining item is randomly selected, and an item chosen for palletization is also selected. They are exchanged if feasible, and the solution is elementary transformed.

The algorithm is terminated after the number of trials or the time limit is reached. These numbers must be adequately set to find good solutions of acceptable quality and time frame.

Just to remember some notations: $f(G)$ is the objective function value for graph G. E is a set of edges. M is a set of pallets. N^{π_k} is the set of items in node π_k.

Algorithm 6 The Noising Methods

```
 1: procedure $NM(\pi_k,\ G_1,\ nodeTime)$
 2:     $\tau^{\pi_k} \leftarrow 0$
 3:     $G^*,\ \tau^{\pi_k},\ \eta_1 \leftarrow Greedy(\pi_k,\ G_1,\ \tau^{\pi_k},\ 1.0)$    ▷ a initial greedy solution based on $\theta_{ij}^{\pi_k}$
 4:     $r_{init} \leftarrow 35.0$    ▷ value found empirically
 5:     $numIters \leftarrow n/5$    ▷ expressions found empirically
 6:     $numTrials \leftarrow numIters/5$    ▷ expression found empirically
 7:     $step \leftarrow r_{init}/(numTrials - 1)$
 8:     $r \leftarrow r_{init}$
 9:     for $trial \in \{1, 2, 3, ..., numTrials\}$ do
10:         $G' \leftarrow G^*$
11:         for $iter \in \{1, 2, 3, ..., numIters\}$ do
12:             if elapsed time > nodeTime then
13:                 $trial \leftarrow numTrials + 1$    ▷ forces termination
14:                 $iter \leftarrow numIters + 1$
15:             $G'' \leftarrow Transform(G')$    ▷ See Algorithm 7
16:             $\delta \leftarrow f(G'') - f(G') + u * r$
17:             if $\delta > 0$ then    ▷ if randomly accepted
18:                 $G' \leftarrow G''$
19:         if $f(G') > f(G^*)$ then
20:             $G^* \leftarrow G'$
21:         $r \leftarrow r - step$
22:     return the best solution found $G^*$
```

Noise is added to the variation of the function $\boldsymbol{f}$ between the current solution G' and a noised solution $G^{''}$. We defined this variation as in line 16, where u is a random number uniformly draw from $[-1,\ 1]$ and r is the noise rate.

In other words, if δ is positive and bigger than $|r * u|$ it returns True; if δ is negative and smaller than $|r * u|$ it returns False; and if $|\delta|$ is smaller than $|r * u|$ it may return

True or False.

As ACLP+RPDP is a maximization problem, a noised solution is accepted if $\delta \geq 0$ (line 17).

Many heuristics applied to combinatorial optimization problems are based on elementary transformations. We call transformation any operation that changes a solution S' into a solution S''.

Algorithm 7 is used by *NM* and the *Tabu Search* to make elementary transformations into a solution.

Algorithm 7 The elementary transformation method

```
 1: procedure $Transform(G')$
 2:     $transformed \leftarrow$ False
 3:     $tested \leftarrow [\,]$
 4:     while not $transformed$ do
 5:         $p \leftarrow$ random choice $(G'_{pallets})$                    ▷ choose a pallet randomly
 6:         if pallet $p \notin tested$ then
 7:             $tested \leftarrow tested \cup p$                        ▷ mark pallets as tested
 8:             $remaining \leftarrow$ candidate items in this node
 9:             $palletized \leftarrow$ items assigned to $p$
10:             while not $transformed$ and $|remaining| > 0$ do
11:                 $item_0 \leftarrow$ random choice $(remaining)$
12:                 $remaining \leftarrow remaining \setminus item_0$
13:                 if $|palletized| > 0$ then
14:                     $item_1 \leftarrow$ random choice $(palletized)$
15:                     $palletized \leftarrow palletized \setminus item_1$
16:                     $f(G') \leftarrow f(G') - item_1^{score}$      ▷ update the score with $item_1$ removal
17:                     if is $item_0$ inclusion is feasible then
18:                         $palletized \leftarrow palletized \cup item_0$
19:                         $f(G') \leftarrow f(G') + item_0^{score}$  ▷ update the score with $item_0$ inclusion
20:                         update $G'$
21:                         $transformed \leftarrow$ True                ▷ end of this procedure
22:                     else
23:                         $palletized \leftarrow palletized \cup item_1$
24:                         $f(G') \leftarrow f(G') + item_1^{score}$
25:         if |tested| = |pallets| then                    ▷ check if all pallets have been tested
26:             return $G'$
27:     return $G'$
```

4.7 Tabu Search

TS is a local search meta-heuristic (GLOVER, 1986) which avoids entrapment in local minima and achieves an effective balance of intensification and diversification. This is usually obtained by keeping track of the last solutions in a *Tabu Queue* (TQ).

(XIA *et al.*, 2018)'s work explores the vehicle routing problem (VRP) aiming to reduce costs, the distance-constrained and capacitated VRP with split deliveries by order was studied. A new *Tabu Search* algorithm is designed to solve the problem, and the examples tested show the efficiency of the proposed heuristic.

Algorithm 8 presents our TS implementation.

To diversify the generation of solutions, the initial G^* is created using a greedy algorithm with a roulette wheel based on $\theta_{ij}^{\pi_k}$ (line 2).

Empirically, we defined the size of the *Tabu Queue* $|TQ| = n^{\pi_k} * m/50$, 2% of the problem size and, in a search for a operational acceptable performance, we defined the number of iterations as a multiple of the TQ size (lines 3-4).

The neighbors of $G^{'}$ are generated in lines 9-14, where our implementation ensures distinct solutions. In line 13, each of these neighbors, if not in TQ, is compared to the best current solution. In case of improvement, the best solution among all neighbors is enqueued in TQ and compared with G^* (lines 15-23).

Algorithm 8 Tabu Search

```
 1: procedure $TS(\pi_k,\ G_1,\ nodeTime)$
 2:     $G^*,\ \tau^{\pi_k},\ \eta_1 \leftarrow Greedy(\pi_k,\ G_1,\ \tau^{\pi_k},\ 1.0)$     ▷ a initial greedy solution based on $\theta_{ij}^{\pi_k}$
 3:     $L_{tq} \leftarrow n^{\pi_k} * m/50$     ▷ 2% the size of the problem
 4:     $numIters \leftarrow 20 \cdot L_{tq}$
 5:     $TQ \leftarrow \{\}$
 6:     $stagnant \leftarrow 0$
 7:     while $stagnant < 3$ and $run\ time < nodeTime$ do
 8:         $G' \leftarrow G^*$
 9:         for $iter \in \{1, 2, 3, ..., numIters\}$ do
10:             if elapsed time $> nodeTime$ then
11:                 $iter \leftarrow numIters + 1$     ▷ forces termination
12:             $G'' \leftarrow Transform(G')$     ▷ See Algorithm 7
13:             if $G^{''} \notin TQ$ and $f(G^{''}) > f(G')$ then
14:                 $G' \leftarrow G^{''}$
15:         if $G' \notin TQ$ then
16:             $TQ \leftarrow TQ \cup G$
17:         if $|TQ| > L_{tq}$ then
18:             $TQ \leftarrow TQ \setminus TQ[0]$
19:         if $f(G') > f(G^*)$ then
20:             $G^* \leftarrow G'$
21:             $stagnant \leftarrow 0$
22:         else
23:             $stagnant \leftarrow stagnant + 1$
24:     return the best solution found $G^*$
```

4.8 Greedy Randomized Adaptive Search Procedure

GRASP (FEO; RESENDE, 1989) is a meta-heuristic that takes advantage of two strategies: the high variability of random solutions and the good quality of greedy solutions. It uses a greedy-probabilistic process, inserting randomness in the generation of initial solutions and ranking neighboring solutions according to a greedy function. **Algorithm 9** presents GRASP implemented in this work.

In other words, GRASP may be viewed as a repetitive sampling technique. A solution is built iteratively by adding one component at a time from a candidate set. In each iteration, the next piece to be added to the solution-in-progress is chosen at random from a list of limited possibilities, the *Restricted Candidates List (RCL).*

(GUTIERREZ *et al.*, 2018) proposed a hybrid GRASP to solve the Vehicle Routing Problem with Stochastic Demands. The effect of adding the GRASP is investigated and it is shown that it yields better the results. They stated that their proposed method outperforms the state-of-the-art metaheuristics and that GRASP efficiently solves big instances in small computational times.

A new greedy solution G^* is created in line 3, which is improved along the search.

Line 4 contains the edges in node π_k in non-ascending order of its heuristic attractiveness; line 6 defines a temporary set of edges for the local search. Edges that remain in the RCL are inserted in this set; and line 5 defines the size of the RCL as the ratio between the size of the problem $(n^{\pi_k} * m)$ and the number of edges $(|E^*|)$ of the greedy solution G^*.

A new solution G from G^+ is created in line 9, that may be improved or not along the search. In lines 11-12, the first L_{rcl} elements from E' are removed and its edges are inserted in RCL; in line 17 one of the RCL edges is randomly selected and inserted in G, if feasible; and in line 26 the temporary edges collected in the internal loop are used to restart the search, until there are insufficient edges.

Unlike the other metaheuristics implemented, GRASP does not use the proportional roulette wheel mechanism. Edges are chosen in the RCL uniformly at random (line 17).

Algorithm 9 The Greedy Randomized Adaptive Search Procedure

```
1: procedure $GRASP(\pi_k,\ G_1,\ nodeTime)$
2:     $\tau^{\pi_k} \leftarrow 0$
3:     $G^*,\ \tau^{\pi_k},\ \eta_1 \leftarrow Greedy(\pi_k,\ G_1,\ \tau^{\pi_k},\ 1.0)$        ▷ a initial greedy solution based on $\theta_{ij}^{\pi_k}$
4:     $E_k \leftarrow$ set with $n^{\pi_k} * m$ edges sorted by $\theta_{ij}^{\pi_k}$ in non-ascending order
5:     $L_{rcl} = n^{\pi_k} * m/|E^*|$
6:     $E^{temp} \leftarrow \{\}$        ▷ a temporary set of edges
7:     $stagnant \leftarrow 0$
8:     while $stagnant < 5$ and $runtime < nodeTime$ do
9:         $G(E) \leftarrow G^+$
10:        $E' \leftarrow E_k$
11:        $RCL \leftarrow E'(1, 2, ..., L_{rcl})$
12:        $E' \leftarrow E' \setminus RCL$
13:        while true do
14:            while true do
15:                if $|RCL| < L_{rcl}$ then
16:                    get out
17:                $e_{ij} \leftarrow random(RCL)$
18:                $RLC \leftarrow RCL \setminus e_{ij}$
19:                $E^{temp} \leftarrow E^{temp} \cup RCL$
20:                if $E \cup \{e_{ij}\}$ is feasible then
21:                    $E \leftarrow E \cup \{e_{ij}\}$
22:                $RCL \leftarrow E'(1, 2, ..., L_{rcl})$
23:                $E' \leftarrow E' \setminus RCL$
24:            if $|E^{temp}| < L_{rcl}$ then
25:                get out
26:            $E' \leftarrow E^{temp}$
27:            $RCL \leftarrow E'(1, 2, ..., L_{rcl})$
28:            $E' \leftarrow E' \setminus RCL$
29:        if then$f(G) > f(G^*)$
30:            $G^* \leftarrow G$
31:            $stagnant \leftarrow 0$
32:        else
33:            $stagnant \leftarrow stagnant + 1$
34:    return the best solution found $G^*$
```

4.9 Genetic Algorithm

Evolutionary algorithms (EA) hold significant importance within the field of metaheuristics. Through the restriction of the search space, (CHU; BEASLEY, 1998) achieved the first successful implementation of genetic algorithms. However, for very big problems, the population size required to adequately cover the search area may become unfeasibly enormous. This can result in a decreased rate of convergence towards an optimal or nearly optimal solution, as it may need a larger number of generations to sufficiently explore the solution space. The computational expense of assessing the complete population across numerous generations can become impractical for exceedingly big issues. As the magnitude of the challenge grows, it becomes increasingly difficult to guarantee both diversity and efficient exploration.

The Genetic Algorithms (GA) are part of the EA that employ approaches inspired by natural evolution, such as inheritance, mutation, selection, and crossover, to find solutions for optimization problems. Individuals with the highest level of strength in a community are more likely to successfully pass on their genetic material to the following generation.

GA involves the evolution of a population of potential solutions to an optimization problem, with the aim of improving the quality of these answers. Every candidate solution possesses a collection of characteristics that can be modified and changed. Typically, solutions are depicted in binary form, consisting of sequences of 0s and 1s, although alternative encodings are also feasible. In this work, we adopt a flat array indexed by j, with values ranging in $\{1, 2, 3, ..., m\}$ to indicate item-pallet associations.

Evolution typically commences with a populace of individuals (item-pallet associations) that are formed randomly and occur across successive generations. During each generation, the level of fitness (the sum of item scores) for each member of the population is assessed. Individuals with higher fitness are chosen randomly from the current population, and their genetic makeup is altered (via recombination and potentially random mutations) to create a new population.

The subsequent iteration of the algorithm utilizes the newly updated population. Typically, the algorithm stops running when it has either generated a maximum number of generations or achieved a desirable fitness level for the population.

Reproduction can be achieved using three distinct methods: (1) Clonal Reproduction: The individual is replicated without any changes and passed on to the succeeding generation. (2) Crossover: Two individuals are chosen, and their genetic material is combined at a specific location. The resulting offspring inherits the initial portion of its genetic makeup from one parent and the final portion from the other parent. (3) Mutation: A single individual is chosen, and one bit (an item-pallet edge) is altered.

Algorithm 10 depicts an approximate description of what we implemented by means of the DEAP framework.

Algorithm 10 Genetic Algorithm

```
1: procedure GA($\pi_k$, $N^{\pi_k}$, $M$, $ng, np$)
2:     $toolbox \leftarrow$ A DEAP Toolbox that contains the evolution operators.
3:     $pop = toolbox.population(np)$                    ▷ a list of individuals
4:     $cxpb \leftarrow [0.7 - 0.8$        ▷ The probability range of mating two individuals
5:     $mutpb \leftarrow [0.02 - 0.2$      ▷ The probability range of mutating an individual
6:     for $g \in \{1, 2, 3, ..., ng\}$ do
7:         $pop \leftarrow select(pop)$
8:         $brood \leftarrow eaSimple(pop, toolbox, cxpb, mutpb)$     ▷ get offsprings
9:         $toolbox_{score} \leftarrow Evaluate(brood)$               ▷ Algorithm 11
10:    $hallOfFame \leftarrow pop$      ▷ An object that will contain the best individuals
```

Algorithm 10 is expected to have at most $\mathcal{O}(ng \cdot np \cdot n^{\pi_k} \cdot m)$ execution time. Where ng is the number of generations and np the size of the population.

Lines 4 and 5 present the tested ranges of crossover and mutation probabilities.

Evaluate solution

This procedure receives a graph G with its features and returns 0 in case the solution is infeasible, or the solution score in case the solution is feasible.

This procedure is used by only the Genetic Algorithm.

. M set of pallets

. Q^{π_k} *Packed* on board at node π_k

. N^{π_k} set of the items at node π_k

. $E_Q^{\pi_k} \cup E_N^{\pi_k}$ set of edges connecting *Packed* or items to pallets.

Algorithm 11 Evaluate a node solution

```
1: procedure $Evaluate(G)$
2:     $solution \leftarrow G(M,\ Q^{\pi_k},\ N^{\pi_k},\ E_Q^{\pi_k} \cup E_N^{\pi_k})$    ▷ the solution graph is interpreted as a vector
3:     $score \leftarrow 0$
4:     $weights \leftarrow \{\ 0 \mid for\ i \in \{1, 2, 3, ..., m\}\ \}$
5:     $volumes \leftarrow \{\ 0 \mid for\ i \in \{1, 2, 3, ..., m\}\ \}$
6:     $itemcount \leftarrow \{\ 0 \mid for\ j \in \{1, 2, 3, ..., n\}\ \}$
7:     $torque \leftarrow 0$
8:     for $i \in \{1, 2, 3, ..., m\}$ do
9:         for $j \in \{1, 2, 3, ..., n\}$ do
10:            if $solution_j = i$ then    ▷ if item $j$ is allocated to pallet $i$
11:                $score \leftarrow score + s_j$
12:                $itemcount_j \leftarrow itemcount_j + 1$
13:                if $itemcount_j > 1$ then    ▷ item allocated to more than 1 pallet
14:                    return 0
15:                $weights_i \leftarrow weights_i + w_j$
16:                if $weights_i > W_i$ then    ▷ weight sum is greater than the pallet capacity
17:                    return 0
18:                $volumes_i \leftarrow volumes_i + v_j$
19:                if $volumes_i > V_i$ then    ▷ volume sum is greater than the pallet capacity
20:                    return 0
21:                $torque \leftarrow torque + w_j \cdot D_i$    ▷ item contribution to the total node torque
22:                if $|torque| > maxTorque$ then    ▷ aircraft torque out of the permitted range
23:                    return 0
24:                if $to_j \neq TO_i$ then    ▷ item and pallet destinations are different
25:                    return 0
26:     return $score$
```

A try for a more efficient GA

As the current GA approach did not yield any results under the constraints assessment, we implemented another, more straight-forward version without using any available packages.

We developed the new GA package as follows:

1. A method to randomly initialize the population of individuals (array indexed by j, with values ranging in $\{1, 2, 3, ..., m\}$ to indicate item-pallet associations).
2. Used the procedure in Algorithm 11 to evaluate each solution. This method returns the fitness.
3. A procedure to select a set of elite parents for the next generation. This procedure receives the population in a non-ascending list of fitness values and the elite size. The best individuals constitute the elite.
4. A procedure to ensure uniform crossover between individuals.

5. A procedure to mutate individuals in place according to a mutation rate. To the expression "in place", we mean mutate the same individual received without making a copy, making this procedure prone to efficiency.
6. A procedure to adaptively set the mutation rate. The mutation rate decreases with each generation. This insight came from the noising rate decreasing from the Noising Methods optimization.
7. A batch evaluation procedure to call Algorithm 11 in a process-based parallelism by chunks of the population in a try to improve efficiency.

Although this new package significantly improved the performance of that using the DEAP package, its solutions were poor, even increasing the number of generations or the population size. The code is available at:

`github.com/celiomesquita/ACLP_RPDP_P/blob/main/genetic.py`

4.10 The Shims heuristic

In this last section, we present a new heuristic designed specifically for ACLP+RPDP, which we named *Shims*. This strategy is based on a practical observation: usually, smaller and lighter items are saved for later adjustments to the remaining clearances.

Shim sets are composed of different thicknesses and are mainly used in plant engineering and maintenance. By combining sheets of different thicknesses, the smallest gaps can be bridged precisely.

`(peel-plate.com/en/products/shims/shim-sets/)`

Figure 4.3 – Shims of different thicknesses

A brief story about the insight this author had as ideas to solve the ACLP part of the ACLP-RPDP problem, which led to *Shims* development:

Upon the author's retirement from the Brazilian Air Force, residency was subsequently established not in Brasília, the nation's capital, but in another city. In the preparatory phase of relocation, an observation was made during the placement of the packaged furniture adjacent to the contracted truck. It was noted that the truck owner engaged in the selection and organization of the cargo with exceptional precision. His methodical commands resulted in the near-optimal utilization of *space within the vehicle, leaving only negligible gaps that were too small to accommodate any additional, smaller boxes.*

The accuracy with which the cargo was organized in the truck were eerily similar to the results obtained by 3-dimensional packing algorithms. This shows an impressive example of how experienced professionals can use sight measurement and spatial analysis in real-life packing tasks. It is also worth mentioning the driver's insistence on loading the heaviest boxes first, demonstrating his concern for the vehicle's stability.

Shims, as described on algorithms 4 and 13, has two phases for each pallet: the greedy phase and the shims composition and selection phase. Throughout this heuristic, pallets p_i are considered in ascending order of $|p_i.D|$ (the closest to CG first), and the n possible edges e_{ij} in non-ascending order of $\theta_{ij}^{\pi_i}$.

Figure 4.4 represents the n^{π_k} possible edges $e_{ij}^{\pi_k}$ of the current pallet p_i sorted by $\theta_{ij}^{\pi_k}$, where η_1^i is the last edge index inserted in solution during the first phase (Algorithm 4).

In other words, *Shims* starts with a greedy solution, then stops close to the local optima for pallet i (η_1^i) and, in a radius defined by the problem size, makes a local search within all possible complements in the surroundings. *Shims* finally selects the best alternative shim to fit the volume remaining capacity.

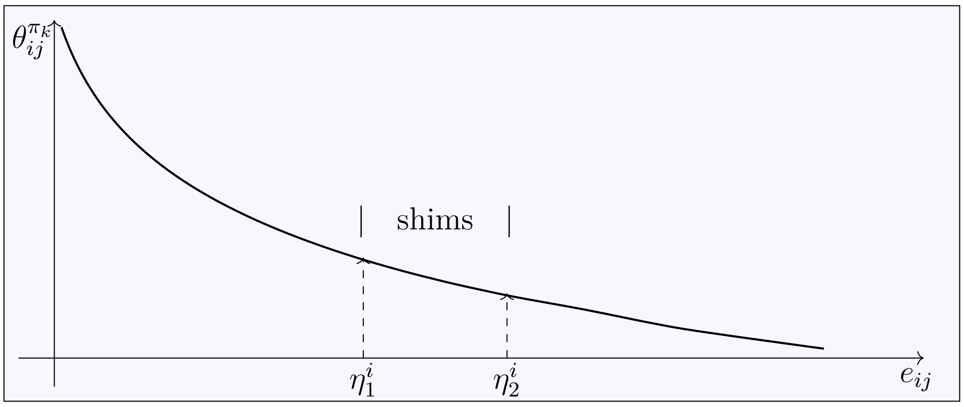


Figure 4.4 – n^{π_k} edges e_{ij} of p_i sorted by $\theta_{ij}^{\pi_k}$ in non-ascending order (where n^{π_k} is the number of items in node π_k)

Put differently, as item-pallet associations (e_{ij}) with indexes before η_1^i are very likely to belong to the best solutions, they are greedily included in the *Shims* solution. As the associations after η_2^i are presumably to not be part of the best solutions, they are set apart for an eventual final local search. As a result, *Shims* operates between these index thresholds.

The *Shims* main process is described by Algorithm 12. In line 7 a partial greedy solution is generated according to the relative volume $level_1$, and determines the values for array η_1 referred in Figure 4.4.

Algorithm 12 *Shims* main process

```
 1: procedure $Shims(N^{\pi_k},\ \pi_k,\ G,\ level_1,\ level_2)$
 2:     Let $G(M^{\pi_k},\ Q^{\pi_k},\ E^{\pi_k})$                         ▷ the initial solution
 3:     $\tau^{\pi_k} \leftarrow 0$                                          ▷ $\pi_k$ aircraft torque
 4:     $\eta_1 \in \{0 \mid i \in \{1, 2, 3, ...,\ m\}\}$
 5:     $\eta_2 \in \{0 \mid i \in \{1, 2, 3, ...,\ m\}\}$
 6:     $vols_{max} \leftarrow \{V_i \cdot level_2 \mid i \in \{1, 2, 3, ...,\ m\}\}$
 7:     $G_1,\ \tau^{\pi_k},\ \eta_1 \leftarrow Greedy(\pi_k,\ G,\ \tau^{\pi_k},\ level_1)$   ▷ a initial greedy solution based on $\theta_{ij}^{\pi_k}$
 8:     for $i \in \{1, 2, 3, ...,\ m\}$ do                                   ▷ $\mathcal{O}(m \log m)$
 9:         $\eta_2^i \leftarrow \eta_1^i$
10:         $vol \leftarrow 0$
11:         while $vol \leq vols_{max}^i$ do                                  ▷ calculate $\eta_2[i]$
12:             $\eta_2^i \leftarrow \eta_2^i + 1$
13:             $vol \leftarrow vol + v_j$
14:         $getBestShims(i,\ \eta_1,\ \eta_2,\ E,\ \pi_k,\ 1 - level_1)$        ▷ pallet gap = $1 - level_1$
15:     $G,\ \tau^{\pi_k},\ \eta_1 \leftarrow Greedy(\pi_k,\ G_1,\ \tau^{\pi_k},\ 1.0)$       ▷ a final greedy local search
16:     return $G(M,\ Q^{\pi_k},\ N^{\pi_k},\ E_Q^{\pi_k} \cup E_N^{\pi_k})$
```

The *Shims* first phase is executed by Algorithm 4, that generates a greedy allocation of the items available in node π_k, according to the non-ascending order of $\theta_{ij}^{\pi_k}$, and considering the *Packed* already shipped ($E_Q^{\pi_k}$).

In line 15, the last phase is a local search accomplished by the greedy method, when it tries to include more items, especially the items beyond η_2.

Algorithm 12 will take at the worst case $\mathcal{O}(n^{\pi_k} \cdot m + m^2 + m \cdot \log m)$ steps, which may be simplified to $\mathcal{O}(n^{\pi_k} \cdot m)$, considering that $n^{\pi_k} \cdot m >> m$.

Algorithm 13 Mount shims of items that fills each pallet gap and return the best shims

```
1: procedure getBestShims($i,\ \eta_1,\ \eta_2,\ E,\ \pi_k,\ slack^i$)
2:     $volume \leftarrow 0$
3:     $b \leftarrow 1$
4:     $shims_b \leftarrow \{\}$
5:     $Set \leftarrow Set \cup \{shims_b\}$
6:     for $x \in \{\eta_1^i, ..., \eta_2^i\}$ do
7:         $createShims \leftarrow$ True
8:         $e_{ij}^{\pi_k} \leftarrow E_x{}^i$
9:         for $shims \in Set$ do
10:            if $e_{ij}^{\pi_k} \notin (E_N^{\pi_k} \cup shims)$ and $e_{ij}^{\pi_k}$ is feasible and $(v_j^{\pi_k} + volume) \leq slack^i$ then
11:                $shims \leftarrow shims \cup \{e_{ij}^{\pi_k}\}$
12:                $volume \leftarrow volume + v_j^{\pi_k}$
13:                $createShims \leftarrow$ False
14:                break
15:        if $createShims$ then
16:            $volume \leftarrow 0$
17:            $b \leftarrow b + 1$
18:            $shims_b \leftarrow \{\}$
19:            $shims_b \leftarrow shims_b \cup \{e_{ij}^{\pi_k}\}$
20:            $Set \leftarrow Set \cup \{shims_b\}$
21:    $sh_w \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab}^{\pi_k} \in shims} w_b^{\pi_k}$ is maximum
22:    $sh_v \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab}^{\pi_k} \in shims} v_b^{\pi_k}$ is maximum
23:    $sh_{best} \leftarrow shims$, where $shims \in \{sh_w, sh_v\}$ and $\sum_{e_{ab}^{\pi_k} \in x} s_b^{\pi_k}$ is maximum
24:    return $sh_{best}$
```

Algorithm 13 describes the procedure to get the top *Shims*, the best subset of edges to fill each pallet gap. This algorithm follows the logic of the *First-Fit Decreasing* method as described by (JOHNSON; GAREY, 1985). Although it does not aim to minimize the number of bins, its mechanism helped build the set of *Shims* to allow the best to be chosen.

Line 4 creates the first empty shims; line 5 produce the first set of shims from where the best shims will be chosen; and line 6 iterate in the edges indexes range to find subsets of shims that fit into each pallet gap.

Initially, new shims creation is permitted (7), but it will only be created if the last edge could not be included in any shims from the set (15). For each edge, iterate in all sets in a try to include it in any of the previous shims (9). As the last edge was included, forbid a new shims creation (13).

Finally, select the best weight shims (21), the best volume shims (22), and the best score shims (23) between these 2. The best score shims edges will be returned.

We also tried the *Best-Fit* algorithm, which presented a slight improvement in the overall results. *Best-Fit* is an online algorithm for bin packing (shims set creation in this work). Its input is a list of items of different volumes, and its output is a subset of the items that fit in a pallet's slack. The *Best-Fit* algorithm was used to create shims and follow the following steps:

- It keeps a list of open shims, which are initially empty.
- When an item arrives, it finds the shim set with the maximum load into which the item can fit, if any. The load of a shim set is the sum of the volumes existing in the shim set before placing a new item.
- If such a shim set is found, the new item is placed inside it. Otherwise, a new shim set is created, and the coming item is placed inside it.

The code is available in:

`github.com/celiomesquita/ACLP_RPDP_P/blob/main/mpShims.py`

5 Parallelism and 3-D packing

(MESQUITA; SANCHES, 2024) reflects the contents of the previous chapters, with all implementations using sequential algorithms. From this point on, we dive into the application of process-based parallelism to enhance the algorithm's performance and, eventually, the quality of the results.

In this chapter, we raise three hypotheses for improvements:

(1) Computer parallelism could improve *Shims*'s performance.
(2) A procedure for 3-D packing running in process-based parallel computing mode would keep the run time adequate for operational use.
(3) To reduce ground handling costs, it could be beneficial to minimize the distances required to move pallets during unloading at the next location.

This chapter seeks to improve (MESQUITA; SANCHES, 2024)'s results in terms of quality and performance by using computer parallelism, minimizing the distance between the next node-destined pallets and the cargo door, and performing a 3-D packing procedure on each pallet.

These improvements in their heuristics are meant to reduce the stress that transport planners are subjected to because they have to deal with a lot of information in planning the aircraft route, assembling the pallets, and picking up and delivering at each node. To the best of our knowledge, this is the first time that an air cargo transport problem that simultaneously involves APP, WBP, PDP, TSP, and 3-D packing has been addressed.

5.1 Parallelism topologies

By the *Amdahl's law* (HILL; MARTY, 2008), there is an optimal number of parallel executions for each problem in each environment. While working at IBM in 1967, Gene Amdahl developed the foundation for what became known as *Amdahl's Law* or *Amdahl's Argument*. Essentially, the law states that while a process can be decomposed into steps that may then be run in parallel, the time taken for the whole process will be significantly limited by the steps that remain serialized.

In this work's case, the aircraft torque must be shared with all processes for them to keep it within its permitted range.The same applies to the list of items to be embarked on. All processes will have to access the same list to prevent an item from being assigned to different pallets.

According to (MANFRIN *et al.*, 2006, p.226), there are five topologies for multiprocessing interchange of information: *fully-connected*, where the master process broadcasts to all remaining child processes; *replace-worst*, where the best-so-far solution process broadcasts only to the current worst solution process; *hypercube*, where processes are connected as a hypercube, and a vertex process broadcasts only to the connected vertices; *ring*, in which one process only sends a message to the next process connected to it; and *parallel independent runs*, in which there are no communication costs and the best solution is chosen among all processes.

In this thesis, we had to select the *fully-connected* topology because the list of items to embark and the aircraft torque must be accessible for reading and writing by all child processes. This decision require an effective race condition management.

> *When threads or processes attempt to simultaneously access a shared resource, and the accesses can result in an error, we often say the program has a race condition, because the threads or processes are in a "race" to carry out an operation. (PACHECO; MALENSEK, 2021, p. 53).*

For the multiprocessing 3-D packing procedure *mp3Dpacking*, the unique shared information is the current aircraft torque.

5.2 Multiprocessing

According to (BRESHEARS, 2009, p.271), a *Process* is the operating system's spawned and controlled entity that encapsulates an executing application.

It is known that working concurrently opens up synchronization issues. But the *Multiprocessing* package (*mp*) offers both local and remote concurrency, effectively side-stepping the Python global interpreter lock by using sub-processes instead of threads. Because of this, the *Multiprocessing* module lets the programmer take full advantage of the fact that a machine has more than one processor, normally capable of 2 threads each.

As in the larger aircraft the number of pallets is bigger than the number of threads available in a common handheld computer, for parallelization, we used the *Python Multiprocessing* package, which implements *process-based parallelism* or *parallel shared memory algorithm.* More details on *process-based parallelism* may be found in (Python Software Foundation., 2022).

5.3 The process-based *SolveTour* algorithm

In this section we present implementations of $SolveNode(...)$, where *method* corresponds to a heuristic, k is the index of the current node π_k and G is the allocation graph of the *Packed* that remain on board at π_k.

Below, we present the implementation for $SolveNode$, the new process-based parallel heuristic that we propose called *Multiprocessing Shims (mpShims)*.

In addition to the set of nodes, pallets, costs and items, $SolveTour$, described in Algorithm 14, receives the parameter *method*, which corresponds to a heuristic for solving the node-by-node subproblems, and the parameter π, which is a permutation that defines the order of visits in this tour.

Observe that this algorithm is a copy of Algorithm 2, whose working is described in Section 4.3 until line 22.

We describe the working differences from line 19 on, that was included in this chapter.

Algorithm 14 Solves the sequence of nodes of tour π

```
 1: procedure $SolveTour(\pi, L, M, C, N, method, timeLim)$
 2:     $\pi_0 \leftarrow 0$
 3:     $\pi_{K+1} \leftarrow 0$
 4:     $score \leftarrow 0$
 5:     $cost \leftarrow 0$
 6:     $G^\pi \leftarrow \{\ \}$
 7:     for $k \in \{1, 2, 3, ...K\}$ do
 8:         for $i \in \{1, 2, 3, ...m\}$ do
 9:             $T_i^{\pi_k} \leftarrow -1$                                  ▷ reset this pallet destination
10:         if $k = 0$ then
11:             $L^{\pi_0} \leftarrow L$
12:             $Q^{\pi_0} \leftarrow \varnothing$
13:         else
14:             $L^{\pi_k} \leftarrow L^{\pi_k} - \{\pi_k\}$
15:             $Q^{\pi_k} \leftarrow UpdatePacked(\pi_k)$
16:         $G_1 \leftarrow InitialSolution(M,\ N^{\pi_k},\ \pi_k,\ L^{\pi_k},\ Q^{\pi_k})$
17:         $G_2 \leftarrow SetPalletsDestinations(G_1)$
18:         $G_3 \leftarrow SolveNode(method, \pi_k, G_2, timeLim)$
19:         $G_4 \leftarrow mp3Dpacking(G_3)$
20:         $G_5 \leftarrow minRampDist(G_4)$
21:         $s, \epsilon \leftarrow ScoreAndDeviation(\pi_k, G_5)$
22:         $score \leftarrow score + s$
23:         $cost \leftarrow cost + c_{\pi_k, \pi_{k+1}} * (1 + c_g * |\epsilon|)$
24:         $G^\pi \leftarrow G^\pi \cup \{G_5\}$
25:     $f(G^\pi) \leftarrow score/cost$
26:     return $G^\pi(L,\ C,\ M,\ N,\ E_L,\ E_M^N),\ f(G^\pi)$
```

In line 19, $mp3Dpacking(G)$ packs the items previously assigned to a pallet, which is feasible in terms of volume, weight, and torque but still not guaranteed to fit into the pallet. In this procedure, some unfit items are excluded from the solution.

In line 20, $minRampDist(G)$ minimizes the distance of the next node destined pallets, keeping the CG in its operational range. This procedure is implemented in integer programming. This method does not affect palletization or the objective function value.

The score and the CG deviation of this graph are calculated (line 21) and accumulated (lines 22-23), allowing the final result of this tour (line 25).

5.4 Multiprocessing Shims - *mpShims*

Finally, we present a new multiprocessing heuristic derived from *Shims*, which we named *mpShims*.

Although the second phase solves a *Knapsack Problem*, a faster algorithms for solving a *Bin Packing Problem* was used (the *Best-Fit* algorithm), and the most well scored bin (*Shims*) was selected.

We considered each pallet as a parallel process, taking advantage of nowadays multi-core computers, being executed in a multiprocessing run-time, accessing concurrently, reading and writing the list of items and the current value of the CG deviation.

Line 2 the initial solution comes with some *Packed* on board, a bipartite graph between pallets and *Packed*; line 4 creates a lock feature for shared data among processes to avoid race condition; line 11 calls the greedy method which updates p_i, N^{π_k}, τ^{π_k}, and G; and line 14 wait for all *Greedy* processes to finish.

Algorithm 15 *mpShims* main process

```
procedure $mpShims(N^{\pi_k}, \pi_k, level_1, G, tmax, level_1, level_2)$
    Let $G^{\pi_k}(M^{\pi_k},\ Q^{\pi_k},\ E_Q^{\pi_k})$    ▷ the initial solution
    $\tau^{\pi_k} \leftarrow 0$    ▷ $\pi_k$ aircraft torque
    $lock \leftarrow$ multiprocessing lock    ▷ avoid race condition
    $proc \leftarrow \{p_i \mid i \in \{1,2,3,...,m\}\}$    ▷ each pallet has its own process
    $\eta_1 \leftarrow \{0 \mid i \in \{1,2,3,...,m\}\}$
    $\eta_2 \leftarrow \{0 \mid i \in \{1,2,3,...,m\}\}$
    $vols_{max} \leftarrow \{V_i \cdot level_2 \mid i \in \{1,2,3,...,m\}\}$
    $slack \leftarrow \{V_i \cdot (1 - level_1) \mid i \in \{1,2,3,...,m\}\}$
    for $i \in \{1,2,3,...,m\}$ do
        $proc_i \leftarrow mpGreedy(i,\ \pi_k,\ G,\ level_1,\ lock,\ \tau^{\pi_k},\ \eta_1^i)$
        $fork_i$    ▷ trigger a process for each pallet
    for $i \in \{1,2,3,...,m\}$ do
        $join_i$    ▷ wait for all processes to finish
    for $i \in \{1,2,3,...,m\}$ do
        $\eta_2^i \leftarrow \eta_1^i$
        $vol \leftarrow 0$
        while $vol \leq vols_{max}^i$ do
            $\eta_2^i \leftarrow \eta_2^i + 1$
            $vol \leftarrow vol + v_j$
        $proc_i \leftarrow getBestShims(i,\ \eta_1,\ \eta_2,\ E,\ \pi_k,\ slack_i,\ lock)$
        $fork_i$
    for $i \in \{1,2,3,...,m\}$ do
        $join_i$
    return $G^{\pi_k}(M,\ Q^{\pi_k},\ N^{\pi_k},\ E_Q^{\pi_k} \cup E_N^{\pi_k})$
```

Line 21 calls the *getBestShims*() method which updates p_i, N^{π_k}, τ^{π_k}, and G; line 24 wait for all *getBestShims*() processes to finish; and line 25 returns a complete solution, a bipartite graph between pallets and *Packed* or items.

Algorithm 16 generates a greedy allocation of the items available in node π_k, according to the non-ascending order of $\theta_{ij}^{\pi_k}$, and considering the *Packed* already shipped ($E_Q^{\pi_k}$). Edges are included in this allocation as long as they respect feasibility constraints. Furthermore, the volume of each pallet cannot exceed $V_i \cdot limit$, where $0 < limit \leq 1$ is a given parameter.

Algorithm 16 Mount a greedy pallet partial solution until the volume and torque limits

```
1: procedure $mpGreedy(i,\ \eta_1,\ \pi_k,\ G,\ level_1,\ lock,\ \tau^{\pi_k})$
2:     Let $G^{\pi_k}(M,\ Q^{\pi_k}, E_Q^{\pi_k})$
3:     $volume \leftarrow 0$
4:     $\tau_{max} \leftarrow W_{max} \cdot limit_{long}^{CG}$          ▷ maximum aircraft torque
5:     while $lock$ is unavailable do          ▷ waits for the lock to be available
6:         nothing
7:     seize $lock$
8:     if $(p_i, a_i^{\pi_k}) \in E_Q^{\pi_k}$ then
9:         $volume \leftarrow volume + v_i^{\pi_k}$
10:        $\Delta\tau \leftarrow w_i^{\pi_k} \cdot D_i$
11:        $\tau^{\pi_k} \leftarrow \tau^{\pi_k} + \Delta\tau$
12:    $E_N^{\pi_k} \leftarrow \varnothing$
13:    for each $e_{ij}^{\pi_k}$ in non-ascending order of $\theta_{ij}^{\pi_k}$ do
14:        $\Delta\tau \leftarrow w_j \cdot D_i$
15:        $\tau_{new} \leftarrow |\tau^{\pi_k} + \Delta\tau|$
16:        if $(E_N^{\pi_k} \cup \{e_{ij}^{\pi_k}\}$ is feasible$)$ and $(volume \leq V_i \cdot level_1)$ and $(\tau_{new} \leq \tau_{max})$ then
17:            $E_N^{\pi_k} \leftarrow E_N^{\pi_k} \cup \{e_{ij}^{\pi_k}\}$
18:            $volume \leftarrow volume + v_j^{\pi_k}$
19:            $\eta_1^i \leftarrow \eta_1^i + 1$
20:            $\tau^{\pi_k} \leftarrow \tau^{\pi_k} + \Delta\tau$
21:    release $lock$
22:    return $G^{\pi_k}(M,\ Q^{\pi_k},\ N^{\pi_k}, E_Q^{\pi_k} \cup E_N^{\pi_k})$
```

The set η_1 is returned together with the greedy solution. This algorithm returns a graph with vertices M, Q^{π_k}, and N^{π_k}; and edges $E_Q^{\pi_k} \cup E_N^{\pi_k}$.

mpShims' second phase (Algorithm 17) is almost the same as *Shims*' second phase, having the only difference in it's application, as *mpShims* is run in multiprocessing runtime. Thus, I repeat it here to address the *lock* argument.

Algorithm 17 Mount shims of edges that fills each pallet gap and return the best shims

```
1: procedure $getBestShims(i,\ \eta_1,\ \eta_2,\ E,\ \pi_k,\ slack,\ lock)$
2:     $volume \leftarrow 0$
3:     $b \leftarrow 1$
4:     $shims_b \leftarrow \{\}$
5:     $Set \leftarrow Set \cup \{shims_b\}$
6:     while $lock$ is unavailable do                          ▷ waits for the lock to be available
7:         nothing
8:     seize $lock$                                            ▷ Avoid race condition
9:     for $x \in \{\eta_1^i, ..., \eta_2^i\}$ do
10:        $createShims \leftarrow$ True
11:        $e_{ij}^{\pi_k} \leftarrow E_x^{\pi_k}$
12:        for $shims \in Set$ do
13:            if $e_{ij}^{\pi_k} \notin (E_N^{\pi_k} \cup shims)$ and $e_{ij}^{\pi_k}$ is feasible and $(v_j^{\pi_k} + volume) \leq slack_i$ then
14:                $shims \leftarrow shims \cup \{e_{ij}^{\pi_k}\}$
15:                $volume \leftarrow volume + v_j^{\pi_k}$
16:                $createShims \leftarrow$ False                ▷ do not create a new Shims
17:                break
18:        if $createShims$ then
19:            $volume \leftarrow 0$
20:            $b \leftarrow b + 1$
21:            $shims_b \leftarrow \{\}$
22:            $shims_b \leftarrow shims_b \cup \{e_{ij}^{\pi_k}\}$
23:            $Set \leftarrow Set \cup \{shims_b\}$
24:    release $lock$
25:    $sh_w \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab}^{\pi_k} \in shims} w_b^{\pi_k}$ is maximum
26:    $sh_v \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab}^{\pi_k} \in shims} v_b^{\pi_k}$ is maximum
27:    $sh_{best} \leftarrow shims$, where $shims \in \{sh_w, sh_v\}$ and $\sum_{e_{ab}^{\pi_k} \in x} s_b^{\pi_k}$ is maximum
28:    return $sh_{best}$
```

5.5 Multiprocessing 3-D packing

This section's content is not present in (MESQUITA; SANCHES, 2024), and is designed to be used with a parallel procedure to pack the previously allocated items onto each pallet.

We have not planned to solve this part of the problem with an exact approach due to its complexity. These problems are, in the strict sense, combinatorial optimization problems, thus NP-hard, that are particularly difficult to solve. Consequently, only a very few exact algorithms exist.

The following assumptions were established for the packing problem:

Assumption 1	→ Items packed on the pallet cannot overlap, occupying the same space.
Assumption 2	→ The placement of items must be orthogonal and parallel to the pallet side.
Assumption 3	→ Each item can freely be rotated in the pallet.
Assumption 4	→ Items are allowed to be placed on top of each other without restrictions.

More details may be found in (OCLOO *et al.*, 2020).

According to (BRANDT; NICKEL, 2019):

Technically the task to solve is a three-dimensional Container Loading Problem (CLP) with a multitude of constraints. The CLP alone is known to be NP-hard and extremely hard to solve even for instances of moderate size. The typical number of items per transport segment ranges between 300 and 800 items.

This thesis also adopts a similar range as items quantities, from 300 to 1000 items per node.

To confirm or not Hypothesis (2), we decided to adopt the model for three-dimensional (3-D) packing based in the instructions on (DUBE; KANAVATHY, 2006), to pack the items selected for each pallet.

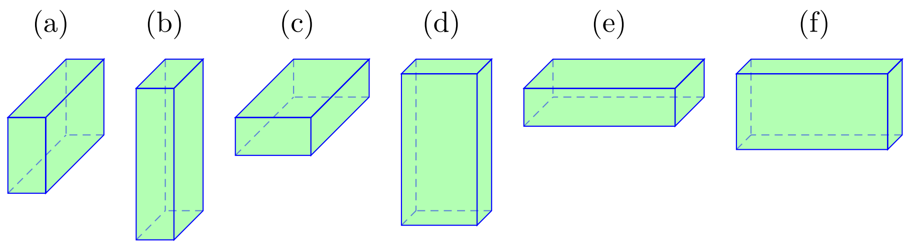
(DUBE; KANAVATHY, 2006) state that the system utilized the *First Fit Decreasing* and the *Best Fit* as the two main heuristic bin packing techniques. They were selected above other heuristic algorithms due to their higher computational speed and ability to generate solutions that closely approach the best answer compared to most other heuristic algorithms.

The main logic of the 3-D packing procedure (3*Dpacker*) is based on an approximate algorithm like follows:

1. From a list of items assigned to a pallet, items are sorted from the biggest to the smallest and placed in such an order on a pallet. Each item has 1 to 6 orientations (Figure 5.1) to choose from in the moment of placement. The orientation procedure can select the best direction type among possible orientations.
2. A pivot point is used to determine item's position. The pivot is an $(X_j,\ Y_j,\ Z_j)$ coordinate which represents a point in the pallet at which an attempt to pack an item will be made.
3. The back lower left corner of the item will be placed at the pivot. If the item cannot be packed at the pivot position then it is rotated until it can be packed at the pivot point or until we have tried all 6 possible rotation types (first fit).
4. If, after rotating it, the item still cannot be packed at the pivot point, then we move on to packing another item and add the unpacked item to a list of items that will be packed after an attempt to pack the remaining items is made.

5. The first pivot in the empty pallet is always (0, 0, 0). When one item can be placed into multiple optimal pivot point, the placement selection module can help make a choice (best fit).

Figure 5.1 – Items possible orientations



The code is available at: `github.com/enzoruiz/3dbinpacking`

This code, which was based on the work of (DUBE; KANAVATHY, 2006), was not planned to run within a multiprocessing framework. We innovate by integrating their solution into the Algorithm 18.

Algorithm 18 presents the multiprocessing solution method for 3-D packing to be applied to an existent solution, to guarantee that the item to be palletized actually fit into the pallet.

Algorithm 18 Multiprocessing 3-D packing procedure

```
1: procedure $mp3Dpacking(\pi_k,\ G)$
2:     Let $G(M,\ Q^{\pi_k},\ N^{\pi_k}, E_Q^{\pi_k} \cup E_N^{\pi_k})$
3:     $proc \leftarrow \{p_i \mid i \in \{1,2,3,...,m\}\}$
4:     $unfit \leftarrow \{\varnothing |\ i \in \{1,2,3,...,m\}\}$        ▷ for each pallet a set of unfit items is initialized
5:     $lock \leftarrow$ multiprocessing lock
6:     for $i \in \{1,2,3,...,m\}$ do
7:         $proc_i \leftarrow 3Dpacker(p_i,\ N_i^{\pi_k},\ \pi_k,\ G,\ lock,\ unfit_i)$
8:         $fork_i$                                                   ▷ pallet $p_i$ packing process is started
9:     for $i \in \{1,2,3,...,m\}$ do
10:        $join_i$                                                   ▷ waits for the packing process to finish
11:        $E_{N_i}^{\pi_k} \leftarrow E_{N_i}^{\pi_k} \setminus unfit_i$
12:    return $G(M,\ Q^{\pi_k},\ N^{\pi_k}, E_Q^{\pi_k} \cup E_N^{\pi_k})$
```

According to line 2, this method receives an initial solution produced by a previous resolution, and according to line 3, each pallet has its own process.

In line 7, $N_i^{\pi_k}$ is the set of items previously assigned to pallet i and the method $3Dpacker$ conforms to this model.

Line 11 excludes the unfit items from the solution.

The running time for the method that composes the $3Dpacker$, *Best Fit*, is $(n \cdot log\ n)$, and for *First Fit Decreasing*, it is $(n \cdot log\ n)$, excluding the running time for sorting.

5.6 Minimization of the unloading costs

(LURKIN; SCHYNS, 2015) considered an aircraft with two doors, and the minimization of loading and unloading costs at the intermediate node was modeled through a container sequencing problem. But, as both aircraft dealt with in this thesis have only the ramp door, we tried to minimize the distance of travel of pallets inside the cargo bay, favoring unloading in the next node, by a post optimization of pallets positions to favor this distance minimization.

To confirm or not Hypothesis (3), that the distances to move pallets on unloading in the next node could be minimized to benefit unloading costs. We modeled this improvement as an integer programming model.

Let D_{max} be the distance from the last ramp pallet to the center of gravity.

Let $Q^{\pi_k} = \{a_1^{\pi_k}, a_2^{\pi_k}, \ldots, a_m^{\pi_k}\}$ be the *Packed* assembled on the set of m pallets in node π_k.

Let $Z_{iq}^{\pi_k}$ be the set of binary decision variables that relates pallet i to *Packed* q in node π_k.

$$\text{minimize} \sum_{i=1}^{m} \sum_{q=1}^{m} Z_{iq}^{\pi_k} \cdot (D_{max} - D_i),\ \textbf{if}\ to_q^{\pi_k} = \pi_{k+1} \tag{5.1}$$

s.t.:

$$\sum_{i=1}^{m} Z_{iq}^{\pi_k} = 1,\ q \in \{1, 2, \ldots, m\} \tag{5.2}$$

$$\sum_{q=1}^{m} Z_{iq}^{\pi_k} = 1,\ i \in \{1, 2, \ldots, m\} \tag{5.3}$$

$$\left| \sum_{i=1}^{m} Z_{iq}^{\pi_k} \cdot D_i \cdot (140 + w_q^{\pi_k}) \right| \leq W_{max} \cdot limit_{long}^{CG} \tag{5.4}$$

Equation 5.1 minimizes the sum of distances of the *Packed* in a node π_k destined to the next node π_{k+1};

Equation 5.2 states that each *Packed* will be assigned to exactly one pallet;

Equation 5.3 states that each pallet will receive one *Packed*;

Equation 5.4 guarantees that the torque limit will not be exceeded. 140kg is the empty pallet weight.

The instruction referred in Algorithm 14, line 20 implements this model.

6 Preparing for solution

This Chapter presents the preparation procedures required for any solution method to be applied to ACLP+RPDP.

All testing is planned to be accomplished on the testing scenarios from Table 6.1, with 7 instances each. The number of items in each node varies randomly between 250 and 1000, depending on the volume surplus chosen.

Table 6.1 – Testing scenarios

Scenario	***K***	**Acft**	***L***
1	2	smaller	$\{0, 1, 2\}$
2	2	larger	$\{0, 1, 2\}$
3	3	larger	$\{0, 1, 2, 3\}$
4	4	larger	$\{0, 1, 2, 3, 4\}$
5	5	larger	$\{0, 1, 2, 3, 4, 5\}$
6	6	larger	$\{0, 1, 2, 3, 4, 5, 6\}$

Column **Acft** tells the size of the aircraft used.
Column ***L*** present sets of nodes.

The best outcome among all $K!$ tours or any set of previously generated tours will be the answer for a combination *scenario*, *surplus* and *method* submitted to a solution method.

The *surplus* is a constant in $\{1.2,\ 1.5,\ 2.0\}$ that is multiplied by the aircraft volumetric capacity to make all solutions difficult to obtain, i.e., in each node many items (16.6%, 33.3%, and 50.0%, respectively) will not be loaded due to the pallet volume constraints.

6.1 The items generation

As we are dealing with a new problem, which until now had not been modeled in the literature, we had to create the benchmarks. For this, we based on the characteristics of real airlifts carried out by the *Brazilian Air Force*, as described below.

In the military airlift carried out in Brazil from 2008 to 2010, 23% of the items weighed between $10kg$ and $20kg$, 22% from $21kg$ to $40kg$, 24% from $41kg$ to $80kg$, 23% from $81kg$ to $200kg$, and 8% between $201kg$ and $340kg$. These five groups of items are described in Table 6.2. On the other hand, as items volumes were known, the average density of these items is approximately $246kg/m^3$.

Table 6.2 – Items weight distributions

$\boldsymbol{x}$	$\boldsymbol{P}$	$\boldsymbol{low}$ (kg)	$\boldsymbol{high}$ (kg)
1	0.23	10	20
2	0.22	21	40
3	0.24	41	80
4	0.23	81	200
5	0.08	201	340

Procedure *roulette*(x) accesses this table's contents to choose a row at random that is biased toward column P.

The procedure *createItems*, which generates N, is described in Algorithm 19. The parameter *scenario* defines L and M (line 2), and the parameter *surplus* sets a percentage limit on the total volume of items at each node (line 3).

In line 3, surplus will receive values 1.2, 1.5, and 2.0, representing the extra volume that exceeds the aircraft capacity in each node. This will generate three types of cargo data to permit more testing variability and to better assess the robustness of the methods to be developed.

For each generated $item_j^k$, its destination is uniformly random selected (line 12), its weight has a distribution according to Table 6.2 (lines 15-16), its score varies 100 (highest) and 5 (lowest) according to a logarithmic scale (line 14, and its volume is randomly defined from the density, where we allow a variation of 40% more or less than the average density of $246kg/m^3$ (line 17).

In commercial aviation the score calculation from line 14 may be calculated differently, perhaps more related to the marginal profit of each item transportation.

From line 18 to 20, as the items dimensions were not available in the Brazilian Air Force transport data, the items' dimensions were determined randomly from the item volume, assuming that all items are parallelepipeds. This is essential to permit the application of a 3-dimensional (3-D) packing procedure.

Algorithm 19 Procedure to create the items to be transported

```
1: procedure $createItems(scenario,\ surplus,\ instance)$
2:     Let $L$ and $M$ be according to $scenario$
3:     $limit \leftarrow surplus \cdot \sum_{i=1}^{m} V_i$
4:     $N \leftarrow \{\}$
5:     for $k \leftarrow 0$ to $K$ do
6:         $j \leftarrow 0$
7:         $N_k \leftarrow \{\}$
8:         $vol \leftarrow 0$
9:         while $vol < limit$ do
10:            $j \leftarrow j + 1$
11:            repeat
12:                $to_j^k \leftarrow RandomInt(0, K)$
13:            until $to_j^k \neq k$
14:            $s_j^k \leftarrow \lfloor 100 \cdot (1 - \log_{10}(RandomInt(1, 9))) \rceil$
15:            $x = roulette()$ biased through $P$                          ▷ see Table 6.2
16:            $w_j^k \leftarrow RandomReal(low(x), high(x))$               ▷ see Table 6.2
17:            $v_j^k \leftarrow w_j^k / RandomReal(148, 344)$              ▷ items density range
18:            $wh_j^k \leftarrow RandomReal(0.5, 1.0) \cdot \sqrt[3]{v_j^k}$   ▷ item width
19:            $dh_j^k \leftarrow RandomReal(1.0, 3.0) \cdot \sqrt[3]{v_j^k}$   ▷ item depth
20:            $h_j^k \leftarrow v_j^k / (wh_j^k * dh_j^k)$                  ▷ item height
21:            $vol \leftarrow vol + v_j^k$
22:            $item_j^k \leftarrow (wh_j^k, dh_j^k, h_j^k, w_j^k, v_j^k, to_j^k, s_j^k)$
23:            $N_k \leftarrow N_k \cup \{item_j^k\}$
24:        $N \leftarrow N \cup N_k$
25:    return $N$
```

The procedure *createItems* is executed once for each *scenario*, *surplus*, and *instance.* Its generated files are to be retrieved every time a solution method is executed.

The files produced with this procedure are available in: `https://github.com/celiomesquita/ACLP_RPDP_P`, in folders *surplus20*, *surplus50*, *surplus100*, for *surplus* values in {1.2, 1.5, 2.0}, respectively.

6.2 The memory structures and data loading

Before the application of any solution method in each node, some global scope parameters must be set. Hence, a preparation described in Algorithm 20 is applied.

In line 6, as the number of nodes is small (at most 7), instead of solving the TSP with some more efficient method, we decided to test all permutations of nodes, as it is of low computational cost.

Algorithm 20 *Memory structures data*

```
1: $method \in \{MIP,\ ACO,\ NM,\ GRASP,\ TS,\ GA,\ Shims,\ mpShims\}$
2: $scenario \leftarrow$ select in Table 6.1
3: $surplus \in \{1.2,\ 1.5,\ 2.0\}$    ▷ choose a volume surplus
4: $N \leftarrow getItems(scenario,\ surplus,\ instance)$    ▷ retrieve items from the text files
5: $L \leftarrow$ set of $K$ nodes    ▷ according to $scenario$
6: $\Pi_K \leftarrow$ set of $K!$ permutations of $K$ nodes
7: $M \leftarrow$ set of $m$ pallets    ▷ according to $scenario$
8: $W_{sum} \leftarrow \sum_{i=1}^{m} W_i$
9: $T_{max} \leftarrow payload \cdot limit_{long}^{CG}$    ▷ according to $scenario$
10: $W_{max} \leftarrow \min(W_{sum},\ payload)$    ▷ due to different aircraft sizes
11: $V_{max} \leftarrow \sum_{i=1}^{m} V_i$
```

In line 1, MIP is an mixed-integer capable programming solver; ACO is the Ant Colony Optimization; NM the Noising Methods; GRASP is the Greedy Randomized Adaptive Search Procedure; TS is the Tabu Search; GA is the Genetic Algorithm; *Shims* is a new heuristic proposed in this thesis; and *mpShims* is a multiprocessing parallel version of *Shims*.

6.3 An integer programming attempt

The development of efficient algorithms has significantly advanced the capability to solve large-scale linear programs that incorporate numerous variables and constraints. Nevertheless, the introduction of integer variables complicates these problems, necessitating distinct methodologies for optimal resolution. In these situations, which are called Mixed-Integer Programming (MIP) problems, you need to use special methods like branch-and-bound, branch-and-cut, or heuristic algorithms to find the best or almost best solutions in the larger solution space.

The following are currently the possible framings used in solving Integer programming models:

1. **An exact algorithm:** with this method, we obtain a guaranteed optimal solution, but it may take an exponential number of iterations to obtain it. Examples of the exact algorithms are cutting-plane, branch-and-bound, branch-and-cut, and dynamic programming.
2. **Heuristic algorithms:** these methods find a suboptimal solution. Empirical evidence suggests that these algorithms tend to find a good solution quickly.
3. **Approximation algorithms:** these methods involve a practical running time suboptimal solution obtained together with a bound on the degree of suboptimality.

In this section, we analyze the performance of a commercial MIP tool as a solution alternative to the problem modeled. We used the well known *Gurobi* (`www.gurobi.com`, with an individual academic license).

In Table 6.3, we solve by a node-by-node approach, because it solves this problem a node at a time, starting at the base and ending at the last node before returning to the base.

It can be observed in Table 6.3 that the MIP solver is not capable of solving all scenarios of ACLP+RPDP in an operationally acceptable time.

Scenarios 5 and 6 were not submitted to the MIP solver due to the anticipated long time (over 4 h and 24 h, respectively) to solve all tours.

Table 6.3 – MIP solver results

		Run time
Scenarios	**f**(G^*)	(60s per node)
1	12.03	3min 1s
2	7.65	6min 8s
3	11.14	18min 39s
4	12.37	1h 30min
5		$>$ 4h
6		$>$ 24h

As the ACLP+RPDP subproblems are HP-hard, we conclude that ACLP+RPDP is also NP-hard, which was confirmed by the results found in this chapter. This led us to assess the performance of five metaheuristics and a special method to solve ACLP+RPDP, which are presented in Chapter 4.

6.4 The pallets destinations subproblem

Algorithm 21 pre-determines the destinations for pallets by considering the total volume of items in the nodes yet to be visited. This aligns with the requirements of the mathematical model in Chapter 2.

Algorithm 21 Algorithm to set pallets destinations

```
 1: procedure $SetPalletsDestinations(\pi_k)$
 2:     $vol \leftarrow \{ 0 \mid 0$ to $K\}$                              ▷ array to store the accumulated volumes
 3:     $dest_{max} \leftarrow 0$                                      ▷ destination with maximum volume
 4:     $vol_{max} \leftarrow 0$
 5:     $total \leftarrow 0$
 6:     for $j \leftarrow 1$ to $n^{\pi_k}$ do
 7:         $d \leftarrow to_j^{\pi_k}$                                      ▷ item destination
 8:         if $d \in L^{\pi_k}$ then                                 ▷ destination not visited yet
 9:             $vol_d \leftarrow vol_d + v_j^{\pi_k}$                  ▷ accumulate the volume in destination $d$
10:             $total \leftarrow total + v_j^{\pi_k}$
11:             if $vol_d > vol_{max}$ then
12:                 $dest_{max} \leftarrow d$                  ▷ update the destination with maximum volume
13:                 $vol_{max} \leftarrow vol_d$                          ▷ update the maximum volume
14:     $numEmpty \leftarrow$ count$(T_i^{\pi_k} = -1,\ for\ i \in \{1, 2, 3, ..., m\})$     ▷ the number of empty pallets
15:     for $x \leftarrow 0$ to $K$ do
16:         if $vol_x \neq 0$ then
17:             $needed \leftarrow \max\{1, \lfloor numEmpty \cdot vol_x / total \rfloor\}$     ▷ the number of pallets needed
18:             $used \leftarrow 0$                                     ▷ the number of pallets used
19:             for $i \leftarrow 1$ to $m$ do
20:                 if $used < needed$ and $T_i^{\pi_k} = -1$ then
21:                     $T_i^{\pi_k} \leftarrow x$                              ▷ set a pallet destination
22:                     $used \leftarrow used + 1$                 ▷ update the number of pallets used
23:     for $i \leftarrow 1$ to $m$ do
24:         if $T_i^{\pi_k} = -1$ then
25:             $T_i^{\pi_k} \leftarrow dest_{max}$
26:     return $M$                                          ▷ return the modified set of pallets
```

This preliminary procedure is run to set pallets destinations according to each item disembarkation node. The number of pallets reserved for each destination is proportional to the volume demand for each delivery point. It could be related to weight but, as volume is more constrictive in airlift, we decided to use volume.

As pallets positions and destinations are defined, they are assembled considering the position they will occupy along the cargo bay. This fosters more freedom for the algorithm to fine tune weight balance, as each item allocation contributes to torque. This is an innovation, as others works (BROSH, 1981), (LIMBOURG *et al.*, 2012), (VERSTICHEL *et al.*, 2011), and (VANCROOENENBURG *et al.*, 2014)) play with pallets positions to minimize CG shift.

In line 24, if there is still some pallet without destination, set its destination to the node of maximum accumulated volume.

Some important considerations for this procedure are:

(1) it makes the node-by-node solution possible in practical running time;

(2) it is run prior to all methods; and

(3) it may be a limitation if $m < K$, as it could be impossible to set the pallets destinations to all nodes.

Regarding consideration (3), it is not a limitation for this thesis because the number of nodes is always less than the number of pallets in the scenarios considered.

6.5 The boarded cargo CG displacement subproblem

When not at the base, some *Packed* contents may have to remain on board until their delivery point. Before completing the cargo with current node items by building a new *Packed*, it is reasonable that the current torque may temporarily exceed the operational range. So, something must be done to make the next mission leg feasible. This may be done by simply building new *Packed* in a way to make the next cargo plan feasible.

Another option is to rearrange pallets to minimize torque before starting to compose the new *Packed*. We have decided on this second option because it was noted that in some instances, and in some nodes, poor solutions were generated before this approach was adopted.

This subproblem must be solved in each node before the solution method is executed, as after pallet disembarkation, the cargo tends to be extremely unbalanced.

The present author uses a MIP tool to solve this, which is implemented according to the mathematical formulation ahead and takes a very short time as there are only two variables: the number of pallets and the quantity of *Packed* items.

Let $M = \{p_1, p_2, \dots, p_m\}$ the set of m pallets. Each pallet p_i, $1 \leq i \leq m$, has its centroid distance to the CG of the aircraft D_i.

Let $Q^{\pi_k} = \{a_1^{\pi_k}, a_2^{\pi_k}, \dots, a_c^{\pi_k}\}$ the set of c^{π_k} *Packed* that remain on board of the aircraft in node π_k. Each *Packed* $a_q^{\pi_k}$, $1 \leq q \leq c^{\pi_k}$ in node π_k, has its weight $w_q^{\pi_k}$, and is stacked on some pallet.

Let Y_{iq} be binary variable, where $1 \leq q \leq c^{\pi_k}$ and $1 \leq i \leq m$. $Y_{iq} = 1$ if *Packed* q is assigned to pallet i, $Y_{iq} = 0$, otherwise.

$$minimize\ torque \left| \sum_{i=1}^{m} \sum_{q=1}^{c^{\pi_k}} Y_{iq} \cdot D_i \cdot w_q^{\pi_k} \right| \tag{6.1}$$

s.t.:

$$\sum_{i=1}^{m} Y_{iq} = 1,\ q \in (1, 2, 3, ..., c^{\pi_k}) \tag{6.2}$$

$$\sum_{q=1}^{c^{\pi_k}} Y_{iq} \leq 1,\ i \in (1, 2, 3, ..., m) \tag{6.3}$$

The expression 6.1 represents the boarded *Packed* torque minimization. Its absolute value must be minimized because torque may assume negative (section ahead of the CG) or positive values (behind of the CG).

The Equation 6.2 guarantees that all *Packed* remain on board, and the equation 6.3 guarantees that at most 1 *Packed* is assigned to a pallet.

Given the bounded size of ACLPP (m and c^{π_k} are at most 18), the complexity of solving this integer programming model is expected to be low for state-of-the-art solvers. The problem is well within the capabilities of modern computational tools to solve efficiently, making it feasible to expect quick and optimal solutions in practice, despite the theoretical NP-hardness of IP problems in general.

It is important to notice that the constraints defined in the mathematical model handle torque constraints by moving pallets or reassigning items to pallets to keep the aircraft CG in its operational range.

7 Results and discussions

In this chapter, we present the results of all solution procedures, as well as the preliminary configurations needed for each method. All methods solve each tour problem using a node-by-node approach and save the results for later best-tour selection.

The experiments are performed on a 64-bit, 16GiB, 3.4GHz, 8-core ®Intel i7-3770 CPU, 2 threads per core, with *Linux Ubuntu 22.04 LTS* as the operational system and *Python 3.10.6* as the programming language.

7.1 Parameters tuning

A relevant problem with meta-heuristics is the parameter's adjustment to extract the best possible efficiency in the optimization. The bulk of the tools to fine-tune these parameters utilize a statistical backdrop that enables them to make accurate predictions regarding the differences in candidate configurations' performance. Inadequacies in the design of the statistical experiment may therefore lead to erroneous results from the statistical tests and, subsequently, cause the approach to produce inaccurate comparisons of candidate configurations.

To determine the parameters $level_1$ and $level_2$ used by *Shims*, We previously carried out some experiments with the *irace* tool (LÓPEZ-IBÁÑEZ *et al.*, 2016), the results of which are presented in Table 7.1. In these tests, of every 7 instances generated for each value of *surplus*, 4 were used as the training set and 3 as the testing set. We provided the ranges [0.8, 1.0] and [1.0, 2.0] for $level_1$ and $level_2$, respectively. In each experiment, there was a maximum of 3,000 runs so that *irace* would have enough data for its statistical tests. For more details, see `cran.r-project.org/web/packages/irace/`.

Table 7.1 – *irace* results for *Shims* parameters

surplus	$level_1$	$level_2$	run time (min)
1.2	0.8621	1.0539	47
1.5	0.9199	1.1399	59
2.0	0.9617	1.5706	63

We did not use this tool to fine-tune *ACO*, *GRASP*, *NM*, and *TS* because it was expected that, although the quality of the results would have improved, their run time would remain much larger than that of *Shims*. We adjusted their parameters through trial and error, using the defaults found in the literature for the multidimensional multiple knapsack problem. We did it for *Shims* because, as it was the fastest method, we wanted to compare its quality results with those of the MIP solver.

7.2 Metaheuristics results

In the following tables we present columns or rows with the expressions: **Norm.** and **Speedup**. The average tour values were presented for f and, for the run time, the worst result obtained. To facilitate the comparison between the methods, we added these expressions:

- **Norm.**: value between 0 and 1, which corresponds to the ratio between the sum of $\boldsymbol{f}$ values obtained by the method in all scenarios and the sum of the best values obtained among all methods in all scenarios. The higher the value of **Norm.**, the closer the method approached the best solutions found.
- **Speedup**: ratio of the sums of the worst run times of all scenarios and the sum of the method run times in all scenarios. The method with the highest **Speedup** is the fastest.

For the term *sequential*, we mean not using multiprocessing parallelism.

All metaheuristics got to feasible solutions with all parameters within their constrained values, as stated in the mathematical model.

The metaheuristics implemented follow a node-by-node approach, considering only the items in the node, and the *Packed* already on board, whose destinations are in L^{π_k}, the subset of possible destinations departing from node π_k.

It is important to note that this node-by-node strategy is a way of quickly arriving at a feasible solution for two reasons:

(1) because the heuristics are fast, and
(2) because the number of destinations is at most 6.

In an attempt to keep metaheuristics run times within an acceptable operational limit, each node had its solving duration limited to 0.7 seconds. Most of the metaheuristics yielded good solutions within this run time limit. Also, by adjusting their parameters, efforts were made to keep all metaheuristics performing similarly, with at least scenario 5 having a reasonable run time.

There is no column for GA in Table 7.2 because its solutions were very poor, even increasing the number of generations or the population size.

- in our first attempt, considering only weight and volume constraints, our implementation was able to solve problems of up to 100 items being allocated on 18 pallets in less than 10 seconds.
- when we included the item count constraint to prevent the same item from being allocated to more than one pallet, GA was able to solve only small problems (20 items on 2 pallets).
- we increased the number of items to 150 and the number of pallets to 18, but even adjusting the GA parameters for more generations (2,400) and more individuals (1,200), GA did not generate any viable solutions. The run time for this attempt was about 10 minutes.
- we implemented a new version without using the DEAP package. The performance improved, but the results continued to be very poor.

We segmented Table 7.2 in 2 sets of rows: the upper set of rows presents the absolute values of $\boldsymbol{f}$, and the average times to solve one instance; in the lower set of rows, we divided each $\boldsymbol{f}$ value by the maximum $\boldsymbol{f}$ value of the row.

Table 7.2 – Metaheuristics results with *surplus* = 1.2, a time limit of 3600s

	NM		**ACO**		**GRASP**		**TS**	
Scenarios	***f***	**run time (s)**	***f***	**run time (s)**	***f***	**run time (s)**	***f***	**run time (s)**
1	11.260	1	**11.578**	3	11.397	1	11.260	5
2	8.382	4	8.399	17	**8.402**	12	8.394	4
3	11.206	17	**11.293**	56	11.238	47	11.118	18
4	13.117	87	**13.253**	278	13.131	258	13.033	90
5	**13.322**	520	13.728	1516	13.396	1690	13.225	586
6	12.197	3582	**12.358**	3602	12.155	3292	11.214	2627
	f	**Speedup**	***f***	**Speedup**	***f***	**Speedup**	***f***	**Speedup**
Norm.	0.990	1.29	1.00	1.00	0.993	1.03	0.972	1.63

We colored in red the run times over 20 minutes (1200s).

Different meta-heuristics handle time constraints differently.

Genetic Algorithm (GA): This method requires evolving a population of solutions through multiple generations. The more generations needed to explore the solution space thoroughly, the longer it takes to run.

Ant Colony Optimization (ACO): ACO relies on many ants exploring various pheromone trails. Finding the best solution requires enough time for this exploration to happen.

Noising Methods (NM): These methods involve running many iterations at different noise levels. The number of iterations and noise levels directly impacts the running time.

Tabu Search (TS): Similar to other methods, TS requires numerous iterations and updates to its Tabu list, which can be time-consuming for complex problems.

Greedy Randomized Adaptive Search Procedure (GRASP): This method also relies on multiple iterations, exploring solutions from the Restricted Candidates List. The more iterations and list size, the longer it takes to find a solution.

It is important to emphasize that each tour problem is solved, and the result is saved for later best-tour selection.

Table 7.3 presents the overall result of the metaheuristics, extracted from Table 7.2.

Table 7.3 – Overall results

Method	Best scenarios	Worst scenarios	Worst run times (min)
NM	5	6	60
ACO	1, 3, 4	5, 6	25, 61
GRASP	2	5, 6	28, 55
TS	-	6	44
GA	-	1,2,3,4,5,6	did not solve

In (MESQUITA; SANCHES, 2024), these scenario numbers are different because in that article there was no scenario with the smaller aircraft.

7.3 *Shims* x Metaheuristics

From this point forward, dividing the total time limit by the quantity of tours yields the time limit for solving a tour. The time limit to solve a node is proportional to the volume to be embarked on at each node. It is different from the previous section, where we established the node limit at 0.7 seconds.

It is important to highlight that, for each table, all experiments with seven instances were run again with the methods under the test to enforce a fair comparison.

As we ventured into the realm of processed-based parallelism, the *Multiprocessing* Python package required the use of typed variables and their passing as arguments as Python dictionaries. This forced us to make a complete code refactor, which resulted in a great gain in efficiency for almost all metaheuristics (except GA).

We run the $surplus = 1.2$ with a time limit of $1200s$ experiments, which are presented in Table 7.4.

Table 7.4 – Metaheuristics results with $surplus = 1.2$, a time limit of $1200s$

	NM		ACO		GRASP		TS		Shims	
Scenarios	***f***	**time (s)**	***f***	**time (s)**	***f***	**time (s)**	***f***	**time (s)**	***f***	**time (s)**
1	12.33	1	12.81	5	12.76	1	13.16	1	**13.19**	1
2	8.10	81	**8.52**	145	8.48	82	8.48	81	8.47	1
3	11.70	310	**12.28**	594	12.22	261	12.21	260	12.10	4
4	12.84	985	**13.40**	1187	13.33	985	13.34	985	13.37	17
5	13.83	993	14.51	1206	14.42	999	14.41	999	**14.66**	21
6	50.26	920	52.04	1230	51.32	894	49.36	810	**52.17**	82
	f	**Speedup**	***f***	**Speedup**	***f***	**Speedup**	***f***	**Speedup**	***f***	**Speedup**
Norm.	0.955	1.11	0.994	1.00	0.985	1.36	0.971	1.39	**0.998**	**34.7**

It can be observed in Table 7.4 that, after the code refactoring, almost all methods got to acceptable results in less than 20 minutes.

It is also necessary to highlight the quality of the Shims and ACO metaheuristics, because both got three of the best results. Shims was remarkably faster because the main purpose of this research was to find the fastest possible solution method for the ACLP+RPDP.

As to the code refactoring, we applied some common techniques like optimizing algorithms, removing redundant code, improving variable and function names, modularizing code into smaller pieces, and applying known design patterns. We also had to set all variables to be statically typed to enhance memory access operations. It is simpler to see the effects of those changes when variables are statically typed.

(TRAINI *et al.*, 2021) conducted a large-scale study analyzing 20 software systems and found that refactoring can significantly affect execution time. They conclude that no single refactoring type is "safe" for performance, and caution is needed, especially for types that break down complex code structures.

We also followed some strategies to enhance the Python code performance:

- We moved to more efficient data structures, minimal use of global variables, and the use of built-in functions and libraries.
- Python's built-in functions and standard libraries, like *itertools* and *functools*, that are optimized for performance and can handle operations more efficiently than custom, manually looped solutions.
- Vectorization with *NumPy* arrays for numerical computations can lead to significant speedups due to *NumPy*'s internal optimizations and use of *C* libraries.
- Concurrency and parallelism can improve the performance of IO-bound or CPU-bound tasks.
- Caching results with memoization can prevent redundant calculations by storing and reusing results.

- Minimal looping over data, using list comprehensions, generator expressions, or map/filter functions, can reduce the overhead of loops.

7.4 *Shims* x MIP solver

We ran Algorithm 1 (the main process to solve the ACLP+RPDP) considering the 5 scenarios from Table 6.1, 3 values for *surplus* from $\{1.2, 1.5, 2.0\}$, 4 values for *tmax* from $\{240s, 1200s, 2400s, 3600s\}$, and two methods: a MIP solver with *Gurobi* and *Shims*.

For the MIP solver to be able to solve the largest possible number of tests without memory overflow, its parameter *MIPgap* was set to 1%. i.e., the relative difference between the primal and the dual solutions ($MIPgap = |z_P - z_D|/|z_P|$).

This shortens its run time, in addition to ensuring that its objective function f is at most 1% of the optimal solution. For more details, see `www.support.gurobi.com`.

For each *scenario*, *surplus* and *tmax* tested, 7 different instances were employed.

We indicate the adopted strategies of dedicating all the processing time to the $ntours = 2$ shortest tours or distributing it among all $ntours = K!$ tours.

The results obtained with $tmax = 3600s$, which is the highest tested run time limit, are in tables 7.5, 7.6 and 7.7, with *surplus* values of 1.2, 1.5 and 2.0, respectively.

It is also important to highlight that all methods, under this new code structure, are viable for operational use. But there is no doubt that the Shims' speedup puts it in an advantageous position among the other methods.

We mark with an **x** the cases where the MIP solver did not find a feasible solution within this run time limit or had to be aborted due to high random-access memory (RAM) usage.

After Gurobi's *MIPgap* parameter setting, it managed to solve two tours with a very short run time, but Shims was 7 to 40 times faster, having tied with Gurobi with the values of the **f** function.

Table 7.5 – Solutions with *surplus* = 1.2 and *tmax* = 3600s

ntours	*method*	*scenario*	**1**	**2**	**3**	**4**	**5**	**6**	**Normalized Speed-up**
2	MIP solver	f	12.81	8.53	11.79	13.14	13.52	12.36	0.9998
		time (s)	5	29	28	25	27	27	1.0
	Shims	f	12.80	8.54	11.78	13.06	13.51	12.34	0.9980
		time (s)	1	1	1	1	1	2	**22.7**
$K!$	MIP solver	f	12.81	8.60	12.20	13.66	15.00	x	0.9998
		time (s)	5	30	35	123	314	x	1.0
	Shims	f	12.81	8.61	12.28	13.46	14.99	13.35	0.9958
		time (s)	1	1	1	4	10	51	**7.49**

Table 7.6 – Solutions with *surplus* = 1.5 and *tmax* = 3600s

ntours	*method*	*scenario*	**1**	**2**	**3**	**4**	**5**	**6**	**Normalized Speed-up**
2	MIP solver	f	17.99	11.83	16.73	18.07	18.83	16.86	0.9996
		time (s)	6	55	64	39	40	88	1.0
	Shims	f	17.98	11.85	16.72	18.05	18.80	16.87	0.9993
		time (s)	1	1	1	2	2	2	**35.8**
$K!$	MIP solver	f	17.99	11.83	16.93	18.40	20.95	17.60	0.9999
		time (s)	6	63	59	195	472	2258	1.0
	Shims	f	17.98	11.85	16.91	18.36	20.93	17.50	0.9976
		time (s)	1	1	2	5	15	100	**23.8**

Table 7.7 – Solutions with *surplus* = 2.0 and *tmax* = 3600s

ntours	*method*	*scenario*	**1**	**2**	**3**	**4**	**5**	**6**	**Normalized Speed-up**
2	MIP solver	f	26.33	17.70	24.20	26.39	27.17	24.20	0.9995
		time (s)	10	168	98	79	70	72	1.0
	Shims	f	26.32	17.74	24.22	26.32	27.07	23.13	0.9896
		time (s)	1	1	2	2	3	4	**40.6**
$K!$	MIP solver	f	26.33	17.90	25.44	26.51	29.13	x	0.9994
		time (s)	9	178	143	378	862	x	1.0
	Shims	f	26.33	17.94	25.45	26.44	28.84	26.22	0.9970
		time (s)	1	1	3	10	31	196	**34.7**

From these data, we can draw some conclusions:

1. The strategy of testing all $K!$ tours often provide a better-quality solution, even with less time on each node. This shows that the four sub-problems are interconnected in such a way that it is not enough to solve them separately.
2. The MIP solver fails in some cases when $scenario = 5$ and the strategy is to check all $K!$ tours. This occurs because the run time limit per node is smaller and there tend to be more *Packed* in the aircraft, reducing the space for allocating items and making the solution difficult.
3. When the MIP solver finishes, it finds the best solution, but the one obtained by *Shims* reaches at least 98.96% of that value. Considering only the strategy of testing all $K!$ tours, this value increases to 99.58%.
4. *Shims* always finds a solution, being 7 to 40 times faster.
5. All run times are much lower than the limit because the solution on many nodes can be fast. Anyway, in all the tests performed, the maximum time spent by *Shims* did not reach 4 minutes. On the other hand, when $scenario = 5$ and $surplus = 1.5$, the MIP solver spent almost 40 minutes.

Table 7.8 shows the results obtained with the strategy of testing the $K!$ tours in all scenarios with different *tmax*. We can observe more cases where the MIP solver fails, even in smaller scenarios. When the MIP solver finishes, *Shims* finds a solution of similar quality (99% or better). In all cases, *Shims* finds a solution in less than 4 minutes, being ≈ 27 times faster.

In Table 7.8 we omitted scenarios 1, 2, and 3 because all methods managed to solve them.

Table 7.8 – Solutions testing all $K!$ tours with different run time limits

		surplus		**1.2**			**1.5**			**2.0**	
method	*tmax*	*scenario*	**4**	**5**	**6**	**4**	**5**	**6**	**4**	**5**	**6**
MIP solver	240s	f	13.67	x	x	18.10	x	x	x	x	x
		time (s)	124	x	x	200	x	x	x	x	x
	1200s	f	13.31	10.00	x	18.25	20.64	x	26.49	27.97	x
		time (s)	129	320	x	190	304	x	384	579	x
	2400s	f	**13.67**	**15.00**	**13.41**	**18.03**	**20.95**	**17.60**	**26.15**	**29.13**	x
		time (s)	134	310	1520	199	461	2258	383	786	x
	3600s	f	13.66	15.00	13.41	18.01	20.95	17.60	26.15	29.13	x
		time (s)	123	314	1520	195	472	2258	378	862	x
Shims	240s	f	13.46	14.99	13.35	17.99	20.93	17.50	26.14	28.84	**26.22**
		time (s)	4	10	51	5	15	100	10	31	196

The actual RAM consumption of the MIP solver was over 8.5 GB, and all of *Shims*'s executions consumed at most 1.5 GB of RAM.

7.5 *Shims* with more than 7 nodes

These last results do not correspond to practical cases of air transport, as tours where $K > 6$ very rarely occur. However, it is possible to see that the *Shims* maintains robust behavior as the number of nodes grows; that is, it could be adapted to similar contexts (ships and trucks, for example), where there may be more nodes.

Considering real data from the 15 main Brazilian airports, we implemented a GA-based TSP heuristic. We implemented this heuristic with DEAP (Distributed Evolutionary Algorithms in Python), an evolutionary computation framework. For more details, see (FORTIN *et al.*, 2012) and `github.com/deap/deap`.

We triggered DEAP to return 100 solutions, which we filtered for the diverse ones (which returned from 30 to 50 diverse solutions) to be solved by *Shims* and selected those with the best benefit-cost ratio. This procedure reduced the time to generate solutions to the 7-node scenario to less than 100 ms and 15 nodes to less than 800 ms, making the overall process still faster than that of (MESQUITA; SANCHES, 2024).

This is an innovative approach to solving the Traveling Salesman Problem (TSP) with a practical twist for handling a limited number of nodes efficiently. It moves beyond the exhaustive search method for small node sets (K!) through optimization with a GA-based TSP heuristic.

Figure 7.1 shows the run time curve of *Shims* as the number K of nodes increases, indicating at each point the value found for the objective function f. Runtime and f are the averages obtained from 7 instances generated with $surplus = 2.0$ for each value of K. In Figure 7.2, we indicate one of the tours found by this TSP heuristic when $K = 15$.

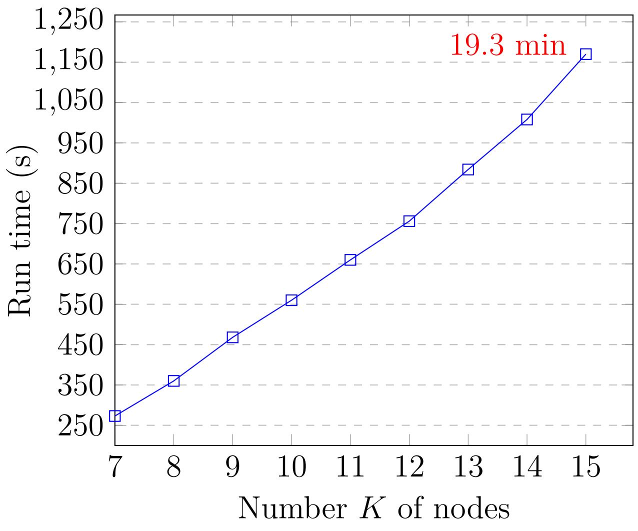


Figure 7.1 – *Shims* performance with *surplus* = 2.0

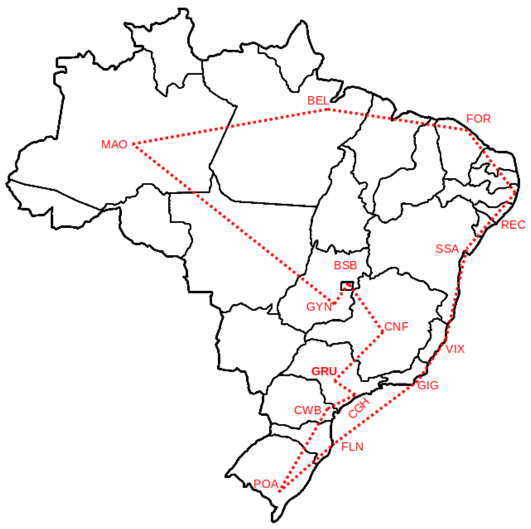


Figure 7.2 – A tour where $K = 15$

7.6 Multiprocessing results

(MESQUITA; SANCHES, 2024) reflect the results with all implementations via sequential algorithms. From this point on, we present the results of the application of process-based parallelism to enhance the algorithm's performance and, as will be noticed, the quality of the results.

In this section, we raised three hypotheses for improvements:

(1) Computer parallelism could improve *Shims*'s performance: as *mpShims* was twice as faster as *Shims*, this hypothesis was confirmed.

(2) The procedure for 3-D packing generated an unfit item rate of, on average, 13%. We are conscious that this unfit rate could be smaller if we had chosen (PAQUAY *et al.*, 2018a) approach, but as we are concerned with an operational application where the running time is crucial, we stick to faster approximate approaches like the *First Fit Decreasing* (first orientation that fits) and the *Best Fit* (best pivot point if more than one is available), like proposed by (DUBE; KANAVATHY, 2006).

(3) The distances to move pallets on unloading in the next node could be minimized to benefit ground handling costs. This hypothesis was confirmed by the results on Table 7.11.

mpShims consider that, theoretically, all items selected for the pallets would fit. But, due to the variability in item dimensions, an actual complete fit is not possible, causing some unfit items to be excluded from the previous solution.

This solution phase was a multiprocessing 3-D packing with each pallet as a process, which decreased this phase's run time by approximate a $n-1$ factor where n is the number of processes. This was possible because each packing process is completely independent; there is no multiprocessing shared memory. It is also important to highlight that, after the items were excluded, the CG deviation was kept within its operational and feasible range.

It is important to emphasize that, as the *Multiprocessing 3-D packing* is a post-optimization procedure, it benefited all solution methods.

The rationale behind this is that, in sequential mode (*Shims*), items with smaller volumes (higher edge attractiveness $\theta_{ij}^{\pi_k}$, Equation 4.3) tend to be assigned first to central pallets, and items with bigger volumes tend to be assigned to the ramp door or to the forward-positioned pallets; these bigger items are more difficult for the 3-D packer actions. On *mpShims*, as all pallets are built in parallel, the size distribution is more uniform, with a better size diversity to be allocated to each pallet, which is less difficult for the 3-D packer actions due to the variety of item dimensions for packing. This explains why *mpShims* exceeded *Shims*' results after the *Multiprocessing 3-D packing*.

We tested all methods with 126 different instances: six operational scenarios with 2 aircraft sizes and 3 to 7 nodes; and seven randomly generated item sets for three volume surpluses (1.2, 1.5, and 2.0 times aircraft volume capacities).

Results before 3-D packing

During the solution process of an integer programming problem, relaxation refers to creating a linear program (LP) by removing the integer constraints from the original problem. This LP acts as a relaxed version of the problem, allowing decision variables to take on continuous values within their bounds. In this context, we call it *Relaxed MIP*.

We ran Algorithm 2 in the 6 scenarios described in Table 6.1, considering 3 methods for node-by-node solution: the state-of-the-art *Gurobi MIP* solver, *Shims*, and *mpShims* (Algorithm 15).

In generating the items in each node, we consider 3 values for the parameter *surplus*. The results obtained for the function f, with the corresponding run time in seconds, are shown in Table 7.9 (*surplus* $= 1.2$, *surplus* $= 1.5$, and *surplus* $= 2.0$).

Note that, in Table 7.9 all *Relaxed MIP* $f(G^*)$ value are italicized to indicate that these values may not be optimal or feasible as the decision variables are not integers.

Table 7.9 – Volume surpluses, methods and scenarios results **before** 3-D packing

Volume *surplus*	**Solution** *method*	**Results**	**1**	**2**	**3**	**4**	**5**	**6**	**Norm. Speedup**
1.2	*Relaxed MIP*	*f*	*12.23*	*8.67*	*12.67*	*15.36*	*16.29*	*16.20*	-
		time (s)	3	13	20	78	198	1103	-
	Shims	*f*	10.34	**8.50**	**12.42**	**15.06**	**15.97**	**15.88**	**0.97**
		time (s)	< 1	3	5	20	58	352	1.00
	mpShims	*f*	**12.23**	7.53	12.28	13.20	13.71	13.44	0.91
		time (s)	< 1	2	2	8	22	120	**2.80**
1.5	*Relaxed MIP*	*f*	*17.01*	*12.45*	*18.02*	*22.03*	*23.09*	*22.35*	-
		time (s)	4	20	32	115	296	1685	-
	Shims	*f*	14.68	**12.21**	**17.67**	**21.01**	**22.64**	**21.91**	**0.99**
		time (s)	< 1	2	3	12	33	209	1.00
	mpShims	*f*	**16.12**	11.81	17.48	17.66	19.81	17.61	0.90
		time (s)	< 1	1	2	7	20	120	**1.70**
2.0	*Relaxed MIP*	*f*	*24.56*	*17.47*	*25.19*	*27.94*	*29.90*	*27.60*	-
		time (s)	4	23	34	119	317	1762	-
	Shims	*f*	21.31	**17.35**	24.24	**27.39**	**29.31**	**27.06**	**0.98**
		time (s)	< 1	2	4	14	35	220	1.00
	mpShims	*f*	**24.33**	17.26	**24.70**	24.81	27.71	24.93	0.96
		time (s)	< 1	2	3	9	26	151	**1.70**

A foundational idea in linear integer programming is that the upper bound is the objective function value of a linear programming relaxation of the problem, demonstrating that no integer viable solution can have an objective value greater (in maximization) than this (VANDERBECK; WOLSEY, 2015).

We solved all scenarios and instances with a MIP solver in a relaxed linear model. This is good for the reader to have an idea of the real performance differences between serial and process-based parallel *Shims*.

Shims had a quality performance very close to the *Relaxed MIP* solver upper bound solution, and, as to the **Speedup**, on average, *mpShims* was 2 times faster than *Shims*. Theoretically, it could be more, but due to the burden of sharing, for reading and writing, the CG deviation and the list of items among the processes (pallets), the *race condition* management limited *mpShims* performance.

Results after 3-D packing

Table 7.10 presents the results of the *Relaxed MIP* solver compared to *Shims* and *mpShims*, after the post-optimization with the multiprocessing 3-D packing.

Although the *Relaxed MIP* solver results are not always feasible, the multiprocessing 3-D packing turns them feasible regarding dimensions and weight. So, in Table 7.10, we considered it's results in the performance comparisons.

Table 7.10 – Volume surpluses, methods and scenarios results **after** 3-D packing (3Dp)

Volume	**Solution**		**Scenarios**						**Norm.**
surplus	*method+3Dp*	**Results**	**1**	**2**	**3**	**4**	**5**	**6**	**Speedup**
1.2	*Relaxed MIP*	*f*	11.47	7.24	**10.78**	12.06	12.97	12.44	0.95
		time (s)	12	51	81	258	601	3161	1.15
	Shims	*f*	9.05	5.11	9.75	10.96	12.86	13.25	0.86
		time (s)	16	68	94	323	719	3549	1.00
	mpShims	*f*	**12.14**	**7.29**	10.36	**12.63**	**14.04**	**13.92**	**0.99**
		time (s)	9	36	54	192	427	2118	**1.70**
1.5	*Relaxed MIP*	*f*	15.83	**10.08**	**15.14**	**16.87**	18.58	17.52	0.97
		time (s)	20	83	142	362	996	4741	1.22
	Shims	*f*	11.55	7.61	12.28	15.00	17.55	17.57	0.84
		time (s)	25	107	156	498	1217	5743	1.00
	mpShims	*f*	**17.05**	10.01	15.01	16.75	**20.09**	**17.85**	**1.00**
		time (s)	18	64	101	351	828	3879	**1.50**
2.0	*Relaxed MIP*	*f*	23.73	**15.23**	21.96	24.22	26.69	24.29	0.99
		time (s)	28	99	185	648	1680	8774	1.00
	Shims	*f*	16.28	10.93	18.16	22.35	25.12	24.53	0.85
		time (s)	37	141	196	653	1453	6851	1.22
	mpShims	*f*	**24.52**	15.18	**21.99**	**24.61**	**26.94**	**24.83**	**1.00**
		time (s)	27	87	146	496	1080	5160	**1.63**

The average value of f obtained by *mpShims* was 0.997, followed by the *Relaxed MIP* with 0.971. *Shims* obtained 0.851, on average.

As to the performance, the average **Speedup** obtained by *mpShims* was 1.61, followed by the *Relaxed MIP* with 1.12. *Shims* reached 1.07, on average.

The *Relaxed MIP solver* results were not always feasible. The infeasible results were made feasible by the *Multiprocessing 3-D packing* method.

In terms of quality and speed, *mpShims* with the *Multiprocessing 3-D packing* superseded the sequential *Shims* results to solve the ACLP-RPDP.

7.7 Minimization of the pallet unloading movements

Although it was expected at most $\mathcal{O}(m^2)$ execution steps, as the number of pallets was small (7 or 18) and it was solved with *Gurobi*, the increased time was negligible (less than 1s).

Table 7.11 – Average distances minimization

Scenario	**Before (m)**	**After (m)**	**Improvement (%)**
1	32.70	28.70	−7.30
2	193.30	188.90	−6.30
3	165.10	153.30	−7.10
4	137.10	132.00	−3.70
5	126.70	122.00	−3.80
6	115.30	110.40	−4.20

Table 7.11 presents the movement minimization in the cargo bay of the next node destined pallets to the ramp door. Columns **Before** and **After** refer to the application of the current method. On average, the distances from the pallets and the last ramp door position were diminished by 5% along the tour.

8 Conclusions

In this thesis, we addressed the complex challenge of optimizing air cargo load planning, routing, pickup, and delivery (ACLP+RPDP). This encompassed intricate subproblems like air cargo palletization, weight and balance, vehicle routing, and pickup/delivery logistics. We analyzed the problem in realistic scenarios involving aircraft payloads of 26,000kg and 75,000kg, employing the standard 463L Master Pallet and considering multi-leg delivery/pickup tours.

We developed a comprehensive process for planning aircraft loading and routing in multi-leg missions with return to base. This required tailoring constraints for each subproblem and implementing an integer programming approach with the Gurobi MIP solver. While this initial attempt wasn't able to solve the full ACLP+RPDP, it paved the way for further development.

Following extensive experimentation with seven methods, six scenarios, and multiple instances (totaling hundreds of experiments with synthetic data based on real Brazilian Air Force data), we identified limitations of existing approaches. Five well-known metaheuristics specifically designed for ACLP+RPDP were then tested.

However, we sought an even more efficient solution. This led to the creation of a novel heuristic named *Shims*. *Shims* boasts exceptionally low search times, making it ideal for multi-leg airlift planning. Notably, it surpassed existing methods in most scenarios, demonstrating its effectiveness.

This thesis offers significant academic contributions:

(1) **The creation of a fast heuristic:** Although *Shims* was already fast, we transformed it into a parallel computing heuristic with a 3-D packing post-optimization process, guaranteeing all items fit onto pallets. This delivers fast and effective solutions for diverse problem sizes. It also has potential for application as an online heuristic for dynamic tours and cargo updates. Additionally, it could be integrated with real-time data analytics for further refinement of loading and routing efficiency.

(2) **Mathematical modeling and validation:** We designed and extensively tested and validated the mathematical model using an integer programming tool.

(3) **A comprehensive solution process:** This innovative approach ensures near-optimal cargo distribution on pallets within the aircraft cargo bay, guaranteeing balance and maximizing efficiency while minimizing fuel consumption. This output, including the tour plan and pallet building/arrangement, significantly enhances airlift safety, ground operation efficiency, and ensures accurate delivery.

(4) **Adaptability:** Our methodology extends beyond aircraft, proving applicable to maritime, road, and rail logistics with adjustments to model constraints and space utilization.

These points clearly align with academic contributions as they contribute to the theoretical foundation, offer new methods, and extend the existing knowledge in the field.

Furthermore, our testing validated three **key hypotheses**:

(1) Parallel computing improves *Shims* performance.

(2) A 3-D packing method becomes operational due to the faster parallel computing heuristic.

(3) Minimizing pallet movement within the cargo bay proves successful.

We also utilized the *irace* package to fine-tune *Shims* parameters, achieving optimal average performance.

The practical impacts include:

(1) A rapid tool for balanced cargo loading and efficient item selection.

(2) Seamless pickup and delivery operations.

(3) Optimized routes maximizing efficiency and minimizing fuel consumption.

(4) A lightweight tool operable on standard computers.

These impacts directly relate to how this research can be applied outside of an academic context, affecting industry practices, operational efficiency, and environmental sustainability.

Future Research Directions:

(1) Optimizing scenarios with non-standard cargo and alternative cargo bay packing methods.

(2) Solving ACLP+RPDP for multiple and diverse aircraft simultaneously.

(3) Addressing sequencing problems in cargo aircraft with two doors.

(4) Integrating real-time data analytics for further refinement of loading and routing efficiency.

(5) Application of the proposed methods to other transportation modes.

This thesis presents a novel approach to ACLP+RPDP, offering valuable contributions to both academic research and practical applications in airlift operations and beyond. The developed methodology and *Shims* heuristic demonstrate significant potential for improving efficiency, safety, and cost-effectiveness in various transportation and logistics domains. Future research directions hold promise for further advancements in this critical field.

Bibliography

ALONSO, M. T.; ALVAREZ-VALDES, R.; PARRENO, F. A grasp algorithm for multi-container loading problems with practical constraints. **A Quarterly Journal of Operations Research**, v. 18, n. 49-72, 2019.

BORTFELDT, A.; WÄSCHER, G. Constraints in container loading: A state-of-the-art review. **European journal of operational research**, Elsevier B.V, Amsterdam, v. 229, n. 1, p. 1–20, 2013. ISSN 0377-2217.

BOWEN, F.; CHEN, X.; DI, X. Learn to tour: Operator design for solution feasibility mapping in pickup-and-delivery traveling salesman problem. **arXiv.org**, Cornell University Library, arXiv.org, Ithaca, 2024. ISSN 2331-8422.

BRANDT, F.; NICKEL, S. The air cargo load planning problem - a consolidated problem definition and literature review on related problems. **European Journal of Operational Research**, Elsevier B.V, v. 275, n. 2, p. 399–410, 2019. ISSN 0377-2217.

BRESHEARS, C. **Chapter 8: A Thread Monkey's Guide to Writing Parallel Applications**. [S.l.]: Kindle 1st Edition, 2009. 271 p.

BROSH, I. Optimal cargo allocation on board a plane: a sequential linear programming approach. **European Journal of Operational Research**, Elsevier, v. 8(1), n. 40-46, 1981.

CHAN, F.; BHAGWAT, R.; KUMAR, N.; TIWARI, M.; LAM, P. Development of a decision support system for air-cargo pallets loading problem: A case study. **Expert Systems with Applications**, v. 31(3), n. 472-485, 2006.

CHARON, I.; HUDRY, O. The noising method: a new method for combinatorial optimization. **Operations research letters**, Elsevier B.V, Amsterdam, v. 14, n. 3, p. 133–137, 1993. ISSN 0167-6377.

CHARON, I.; HUDRY, O. The noising methods: A generalization of some metaheuristics. **European Journal of Operational Research**, Elsevier B.V, v. 135, n. 1, p. 86–101, 2001. ISSN 0377-2217.

CHENGUANG, Y.; HU, L.; YUAN, G. Load planning of transport aircraft based on hybrid genetic algorithm. **MATEC Web of Conferences**, EDP Sciences, Les Ulis, v. 179, p. 1007, 2018. ISSN 2261-236X.

CHU, P.; BEASLEY, J. A genetic algorithm for the multidimensional knapsack problem. **Journal of heuristics**, Springer Nature, DORDRECHT, v. 4, n. 1, p. 63–86, 1998. ISSN 1381-1231.

DORIGO, M. **Optimization, Learning and Natural Algorithms**. PhD Thesis — Politecnico di Milano, Italia, 1992.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. The ant system: optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 26, n. 29-41, 1996.

DUBE, E.; KANAVATHY, L. R. Optimizing three-dimensional bin packing through simulation. In: **Modelling, simulation, and optimization**. Durban, South Africa: [s.n.], 2006. p. 507–034.

ES, G. van. **Analysis of aircraft weight and balance related safety occurrences**: une revue des langages existants (a review of existing languages). [S.l.], 2007. (RR 95-02).

FEILLET, D.; DEJAX, P.; GENDREAU, M. Traveling salesman problems with profits. **Transportation Science**, INFORMS, Linthicum, MD, USA, v. 39, n. 2, p. 188?205, may 2005. ISSN 1526-5447. Available in: <https://doi.org/10.1287/trsc.1030.0079>.

FEO, T. A.; RESENDE, M. G. C. A probabilistic heuristic for a computationally difficult set covering problem. **Operations Research Letters**, v. 8, n. 67-71, 1989.

FIDANOVA, S. **Ant Colony Optimization and Multiple Knapsack Problem**. [S.l.]: J–Ph. Renard editor, 2006. 498–509 p.

FOK, K.; CHUN, A. Optimizing air cargo load planning and analysis. In: **Proceedings of the international conference on computing, communications and control technologies**. [S.l.: s.n.], 2004. p. 0.

FORTIN, F.-A.; De Rainville, F.-M.; GARDNER, M.-A.; PARIZEAU, M.; GAGNÉ, C. DEAP: Evolutionary algorithms made easy. **Journal of Machine Learning Research**, v. 13, p. 2171–2175, jul 2012.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers and Operations Research**, v. 13, n. 533-549, 1986.

GUTIERREZ, A.; DIEULLE, L.; LABADIE, N.; VELASCO, N. A hybrid metaheuristic algorithm for the vehicle routing problem with stochastic demands. **Computers and operations research**, Elsevier Ltd, New York, v. 99, p. 135–147, 2018. ISSN 0305–0548.

HANDBOOK of Metaheuristics. [S.l.]: Springer, 2003. ISBN 978-1-4020-7263-5.

HANDBOOK of Metaheuristics. [S.l.]: Springer, 2010. ISBN 978-1-4419-1665-5.

HEIDELBERG, K. R.; PARNELL, G. S.; AMES, J. E. Automated air load planning. **Naval Research Logistics**, v. 45, n. 751-768, 1998.

HILL, M. D.; MARTY, M. R. Amdahl's law in the multicore era. **IEEE Computer**, IEEE Computer Society, v. 41, n. 7, p. 33–38, 2008.

HOUGHTON, E. L.; CARPENTER, P. W. **Aerodynamics for Engineering Students (5th ed.).** [S.l.]: Butterworth Heinmann, 2003. 18 p. ISBN ISBN 0-7506-5111-3.

IATA. International Air Transport Association: **The importance of air transports to Brazil**. 2017. Available in: <https://www.iata.org/contentassets/870a246ffa224bb8af8146ecb9fce274/benefits-of-aviation-brazil-2017.pdf>. Accessed in: 2022/04/17.

JOHNSON, D. S.; GAREY, M. R. A 7160 theorem for bin packing. **Journal of Complexity**, v. 1, n. 1, p. 65–106, 1985. ISSN 0885-064X.

JU, B.; KIM, M.; MOON, I. Vehicle routing problem considering reconnaissance and transportation. **Sustainability (Basel, Switzerland)**, MDPI AG, Basel, v. 13, n. 6, p. 3188, 2021. ISSN 2071-1050.

KALUZNY, B. L.; SHAW, R. H. A. D. Optimal aircraft load balancing. **International transactions in operational research**, Blackwell Publishing Ltd, Oxford, UK, v. 16, n. 6, p. 767–787, 2009. ISSN 0969-6016.

LAABADI, S.; NAIMI, M.; AMRI, H. E.; ACHCHAB, B. The 0/1 multidimensional knapsack problem and its variants: A survey of practical models and heuristic approaches. **American Journal of Operations Research**, v. 8, n. 395–439, 2018. Https://doi.org/10.4236/ajor.2018.85023.

LARSEN, O.; MIKKELSEN, G. An interactive system for the loading of cargo aircraft. **European Journal of Operational Research**, North-Holland lh.tblishing Company, 1979.

LIMBOURG, S.; SCHYNS, M.; LAPORTE, G. Automatic aircraft cargo load planning. **Journal of the Operational Research Society**, v. 63, n. 1271-1283, 2012.

LURKIN, V.; SCHYNS, M. The airline container loading problem with pickup and delivery. **European Journal of Operational Research**, Elsevier, v. 244(3), n. 955-965, 2015.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, Elsevier, v. 3, n. C, p. 43–58, 2016. ISSN 2214-7160.

MANFRIN, M.; BIRATTARI, M.; STÜTZLE, T.; DORIGO, M. **Parallel Ant Colony for the Traveling Salesman Problem**. [S.l.]: Springer, 2006. 224–234 p.

MESQUITA, A. C. P.; CUNHA, C. B. Uma heurística integrada baseada na meta-heurística busca dispersa para problemas de roteirização de veículos com coleta e entrega simultâneas no contexto da força aérea brasileira (An integrated heuristic based on the Scatter Search metaheuristic for vehicle routing problems with simultaneous delivery and pickup in the context of the Brazilian Air Force). **TRANSPORTES**, Revista Transportes, v. 19, n. 33–42, 2011.

MESQUITA, A. C. P.; SANCHES, C. A. A. Air cargo load and route planning in pickup and delivery operations. **Expert Systems with Applications**, n. 249, p. 123711, 2024. ISSN 0957-4174. Available in: <https://www.sciencedirect.com/science/article/pii/S0957417424005773>.

MIGUEL, J.; MACALINTAL, V.; UBANDO, A. T. Optimal aircraft payload weight and balance using fuzzy linear programming model. **Chemical Engineering Transactions**, v. 103, p. 613–618, 2023.

MONGEAU, M.; BES, C. Optimization of aircraft container loading. **IEEE Transaction on Aerospace and Electronic Systems**, v. 39(1), n. 140-150, 2003.

NACCACHE, S.; CÔTÉ, J.-F.; COELHO, L. C. The multi-pickup and delivery problem with time windows. **European Journal of Operational Research**, Elsevier B.V, v. 269, n. 1, p. 353–362, 2018. ISSN 0377-2217.

NG, K. Y. K. A multicriteria optimization approach to aircraft loading. **Operations Research**, v. 40, p. 1200–1205, 1992.

NIAR, S.; FREVILLE, A. A parallel tabu search algorithm for the 0-1 multidimensional knapsack problem. In: **Proceedings 11th International Parallel Processing Symposium**. [S.l.: s.n.], 1997. p. 512–516.

OCLOO, V.; FÜGENSCHUH, A.; PAMEN, O. **A New Mathematical Model for a 3-D Container Packing Problem**. 01 2020. 28 p.

OLJA, C.; SLOBODAN, G.; LJUBISA, V.; PETAR, M. Analysis of aircraft weight and balance related safety occurrences. In: . [s.n.], 2010. Available in: <https://api.semanticscholar.org/CorpusID:115323939>.

ÖZTAs, T.; TUs, A. egül. A hybrid metaheuristic algorithm based on iterated local search for vehicle routing problem with simultaneous pickup and delivery. **Expert Systems with Applications**, v. 202, p. 117401, 2022. ISSN 0957-4174. Available in: <www.sciencedirect.com/science/article/pii/S095741742200745X>.

PACHECO, P.; MALENSEK, M. **An Introduction to Parallel Programming 2nd Edition**. [S.l.]: Morgan Kaufmann, 2021. 496 p. ISBN 978-0128046050.

PANTUZA, G.; SOUZA, M. C. de. Formulations and a lagrangian relaxation approach for the prize collecting traveling salesman problem. **International transactions in operational research**, Wiley Subscription Services, Inc, Oxford, v. 29, n. 2, p. 729–759, 2022. ISSN 0969-6016.

PAQUAY, C.; LIMBOURG, S.; SCHYNS, M. A tailored two-phase constructive heuristic for the three-dimensional multiple bin size bin packing problem with transportation constraints. **European Journal of Operational Research**, v. 267, n. 1, p. 52–64, 2018. ISSN 0377-2217. Available in: <https://www.sciencedirect.com/science/article/pii/S0377221717310214>.

PAQUAY, C.; LIMBOURG, S.; SCHYNS, M.; OLIVEIRA, J. F. Mip-based constructive heuristics for the three-dimensional bin packing problem with transportation constraints. **International Journal of Production Research**, Taylor and Francis, v. 56, n. 4, p. 1581–1592, 2018. Available in: <https://doi.org/10.1080/00207543.2017.1355577>.

PAQUAY, C.; SCHYNS, M.; LIMBOURG, S. A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. **International Transactions in Operational Research**, v. 23, n. 187-213, 2016.

Python Software Foundation. **Python 3.10.10 documentation: Process-based parallelism**. 2022. Available in: <https://docs.python.org/3/library/multiprocessing.html>.

REINELT, G. **The Traveling Salesman**: Computational solutions for tsp applications. [S.l.]: Springer Berlin, Heidelberg, 1994.

ROESENER, A.; BARNES, J. An advanced tabu search approach to the dynamic airlift loading problem. **Logistics Research**, v. 9(1), n. 1–18, 2016.

ROESENER, A.; HALL, S. A nonlinear integer programming formulation for the airlift loading problem with insufficient aircraft. **Journal of Nonlinear Analysis and Optimization: Theory and Applications**, v. 5, n. 1, p. 125–141, Jan. 2014. Available in: <http://www.math.sci.nu.ac.th/ojs302/index.php/jnao/article/view/298>.

SHAH, S. **Genetic Algorithm for the 0/1 Multidimensional Knapsack Problem**. 2020.

TANG, C.-H. A scenario decomposition-genetic algorithm method for solving stochastic air cargo container loading problems. **Transportation research. Part E, Logistics and transportation review**, Elsevier India Pvt Ltd, Exeter, v. 47, n. 4, p. 520–531, 2011. ISSN 1366-5545.

TRAINI, L.; POMPEO, D. D.; TUCCI, M.; LIN, B.; SCALABRINO, S.; BAVOTA, G.; LANZA, M.; OLIVETO, R.; CORTELLESSA, V. How software refactoring impacts execution time. **ACM Trans. Softw. Eng. Methodol.**, Association for Computing Machinery, New York, NY, USA, v. 31, n. 2, dec 2021. ISSN 1049-331X. Available in: <https://doi.org/10.1145/3485136>.

VANCROONENBURG, W.; VERSTICHEL, J.; TAVERNIER, K.; BERGHE, G. V. Automatic air cargo selection and weight balancing: A mixed integer programming approach. **Transportation research. Part E, Logistics and transportation review**, Elsevier India Pvt Ltd, Exeter, v. 65, p. 70–83, 2014. ISSN 1366-5545.

VANDERBECK, F.; WOLSEY, L. A. **Introduction to linear optimization and interior point methods**. [S.l.: s.n.], 2015. (Springer Series in Operations Research and Financial Engineering). ISBN 978-1489979560.

VERSTICHEL, J.; VANCROONENBURG, W.; SOUFFRIAU, W.; BERGHE, G. V. A mixed integer programming approach to the aircraft weight and balance problem. **Procedia Social and Behavioral Sciences**, v. 20, n. 1051-1059, 2011.

WONG, E. Y.; LING, K. K. T. A mixed integer programming approach to air cargo load planning with multiple aircraft configurations and dangerous goods. In: **2020 7th International Conference on Frontiers of Industrial Engineering (ICFIE)**. [S.l.: s.n.], 2020. p. 123–130.

WONG, E. Y. C.; MO, D. Y.; SO, S. Closed-loop digital twin system for air cargo load planning operations. **International Journal of Computer Integrated Manufacturing**, Taylor and Francis, v. 34, n. 7-8, p. 801–813, 2021. Available in: <https://doi.org/10.1080/0951192X.2020.1775299>.

XIA, Y.; FU, Z.; PAN, L.; DUAN, F. Tabu search algorithm for the distance-constrained vehicle routing problem with split deliveries by order. **PloS one**, Public Library of Science, United States, v. 13, n. 5, p. e0195457–e0195457, 2018. ISSN 1932–6203.

ZHAN, S.; WANG, L.; ZHANG, Z.; ZHONG, Y. Noising methods with hybrid greedy repair operator for 0-1 knapsack problem. **Memetic Computing**, v. 12, n. 37-50, 2020.

ZHAO, X.; DONG, Y.; ZUO, L. A combinatorial optimization approach for air cargo palletization and aircraft loading. **Mathematics**, v. 11, n. 13, p. 1–16, 2023.

ZHAO, X.; YUAN, Y.; DONG, Y.; ZHAO, R. Optimization approach to the aircraft weight and balance problem with the centre of gravity envelope constraints. **IET Intelligent Transport Systems**, IET, v. 15, n. 10, p. 1269–1286, 2021. Available in: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/itr2.12096>.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO TD	2. DATA 08 de maio de 2024	3. DOCUMENTO N. DCTA/ITA/TD-014/2024	4. N. DE PÁGINAS 110

5. TÍTULO E SUBTÍTULO:
Air Cargo Load and Route Planning in Pickup-Delivery Operations

6. AUTOR:
Antonio Celio Pereira de Mesquita

7. INSTITUIÇÃO
Instituto Tecnológico de Aeronáutica – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:
Air Cargo Load Planning; OR in Airlines; Vehicle Routing; Weight and Balance; Pickup and Delivery; Mixed-Integer Programming

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:
Carga aérea; Planejamento de rotas; Métodos heurísticos; Logística; Otimização; Programação matemática; Computação.

10. APRESENTAÇÃO: (X) **Nacional** ( ) **Internacional**
ITA, São José dos Campos. Curso de Doutorado. Programa de Pós-Graduação em Engenharia Eletrônica e Computação. Área de Informática. Orientador: Prof. Dr. Carlos Alberto Alonso Sanches. Defesa em 29/04/2024. Publicada em 08/05/2024.

11. RESUMO:
This thesis addresses the critical challenge of planning and executing aerial pickup and delivery of goods across Brazil's vast territory, a complex process often jeopardized by rapid loading requirements and the risks of cargo unbalancing and misdelivery. Faced with the urgency of rapid takeoffs and the complexities of distributing goods over Brazil's extensive territory, this research tackles the unexplored problem of optimizing air cargo load planning with routing, pickup, and delivery, balancing utility scores, cargo weight, and fuel consumption against the constraints of aircraft capacity and center of gravity. We introduce a comprehensive model that integrates air palletization, weight and balance, pickup and delivery, and vehicle routing into a unified framework. Through rigorous mathematical modeling and the development of innovative sequential and parallel heuristics, our approach minimizes both ground handling times and fuel consumption, directly contributing to reduced carbon emissions. Our methods were validated through extensive testing with both commercial solvers and metaheuristics, using data reflective of real-world scenarios from the Brazilian Air Force. These data contain a variety of items, like aircraft components and supplies, medication, and other supplies for remote population needs in isolated regions in the Brazilian territory, as well as for resupplying the defense forces near the Brazilian border. Despite issues such as incomplete data and historical records, our findings demonstrate the practical applicability and adaptability of our solutions to a broad range of logistical and optimization challenges. This research not only advances the field of aerial logistics but also offers adaptable tools for tackling diverse optimization problems across military and civilian contexts, promising significant improvements in efficiency and sustainability. Future work may explore the integration of real-time data analytics to further refine loading and routing efficiency.

12. GRAU DE SIGILO:
(X) **OSTENSIVO** ( ) **RESERVADO** ( ) **SECRETO**